



# **Revisões sobre Probabilidades, Variáveis Aleatórias e Processos Estocásticos**

Desempenho e Dimensionamento de Redes

Prof. Amaro de Sousa ([asou@ua.pt](mailto:asou@ua.pt))

DETI-UA, 2017/2018

# Experiência aleatória

- Numa experiência aleatória, o espaço de resultados,  $S$ , é o conjunto de todos os resultados possíveis da experiência
- Qualquer subconjunto  $E$  do espaço de resultados  $S$  designa-se por evento ou acontecimento
- Dados dois acontecimentos  $E$  e  $F$ , podem-se definir outros acontecimentos:
  - A união dos acontecimentos,  $E \cup F$
  - A intersecção dos acontecimentos,  $EF$
- Quando  $EF = \emptyset$  ( $\emptyset$  é o conjunto vazio) os acontecimentos dizem-se mutuamente exclusivos
- O complemento de  $E$ ,  $E^c$ , é o conjunto de elementos de  $S$  que não pertencem a  $E$

# Probabilidades definidas sobre acontecimentos

- Para cada acontecimento  $E$  de  $S$ , admite-se a existência de um número  $P(E)$  designado por probabilidade de  $E$ , que satisfaz as seguintes condições:

$$(1) \quad 0 \leq P(E) \leq 1$$

$$(2) \quad P(S) = 1$$

(3) Para qualquer conjunto de acontecimentos mutuamente exclusivos  $E_1, E_2, E_3, \dots$

$$P\left(\bigcup_i E_i\right) = \sum_i P(E_i)$$

- Corolários:

$$P(E) + P(E^c) = 1$$

$$P(E \cup F) = P(E) + P(F) - P(EF)$$

# Probabilidades condicionadas

- Dados dois acontecimentos  $E$  e  $F$ , a probabilidade condicionada de  $E$  ocorrer dado que  $F$  ocorreu designa-se por  $P(E|F)$  e é dada por

$$P(E|F) = P(EF)/P(F)$$

- Dois acontecimentos  $E$  e  $F$  dizem-se acontecimentos independentes se

$$P(EF) = P(E)P(F)$$

- Se  $E$  e  $F$  são independentes, então:

$$P(E|F) = P(E) \quad \text{e} \quad P(F|E) = P(F)$$

ou seja, se o conhecimento que um acontecimento ocorreu não afetar a probabilidade do outro ocorrer.

- Sejam  $F_1, F_2, \dots, F_n$  acontecimentos mutuamente exclusivos tais que a sua união forma o espaço de resultados  $S$ . Então,

$$P(E) = \sum_{i=1}^n P(EF_i) = \sum_{i=1}^n P(E|F_i)P(F_i)$$

# Regra de Bayes

Sejam  $F_1, F_2, \dots, F_n$  acontecimentos mutuamente exclusivos tais que a sua união forma o espaço de resultados  $S$ .

Tendo ocorrido o acontecimento  $E$ , a probabilidade de  $F_j (j = 1, 2, \dots, n)$  ter ocorrido é dada por:

$$P(F_j | E) = \frac{P(EF_j)}{P(E)} = \frac{P(E | F_j)P(F_j)}{P(E)} = \frac{P(E | F_j)P(F_j)}{\sum_{i=1}^n P(E | F_i)P(F_i)}$$

## Probabilidades condicionadas – Exemplo 1

Num teste de escolha múltipla, um estudante sabe a resposta certa com probabilidade  $p$  e adivinha a resposta com probabilidade  $1-p$ . Ao adivinhar a resposta, o estudante acerta com probabilidade  $1/m$ , sendo  $m$  o número de alternativas de escolha múltipla.

Determine a probabilidade de um estudante (a) responder corretamente e (b) saber a resposta dado que a respondeu corretamente.

Acontecimentos:  $E$  – o aluno responde corretamente

$F$  – o aluno sabe a resposta ( $F^c$  – não sabe a resposta)

$$\begin{aligned}(a) P(E) &= P(E|F)P(F) + P(E|F^c)P(F^c) = \\&= 1 \times p + 1/m \times (1 - p) = \\&= p + (1 - p)/m\end{aligned}$$

$$\begin{aligned}(b) P(F|E) &= P(E|F)P(F) / P(E) = \\&= 1 \times p / [p + (1 - p)/m] = \\&= p m / [1 + (m - 1) p]\end{aligned}$$

Se  $p = 50\%$  e  $m = 4$ , então (i)  $P(E) = 62.5\%$  e (ii)  $P(F|E) = 80\%$

## Probabilidades condicionadas – Exemplo 2

Numa ligação sem fios (wireless) entre dois equipamentos, a probabilidade dos pacotes de dados serem recebidos com erros é de 0.1% em condições normais ou de 10% quando há interferências. A probabilidade de haver interferência é de 2%. Os equipamentos têm a capacidade de verificar na receção se os pacotes de dados foram recebidos com erros ou não.

- (a) Se uma mensagem for recebida com erros, qual a probabilidade da ligação estar com interferência.
- (b) Os terminais decidem que a ligação está com interferência quando recebem 2 pacotes seguidos com erros. Qual a probabilidade desta decisão estar correta?

Fazer em casa!

# Variáveis aleatórias

- Uma variável aleatória  $X$  é uma função que atribui um número real a cada ponto do espaço de resultados  $S$  de uma experiência aleatória.
- A função distribuição (ou função de distribuição cumulativa) da v.a.  $X$  é:

$$F(x) = P(X \leq x) , -\infty < x < +\infty$$

- Propriedades da função distribuição:
  - (1)  $0 \leq F(x) \leq 1$  para todo o  $x$
  - (2) se  $x_1 \leq x_2$  então  $F(x_1) \leq F(x_2)$  (função não decrescente)
  - (3)  $\lim_{x \rightarrow -\infty} F(x) = 0$  e  $\lim_{x \rightarrow +\infty} F(x) = 1$
  - (4)  $P(a < X \leq b) = F(b) - F(a)$ , para  $a < b$

# Variáveis aleatórias discretas

- Uma variável aleatória  $X$  diz-se discreta se puder tomar, quando muito, um número contável de valores  $x_1, x_2, \dots, x_i, \dots$
- Define-se função probabilidade (ou função massa de probabilidade) da v.a discreta  $X$  por

$$f(x_i) = P(X=x_i) \quad \text{para todos os valores de } i = 1, 2, 3, \dots$$

- Obrigatoriamente, tem de acontecer que:  $\sum_{i=1}^{\infty} f(x_i) = 1$
- A função distribuição da v.a discreta  $X$  é:

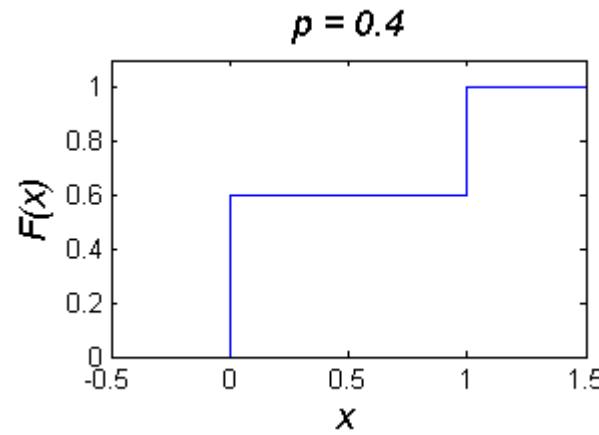
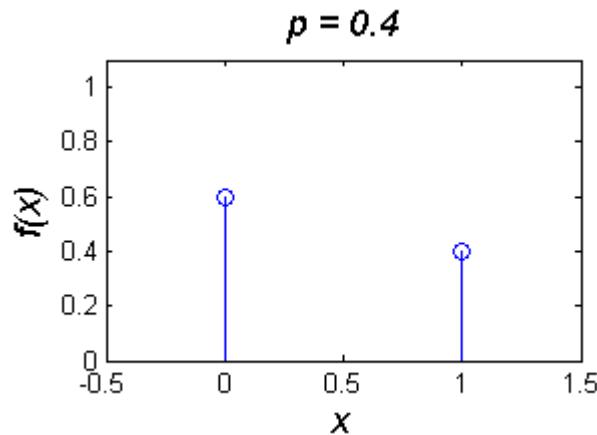
$$F(x) = \sum_{x_i \leq x} f(x_i) \quad , -\infty < x < +\infty$$

# Variáveis aleatórias discretas

Variável aleatória de Bernoulli: experiência que pode resultar em sucesso com probabilidade  $p$  ou insucesso com probabilidade  $1 - p$ .

Se  $X = 1$  representar um sucesso e  $X = 0$  um insucesso, a função probabilidade é:

$$f(i) = p^i (1-p)^{1-i}, i = 0, 1$$

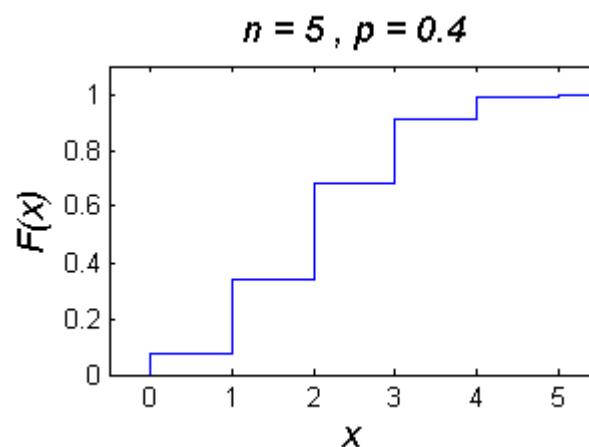
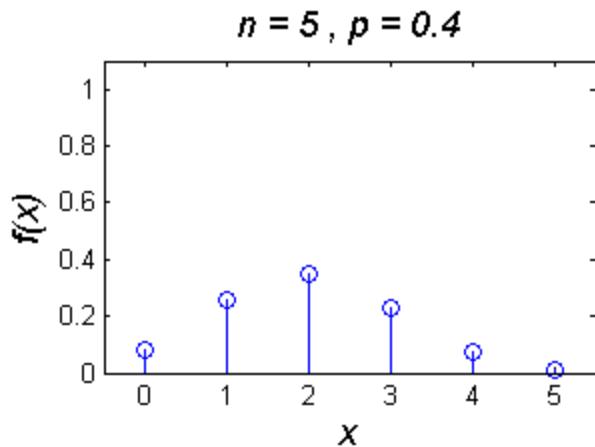


# Variáveis aleatórias discretas

Variável aleatória binomial: admita uma sequência de  $n$  experiências de Bernoulli independentes, cada uma das quais resulta num sucesso com probabilidade  $p$  ou num insucesso com probabilidade  $1 - p$ .

Se  $X$  representar o número de sucessos em  $n$  experiências, a função probabilidade é:

$$\text{onde } \binom{n}{i} = \frac{n!}{i!(n-i)!} \quad f(i) = \binom{n}{i} p^i (1-p)^{n-i}, i = 0, 1, 2, \dots, n$$

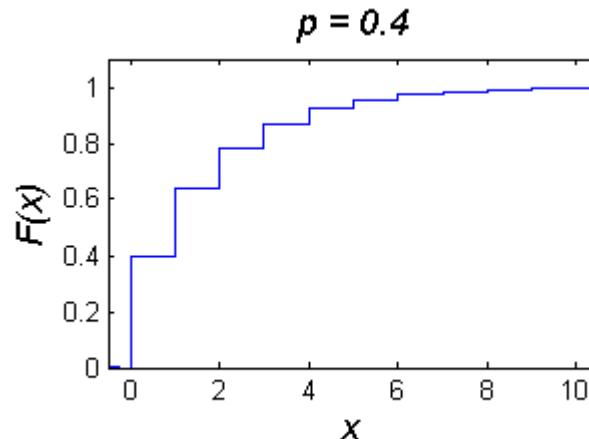
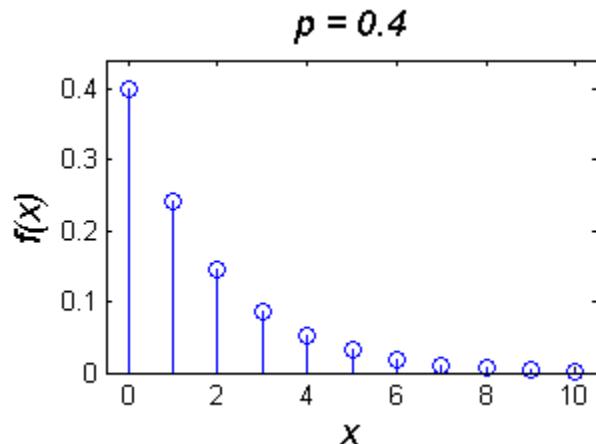


# Variáveis aleatórias discretas

Variável aleatória geométrica: admite que são realizadas experiências de Bernoulli independentes com parâmetro  $p$  (probabilidade de sucesso) até que ocorra um sucesso.

Se  $X$  representar o número de insucessos antes do primeiro sucesso, a função probabilidade é

$$f(i) = (1-p)^i p , i = 0, 1, 2, \dots$$



Se  $X$  representar o número de experiências até ao primeiro sucesso, a função probabilidade é

$$f(i) = (1-p)^{i-1} p , i = 1, 2, \dots$$

## Variáveis aleatórias discretas – Exemplos 3 e 4

Considere a experiência aleatório do lançamento de 2 dados equilibrados. O resultado da experiência é dado pela soma das 2 faces que ficam para cima.

Determine a função probabilidade  $f(x)$  e a função distribuição  $F(x)$  da variável aleatória associada a esta experiência.

Numa dada ligação de dados, a probabilidade de erro de bit (BER – *Bit Error Rate*) no estado normal é dada por  $q$  e os erros em diferentes bits são estatisticamente independentes.

Determine: (i) a probabilidade de um pacote de dados de tamanho 100 Bytes ser recebido sem erros se  $q = 10^{-5}$ ; (ii) a probabilidade de um pacote de dados de 1000 Bytes ser recebido com 2 ou mais erros se  $q = 10^{-7}$ .

Fazer em casa!

# Variáveis aleatórias contínuas

- Uma variável aleatória  $X$  diz-se contínua se existir uma função não negativa  $f(x)$  tal que para qualquer conjunto de números reais  $B$ :

$$P(X \in B) = \int_B f(x)dx \quad \int_{-\infty}^{+\infty} f(x)dx = 1$$

$f(x)$  é a função densidade de probabilidade da v.a contínua  $X$

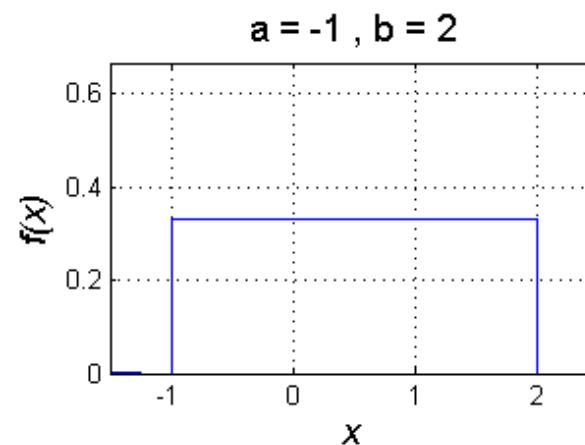
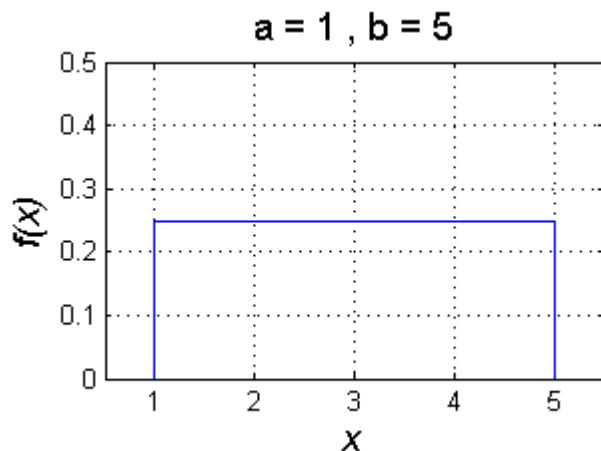
- Resulta então que:  $P\{a \leq X \leq b\} = \int_a^b f(x)dx$
- A função distribuição da v.a contínua  $X$  é:

$$F(x) = P(X \in [-\infty, x]) = \int_{-\infty}^x f(y)dy$$

# Exemplos de variáveis aleatórias contínuas

Variável aleatória com Distribuição Uniforme: uma v.a. diz-se uniformemente distribuída no intervalo  $[a,b]$  se a função densidade de probabilidade for dada por

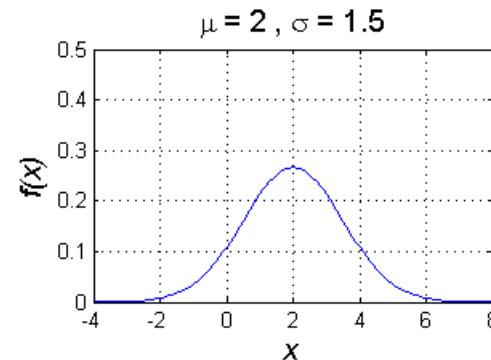
$$f(x) = \begin{cases} \frac{1}{b-a} & , a < x < b \\ 0 & , \text{cc} \end{cases}$$



# Exemplos de variáveis aleatórias contínuas

Variável aleatória com Distribuição Gaussiana (ou Normal): Uma v.a.  $X$  tem uma distribuição Gaussiana com média  $\mu$  e desvio padrão  $\sigma$  se a função densidade é dada por:

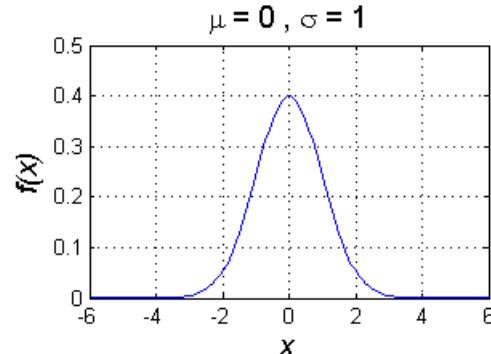
$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(x-\mu)^2/2\sigma^2}$$



Designa-se por distribuição Gaussiana (ou Normal) padrão à distribuição Gaussiana com média 0 e desvio padrão 1.

Neste caso:

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$$



# Média de uma variável aleatória

- Média ou valor esperado de uma v.a.  $X$ :

$$E[X] = \begin{cases} \sum_{j=1}^{\infty} x_j f_X(x_j) & \text{se } X \text{ discreta} \\ \int_{-\infty}^{+\infty} x f_X(x) dx & \text{se } X \text{ continua} \end{cases}$$

- Propriedades importantes:

$$E[cX] = cE[X]$$

$$E\left[\sum_{i=1}^n c_i X_i\right] = \sum_{i=1}^n c_i E[X_i]$$

- Média da v.a.  $Y = g(X)$ :

$$E[g(X)] = \begin{cases} \sum_{j=1}^{\infty} g(x_j) f_X(x_j) & \text{se } X \text{ discreta} \\ \int_{-\infty}^{+\infty} g(x) f_X(x) dx & \text{se } X \text{ continua} \end{cases}$$

# Variância de uma variável aleatória

- Variância de uma v.a.  $X$ :

$$\text{Var}[X] = E[(X - E[X])^2] = E[X^2] - E[X]^2$$

- Propriedades importantes:

$$\text{Var}[X] \geq 0$$

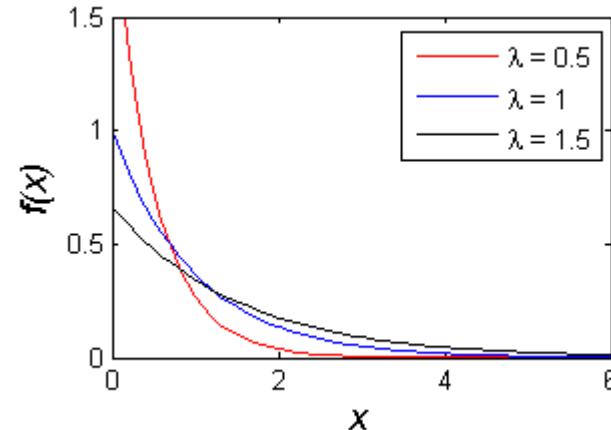
$$\text{Var}[cX] = c^2 \text{Var}[X]$$

$$\text{Var}\left[\sum_{i=1}^n X_i\right] = \sum_{i=1}^n \text{Var}[X_i] \quad \text{se } X_i \text{ forem independentes}$$

# Distribuição exponencial

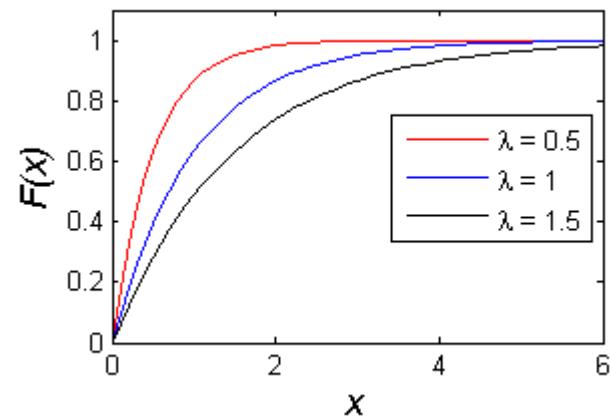
- Uma v. a.  $X$  tem uma distribuição exponencial com parâmetro  $\lambda$ ,  $\lambda > 0$ , se a sua função densidade de probabilidade for:

$$f(x) = \begin{cases} \lambda e^{-\lambda x}, & x \geq 0 \\ 0, & x < 0 \end{cases}$$



- A função de distribuição é dada por:

$$F(x) = \begin{cases} 1 - e^{-\lambda x}, & x \geq 0 \\ 0, & x < 0 \end{cases}$$



# Distribuição exponencial

- A média e a variância de uma distribuição exponencial são:

$$E[X] = \frac{1}{\lambda} \quad \text{Var}(X) = \frac{1}{\lambda^2}$$

- A distribuição exponencial não tem memória, isto é,

$$P\{X > s + t \mid X > t\} = P\{X > s\}$$

- Se  $X_1$  e  $X_2$  são v. a. independentes e exponencialmente distribuídas com médias  $1/\lambda_1$  e  $1/\lambda_2$  respectivamente, então

$$P\{X_1 < X_2\} = \frac{\lambda_1}{\lambda_1 + \lambda_2}$$

## Distribuição exponencial – Exemplo 5

Uma pessoa entra num banco e encontra os 2 empregados do banco ocupados a servir clientes. Não existem outros clientes no banco, pelo que a pessoa começará a ser atendida logo que um dos clientes que se encontra a ser atendido deixe o banco.

O empregado A serve os clientes segundo uma distribuição exponencial à taxa  $\lambda_A = 12$  clientes por hora e o empregado B serve os clientes segundo uma distribuição exponencial à taxa  $\lambda_B = 8$  clientes por hora. Determine a probabilidade da pessoa que entrou ser o segundo cliente a ser servido.

Evento A – o empregado A termina de servir o seu cliente antes do empregado B

Evento B – o empregado B termina de servir o seu cliente antes do empregado A

Evento C – a pessoa que entrou é o segundo cliente a ser servido

$$\begin{aligned} P(C) &= P(C | A) \times P(A) + P(C | B) \times P(B) = \\ &= \frac{\lambda_A}{\lambda_A + \lambda_B} \times \frac{\lambda_A}{\lambda_A + \lambda_B} + \frac{\lambda_B}{\lambda_A + \lambda_B} \times \frac{\lambda_B}{\lambda_A + \lambda_B} = \\ &= \frac{12}{12+8} \times \frac{12}{12+8} + \frac{8}{12+8} \times \frac{8}{12+8} = 0.52 = 52\% \end{aligned}$$

# Processos estocásticos

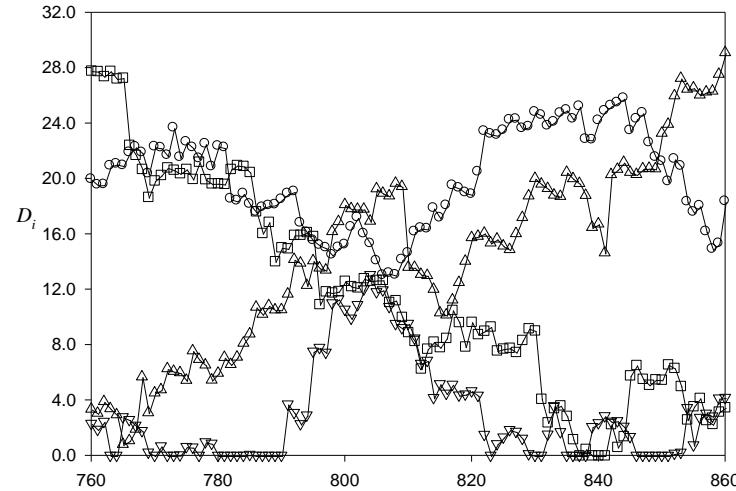
- Um processo estocástico  $\{X(t), t \in T\}$  é um conjunto de variáveis aleatórias: para cada  $t \in T$ ,  $X(t)$  é uma variável aleatória.
- O índice  $t$  é frequentemente interpretado como tempo. Nesta interpretação,  $X(t)$  é o estado do processo no instante  $t$ .
- O conjunto  $T$  é o conjunto de índices do processo.
  - (1) se  $T$  é um conjunto contável, designa-se o processo estocástico como sendo em tempo discreto
  - (2) se  $T$  é um intervalo da reta real, designa-se o processo estocástico como sendo em tempo contínuo
- O espaço de estados é o conjunto de todos os valores que as variáveis aleatórias  $X(t)$  podem tomar.

# Exemplos de processos estocásticos

Considere um sistema com uma fila de espera e um servidor. A este sistema chegam clientes para serem servidos.

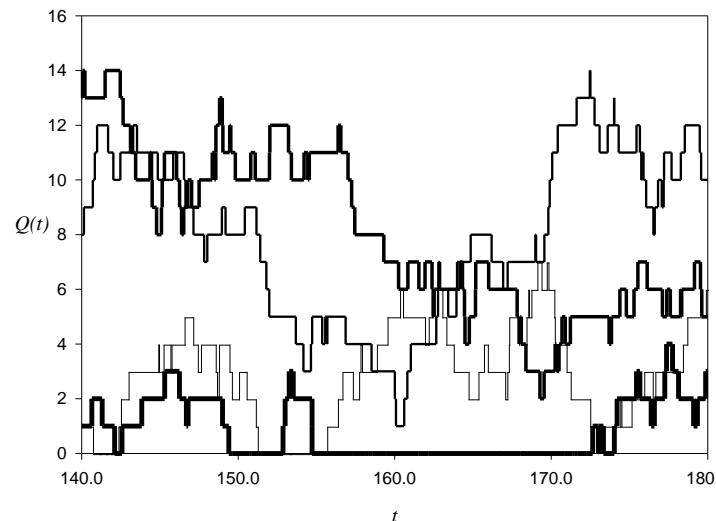
Atrasos sofridos por cada cliente na fila de espera

- (1) é um processo estocástico em tempo discreto
- (2) o estado é uma variável contínua



O número de clientes em espera

- (1) é um processo estocástico em tempo contínuo
- (2) o estado é uma variável discreta





# **Processos de Poisson, Cadeias de Markov em Tempo Contínuo e Sistemas de Filas de Espera**

Desempenho e Dimensionamento de Redes

Prof. Amaro de Sousa ([asou@ua.pt](mailto:asou@ua.pt))

DETI-UA, 2017/2018

# Processos de contagem

- Um processo estocástico  $\{N(t), t \geq 0\}$  diz-se um processo de contagem se  $N(t)$  representar o número total de eventos que ocorreram até ao instante  $t$ .
- Um processo de contagem satisfaz as seguintes condições:
  - (1)  $N(t) \geq 0$ .
  - (2)  $N(t)$  toma valores inteiros apenas.
  - (3) Se  $s < t$ , então  $N(s) \leq N(t)$ .
  - (4) Se  $s < t$ , então  $N(t) - N(s)$  é igual ao número de eventos ocorridos no intervalo de tempo  $[s, t]$ .
- Um processo de contagem tem incrementos independentes se o número de eventos em intervalos de tempo disjuntos for independente.
- Um processo de contagem tem incrementos estacionários se a distribuição do número de eventos que ocorre em qualquer intervalo de tempo depender apenas do comprimento do intervalo de tempo.

# Processos de Poisson

- Um processo de contagem diz-se um processo de Poisson com taxa  $\lambda$ ,  $\lambda > 0$ , se:
  - (1)  $N(0) = 0$ ;
  - (2) o processo tem incrementos independentes;
  - (3) o número de eventos num intervalo de duração  $t$  tem uma distribuição de Poisson com média  $\lambda t$ . Isto é, para todo  $s$ ,  $t \geq 0$

$$P\{N(s+t) - N(s) = n\} = e^{-\lambda t} \frac{(\lambda t)^n}{n!}$$

- Um processo de Poisson tem incrementos estacionários e média

$$E[N(t)] = \lambda t$$

razão pela qual  $\lambda$  é designada a taxa do processo de Poisson.

# Propriedades de um processo de Poisson

- **Propriedade 1:** Num processo de Poisson com taxa  $\lambda$  considere-se:
  - $T_1$  o instante do primeiro evento
  - $T_n$ ,  $n > 1$ , o intervalo de tempo entre o  $(n-1)$ -ésimo evento e o  $n$ -ésimo evento
- Então,  $T_n$ ,  $n = 1, 2, \dots$ , são variáveis aleatórias independentes e identicamente distribuídas com distribuição exponencial de média  $1/\lambda$ .
- **Propriedade 2:** Num processo de Poisson  $\{N(t), t \geq 0\}$  com taxa  $\lambda$  considere-se que cada evento é classificado de forma independente em:
  - evento do tipo 1 com probabilidade  $p$
  - evento do tipo 2 com probabilidade  $1-p$
- Assim,  $\{N_1(t), t \geq 0\}$  e  $\{N_2(t), t \geq 0\}$  são o número de eventos de cada tipo que ocorreram no intervalo  $[0, t]$ .
- Então,  $N_1(t)$  e  $N_2(t)$  são ambos processos independentes e de Poisson com taxas  $\lambda p$  e  $\lambda(1-p)$ .

# Propriedades de um processo de Poisson

- **Propriedade 3:** Sejam  $\{N_1(t), t \geq 0\}$  e  $\{N_2(t), t \geq 0\}$  processos de Poisson independentes com taxas  $\lambda_1$  e  $\lambda_2$ ,
- Então, o processo  $N(t) = N_1(t) + N_2(t)$  é também um processo de Poisson com taxa  $\lambda = \lambda_1 + \lambda_2$ .
- **Propriedade 4:** Sabendo-se que num processo de Poisson ocorreram exatamente  $n$  eventos até ao instante  $t$ ,
- Então, os instantes de ocorrência dos eventos são distribuídos independentemente e uniformemente no intervalo  $[0, t]$ . Por esta razão diz-se que num processo de Poisson as chegadas são aleatórias.

# Cadeias de Markov em tempo contínuo

- Considere-se um processo estocástico em tempo contínuo  $\{X(t), t \geq 0\}$  com o espaço de estados definido pelo conjunto dos números inteiros não negativos.
- $X(t)$  é uma cadeia de Markov se para todo o  $s$ ,  $t \geq 0$  e inteiros não-negativos  $i, j$ ,  $x(u)$ ,  $0 \leq u < s$  :

$$P\{X(s+t) = j | X(s) = i, X(u) = x(u), 0 \leq u < s\} = \\ P\{X(s+t) = j | X(s) = i\}$$

- Significa que a distribuição futura  $X(s+t)$  condicionada ao presente  $X(s)$  e ao passado  $X(u)$ ,  $0 \leq u < s$ , depende apenas do presente e é independente do passado (propriedade Markoviana).
- Se  $P\{X(s+t) = j | X(s) = i\}$  for independente de  $s$  então diz-se que a cadeia de Markov em tempo contínuo tem probabilidades de transição estacionárias ou homogéneas:

$$P\{X(s+t) = j | X(s) = i\} = P\{X(t) = j | X(0) = i\}$$

# Cadeias de Markov em tempo contínuo

- Uma cadeia de Markov em tempo contínuo tem como propriedades:
  - (1) Quando o processo entra no estado  $i$ , o tempo de permanência nesse estado, antes de efetuar uma transição para um estado diferente, é exponencialmente distribuído (designamos a média por  $1/q_i$ );
  - (2) Quando o processo deixa o estado  $i$ , entra de seguida no estado  $j$  com uma probabilidade  $P_{ij}$  que satisfaz as seguintes condições

$$P_{ii} = 0 \quad 0 \leq P_{ij} \leq 1 \quad , j \neq i \quad \sum_j P_{ij} = 1$$

NOTA: A propriedade (1) é equivalente a dizer que quando o processo está no estado  $i$ , ele transita para outro estado qualquer a uma taxa  $q_i$ .

- Numa cadeia de Markov em tempo contínuo, o tempo de permanência num estado e o próximo estado visitado são variáveis aleatórias independentes.

# Taxas de transição instantâneas

- Para qualquer par de estados  $i$  e  $j$  seja

$$q_{ij} = q_i P_{ij}$$

$q_i$  - a taxa à qual o processo faz uma transição quando está no estado  $i$

$P_{ij}$  - a probabilidade que a transição seja para o estado  $j$  quando está no estado  $i$

$q_{ij}$  - a taxa à qual o processo faz uma transição para o estado  $j$  quando está no estado  $i$

- As  $q_{ij}$  designam-se por taxas de transição instantâneas. Estas são as grandezas habitualmente representadas nos diagramas de transição de estados.

- Como 
$$q_i = \sum_j q_i P_{ij} = \sum_j q_{ij}$$
 
$$P_{ij} = \frac{q_{ij}}{q_i} = \frac{q_{ij}}{\sum_j q_{ij}}$$

resulta que a especificação das taxas de transição instantâneas determina a cadeia de Markov em tempo contínuo.

## Exemplo 1

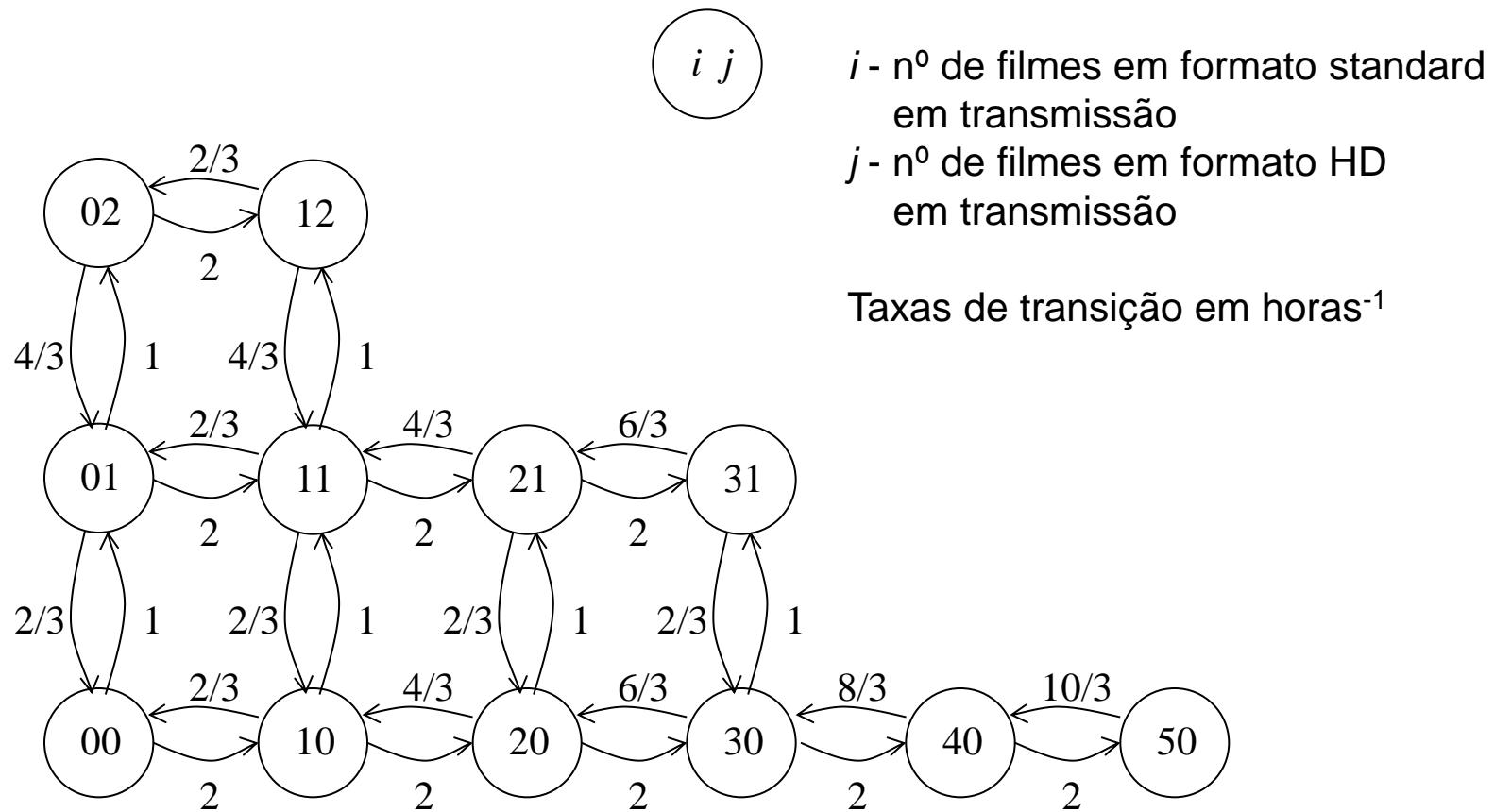
Considere um servidor de *video-streaming* que providencia filmes em formato standard (2.0 Mbps) ou HD (4.0 Mbps) e com uma interface de rede de 10 Mbps.

No período de maior tráfego, a taxa de pedidos de filmes é de 2 filmes/hora em formato standard e 1 filme/hora em formato HD.

Ambos os tipos de filmes têm uma duração exponencialmente distribuída de média 90 minutos. Quando um filme é pedido, ele começa a ser transmitido desde que haja capacidade disponível na interface de rede ou é recusado caso contrário.

Considere o estado do sistema dado pelo número de filmes de cada tipo em transmissão. Qual o diagrama de transição de estados do sistema?

# Diagrama de transição de estados do Exemplo 1



# Probabilidades limite

- Seja  $P_{ij}(t) = P\{X(s+t) = j \mid X(s) = i\}$  a probabilidade de um processo presentemente no estado  $i$  estar no estado  $j$  após um intervalo de tempo  $t$ .

- A probabilidade de uma cadeia de Markov em tempo contínuo estar no estado  $j$  no instante  $t$  converge para um valor limite independente do estado inicial:

$$\pi_j \equiv \lim_{t \rightarrow \infty} P_{ij}(t)$$

- Condição suficiente para a existência de probabilidades limite:
  - (1) a cadeia é irredutível, isto é, começando no estado  $i$  existe uma probabilidade positiva de alguma vez se estar no estado  $j$ , para todo o par de estados  $i, j$
  - (2) a cadeia de Markov é recorrente positiva, isto é, começando em qualquer estado o tempo médio para voltar a esse estado é finito

# Cálculo das probabilidades limite

- As probabilidades limite podem calcular-se resolvendo as equações:

$$q_j \pi_j = \sum_{k \neq j} q_{kj} \pi_k , \quad \text{para todos os estados } j$$

$$\sum_j \pi_j = 1$$

- Estas equações são designadas por equações de balanço:

taxa à qual o sistema transita do estado  $j$

=

taxa à qual o sistema transita para o estado  $j$

- A probabilidade  $\pi_j$  pode ser interpretada como a proporção de tempo em que o processo está no estado  $j$ .
- As probabilidades  $\pi_j$  são designadas por probabilidades estacionárias: se o estado inicial for dado pela distribuição  $\{\pi_j\}$ , então a probabilidade de se estar no estado  $j$  no instante  $t$  é  $\pi_j$ , para todo o  $t$ .

## Exemplo 1

Equações de balanço:

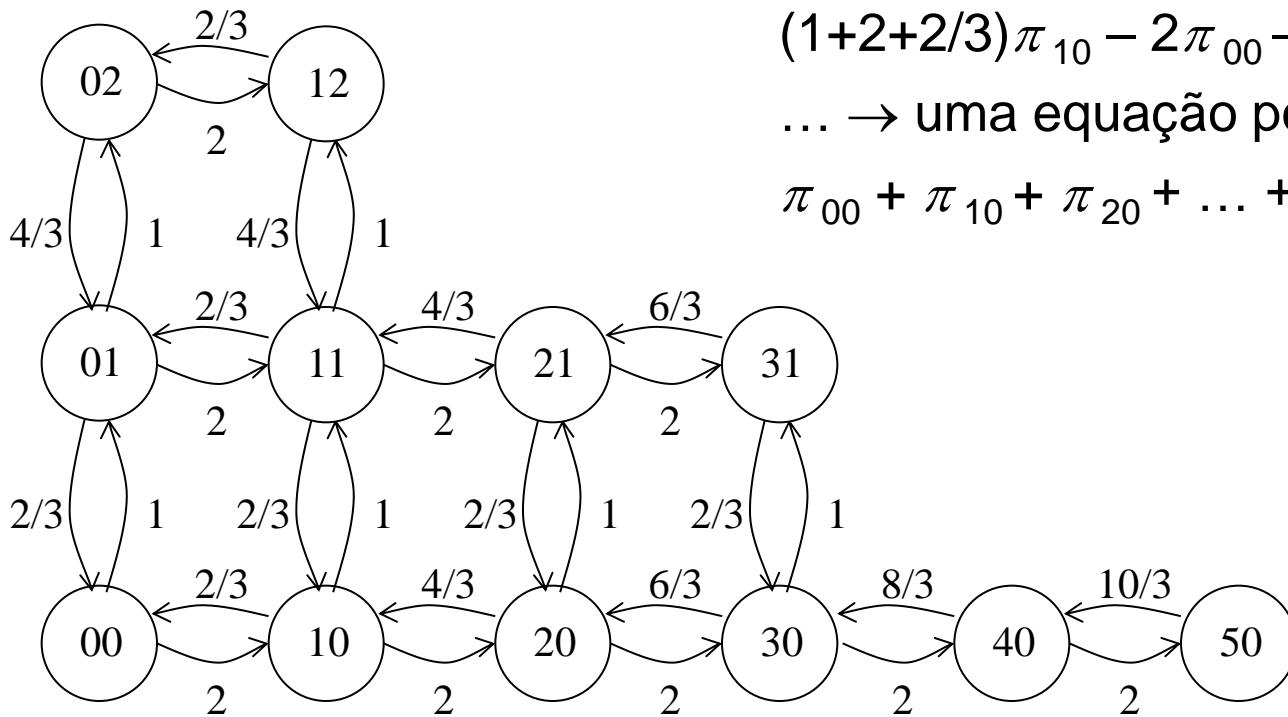
$$q_j \pi_j = \sum_{k \neq j} q_{kj} \pi_k \quad \sum_j \pi_j = 1$$

$$(1+2)\pi_{00} - 2/3\pi_{10} - 2/3\pi_{01} = 0$$

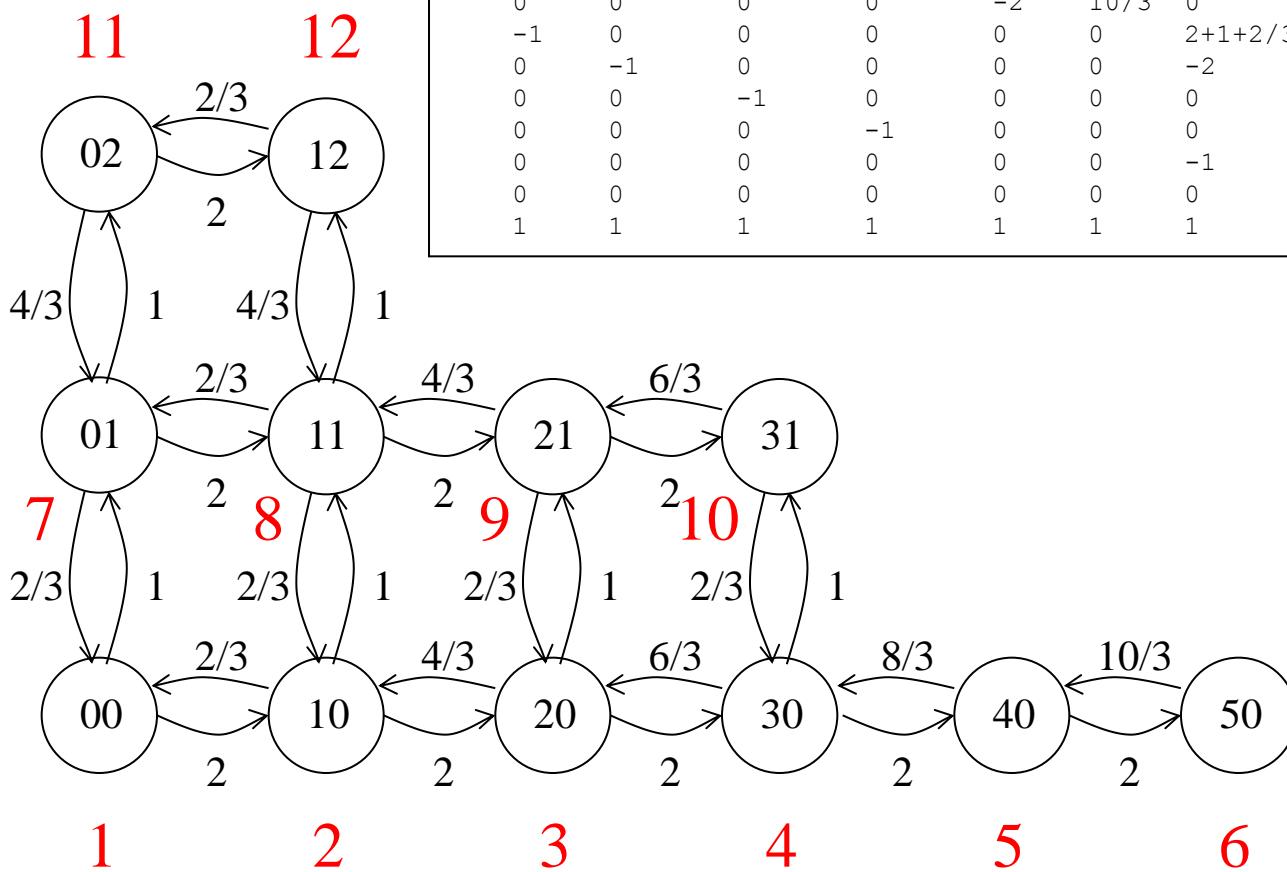
$$(1+2+2/3)\pi_{10} - 2\pi_{00} - 4/3\pi_{20} - 2/3\pi_{11} = 0$$

... → uma equação por cada estado

$$\pi_{00} + \pi_{10} + \pi_{20} + \dots + \pi_{02} + \pi_{12} = 1$$



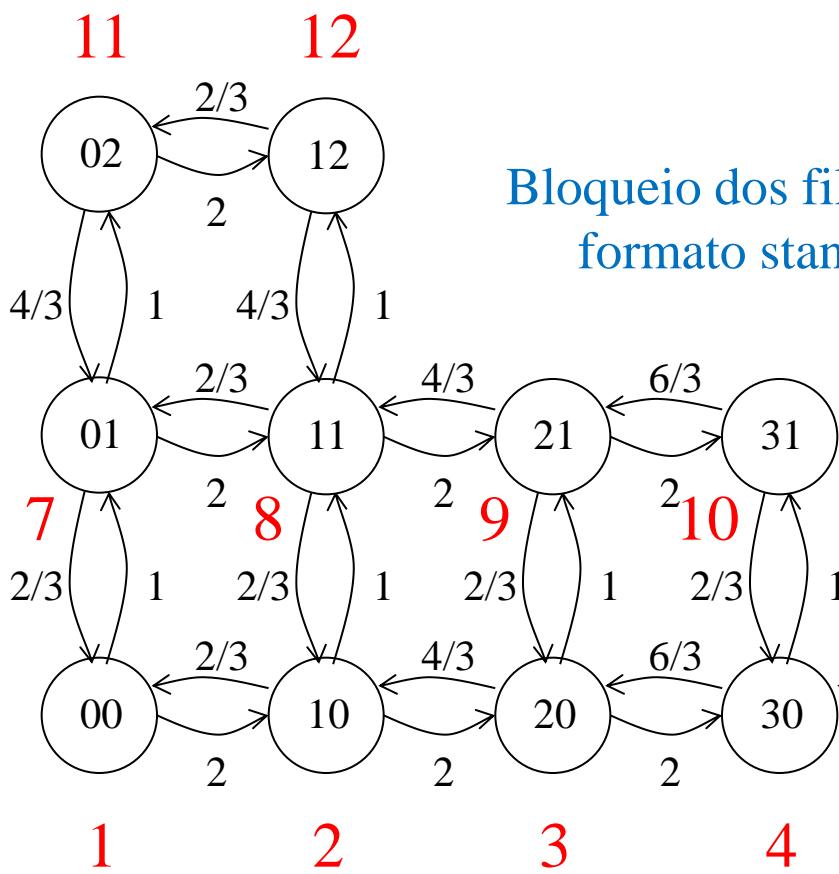
# Exemplo 1 – resolução no MATLAB



```
A= [2+1   -2/3   0     0     0     0     -2/3   0     0     0     0     0
      -2     2+1+2/3 -4/3   0     0     0     0     -2/3   0     0     0     0
       0     -2     2+1+4/3 -6/3   0     0     0     0     -2/3   0     0     0
       0     0     -2     2+1+6/3 -8/3   0     0     0     0     -2/3   0     0
       0     0     0     -2     2+8/3  -10/3  0     0     0     0     0     0
       0     0     0     0     -2     10/3   0     0     0     0     0     0
      -1     0     0     0     0     0     2+1+2/3 -2/3   0     0     -4/3   0
       0     -1     0     0     0     0     -2     2+1+4/3 -4/3   0     0     -4/3
       0     0     -1     0     0     0     0     -2     2+6/3  -6/3   0     0
       0     0     0     -1     0     0     0     0     -2     8/3    0     0
       0     0     0     0     0     0     -1     0     0     0     2+4/3 -2/3
       0     0     0     0     0     0     0     -1     0     0     -2     6/3
       1     1     1     1     1     1     1     1     1     1     1     1]
```

```
B= [0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      1]
```

# Exemplo 1 – resolução no MATLAB



Bloqueio dos filmes em formato standard

Bloqueio dos filmes em formato HD

```
>> x=A\B
```

```
x =
0.0236
0.0708
0.1061
0.1061
0.0796
0.0478
0.0354
0.1061
0.1592
0.1592
0.0265
0.0796
```

```
>> x(6)+x(10)+x(12)
```

```
ans =
0.2866
```

```
>> x(5)+x(6)+x(9)+x(10)+x(11)+x(12)
```

```
ans =
0.5519
```

## Definições do teorema de Little

- Admita-se que se observa um sistema desde o instante  $t = 0$ . Seja:  
 $L(t)$  - o número de clientes no sistema no instante  $t$ ,  
 $N(t)$  - o número de clientes que chegaram no intervalo  $[0, t]$ ,  
 $W_i$  - o tempo despendido no sistema pelo  $i$ -ésimo cliente.
- Média temporal do número de clientes observados até ao instante  $t$ :  
$$L_t = \frac{1}{t} \int_0^t L(\tau) d\tau \quad L = \lim_{t \rightarrow \infty} L_t$$
- Média temporal da taxa de chegada no intervalo  $[0, t]$ :

$$\lambda_t = N(t)/t \quad \lambda = \lim_{t \rightarrow \infty} \lambda_t$$

- Média temporal do atraso dos clientes até ao instante  $t$ :

$$W_t = \frac{\sum_{i=0}^{N(t)} W_i}{N(t)} \quad W = \lim_{t \rightarrow \infty} W_t$$

## Teorema de Little

- O teorema de Little enuncia que

$$L = \lambda W$$

- O teorema de Little traduz a ideia intuitiva de que, para a mesma taxa de chegada de clientes, sistemas mais congestionados ( $L$  elevado) impõem maiores atrasos ( $W$  elevado).
- Num dia de chuva, o mesmo tráfego (mesmo  $\lambda$ ) é mais lento do que normalmente ( $W$  maior) e as ruas estão mais congestionadas ( $L$  maior).
- Um restaurante de refeições rápidas ( $W$  menor) precisa de uma sala menor ( $L$  menor) que um restaurante normal, para a mesma taxa de chegada de clientes (mesmo  $\lambda$ ).

## Propriedade PASTA

- Considere um sistema em que os clientes chegam um de cada vez e são servidos um de cada vez.
- Seja  $L(t)$  o número de clientes no sistema no instante  $t$  e defina-se  $P_n$ ,  $n \geq 0$ , como

$$P_n = \lim_{t \rightarrow \infty} P\{L(t) = n\}$$

$P_n$  é a probabilidade em estado estacionário de existirem exatamente  $n$  clientes no sistema (ou a proporção de tempo em que o sistema contém exatamente  $n$  clientes).

- Considere  $a_n$  a proporção de clientes que ao chegar encontram  $n$  clientes no sistema.
- Considere  $d_n$  a proporção de clientes que ao partir deixam  $n$  clientes no sistema.
- Em qualquer sistema em que os clientes chegam um de cada vez e são servidos um de cada vez verifica-se que

$$a_n = d_n$$

# Propriedade PASTA

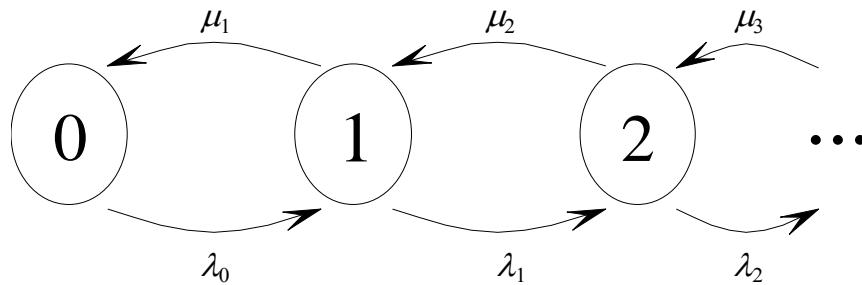
- Propriedade PASTA (*Poisson Arrivals always See Time Averages*):  
As chegadas de Poisson em que o tempo de serviço é estatisticamente independente dos instantes de chegada, vêm sempre médias temporais:

$$a_n = P_n$$

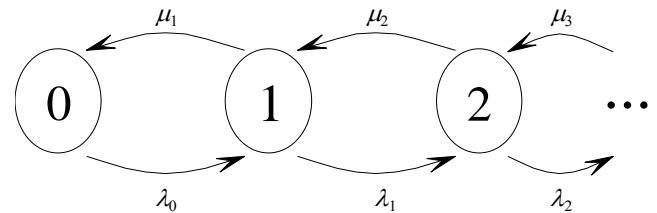
- Contra exemplos:
- Considere que o intervalo entre chegadas é uniformemente distribuído entre 2 e 6 segundos e os tempos de serviço dos clientes são uniformemente distribuídos entre 1 e 2 segundos.
  - As chegadas não são de Poisson
- Considere um sistema em que o processo de chegada de clientes é um processo de Poisson e que o tempo de serviço do  $n$ -ésimo cliente é igual a metade do intervalo entre a chegada do  $n$ -ésimo e do  $(n+1)$ -ésimo cliente.
  - O tempo de serviço não é independente das chegadas

# Processos de nascimento e morte

- Considere um sistema cujo estado representa o número de clientes no sistema.
- Sempre que o sistema tem  $n$  clientes:
  - (1) chegam novos clientes ao sistema a uma taxa exponencial  $\lambda_n$
  - (2) partem clientes do sistema a uma taxa exponencial  $\mu_n$
- Este sistema é designado por processo de nascimento e morte.
- Os parâmetros  $\lambda_n$  ( $n = 0, 1, \dots$ ) e  $\mu_n$  ( $n = 1, 2, \dots$ ) são designados por taxas de chegada (ou de nascimento) e taxas de partida (ou de morte), respetivamente.



# Equações de balanço de processos de nascimento e morte



Estado                    taxa de saída = taxa de entrada

$$0 \quad \lambda_0\pi_0 = \mu_1\pi_1$$

$$1 \quad (\lambda_1 + \mu_1)\pi_1 = \mu_2\pi_2 + \lambda_0\pi_0$$

$$2 \quad (\lambda_2 + \mu_2)\pi_2 = \mu_3\pi_3 + \lambda_1\pi_1$$

$$n, n \geq 1 \quad (\lambda_n + \mu_n)\pi_n = \mu_{n+1}\pi_{n+1} + \lambda_{n-1}\pi_{n-1}$$

Ou, de forma equivalente (por manipulação das equações anteriores):

$$\lambda_n\pi_n = \mu_{n+1}\pi_{n+1}, \quad n \geq 0$$

# Probabilidades limite de processos de nascimento e morte

$$\pi_0 = \frac{1}{1 + \sum_{i=1}^{\infty} \frac{\lambda_0 \lambda_1 \cdots \lambda_{i-1}}{\mu_1 \mu_2 \cdots \mu_i}}$$

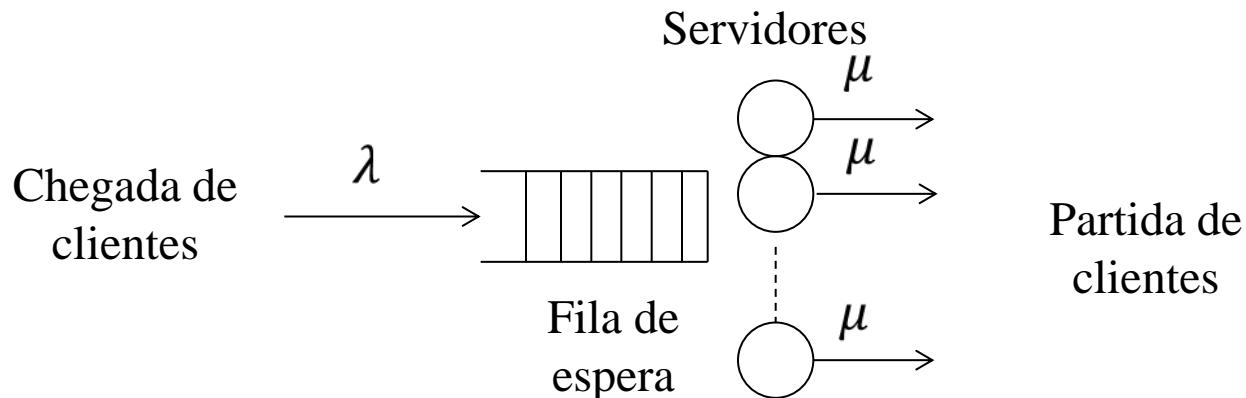
$$\pi_n = \frac{\lambda_0 \lambda_1 \cdots \lambda_{n-1}}{\mu_1 \mu_2 \cdots \mu_n \left( 1 + \sum_{i=1}^{\infty} \frac{\lambda_0 \lambda_1 \cdots \lambda_{i-1}}{\mu_1 \mu_2 \cdots \mu_i} \right)} = \frac{\lambda_0 \lambda_1 \cdots \lambda_{n-1}}{\mu_1 \mu_2 \cdots \mu_n} \cdot \pi_0, \quad n \geq 1$$

Condição necessária para a existência de probabilidades limite:

$$\sum_{i=1}^{\infty} \frac{\lambda_0 \lambda_1 \cdots \lambda_{i-1}}{\mu_1 \mu_2 \cdots \mu_i} < \infty$$

# Sistema de fila de espera

- Um sistema de fila de espera é caracterizado por:
  - um conjunto de  $c$  servidores, cada um com capacidade para servir clientes a uma taxa  $\mu$
  - uma fila de espera com uma determinada capacidade (em nº de clientes)
- A este sistema chegam clientes a uma taxa  $\lambda$
- Quando um cliente chega:
  - ele começa a ser servido por um servidor se houver algum disponível
  - ele é colocado da fila de espera se os servidores estiverem todos ocupados
- Os clientes na fila de espera são atendidos segundo uma disciplina *First-In-First-Out*



# Sistema de fila de espera

- Um sistema de fila de espera é representado por:

$$A/B/c/d$$

*A* – o processo de chegada de clientes:

*M* – Markoviano, *D* – Determinístico, *G* – Genérico

*B* – o processo de atendimento de clientes:

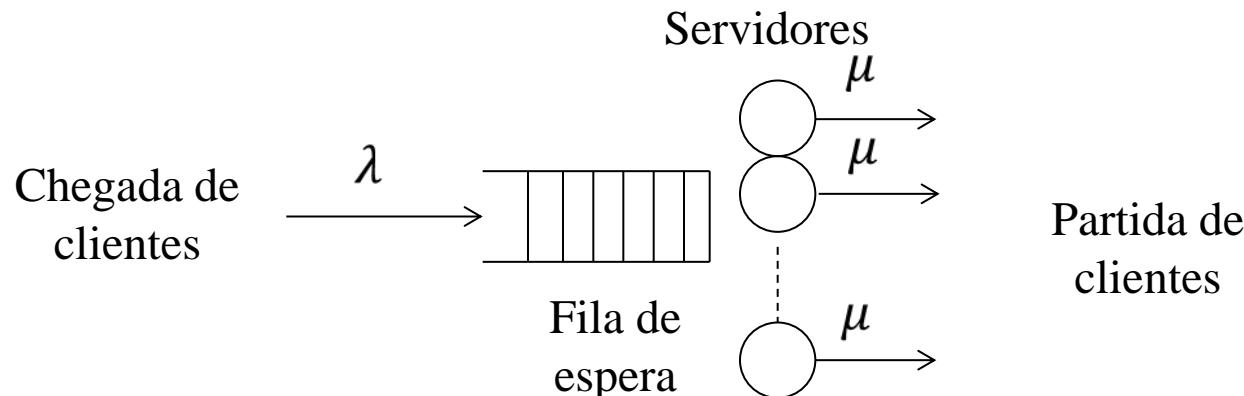
*M* – Markoviano, *D* – Determinístico, *G* – Genérico

*c* – o número de servidores

*d* – capacidade do sistema em nº de clientes:

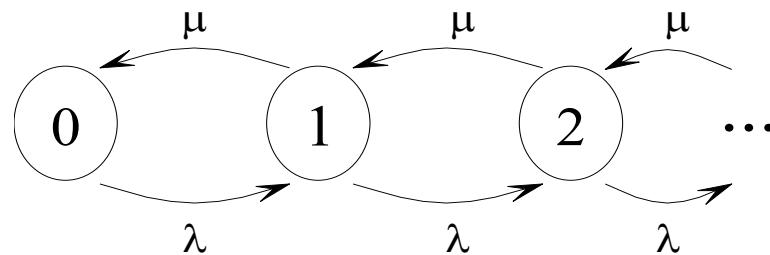
número de servidores + capacidade da fila de espera

- Quando *d* é omissio, a fila de espera tem tamanho infinito.



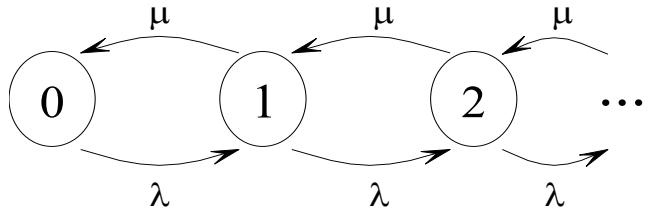
# Sistema $M/M/1$

- Processo de nascimento e morte em que:
  - (1) a chegada de clientes é um processo de Poisson com taxa  $\lambda$
  - (2) o tempo de atendimento de um servidor é exponencialmente distribuído com média  $1/\mu$
  - (3) o sistema tem 1 servidor
  - (4) o sistema acomoda um número infinito de clientes



- Uma ligação ponto-a-ponto com capacidade  $\mu$  pacotes/s e uma fila de espera muito grande onde chegam pacotes a uma taxa de Poisson  $\lambda$  pacotes/s com comprimento exponencialmente distribuído de média  $1/\mu$  é um sistema  $M/M/1$

# Sistema $M/M/1$



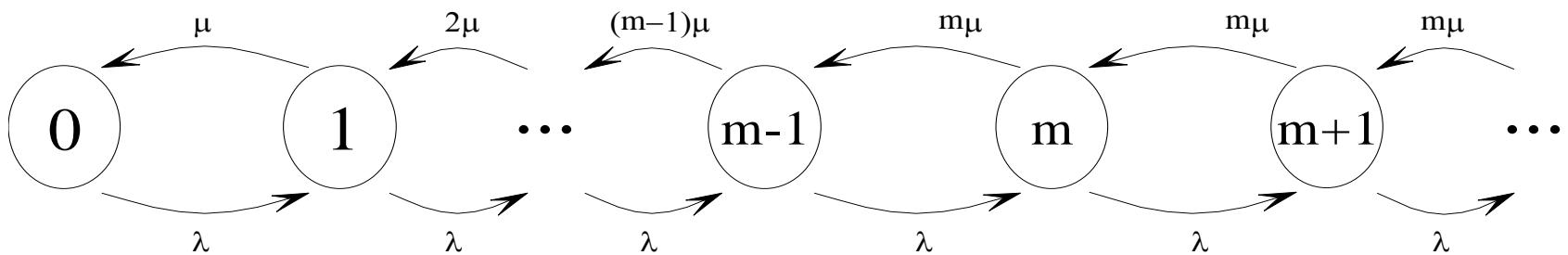
$$P_0 = \frac{1}{1 + \sum_{i=1}^{\infty} \frac{\lambda_0 \lambda_1 \cdots \lambda_{i-1}}{\mu_1 \mu_2 \cdots \mu_i}} = \frac{1}{1 + \sum_{i=1}^{\infty} \left(\frac{\lambda}{\mu}\right)^i}$$

$$P_n = \frac{\lambda_0 \lambda_1 \cdots \lambda_{n-1}}{\mu_1 \mu_2 \cdots \mu_n} \cdot P_0 = \left(\frac{\lambda}{\mu}\right)^n \cdot P_0 = \frac{\left(\frac{\lambda}{\mu}\right)^n}{1 + \sum_{i=1}^{\infty} \left(\frac{\lambda}{\mu}\right)^i}$$

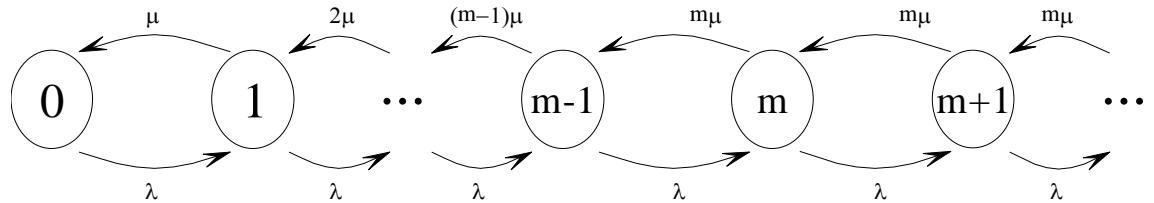
- Número médio de clientes no sistema:  $L = \sum_{n=0}^{\infty} nP_n = \frac{\lambda}{\mu - \lambda}$
- Atraso médio no sistema:  $W = \frac{L}{\lambda} = \frac{1}{\mu - \lambda}$
- Atraso médio na fila de espera:  $W_Q = W - \frac{1}{\mu} = \frac{\lambda}{\mu(\mu - \lambda)}$
- Número médio de clientes na fila de espera:  $L_Q = \lambda W_Q = \frac{\lambda^2}{\mu(\mu - \lambda)}$

# Sistema $M/M/m$

- Processo de nascimento e morte em que:
  - (1) a chegada de clientes é um processo de Poisson com taxa  $\lambda$
  - (2) o tempo de atendimento de um servidor é exponencialmente distribuído com média  $1/\mu$
  - (3) o sistema tem  $m$  servidores
  - (4) o sistema acomoda um número infinito de clientes



## Sistema $M/M/m$



Equações de balanço:

$$\lambda P_{n-1} = n\mu P_n, \quad n \leq m$$

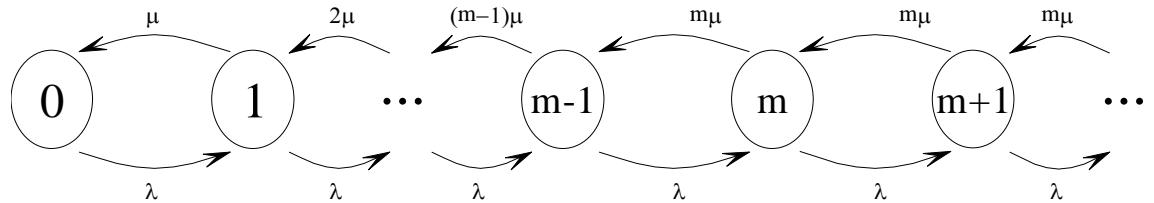
$$\lambda P_{n-1} = m\mu P_n, \quad n > m$$

Probabilidade de  $n$  clientes no sistema em estado estacionário ( $\rho = \lambda/m\mu < 1$ ):

$$P_0 = \frac{1}{\sum_{n=0}^{m-1} \frac{(m\rho)^n}{n!} + \frac{(m\rho)^m}{m!(1-\rho)}}$$

$$P_n = \begin{cases} P_0 \frac{(m\rho)^n}{n!}, & n \leq m \\ P_0 \frac{m^m \rho^n}{m!}, & n \geq 1 \end{cases}$$

## Sistema $M/M/m$



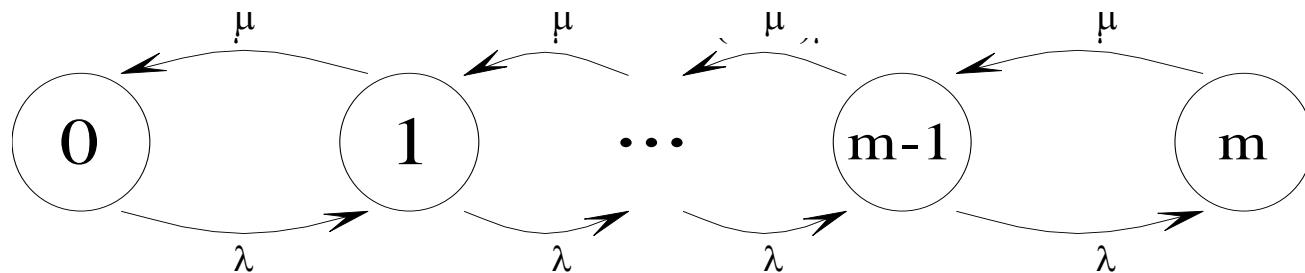
- Probabilidade de uma chegada encontrar todos os servidores ocupados (fórmula de Erlang C):

$$P_Q = \sum_{n=m}^{\infty} p_n = \frac{P_0(m\rho)^m}{m!(1-\rho)}$$

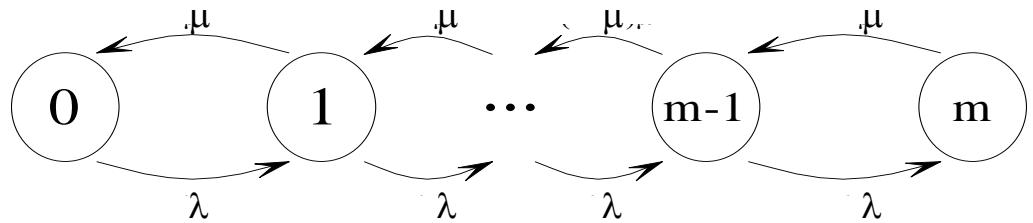
- Número médio de clientes na fila de espera:  $L_Q = \sum_{n=0}^{\infty} n P_0 \frac{m^m \rho^{m+n}}{m!} = P_Q \frac{\rho}{1-\rho}$
- Atraso médio na fila de espera:  $W_Q = \frac{L_Q}{\lambda} = P_Q \frac{\rho}{\lambda(1-\rho)}$
- Atraso médio no sistema:  $W = \frac{1}{\mu} + W_Q = \frac{1}{\mu} + \frac{P_Q}{m\mu - \lambda}$
- Número médio de clientes no sistema:  $L = \lambda W = m\rho + \frac{\rho P_Q}{1-\rho}$

## Sistema $M/M/1/m$

- Processo de nascimento e morte em que:
  - (1) a chegada de clientes é um processo de Poisson com taxa  $\lambda$
  - (2) o tempo de atendimento de um servidor é exponencialmente distribuído com média  $1/\mu$
  - (3) o sistema tem 1 servidor
  - (4) o sistema acomoda no máximo  $m$  clientes (*i.e.*, a fila de espera tem capacidade para  $m - 1$  clientes)



## Sistema $M/M/1/m$



- Equações de balanço:

$$\lambda P_{n-1} = \mu P_n, \quad n = 1, 2, \dots, m$$

- Probabilidade de  $n$  clientes no sistema em estado estacionário:

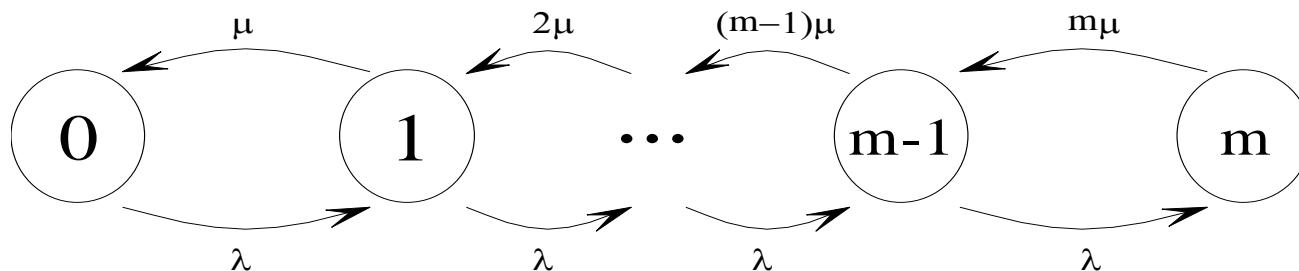
$$P_n = \frac{(\lambda/\mu)^n}{\sum_{i=0}^m (\lambda/\mu)^i} \quad n = 0, 1, \dots, m$$

- Pela propriedade PASTA, a probabilidade de uma chegada encontrar o sistema cheio (*i.e.*, o servidor ocupado e a fila de espera cheia):

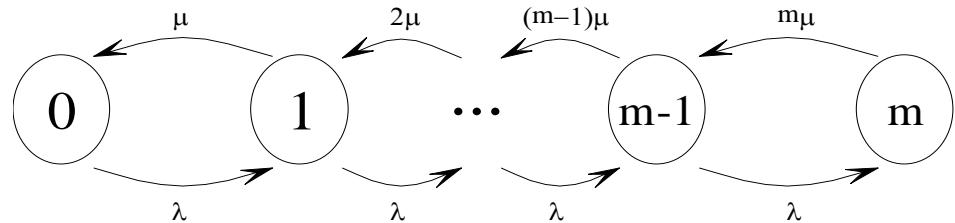
$$P_m = \frac{(\lambda/\mu)^m}{\sum_{i=0}^m (\lambda/\mu)^i}$$

## Sistema $M/M/m/m$

- Processo de nascimento e morte em que:
  - (1) a chegada de clientes é um processo de Poisson com taxa  $\lambda$
  - (2) o tempo de atendimento de um servidor é exponencialmente distribuído com média  $1/\mu$
  - (3) o sistema tem  $m$  servidores
  - (4) o sistema acomoda no máximo  $m$  clientes (*i.e.*, não tem fila de espera)



## Sistema $M/M/m/m$



- Equações de balanço:

$$\lambda P_{n-1} = n\mu P_n, \quad n = 1, 2, \dots, m$$

- Probabilidade de  $n$  clientes no sistema em estado estacionário:

$$P_n = \frac{(\lambda/\mu)^n / n!}{\sum_{i=0}^m (\lambda/\mu)^i / i!} \quad n = 0, 1, \dots, m$$

- Pela propriedade PASTA, a probabilidade de uma chegada encontrar o sistema cheio é (fórmula de Erlang B):

$$P_m = \frac{(\lambda/\mu)^m / m!}{\sum_{i=0}^m (\lambda/\mu)^i / i!}$$

## Sistema $M/G/1$

- Processo de nascimento e morte em que:
  - (1) a chegada de clientes é um processo de Poisson com taxa  $\lambda$
  - (2) o tempo de atendimento  $S$  do servidor tem uma distribuição genérica e independente das chegadas dos clientes
  - (3) o sistema tem 1 servidor
  - (4) o sistema acomoda um número infinito de clientes
- Sendo conhecidos  $E[S]$  e  $E[S^2]$  do tempo de atendimento  $S$ , a fórmula de Pollaczek - Khintchine enuncia que o atraso médio na fila de espera é dado por:

$$W_Q = \frac{\lambda E[S^2]}{2(1 - \lambda E[S])}$$

- O atraso médio no sistema é dado por:

$$W = \frac{\lambda E[S^2]}{2(1 - \lambda E[S])} + E[S]$$

## Sistema $M/G/1$

$$W_Q = \frac{\lambda E[S^2]}{2(1 - \lambda E[S])}$$

- Quando o tempo de serviço é exponencialmente distribuído, o sistema resulta num  $M/M/1$ :

$$E[S] = 1/\mu$$

$$E[S^2] = 2/\mu^2$$

$$W_Q = \frac{\lambda}{\mu(\mu - \lambda)}$$

- Quando os tempos de serviço são iguais para todos os clientes com valor  $1/\mu$ , o sistema resulta num  $M/D/1$ :

$$E[S] = 1/\mu$$

$$E[S^2] = 1/\mu^2$$

$$W_Q = \frac{\lambda}{2\mu(\mu - \lambda)}$$

- Uma ligação ponto-a-ponto com capacidade  $\mu$  pacotes/s e uma fila de espera muito grande onde chegam pacotes a uma taxa de Poisson  $\lambda$  pacotes/s é um sistema  $M/G/1$ .

- Se o comprimento dos pacotes for exponencialmente distribuído, degenera num sistema  $M/M/1$ .
- Se o comprimento dos pacotes for fixo, degenera num sistema  $M/D/1$ .

## Exemplo 2

Considere um sistema de transmissão de pacotes com uma fila de espera seguida de uma linha de transmissão de 64 Kbps. O tráfego oferecido é de Poisson com taxa de 15 pacotes/segundo. O comprimento dos pacotes é exponencialmente distribuído com média de 400 bytes. A fila de espera tem capacidade para 3 pacotes. Um pacote que chegue ao sistema é encaminhado pela linha de transmissão, se estiver livre, ou posto em fila de espera. Se a fila de espera se encontrar cheia, o pacote é perdido. Calcule:

- (a) A percentagem de pacotes perdidos.
- (b) A percentagem de pacotes que não sofre atraso na fila de espera.
- (c) A percentagem de utilização da linha de transmissão.

Fazer em casa!



# **Introdução à Simulação baseada em Eventos Discretos**

Desempenho e Dimensionamento de Redes

Prof. Amaro de Sousa ([asou@ua.pt](mailto:asou@ua.pt))

DETI-UA, 2017/2018

# Simulação baseada em eventos discretos

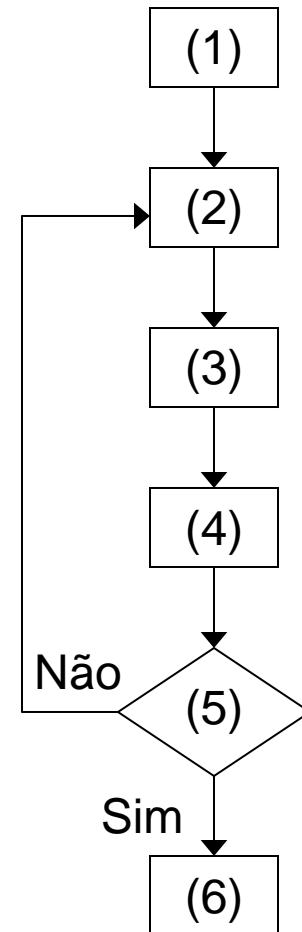
Modelação da evolução temporal de um sistema, através de uma representação na qual as variáveis que descrevem o estado do sistema mudam de valor em instantes discretos no tempo. Os instantes de tempo são aqueles em que ocorre um evento.

## Elementos de um programa de simulação:

- (1) Variáveis de estado: descrevem o estado do sistema
- (2) Contadores estatísticos: variáveis que armazenam informação estatística relativa ao desempenho do sistema
- (3) Relógio de simulação: variável que indica, em cada instante, o tempo de simulação ( $\text{tempo de simulação} \neq \text{tempo de computação}$ )
- (4) Eventos: tipos de ocorrências instantâneas que alteram o estado do sistema e/ou os contadores estatísticos
- (5) Lista de eventos: lista onde são armazenados os instantes de ocorrência dos eventos futuros

# Estrutura básica de um simulador baseado em eventos discretos

- (1) Inicializa as variáveis de estado, os contadores estatísticos e a lista de eventos com o(s) primeiro(s) evento(s);
- (2) Determina qual o próximo evento da lista de eventos;
- (3) Avança o relógio de simulação para o instante de ocorrência desse evento;
- (4) Executa as ações associadas a esse evento (geração de novos eventos e atualização das variáveis de estado e dos contadores estatísticos);
- (5) Determina se a simulação deve terminar; se não, retorna ao passo (2);
- (6) Atualiza os contadores estatísticos e determina as estimativas de interesse.



# **Exemplo: Simulador de um servidor de *video-streaming***

Parâmetros de entrada do simulador:

- (1) Tempo entre pedidos de filmes: variável exponencialmente distribuída com média 60 minutos
- (2) Duração média dos filmes: variável exponencialmente distribuída com média 90 minutos
- (3) Capacidade da ligação de rede do servidor = 2 filmes

Critério de paragem de simulação:

Instante de tempo do 4º pedido de filme (este pedido conta para os contadores estatísticos)

Medidas de desempenho que se pretende estimar:

- (1) Probabilidade de bloqueio: percentagem de pedidos de filmes que são recusados porque a ligação de rede está completamente ocupada
- (2) Ocupação média da ligação de rede (em número de filmes)

# **Exemplo:**

## **Simulador de um servidor de *video-streaming***

Eventos:

CHEGADA: pedido de um filme

PARTIDA: terminação de um filme em transmissão

Variáveis de estado:

ESTADO: número de filmes em transmissão

Contadores estatísticos:

OCUPAÇÃO: integral da ocupação da ligação (em número de filmes) desde o início da simulação até ao instante presente

P\_RECUSADOS: número de pedidos de filmes recusados até ao instante presente

PEDIDOS: número de pedidos de filmes até ao instante presente

# **Instante $t = 0,0$**

## **Início da Simulação**

Relógio de Simulação:

0

LISTA DE EVENTOS  
CHEGADA - 63 min

---

Variável de Estado:

ESTADO  
0

---

Contadores  
Estatísticos:

OCUPAÇÃO  
0

P\_RECUSADOS  
0

PEDIDOS  
0

## Instante $t = 63$ (Primeira CHEGADA)

Relógio de Simulação:

63

### LISTA DE EVENTOS

CHEGADA – 63 min

CHEGADA - 117 min

PARTIDA - 153 min

Variável de Estado:

ESTADO  
1

Contadores  
Estatísticos:

OCUPAÇÃO  
0

$$\leftarrow 0 + 0 \times (63 - 0,0)$$

P\_RECUSADOS  
0

$$\leftarrow 0 + 0$$

PEDIDOS  
1

## Instante $t = 117$ (Segunda CHEGADA)

Relógio de Simulação:

117

Variável de Estado:

ESTADO  
2

Contadores  
Estatísticos:

OCUPAÇÃO  
54

$\leftarrow 0 + 1 \times (117 - 63)$

P\_RECUSADOS  
0

$\leftarrow 0 + 0$

PEDIDOS  
2

### LISTA DE EVENTOS

CHEGADA - 63 min

CHEGADA - 117 min

CHEGADA - 150 min

PARTIDA - 153 min

PARTIDA - 207 min

## Instante $t = 150$ (Terceira CHEGADA)

Relógio de Simulação:

150

Variável de Estado:

ESTADO  
2

Contadores  
Estatísticos:

OCUPAÇÃO  
120

P\_RECUSADOS  
1

PEDIDOS  
3

### LISTA DE EVENTOS

CHEGADA - 63 min  
CHEGADA - 117 min  
CHEGADA - 150 min  
PARTIDA - 153 min  
CHEGADA - 204 min  
PARTIDA - 207 min

$$\leftarrow 54 + 2 \times (150 - 117)$$

$$\leftarrow 0 + 1$$

## Instante $t = 153$ (Primeira PARTIDA)

Relógio de Simulação:

153

Variável de Estado:

ESTADO  
1

Contadores  
Estatísticos:

OCUPAÇÃO  
126

P\_RECUSADOS  
1

PEDIDOS  
3

### LISTA DE EVENTOS

CHEGADA - 63 min

CHEGADA - 117 min

CHEGADA - 150 min

PARTIDA - 153 min

CHEGADA - 204 min

PARTIDA - 207 min

$$\leftarrow 120 + 2 \times (153 - 150)$$

## Instante $t = 204$ (Quarta CHEGADA)

Fim da Simulação

Relógio de Simulação:

204

Variável de Estado:

ESTADO  
2

Contadores  
Estatísticos:

OCUPAÇÃO  
177

P\_RECUSADOS  
1

PEDIDOS  
4

### LISTA DE EVENTOS

CHEGADA - 63 min  
CHEGADA - 117 min  
CHEGADA - 150 min  
PARTIDA - 153 min  
CHEGADA - 204 min  
PARTIDA - 207 min

$$\leftarrow 126 + 1 \times (204 - 153)$$

Probabilidade de bloqueio:

$$P_{\text{RECUSADOS}} / P_{\text{EDIDOS}} = 1/4 = 25\%$$

Ocupação média da ligação:

$$OCUPAÇÃO / t = 177/204 = 0,87 \text{ filmes}$$

# Geração de valores aleatórios com distribuição uniforme entre 0 e 1

Os geradores mais populares são os geradores lineares congruenciais (LCG - *Linear Congruential Generator*)

Método de geração:

(1) Geram-se os inteiros  $Z_1, Z_2, \dots$  de acordo com a fórmula recursiva

$$Z_i = (aZ_{i-1} + c)(\text{mod } m)$$

onde  $m, a, c$  e  $Z_0$  são inteiros não-negativos

(2) Faz-se  $U_i = Z_i / m$

Os números  $U_i$  parecem ser uniformemente distribuídos no intervalo  $[0,1]$

# Exemplo

Exemplo:  $Z_i = (5Z_{i-1} + 3)(\text{mod } 16)$

$$Z_0 = 7$$

$i$	$Z_i$	$U_i$	$i$	$Z_i$	$U_i$
0	7	----	10	9	0.563
1	6	0.375	11	0	0.000
2	1	0.063	12	3	0.188
3	8	0.500	13	2	0.125
4	11	0.688	14	13	0.813
5	10	0.625	15	4	0.250
6	5	0.313	16	7	0.438
7	12	0.750	17	6	0.375
8	15	0.938	18	1	0.063
9	14	0.875	19	8	0.500

Este gerador tem período 16. O gerador do MATLAB tem período  $2^{31}-1$

# Geração de valores com outras distribuições

## Variáveis discretas:

Considere-se que a variável aleatória pode assumir os valores  $X_1, X_2, \dots, X_n$ . A probabilidade do valor  $X_i$  é  $P(X = X_i) = f_i$ .

Método:

- Dividir o intervalo  $[0,1]$  em  $n$  intervalos proporcionais a  $f_i$ , com  $i = 1 \dots n$
- Gerar um valor aleatório  $U$  em  $[0,1]$  com distribuição uniforme
- Retornar  $X_i$  se  $U$  estiver no  $i$ -ésimo intervalo

Por exemplo, a variável aleatória de Bernoulli  $X$  com  $p(0) = 1/4$  e  $p(1) = 3/4$  pode ser gerada pelo algoritmo:

- (1) Gerar  $U \sim U(0,1)$
- (2) Se  $U \leq 1/4$ , retornar  $X = 0$ ; caso contrário, retornar  $X = 1$

# Geração de valores com outras distribuições

Variáveis discretas (exemplo MATLAB)

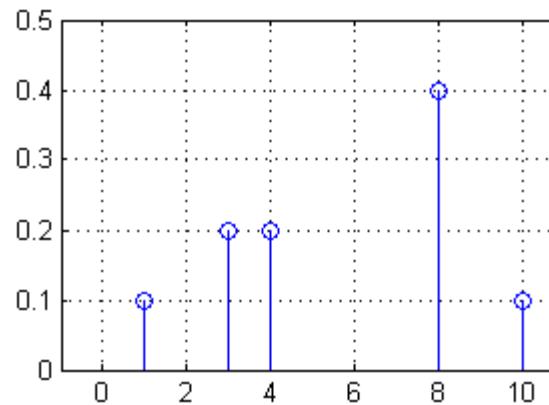
```
x= [1 3 4 8 10];  
f= [0.1 0.2 0.2 0.4 0.1];
```

```
figure(1)  
stem(x,f)  
axis([-1 11 0 0.5])  
grid on
```

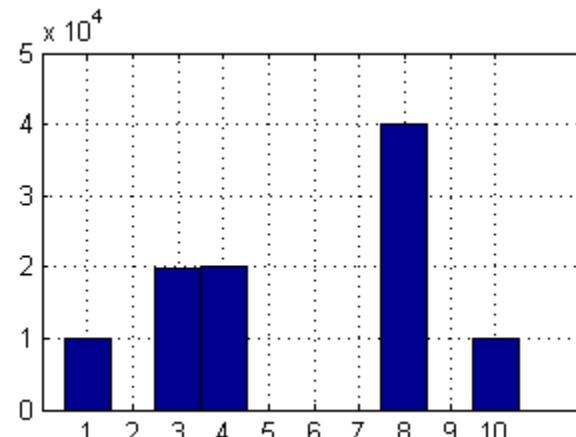
```
f_cum= cumsum(f)
```

```
a= zeros(1,100000);  
for it= 1:100000  
    a(it)= x(sum(rand()>f_cum)+1);  
end
```

```
figure(2)  
hist(a,1:10)  
Grid on
```



```
f_cum =  
  
0.1 0.3 0.5 0.9 1.0
```



# Geração de valores com outras distribuições

Variáveis contínuas:

Algoritmos mais populares baseiam-se no método da transformação inversa.

Considere  $F(X)$  a função distribuição de uma variável aleatória contínua  $X$ . Considere  $F^{-1}(U)$  a sua função inversa.

Método:

- (1) Gerar  $U \sim U(0,1)$
- (2) Retornar  $X = F^{-1}(U)$

Por exemplo, numa variável aleatória com distribuição exponencial de média  $1/\lambda$ :

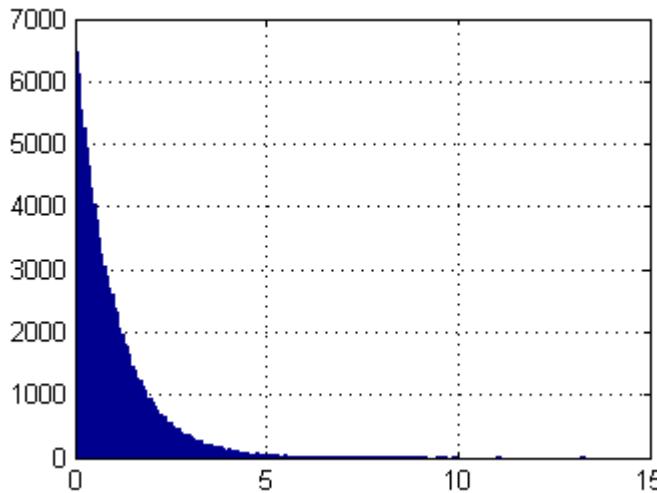
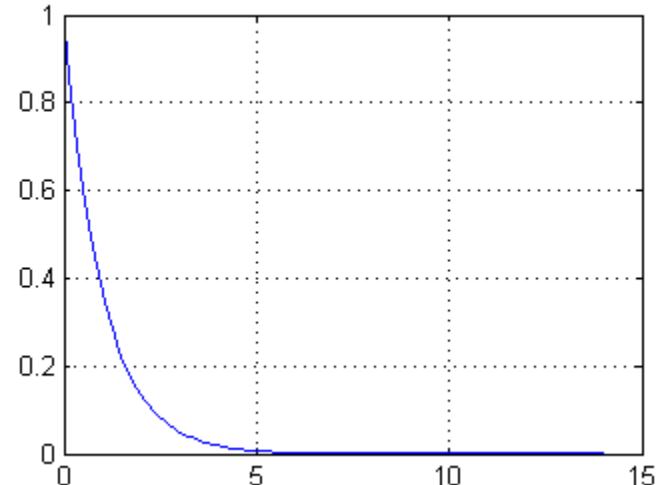
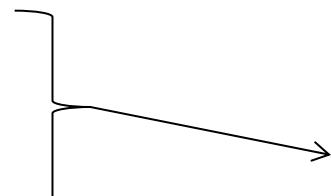
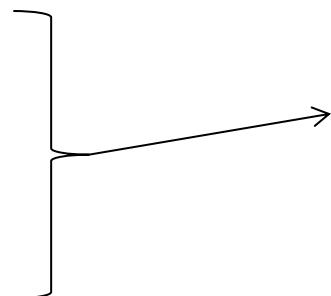
$$F(x) = \begin{cases} 1 - e^{-\lambda x}, & x \geq 0 \\ 0, & x < 0 \end{cases} \quad F^{-1}(U) = -\frac{1}{\lambda} \ln(U)$$

# Geração de valores com outras distribuições

Variável exponencial (exemplo MATLAB)

```
x= 0:0.1:14;  
f=exppdf(x,1)  
figure(1)  
plot(x,f)  
grid on
```

```
a=exprnd(1,1,100000);  
figure(2)  
hist(a,200)  
grid on
```



# Análise dos resultados de uma simulação

Sejam  $X_1, X_2, \dots, X_n$  observações de variáveis aleatórias Independentes e Identicamente Distribuídas (IID) com média  $\mu$  e variância  $\rho^2$  finitas (resultado de diferentes réplicas da simulação do sistema).

A média amostral definida por  
é um estimador de  $\mu$ .

$$\bar{X}(n) = \frac{\sum_{i=1}^n X_i}{n}$$

A variância amostral definida por  
é um estimador de  $\rho^2$ .

$$S^2(n) = \frac{\sum_{i=1}^n (X_i - \bar{X}(n))^2}{n-1}$$

A análise dos resultados de uma simulação é feita normalmente com base no teorema do limite central.

# Análise dos resultados de uma simulação

Seja  $Z_n$  a variável aleatória dada por

$$Z_n = \frac{\bar{X}(n) - \mu}{\sqrt{\sigma^2/n}}$$

e seja  $F_n(z)$  a função distribuição de  $Z_n$  para uma amostra de tamanho  $n$ .

O teorema do limite central enuncia que

$$\lim_{n \rightarrow +\infty} F_n(z) = \Phi(z)$$

em que  $\Phi(z)$  é a função distribuição de uma variável aleatória Gaussiana (ou Normal) padrão (média nula e desvio padrão unitário).

Uma vez que  $\lim_{n \rightarrow +\infty} S^2(n) = \rho^2$  então a v.a.  $\frac{\bar{X}(n) - \mu}{\sqrt{S^2(n)/n}}$

tem uma distribuição aproximadamente Gaussiana padrão.

# Análise dos resultados de uma simulação

Para  $n$  suficientemente elevado,

$$P\left(-z_{1-\alpha/2} \leq \frac{\bar{X}(n) - \mu}{\sqrt{S^2(n)/n}} \leq z_{1-\alpha/2}\right) = \\ P\left(\bar{X}(n) - z_{1-\alpha/2} \sqrt{S^2(n)/n} \leq \mu \leq \bar{X}(n) + z_{1-\alpha/2} \sqrt{S^2(n)/n}\right) \approx 1 - \alpha$$

onde  $z_{1-\alpha/2}$  é o ponto crítico da distribuição Gaussiana padrão ( $z_{1-\alpha/2}$  é o valor de  $z$  tal que  $P(x \leq z) = 1 - \alpha/2$  em que  $x$  é uma variável com distribuição Gaussiana padrão).

Assim, o intervalo de confiança aproximado a  $100(1-\alpha)\%$  para  $\mu$  é dado por

$$\bar{X}(n) \pm z_{1-\alpha/2} \sqrt{S^2(n)/n}$$

# Análise dos resultados de uma simulação

O intervalo de confiança aproximado a  $100(1-\alpha)\%$  para  $\mu$  é dado por

$$\bar{X}(n) \pm z_{1-\alpha/2} \sqrt{S^2(n)/n}$$

Em MATLAB:

```
N = 20; %número de simulações
results= zeros(1,N); %vetor com os N resultados de simulação
for it= 1:N
    results(it)= simulator();
end

alfa= 0.1; %intervalo de confiança a 90%
media = mean(results);
termo = norminv(1-alfa/2)*sqrt(var(results)/N);

fprintf('resultado = %.2e +- %.2e\n',media,termo)
```

# Análise dos resultados de uma simulação

O teorema do limite central requer que as variáveis  $X_1, X_2, \dots, X_n$  sejam independentes e identicamente distribuídas (IID).

- Uma das formas de garantir a independência é replicar a simulação em que cada réplica é iniciada com números aleatórios distintos dando assim origem a observações independentes.

Em geral, os processos estocásticos têm regimes transitórios (que dependem das condições iniciais) antes de atingir o regime estacionário.

- Para garantir que os parâmetros de desempenho são corretamente calculados, é necessário deixar aquecer a simulação (*warm-up*) dando tempo para que os regimes transitórios iniciais se extingam.
- Se o tempo simulado for muito superior ao tempo de *warm-up*, os contadores estatísticos podem ser inicializados no início da simulação.
- Se não, os contadores estatísticos devem ser inicializados apenas após o tempo de *warm-up* (que deverá ser previamente estimado).



## **Partilha de um Recurso de Comunicações**

Desempenho e Dimensionamento de Redes

Prof. Amaro de Sousa ([asou@ua.pt](mailto:asou@ua.pt))

DETI-UA, 2017/2018

# Recurso de comunicações digital

Um recurso de comunicações digital é um elemento de rede que permite a troca de informação em formato digital.

Pode ser visto como um *túnel de bits* através do qual pode ser transmitido/recebido um dado valor em bits/s. Este valor é designado por capacidade do recurso.

Exemplos:

- Cada sentido de uma ligação ponto-a-ponto bidirecional (um cabo Ethernet a ligar um PC a um *switch* Ethernet) é um recurso de comunicações entre os dois elementos.
- Uma interface de rede de um servidor de vídeo é um recurso de comunicações do servidor para os terminais de vídeo.
- Uma ligação de um router de uma casa ao ISP é constituído por dois recursos de comunicação, um em cada sentido, cuja capacidade é normalmente diferente em cada sentido.

# Recurso de comunicações organizado em circuitos

Um recurso de comunicações pode ser explicitamente estruturado em circuitos (ou canais), que correspondem a partições da capacidade total do recurso.

- Um circuito é caracterizado pela sua largura de banda (um valor também em bits/s).
- A estruturação de uma ligação ponto-a-ponto em circuitos pode ser feita através de:
  - (1) multiplexagem no tempo (sistemas TDM)
  - (2) multiplexagem na frequência (sistemas FDM)
  - (3) multiplexagem de comprimentos de onda (sistemas WDM)

Um recurso de comunicações pode também ser implicitamente usado como se um fosse estruturado em circuitos (noção de circuitos virtuais).

- Cada comunicação usa uma partição da capacidade do recurso.

# Recurso de comunicações com estabelecimento de circuitos

Uma chamada, quando é estabelecida, corresponde à atribuição temporária de um circuito (ou mais) com uma determinada largura de banda.

Após estabelecida, a chamada dura um tempo finito após o qual a largura de banda do(s) circuito(s) atribuído(s) é libertada e fica disponível para pedidos futuros de chamadas.

Assim, um recurso de comunicações pode ser partilhado por diferentes fluxos de chamadas.

Uma classe de serviço identifica um fluxo de chamadas com as mesmas características de tráfego e os mesmos requisitos de largura de banda.

# Recurso de comunicações com estabelecimento de circuitos

As características de tráfego são determinadas

- (i) pela descrição estocástica do processo de chegada de chamadas,
- (ii) pela descrição estocástica do tempo das chamadas, e
- (iii) pela largura de banda (em nº de circuitos) requerida por cada chamada.

O parâmetro de qualidade de serviço mais importantes é a probabilidade de bloqueio (probabilidade de um pedido de chamada não ser aceite pela ligação por falta de recursos disponíveis).

Um recurso de comunicações pode ser:

- (1) uni-serviço – se suportar apenas fluxos de chamadas de uma única classe de serviço
- (2) multi-serviço – se suportar fluxos de chamadas de diferentes classes de serviço

A diferenciação de serviço faz-se através da função de controle de fluxos (também designada de controle de admissão de chamadas)

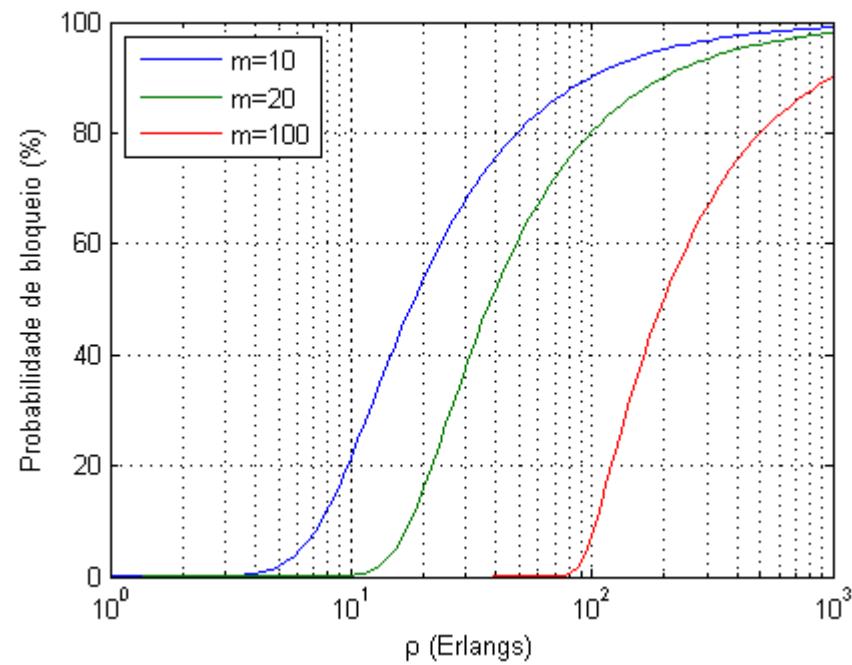
# Recurso de comunicações uni-serviço - Distribuição de ErlangB

Um recurso de comunicações modelado por um grupo de  $m$  circuitos ao qual é oferecido um fluxo de chamadas de Poisson com taxa  $\lambda$  e em que a duração das chamadas é exponencialmente distribuída com média  $1/\mu$  é modelado por um sistema de fila de espera  $M/M/m/m$ .

Designa-se como unidade de intensidade de tráfego,  $\rho = \lambda/\mu$  , o Erlang.

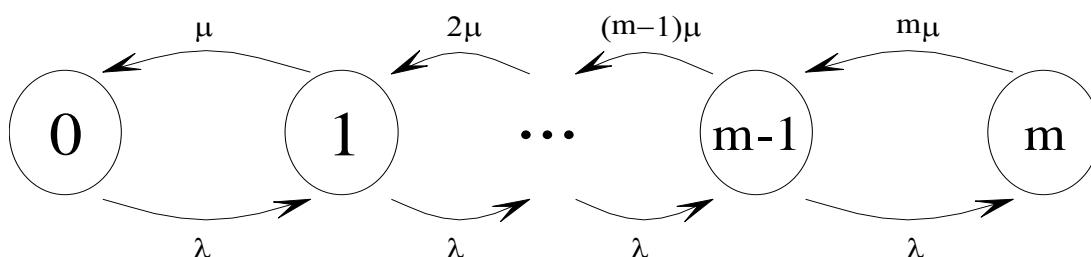
A fórmula de ErlangB, dá a probabilidade de bloqueio:

$$P_m = \frac{\rho^m / m!}{\sum_{n=0}^m \rho^n / n!} = E(\rho, m)$$



# Estabelecimento de circuitos uni-serviço

Cadeia de Markov de um sistema de fila de espera  $M/M/m/m$ :

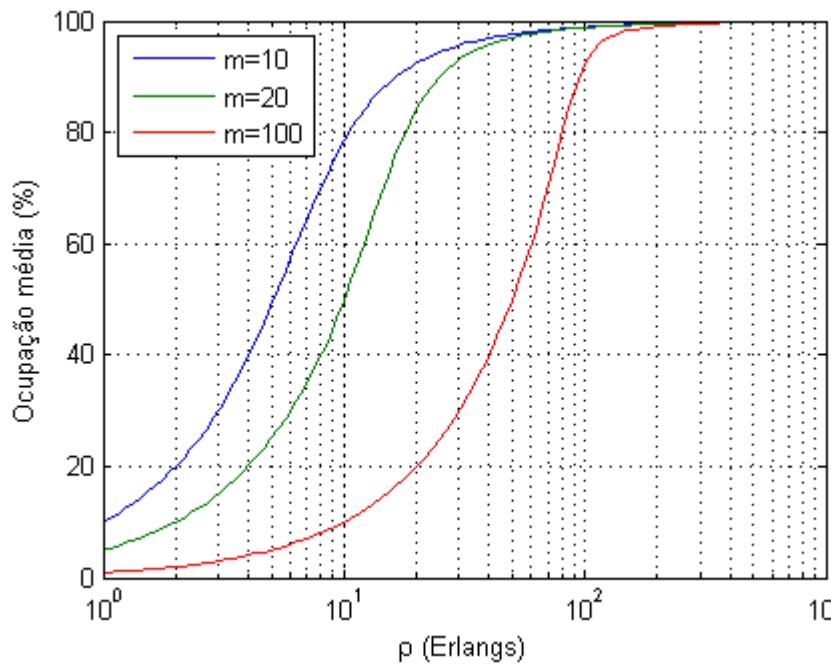


$$\pi_0 = \frac{1}{1 + \sum_{i=1}^m \frac{\lambda_0 \lambda_1 \cdots \lambda_{i-1}}{\mu_1 \mu_2 \cdots \mu_i}}$$

$$\pi_n = \frac{\lambda_0 \lambda_1 \cdots \lambda_{n-1}}{\mu_1 \mu_2 \cdots \mu_n} \cdot \pi_0, \quad n \geq 1$$

A ocupação média da ligação  
é ( $\rho = \lambda/\mu$ ):

$$\sum_{i=0}^m (i \times \pi_i) = \frac{\sum_{i=1}^m \frac{\rho^i}{(i-1)!}}{\sum_{i=0}^m \frac{\rho^i}{i!}}$$



# Recurso de comunicações uni-serviço

## Distribuição de Engset

Considere um recurso de comunicações com capacidade para  $N$  circuitos e que serve  $M$  clientes ( $M > N$ ).

Cada um dos  $M$  clientes está inativo durante um período de tempo exponencialmente distribuído com média  $1/\lambda$  e gera uma chamada com uma duração média  $1/\mu$ .

A cada chamada pedida é atribuído um dos  $N$  circuitos disponíveis; se não houver qualquer circuito disponível a chamada é bloqueada.

Este sistema pode ser modelado por um processo de nascimento e morte com os estados  $n = 0, 1, \dots, N$  (representando o número de circuitos ocupados) e com taxas de nascimento e de morte dadas por:

$$\lambda_n = (M - n)\lambda, \quad 0 \leq n \leq N - 1$$

$$\mu_n = n\mu, \quad 1 \leq n \leq N$$

# Recurso de comunicações uni-serviço

## Distribuição de Engset

A probabilidade de  $n$  chamadas no sistema é

$$\pi_n = \frac{\binom{M}{n} \left(\frac{\lambda}{\mu}\right)^n}{\sum_{n=0}^N \binom{M}{n} \left(\frac{\lambda}{\mu}\right)^n}$$

A probabilidade de bloqueio é

$$P_B = \frac{\binom{M-1}{N} \left(\frac{\lambda}{\mu}\right)^N}{\sum_{n=0}^N \binom{M-1}{n} \left(\frac{\lambda}{\mu}\right)^n}$$

Esta é a chamada distribuição de Engset.

Note-se que a propriedade PASTA não se verifica ( $\pi_n \neq P_B$ ) dado que a taxa de chegada de chamadas não é estatisticamente independente do estado do sistema.

Esta fórmula degenera na fórmula de ErlangB, quando  $M \rightarrow \infty$ ,  $\lambda \rightarrow 0$ , com  $M\lambda$  constante.

# Estabelecimento de circuitos multi-serviço

Considere um recurso de comunicações com  $C$  circuitos que serve as classes de serviço  $k = 1, 2, \dots, K$  ( $K$  é o número de classes de serviço).

A cada classe  $k$  está associada uma taxa de chegada,  $\lambda_k$ , um tempo médio de serviço,  $1/\mu_k$  e uma largura de banda  $b_k$  (em numero de circuitos):

- As chamadas das  $K$  classes chegam de acordo com processos independentes de Poisson à taxa  $\lambda_k$ .
- Uma chamada da classe  $k$  que tenha sido admitida pelo sistema, ocupa  $b_k$  circuitos durante o tempo de serviço da chamada, o qual é exponencialmente distribuído com média  $1/\mu_k$ .

Seja  $\rho_k$  a intensidade de tráfego de cada classe, isto é,  $\rho_k = \lambda_k / \mu_k$ .

Assuma-se que os tempos de serviço das chamadas são independentes entre si e independentes dos processos de chegadas.

# Estabelecimento de circuitos multi-serviço

Seja  $n_k$  o número de chamadas da classe  $k$  no sistema.

Considerem-se os vetores  $\mathbf{n} = (n_1, \dots, n_k)$  e  $\mathbf{b} = (b_1, \dots, b_k)$ .

O número total de circuitos ocupados no sistema é:

$$\mathbf{b} \cdot \mathbf{n} = \sum_{k=1}^K b_k n_k$$

Uma chamada da classe  $k$  é admitida no sistema se  $b_k \leq C - \mathbf{b} \cdot \mathbf{n}$ ; caso contrário é bloqueada e perdida.

O espaço de estados do processo de nascimento e morte multidimensional é:

$$S = \left\{ \mathbf{n} \in I^K : \mathbf{b} \cdot \mathbf{n} \leq C \right\}$$

onde  $I$  é o conjunto dos inteiros não negativos.

Seja  $S_k$  o subconjunto dos estados para os quais uma chamada da classe  $k$  é admitida quando chega à rede, isto é,

$$S_k = \left\{ \mathbf{n} \in S : \mathbf{b} \cdot \mathbf{n} \leq C - b_k \right\}$$

# Estabelecimento de circuitos multi-serviço

As probabilidades limite de cada estado são dadas por:

$$\text{onde } G = \sum_{\mathbf{n} \in S} \prod_{l=1}^K \frac{\rho_l^{n_l}}{n_l!} \quad P(\mathbf{n}) = \frac{1}{G} \prod_{l=1}^K \frac{\rho_l^{n_l}}{n_l!} \quad \mathbf{n} \in S$$

e a probabilidade de bloqueio da classe  $k$  por:

$$B_k = 1 - \frac{\sum_{\mathbf{n} \in S_k} \prod_{l=1}^K \frac{\rho_l^{n_l}}{n_l!}}{\sum_{\mathbf{n} \in S} \prod_{l=1}^K \frac{\rho_l^{n_l}}{n_l!}} \Leftrightarrow B_k = \frac{\sum_{\mathbf{n} \in S \setminus S_k} \prod_{l=1}^K \frac{\rho_l^{n_l}}{n_l!}}{\sum_{\mathbf{n} \in S} \prod_{l=1}^K \frac{\rho_l^{n_l}}{n_l!}}$$

- Primeira expressão: 1 menos a probabilidade dos estados  $S_k$  (estados para os quais uma chamada da classe  $k$  é admitida).
- Segunda expressão: probabilidade dos estados  $S \setminus S_k$  (estados para os quais uma chamada da classe  $k$  é rejeitada).

Este sistema é por vezes designado de *stochastic knapsack*.

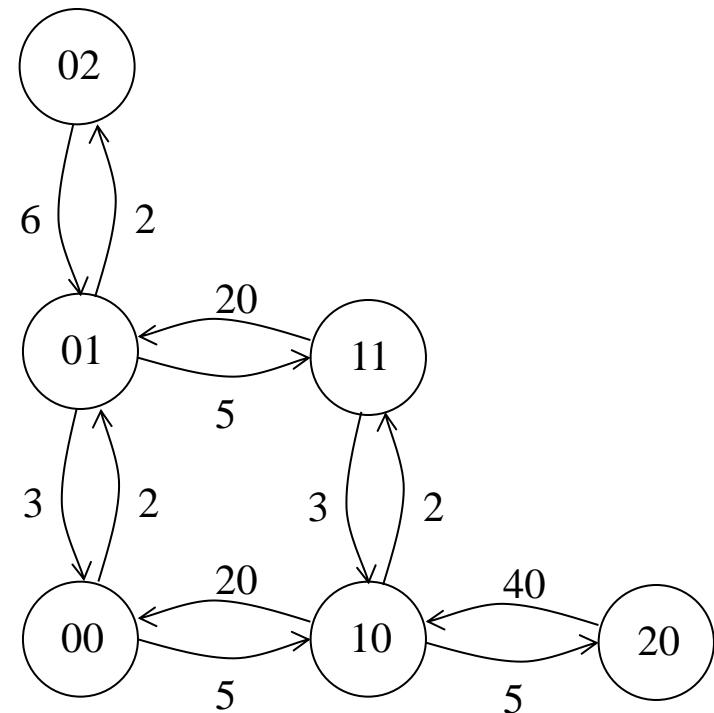
# Exemplo 1

Considere uma rede telefónica VoIP de uma empresa em que o número de chamadas VoIP para o exterior é no máximo 2. Existem dois tipos de chamadas VoIP: chamadas pessoais e chamadas de apoio ao cliente.

O tráfego VoIP de/para a rede pública é um processo de Poisson com taxa de 5 chamadas por hora para chamadas pessoais e de 2 chamadas por hora para as chamadas de apoio ao cliente.

As chamadas pessoais têm duração exponencial com média de 3 minutos e as chamadas de apoio ao cliente têm duração exponencial com média de 20 minutos.

Determine a probabilidade de bloqueio de cada tipo de chamada VoIP.



# Recurso de comunicações com comutação de pacotes

Diferentes fluxos de pacotes podem ser

- (1) suportados por um circuito - multiplexagem estatística,
- (2) suportados por diferentes circuitos - multiplexagem determinística.

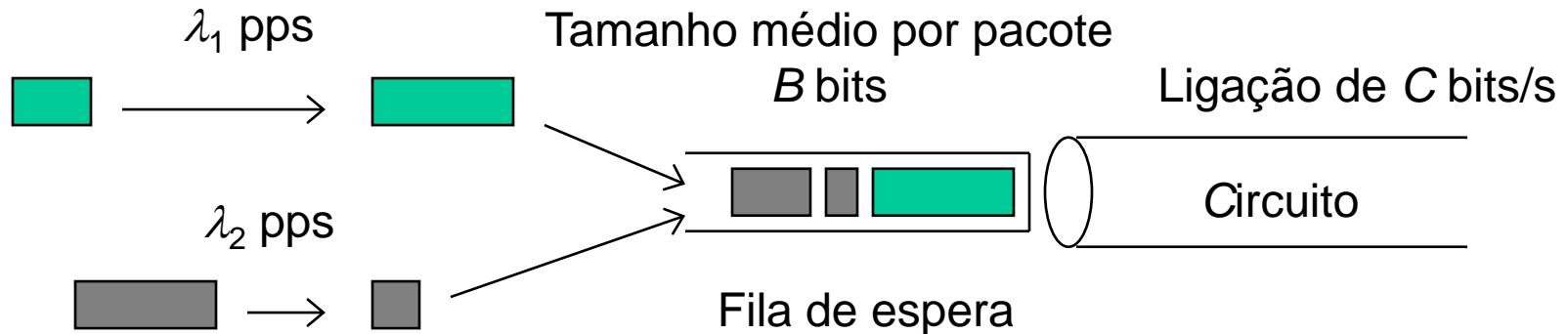
Tal como nos fluxos de chamadas, uma classe de serviço identifica um conjunto de fluxos de pacotes com as mesmas características de tráfego e os mesmos requisitos de qualidade de serviço.

As características de tráfego são determinadas por

- (i) descrição estocástica do processo de chegada dos pacotes, e
- (ii) descrição estocástica do tamanho dos pacotes.

Um dos parâmetro de qualidade de serviço mais importantes é o atraso médio por pacote (existem outros como, por exemplo, a variância do atraso ou a percentagem de perda de pacotes).

# Multiplexagem estatística de fluxos de pacotes



Considere-se que:

- (i) as chegadas de pacotes são processos de Poisson,
- (ii) o tamanho dos pacote é exponencialmente distribuído,
- (iii) a fila de espera é atendida com uma disciplina *First-In-First-Out*.

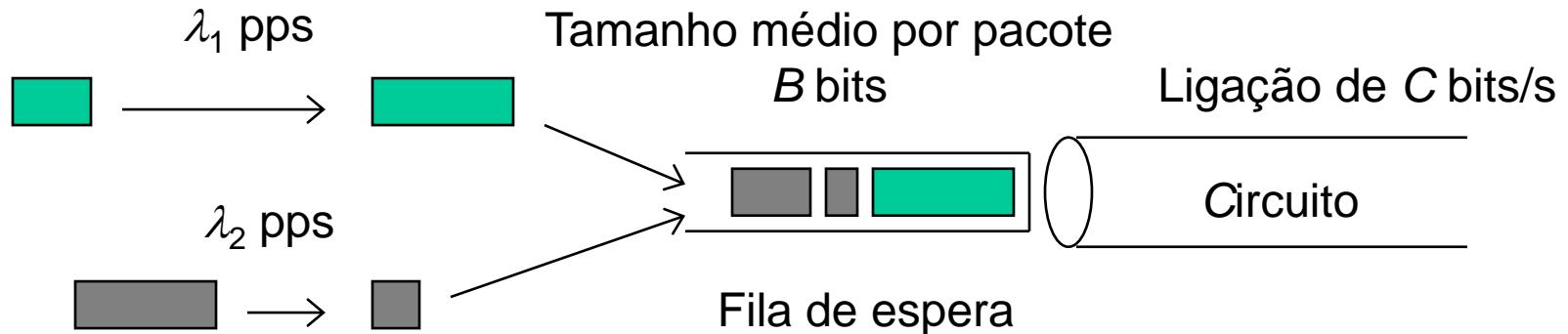
Se a fila de espera for de tamanho infinito, este sistema é modelado pelo sistema de filas de espera  $M/M/1$  em que

$$\begin{aligned}\lambda &= \lambda_1 + \lambda_2 \text{ pps (pacotes por segundo)} \\ \mu &= C/B \text{ pps}\end{aligned}$$

Atraso médio dos pacotes:

$$W = \frac{1}{\mu - \lambda} = \frac{1}{C/B - (\lambda_1 + \lambda_2)}$$

# Multiplexagem estatística de fluxos de pacotes



Se a fila de espera for finita com capacidade para  $m$  pacotes, este sistema é modelado pelo sistema de filas de espera  $M/M/1/m$

Percentagem de perda de pacotes:  $\mu_m = \frac{(\lambda/\mu)^m}{\sum_{i=0}^m (\lambda/\mu)^m}$

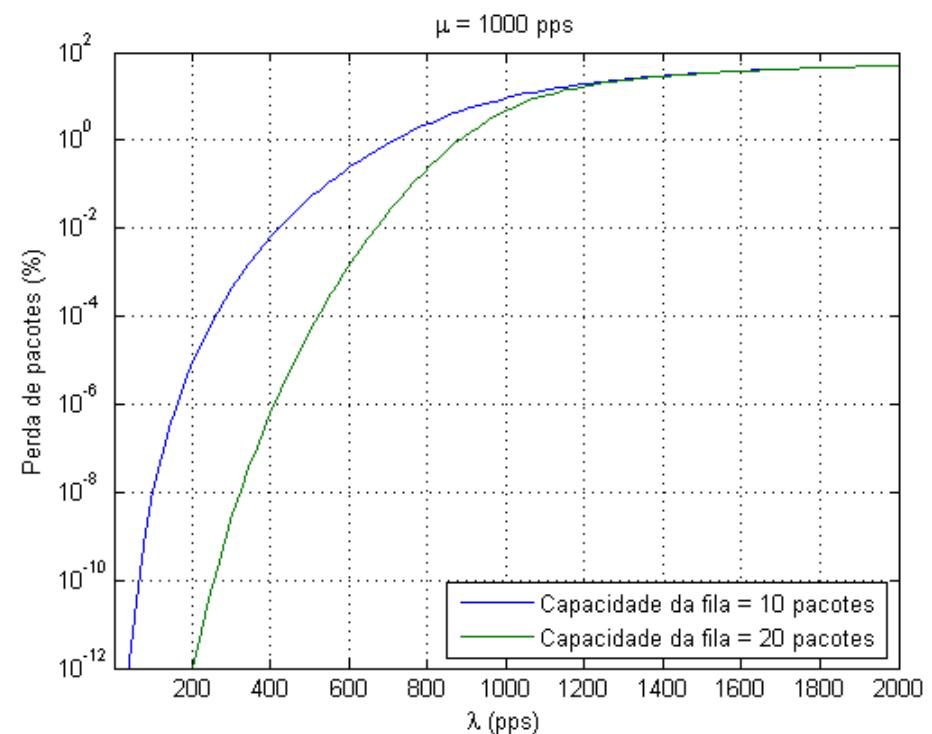
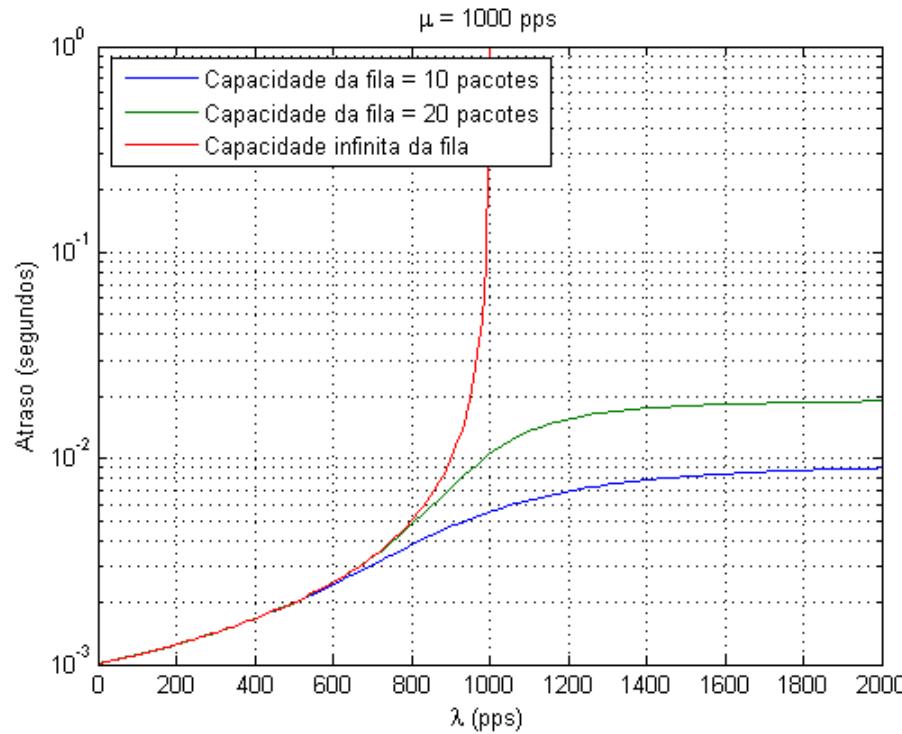
Atraso médio dos pacotes:  $W = \frac{L}{\lambda(1 - \mu_m)} = \frac{\sum_{i=0}^m i \times \pi_i}{\lambda(1 - \mu_m)}$

Teorema de Little

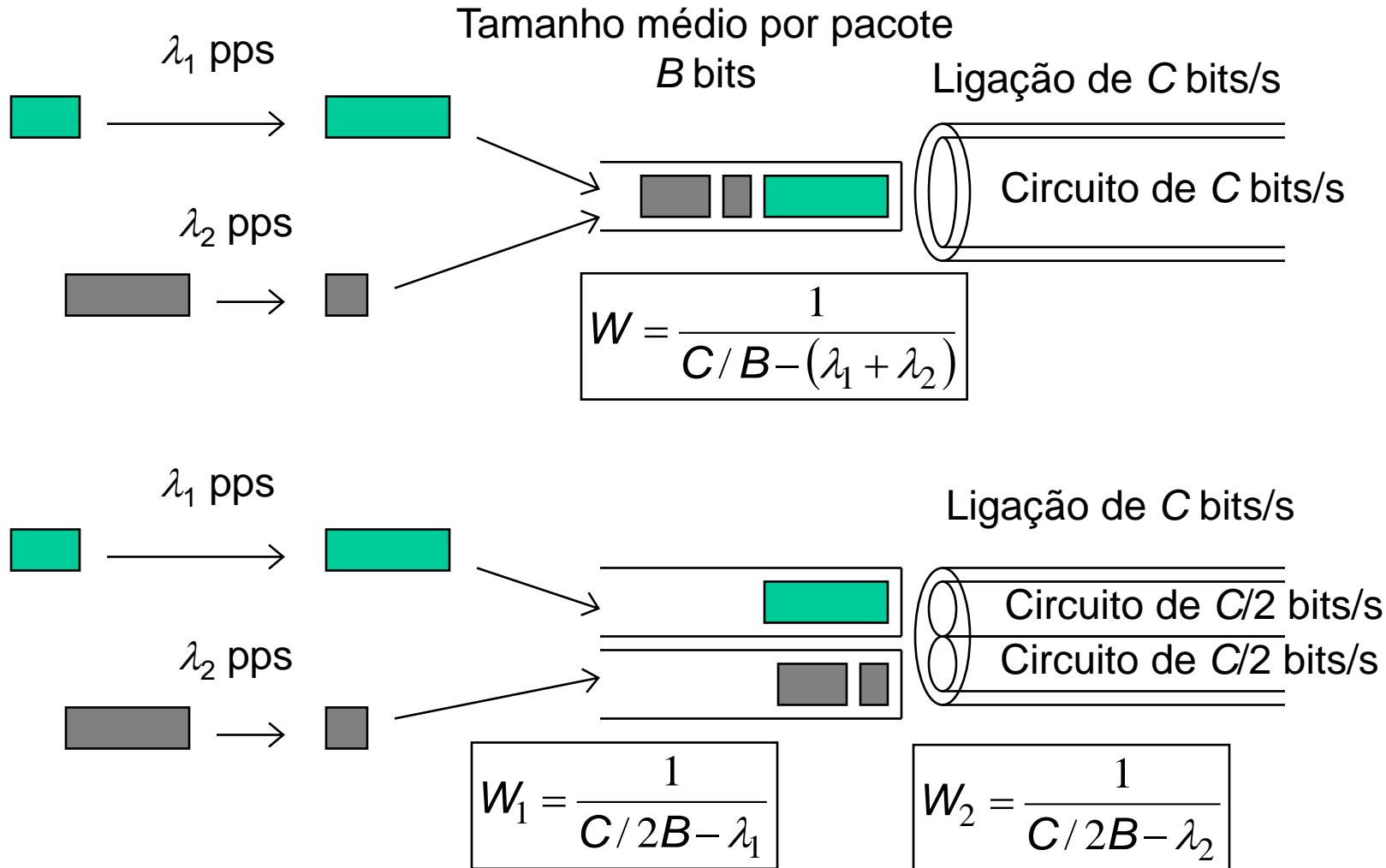
$$= \frac{1}{\lambda(1 - \mu_m)} \times \frac{\sum_{i=0}^m i \times (\lambda/\mu)^i}{\sum_{i=0}^m (\lambda/\mu)^i}$$

# Multiplexagem estatística de fluxos de pacotes

- Se a fila de espera for de tamanho infinito, sistema modelado por  $M/M/1$
- Se a fila de espera tiver capacidade para  $m$  pacotes, sistema modelado por  $M/M/1/m$
- Exemplo:
  - ligação de 10 Mbps e tamanho médio de pacotes de 1250 Bytes
  - $\mu = 10^7 / (1250 \times 8) = 1000 \text{ pps}$



# Multiplexagem Estatística vs. Multiplexagem Determinística



## Exemplo 2

Considere um sistema de transmissão de pacotes constituído por 2 canais de 64 Kbps com uma fila de espera muito grande. Este sistema suporta dois fluxos de pacotes: fluxo A de 1 pacote/segundo e fluxo B de 25 pacotes/segundo. Ambos os fluxos geram pacotes de tamanho exponencialmente distribuído com média de 250 bytes. Calcule o atraso médio por pacotes de cada fluxo quando:

- (a) Ambos os fluxos são multiplexados estatisticamente na capacidade total do sistema de transmissão.
- (b) Cada fluxo é encaminhado separadamente por cada um dos canais do sistema de transmissão.

# **Multiplexagem Estatística vs. Multiplexagem Determinística**

- Em geral, a multiplexagem estatística conduz a atrasos médios por pacote inferiores:  
Na multiplexagem determinística, um circuito dedicado a um fluxo e que esteja momentaneamente livre não pode ser usado pelos outros fluxos.
- No entanto, a multiplexagem determinística conduz a variâncias de atraso menores:  
Na multiplexagem estatística, o número médio de clientes em espera é maior.

# Disciplina com prioridades

Na multiplexagem estatística, os pacotes de cada fluxo podem ser tratados (i.e., transmitidos) de forma diferenciada.

Uma possibilidade é atribuir prioridades aos fluxos, de tal forma que os pacotes de um fluxo com maior prioridade são sempre transmitidos antes do que os pacotes de um fluxo com menor prioridade (esquema designado de *prioritização estrita*).

O sistema M/G/1 com prioridades pode ser utilizado para modelar este sistema.

Considere um sistema  $M/G/1$  em que existem  $n$  classes de serviço.

A  $k$ -ésima classe é determinada por:

(1) taxa de chegadas:  $\lambda_k$

(2) 1º e 2º momentos do tempo de serviço:  $E(S_k) = \frac{1}{\mu_k}$        $E(S_k^2)$

(3) prioridade:  $k$  (quanto menor este valor, maior a prioridade)

# Sistema M/G/1 com prioridades

O sistema transmite primeiro os pacotes das classes com maior prioridade.

Os pacotes de uma mesma classe são transmitidos por ordem de chegada (disciplina *FIFO - First In First Out*).

Considera-se que as chegadas dos pacotes de cada classe são independentes e de Poisson e independentes dos tempos de transmissão.

A transmissão de um pacote não é interrompida pela chegada de um pacote de uma classe com maior prioridade (disciplina de serviço designada por *não-preemptiva*).

O atraso médio na fila de espera correspondente aos pacotes da classe  $k$  é dado por:

$$W_{Qk} = \frac{\sum_{i=1}^n \lambda_i E(S_i^2)}{2(1 - \rho_1 - \dots - \rho_{k-1})(1 - \rho_1 - \dots - \rho_k)} \quad \begin{aligned} \rho_k &= \lambda_k / \mu_k \\ \rho_1 + \dots + \rho_n &< 1 \end{aligned}$$

## Disciplina “*Fair Queuing*”

- Na disciplina FIFO, um fluxo de pacotes que decida momentaneamente aumentar a sua taxa de transmissão captura uma fração arbitrariamente grande dos recursos do sistema.
- Na disciplina com prioridades, enquanto a fila de espera de um fluxo de maior prioridade contiver pacotes, os fluxos de menor prioridade nunca serão servidos.

Uma disciplina que permite evitar estas desvantagens é o *fair queuing*. Nesta disciplina, existe uma fila de espera separada para cada fluxo de pacotes e as filas de espera são servidas proporcionalmente aos pesos que lhe foram atribuídos.

Assim, se a fonte de um fluxo envia pacotes a uma taxa demasiado rápida, a sua fila de espera enche sem afetar os restantes fluxos.

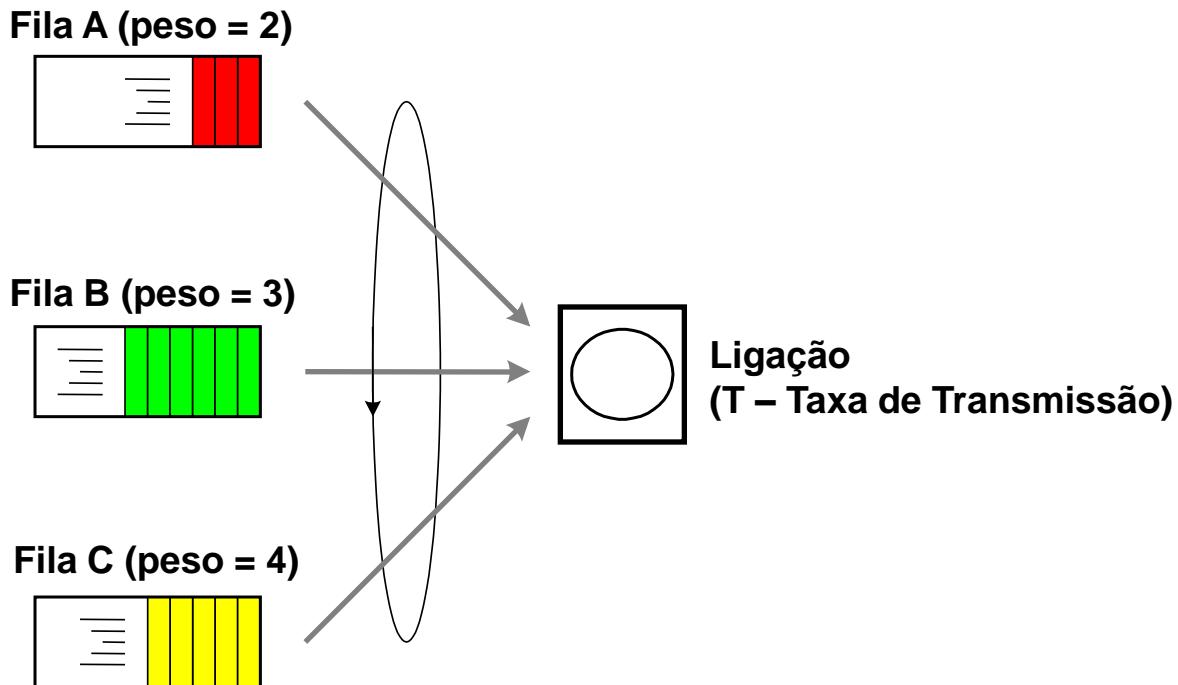
# **Weighted Fair Queuing (WFQ)**

Este algoritmo garante que cada fila de espera consegue uma percentagem da taxa de transmissão do recurso de comunicações pelo menos igual ao seu peso a dividir pela soma dos pesos de todas as filas

$$R_A = \frac{2}{2+3+4} T$$

$$R_B = \frac{3}{2+3+4} T$$

$$R_C = \frac{4}{2+3+4} T$$



## Exemplo 3

Considere um sistema de transmissão de pacotes com capacidade de 120 Kbps. Este sistema suporta dois fluxos de tráfego: fluxo A de 1 pacote/segundo de tamanho exponencialmente distribuído com média de 250 bytes; fluxo B de 5 pacotes/segundo de tamanho exponencialmente distribuído com média de 125 bytes. O fluxo B tem um requisito de qualidade de serviço que exige que o atraso médio por pacote seja 10 milissegundos. Pretende-se implementar um esquema de multiplexagem determinística que permita cumprir com a qualidade de serviço do fluxo B. Calcule:

- (a) a percentagem da capacidade de transmissão que deve ser dedicada ao fluxo B.
- (b) o atraso médio por pacote do fluxo A no sistema resultante.

## Exemplo 4

Considere um sistema de transmissão de pacotes com capacidade de 120 Kbps. Este sistema suporta dois fluxos de tráfego: fluxo A de 1 pacote/segundo de tamanho exponencialmente distribuído com média de 250 bytes; fluxo B de 5 pacotes/segundo de tamanho exponencialmente distribuído com média de 125 bytes.

O sistema atribui maior prioridade ao fluxo B com uma disciplina não preemptiva. Calcule o atraso médio por pacote de cada fluxo.



## **Encaminhamento em Redes com Comutação de Pacotes**

Desempenho e Dimensionamento de Redes

Prof. Amaro de Sousa ([asou@ua.pt](mailto:asou@ua.pt))

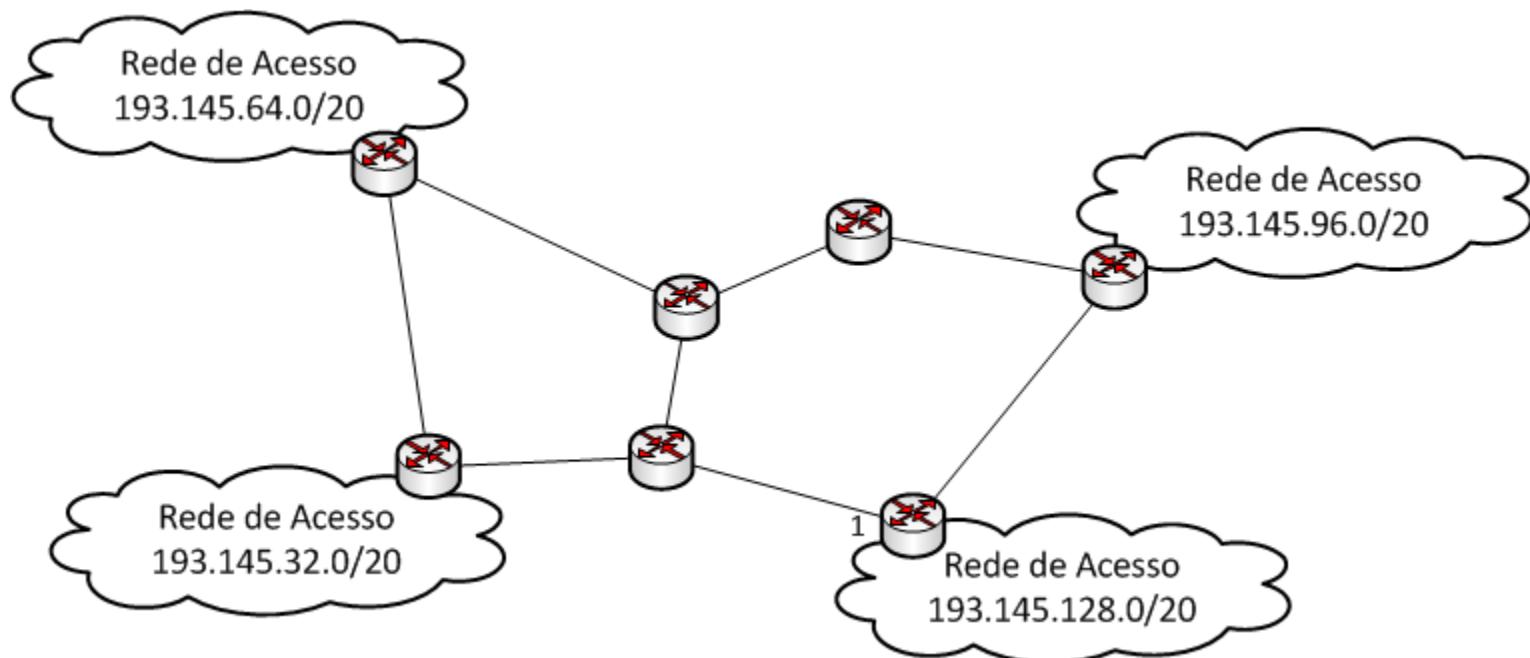
DETI-UA, 2017/2018

# Encaminhamento em redes com comutação de pacotes

Existem 2 tipos de redes com comutação de pacotes:

- redes de circuitos virtuais
- redes de datagramas

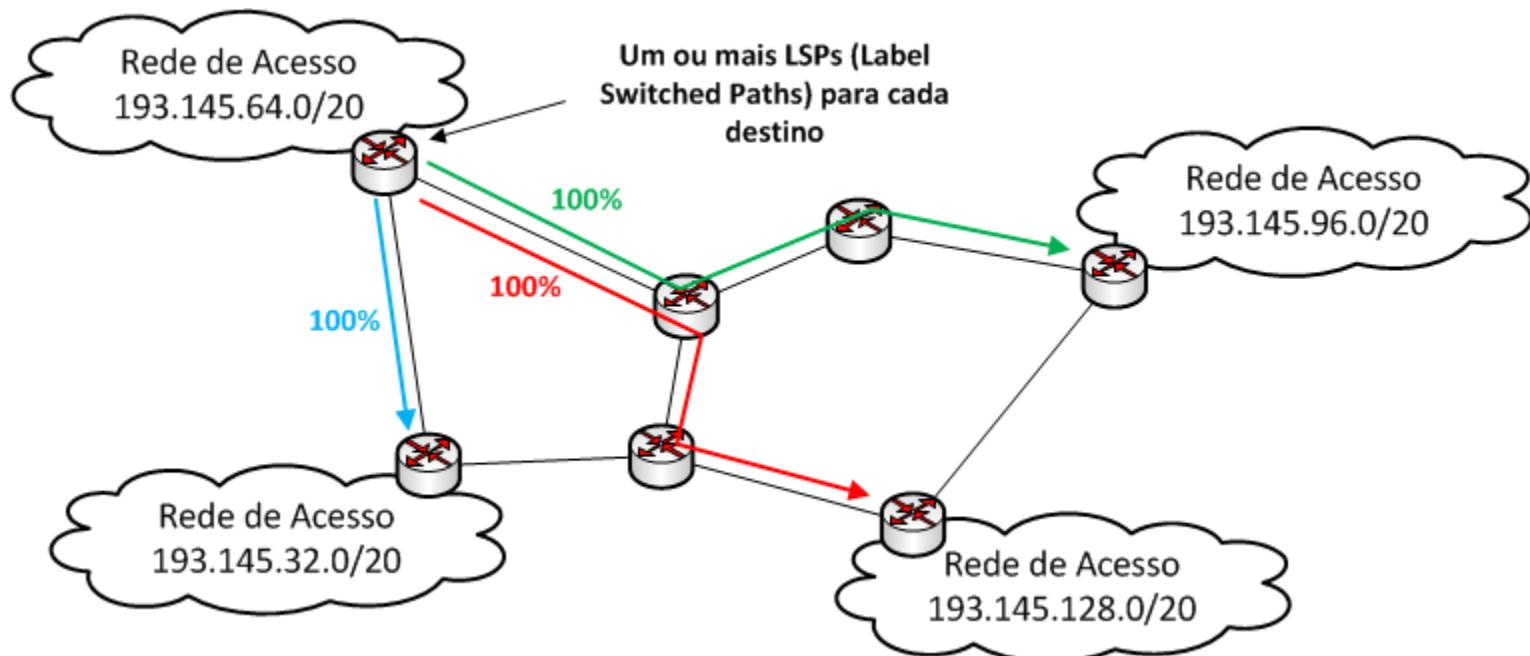
Considere-se o seguinte exemplo de uma rede de um ISP (*Internet Service Provider*)



# Encaminhamento em redes com comutação de pacotes – redes de circuitos virtuais

- Os percursos são definidos na fase de estabelecimento dos circuitos virtuais.
- Após esta fase, os pacotes de cada circuito virtual são encaminhados pelo percurso definido.

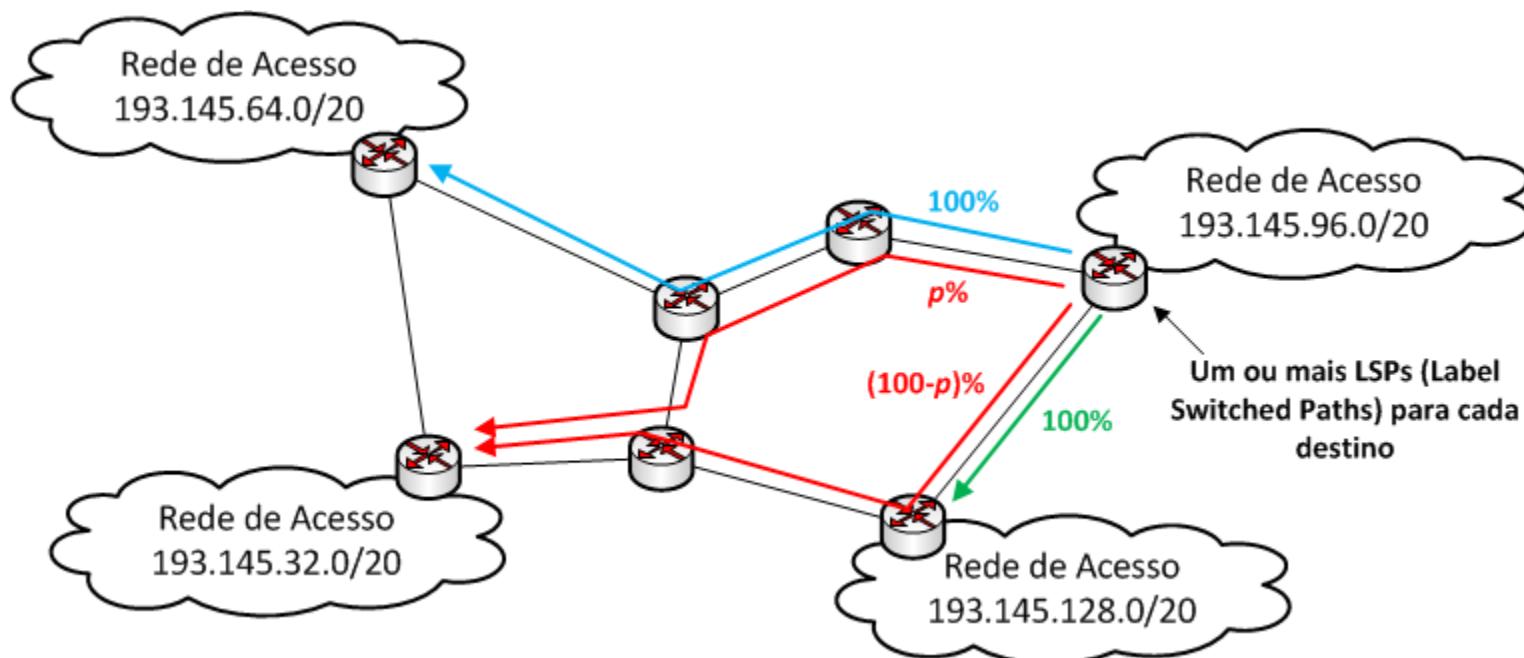
Exemplo: redes IP/MPLS em que os circuitos virtuais se designam por LSPs (*Label Switched Paths*).



# Encaminhamento em redes com comutação de pacotes – redes de circuitos virtuais

- Os percursos são definidos na fase de estabelecimento dos circuitos virtuais.
- Após esta fase, os pacotes de cada circuito virtual são encaminhados pelo percurso definido.

Exemplo: redes IP/MPLS em que os circuitos virtuais se designam por LSPs (*Label Switched Paths*).



# **Encaminhamento em redes com comutação de pacotes – redes de datagramas**

- As decisões de encaminhamento são efetuadas pacote a pacote ou endereço destino a endereço destino.
- Dois pacotes do mesmo par origem-destino podem seguir percursos distintos na rede.

Exemplo: redes IP com o protocolo de encaminhamento RIP ou OSPF.

Nas redes IP, o encaminhamento é baseado em percursos de custo mínimo de cada nó (router) para cada destino

- No OSPF, é atribuído a cada ligação um número positivo designado por custo da ligação.
- No RIP, o custo é 1 para cada ligação.
- Cada percurso de um router para um destino tem um custo igual à soma dos custos das ligações que o compõem.
- Em cada router, cada pacote IP é encaminhado por um dos percursos de custo mínimo para o destino do pacote.

# **Encaminhamento em redes com comutação de pacotes – redes de datagramas**

Cada pacote IP é encaminhado por um dos percursos de custo mínimo para o destino do pacote:

Método estático: o custo das ligações é fixo.

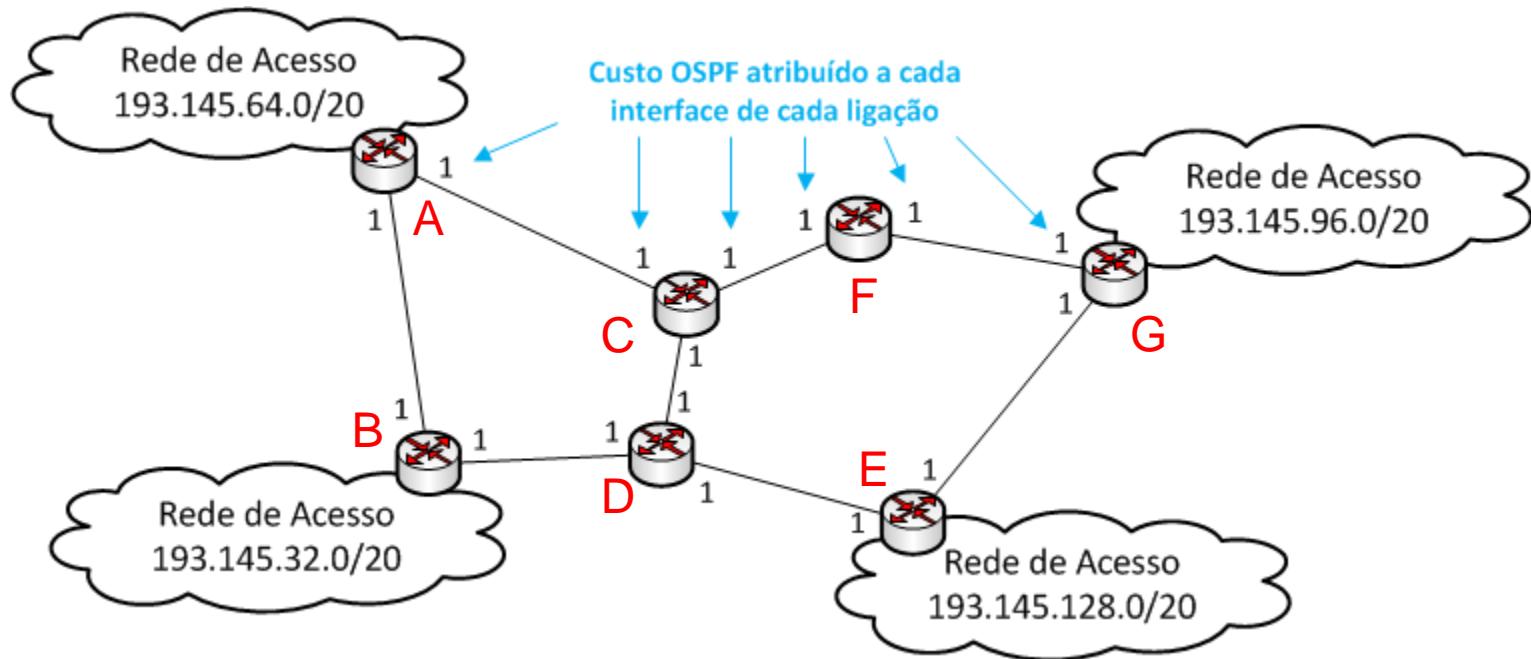
Método dinâmico: o custo das ligações varia ao longo do tempo em função do seu nível de utilização (exemplo: protocolos IGRP e EIGRP)

- o percurso de custo mínimo adapta-se a situações de sobrecarga obrigando os pacotes a evitarem as ligações mais utilizadas
- introduz um efeito de realimentação que pode levar a oscilações indesejáveis.

Quando existem múltiplos percursos de custo mínimo de um nó para um destino, é usada a técnica ECMP (*Equal Cost Multi-Path*):

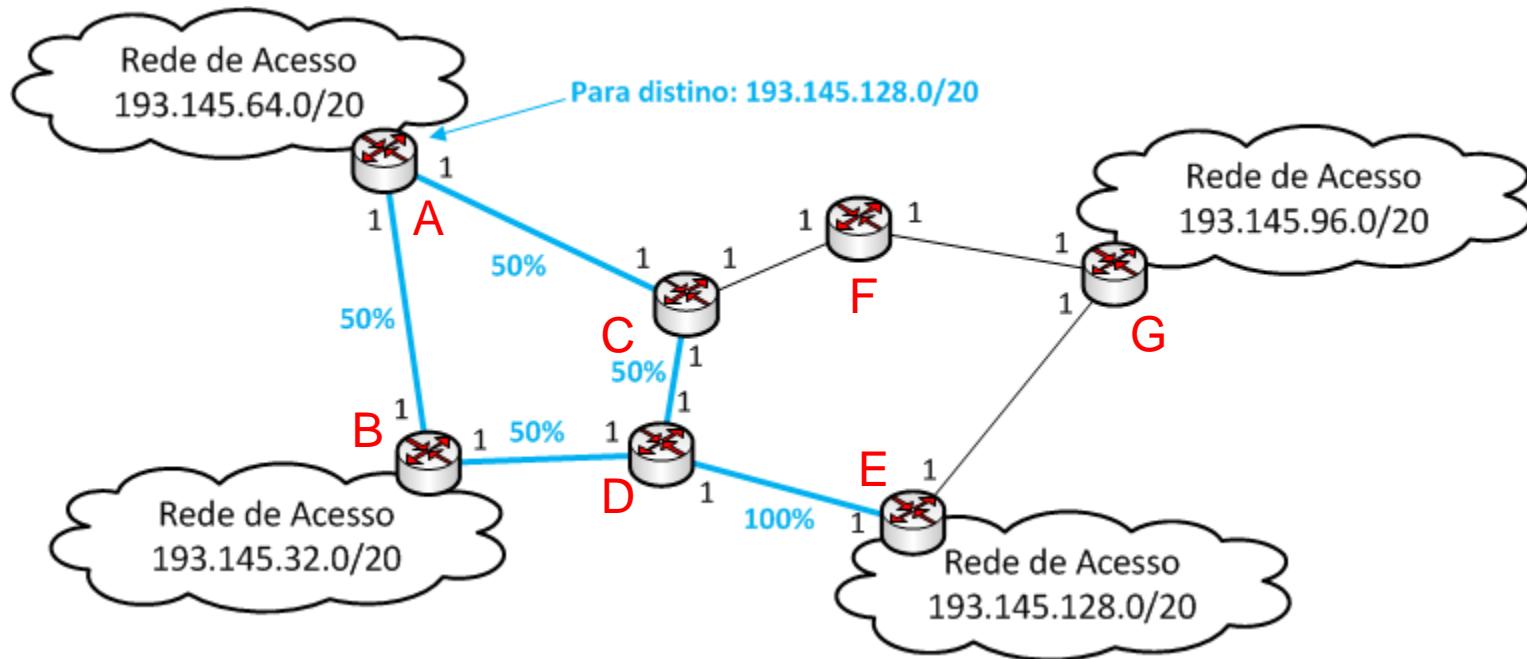
- em cada nó, o tráfego é bifurcado em igual percentagem por todas as ligações de saída que proporcionam percursos de custo mínimo

# Encaminhamento em redes IP com encaminhamento OSPF (I)



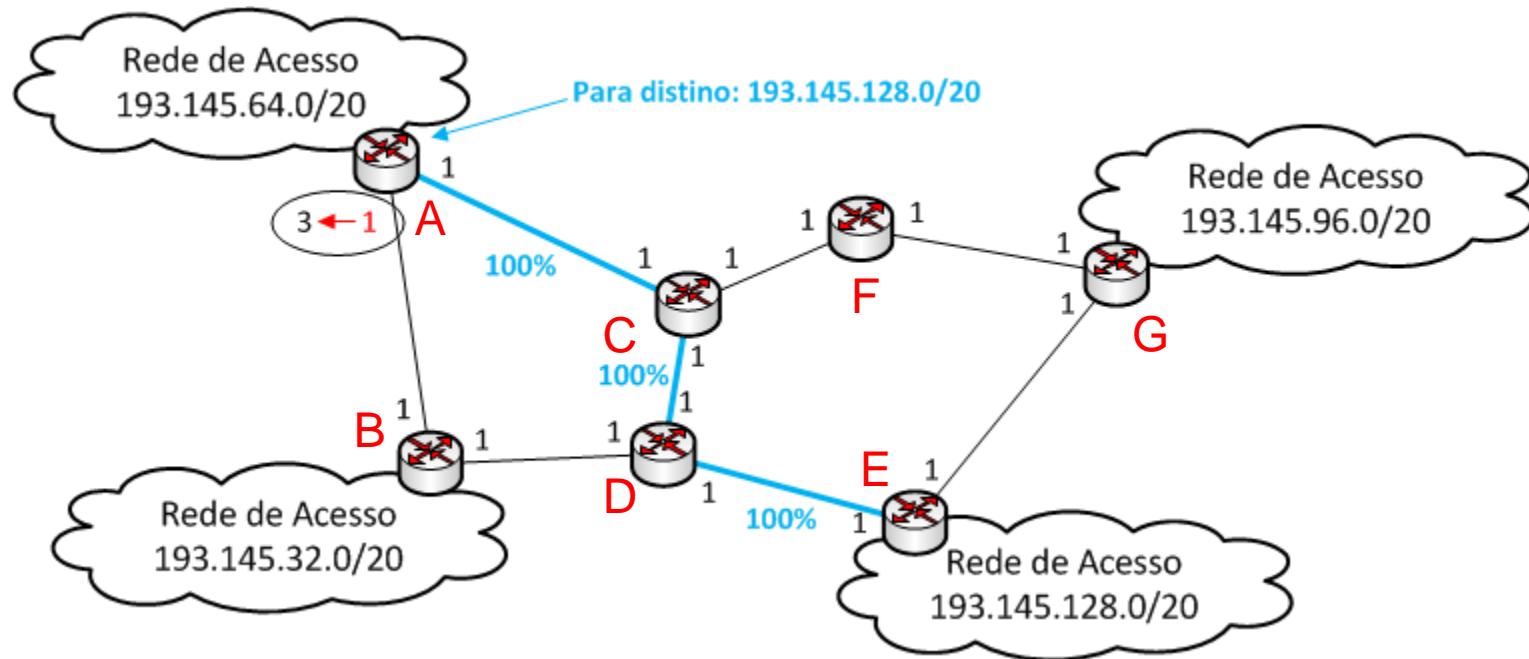
Neste exemplo, todos os custos OSPF estão configurados a 1.

# Encaminhamento em redes IP com encaminhamento OSPF (II)



Pelo ECMP, o router A encaminha os pacotes IP com destino para um endereço IP da rede 193.145.128.0/20 em igual percentagem pelos percursos que passam por B e por C.

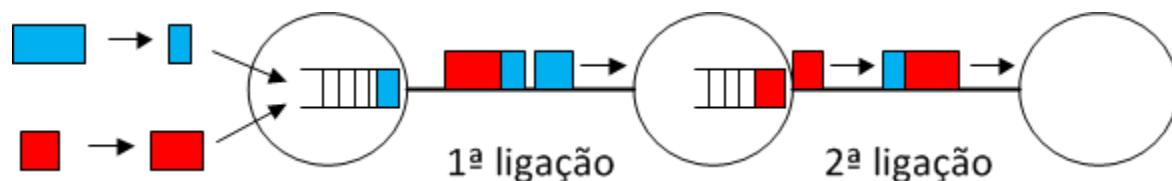
# Encaminhamento em redes IP com encaminhamento OSPF (III)



Mudando o custo da ligação de A para B de 1 para 3, o router A encaminha os pacotes IP com destino para um endereço IP da rede 193.145.128.0/20 pelo único percurso de custo mínimo.

# Redes de ligações ponto-a-ponto

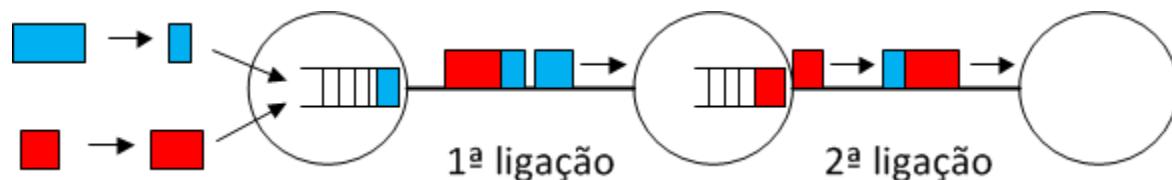
Numa rede de ligações ponto-a-ponto os intervalos entre chegadas de pacotes tornam-se fortemente correlacionados com o comprimento dos pacotes, após passagem pela primeira fila de espera. Este facto dificulta a análise.



Exemplo: Considerem-se duas ligações ponto-a-ponto em cascata. Os pacotes chegam à 1<sup>a</sup> fila de espera a uma taxa de Poisson e o comprimento dos pacotes é exponencialmente distribuído.

# Redes de ligações ponto-a-ponto

Numa rede de ligações ponto-a-ponto os intervalos entre chegadas de pacotes tornam-se fortemente correlacionados com o comprimento dos pacotes, após passagem pela primeira fila de espera. Este facto dificulta a análise.



- a 1<sup>a</sup> fila de espera é do tipo M/M/1
- no entanto, a 2<sup>a</sup> fila de espera não é do tipo M/M/1:
  - o intervalo entre a chegada de dois pacotes consecutivos à 2<sup>a</sup> fila de espera é sempre superior ou igual ao tempo de transmissão do segundo pacote na 1<sup>a</sup> fila de espera;
  - assim, tipicamente pacotes maiores esperam menos tempo na 2<sup>a</sup> fila de espera que pacotes mais pequenos.

# Aproximação de Kleinrock

A aproximação de Kleinrock consiste em admitir que cada ligação pode ser modelada por um sistema M/M/1 (as filas de espera são de tamanho infinito).

Os fluxos de pacotes são unidirecionais e as ligações das redes de comutação de pacotes são bidirecionais. Assim, uma ligação de rede entre os nós de comutação  $i$  e  $j$  é representada pelos pares ordenados  $(i,j)$  e  $(j,i)$  que representam cada um dos seus sentidos.

Considere-se uma rede de ligações ponto-a-ponto, onde existem diversos fluxos de pacotes  $s = 1 \dots S$ .

A cada fluxo  $s$  está associado um percurso único na rede (rede de circuitos virtuais com um circuito virtual por fluxo), formado por uma sequência de ligações  $(i,j)$  definida pelo conjunto  $R_s$ .

Seja  $\lambda_s$ , em pacotes/segundo, a taxa de chegada do fluxo  $s$ .

Então a taxa total de chegada à ligação  $(i,j)$  é:  $\lambda_{ij} = \sum_{s:(i,j) \in R_s} \lambda_s$

# Aproximação de Kleinrock

Considere-se agora o caso em que pode haver múltiplos percursos associados a cada fluxo de pacotes.

Seja  $f_{ij}(s)$  a fração de pacotes do fluxo  $s$  que atravessa a ligação  $(i,j)$  e considere-se que nenhum pacote atravessa duas vezes a mesma ligação (i.e., não há ciclos de encaminhamento).

Neste caso, o conjunto  $R_s$  inclui todas as ligações  $(i,j)$  tais que  $f_{ij}(s) > 0$ .

Então a taxa total de chegada à ligação  $(i,j)$  é:  $\lambda_{ij} = \sum_{s:(i,j) \in R_s} f_{ij}(s) \lambda_s$

Considerando  $\mu_{ij}$  a capacidade da ligação  $(i,j)$  em número médio de pacotes/segundo, o número médio de pacotes em todas as ligações é:

$$L = \sum_{(i,j)} \frac{\lambda_{ij}}{\mu_{ij} - \lambda_{ij}}$$

# Aproximação de Kleinrock

Usando o teorema de Little, o atraso médio por pacote é

$$W = \frac{1}{\gamma} \sum_{(i,j)} \frac{\lambda_{ij}}{\mu_{ij} - \lambda_{ij}} \quad \gamma = \sum_s \lambda_s$$

Nos casos em que os atrasos de processamento dos pacotes nos nós de comutação e os atrasos de propagação nas ligações não são desprezáveis, o atraso médio por pacote passa a ser

$$W = \frac{1}{\gamma} \sum_{(i,j)} \left( \frac{\lambda_{ij}}{\mu_{ij} - \lambda_{ij}} + \lambda_{ij} d_{ij} \right)$$

em que  $d_{ij}$  é o atraso médio de processamento e propagação associado à ligação  $(i, j)$ .

# Aproximação de Kleinrock

No caso em que a cada fluxo está associado um percurso único na rede, o atraso médio por pacote do fluxo de tráfego  $s$  é:

$$W_s = \sum_{(i,j) \in R_s} \left( \frac{1}{\mu_{ij} - \lambda_{ij}} + d_{ij} \right)$$

No caso em que há diferentes percursos associados a cada fluxo de pacotes, o atraso médio por pacote de cada fluxo é a média pesada dos atrasos de cada percurso (dados pela fórmula anterior) em que os pesos são as percentagens do tráfego total que são encaminhados por cada percurso.

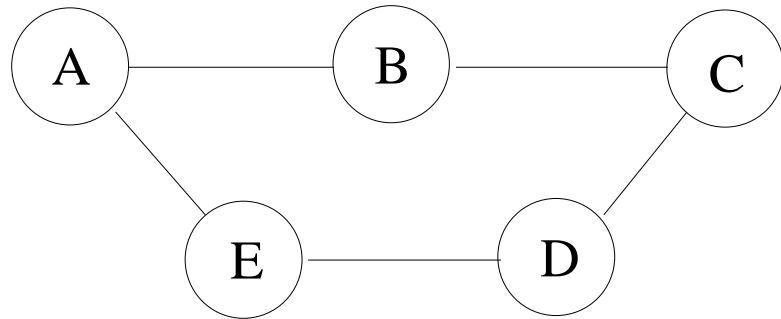
- Nas redes com um percurso por fluxo, a maior fonte de erro associada à aproximação de Kleinrock deve-se à correlação entre os comprimentos dos pacotes e os intervalos entre chegadas.
- Nas redes com múltiplos percursos por fluxo, pode existir um fator adicional de erro, dependendo da forma como os fluxos são bifurcados nos nós.

## Exemplo 1

Considere a rede IP da figura em que todas as ligações são bidirecionais de 10 Mbps. A esta rede são submetidos 4 fluxos de pacotes: de A para C com uma taxa de Poisson de 1000 pps; de A para D com uma taxa de Poisson de 250 pps; de B para D com uma taxa de Poisson de 1000 pps e de B para E com uma taxa de Poisson de 750 pps. O tamanho dos pacotes de todos os fluxos é exponencialmente distribuído com média de 500 bytes. O tempo de propagação das ligações é desprezável em todas as ligações exceto na ligação B-C que é de 10 milissegundos em cada sentido.

O protocolo de encaminhamento nos routers é o RIP. Utilizando a aproximação de Kleinrock, calcule:

- (a) O atraso médio por pacote de cada fluxo.
- (b) O atraso médio por pacote de todos os fluxos.
- (c) A utilização (em percentagem) de cada ligação em cada sentido.

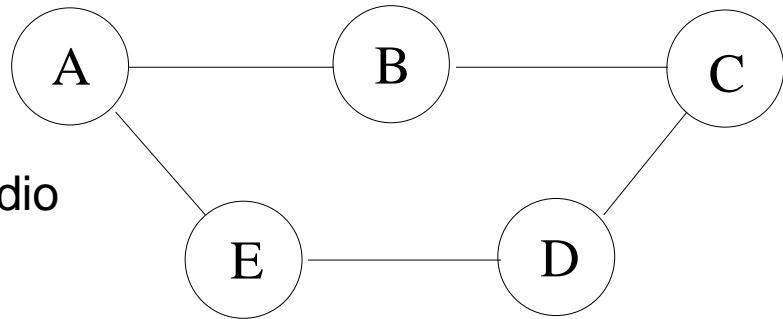


## Exemplo 2

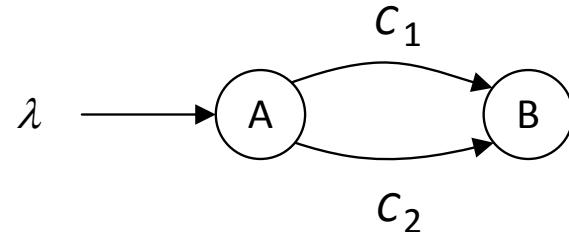
Considere a rede IP da figura em que todas as ligações são bidirecionais de 10 Mbps. A esta rede são submetidos 4 fluxos de pacotes: de A para C com uma taxa de Poisson de 1000 pps; de A para D com uma taxa de Poisson de 250 pps; de B para D com uma taxa de Poisson de 1000 pps e de B para E com uma taxa de Poisson de 750 pps. O tamanho dos pacotes de todos os fluxos é exponencialmente distribuído com média de 500 bytes. O tempo de propagação das ligações é desprezável em todas as ligações exceto na ligação B-C que é de 10 milissegundos em cada sentido.

O protocolo de encaminhamento nos routers é o OSPF.

- (a) Determine os custos OSPF que permitem minimizar a utilização da ligação mais carregada.
- (b) Utilizando a aproximação de Kleinrock, determine o atraso médio por pacote de todos os fluxos na solução da alínea anterior.



# Encaminhamento ótimo com bifurcação de fluxos (exemplo)



Na figura, o fluxo de pacotes  $\lambda$  (pacotes/seg) é bifurcado por duas ligações com capacidades  $C_1$  e  $C_2$  (ambas em pacotes/seg). Designemos os fluxos em cada ligação por  $x_1$  e  $x_2$ , respectivamente ( $\lambda = x_1 + x_2$ ). O número médio de pacotes nesta rede é, pela aproximação de Kleinrock, dado por:

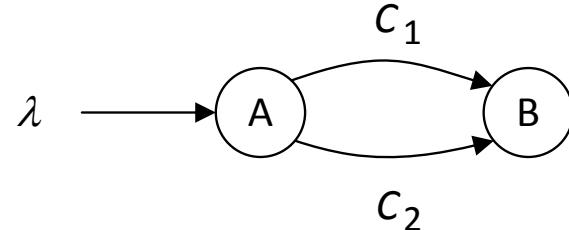
$$L = \frac{x_1}{C_1 - x_1} + \frac{x_2}{C_2 - x_2}$$

Os valores de  $x_1$  e  $x_2$  que minimizam o atraso médio por pacote são os que minimizam o número médio de pacotes na rede (teorema de Little:  $L = \lambda W$ ). Assim, atendendo à restrição  $\lambda = x_1 + x_2$  temos:

$$L = \frac{x_1}{C_1 - x_1} + \frac{\lambda - x_1}{C_2 - (\lambda - x_1)} \quad \frac{\partial L}{\partial x_1} = \frac{C_1}{(C_1 - x_1)^2} - \frac{C_2}{(C_2 - (\lambda - x_1))^2}$$

Relembrar regra das derivadas:  $\left(\frac{u}{v}\right)' = \frac{u'v - uv'}{v^2}$

# Encaminhamento ótimo com bifurcação de fluxos (exemplo)



Fazendo

$$\frac{\partial L}{\partial x_1} = \frac{C_1}{(C_1 - x_1)^2} - \frac{C_2}{(C_2 - (\lambda - x_1))^2} = 0$$

temos

$$x_1^* = \frac{\sqrt{C_1} \left[ \lambda - (C_2 - \sqrt{C_1 C_2}) \right]}{\sqrt{C_1} + \sqrt{C_2}}$$

$$x_2^* = \frac{\sqrt{C_2} \left[ \lambda - (C_1 - \sqrt{C_1 C_2}) \right]}{\sqrt{C_1} + \sqrt{C_2}}$$

Assumindo que  $C_1 \geq C_2$  temos dois casos possíveis:

$\lambda > C_1 - \sqrt{C_1 C_2}$  →  $0 < x_1^* < \lambda$  ,  $0 < x_2^* < \lambda$  solução ótima: fluxo bifurcado por  $C_1$  e  $C_2$

$\lambda < C_1 - \sqrt{C_1 C_2}$  →  $x_1^* = \lambda$  ,  $x_2^* = 0$  solução ótima: fluxo encaminhado apenas por  $C_1$

## Encaminhamento ótimo - caso geral

No encaminhamento ótimo, os fluxos em cada percurso são definidos por forma a otimizar uma função de custo que representa o desempenho da rede:

$$\sum_{(i,j)} D_{ij}(F_{ij})$$

onde  $F_{ij}$  representa o fluxo na ligação  $(i, j)$  e a função  $D_{ij}$  é monótona crescente.

Uma função  $D_{ij}$  usada com frequência é

$$D_{ij}(F_{ij}) = \frac{F_{ij}}{C_{ij} - F_{ij}} + d_{ij} F_{ij}$$

onde  $C_{ij}$  é a capacidade da ligação  $(i,j)$  e  $d_{ij}$  o atraso de propagação e processamento na ligação  $(i,j)$ .

Neste caso, a função de custo corresponde ao número médio de pacotes na rede, obtido com base na aproximação de Kleinrock.

## Encaminhamento ótimo - caso geral

$W$  - conjunto de todos os pares OD (origem - destino)  $w$ ;

$\lambda_w$  - fluxo de entrada do par OD  $w$ ;

$P_w$  - conjunto de todos os percursos dirigidos que ligam o nó origem ao nó destino do par OD  $w$ ;

$x_p$  - fluxo no percurso  $p$ .

O encaminhamento ótimo é dado pelo seguinte problema de otimização:

$$\text{Minimizar: } D(x) = \sum_{(i,j)} D_{ij} \left( \sum_{\substack{\text{todos os percursos } p \\ \text{contendo } (i,j)}} x_p \right)$$

Sujeito a:

$$\sum_{p \in P_w} x_p = \lambda_w , \forall w \in W$$

$$x_p \geq 0 , \forall p \in P_w, \forall w \in W$$

# Solução para o encaminhamento ótimo

Define-se o comprimento da primeira derivada do percurso  $p \in P_w$  dado por:

$$\frac{\partial D(x)}{\partial x_p} = \sum_{\substack{\text{todas as ligações } (i,j) \\ \text{no percurso } p}} D'_{ij}$$

Prova-se que um vetor de fluxos  $x^* = \{x_p^*, \forall p \in P_w\}$  para o par OD  $w$  é ótimo se e só se:

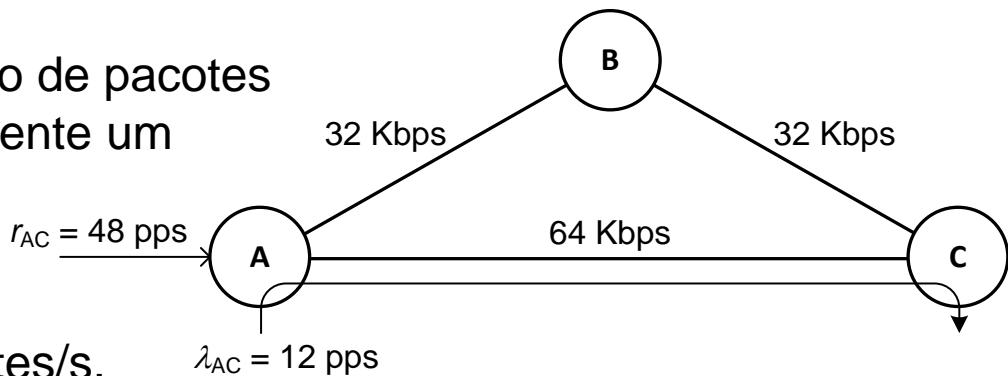
$$x_p^* > 0 \Rightarrow \frac{\partial D(x^*)}{\partial x_{p'}} \geq \frac{\partial D(x^*)}{\partial x_p}, \forall p' \in P_w$$

O fluxo ótimo é positivo apenas nos percursos  $p \in P_w$  com um comprimento de primeira derivada mínimo.

Assim, os percursos usados no encaminhamento ótimo têm comprimento de primeira derivada igual.

## Exemplo 3

Considere a rede com comutação de pacotes da figura. A rede suporta inicialmente um único fluxo de 12 pacotes/s no percurso direto AC. Admita que é oferecido um novo fluxo do nó A para o nó C, de 48 pacotes/s.

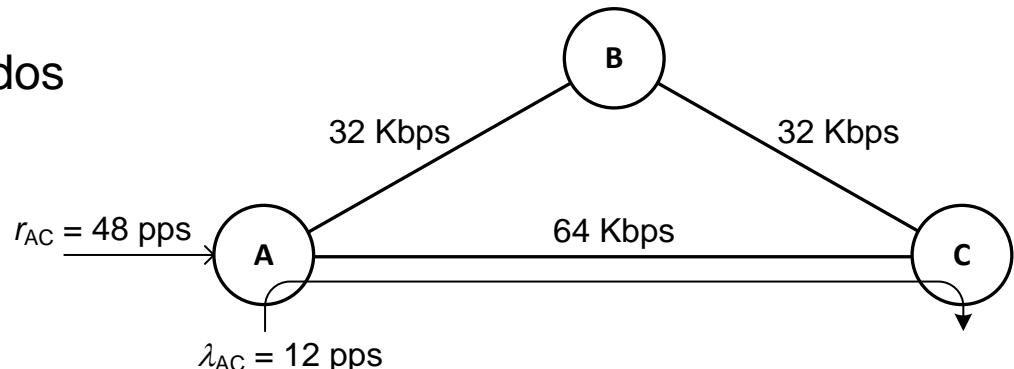


Assuma que ambos os fluxos são caracterizados por intervalos entre chegadas e comprimentos de pacotes independentes e exponencialmente distribuídos, e que o comprimento médio dos pacotes é 125 bytes.

- Calcule o atraso médio total dos pacotes (isto é, o atraso médio calculado sobre todos fluxos), quando o novo fluxo é encaminhado em igual percentagem pelos dois percursos possíveis.
- Admitindo que o novo fluxo (e apenas o novo) pode ser bifurcado pelos dois percursos possíveis, calcule os fluxos ótimos que minimizam o atraso médio total dos pacotes e determine o atraso médio resultante.

## Resposta ao Exemplo 3(a)

(a) Calcule o atraso médio total dos pacotes (isto é, o atraso médio calculado sobre todos fluxos), quando o novo fluxo é encaminhado em igual percentagem pelos dois percursos possíveis.



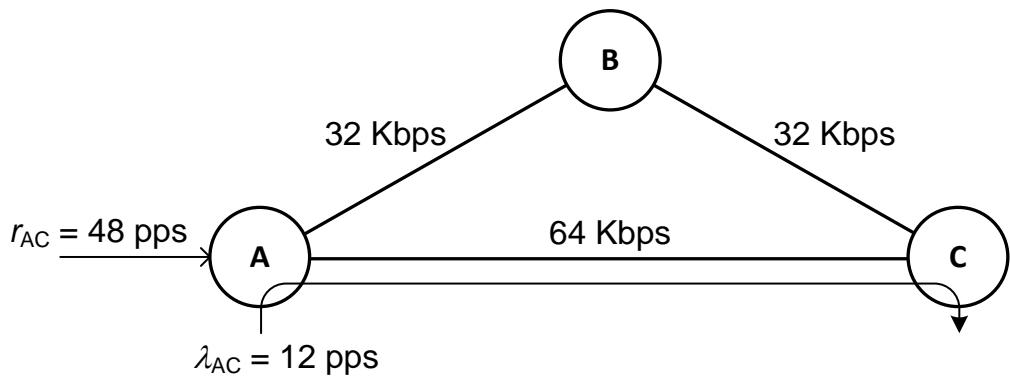
$$\mu_{AB} = \mu_{BC} = \frac{32000}{125 \times 8} = 32 \text{ pps}$$

$$\mu_{AC} = \frac{64000}{125 \times 8} = 64 \text{ pps}$$

$$W = \frac{L}{\lambda} = \frac{L_{AB} + L_{BC} + L_{AC}}{\lambda} = \frac{\frac{24}{\mu_{AB} - 24} + \frac{24}{\mu_{BC} - 24} + \frac{24 + 12}{\mu_{AC} - (24 + 12)}}{48 + 12} = 0.121 \text{ seg.}$$

## Resposta ao Exemplo 3(b)

(b) Admitindo que o novo fluxo (e apenas o novo) pode ser bifurcado pelos dois percursos possíveis, calcule os fluxos ótimos que minimizam o atraso médio total dos pacotes e determine o atraso médio resultante.



$$L = \frac{x_1}{32 - x_1} + \frac{x_1}{32 - x_1} + \frac{x_2 + 12}{64 - (x_2 + 12)}$$

$$\frac{\partial L}{\partial x_1} = \frac{32}{(32 - x_1)^2} + \frac{32}{(32 - x_1)^2} + 0 = \frac{64}{(32 - x_1)^2}$$

$$\frac{\partial L}{\partial x_2} = 0 + 0 + \frac{64}{(52 - x_2)^2} = \frac{64}{(52 - x_2)^2}$$

$$\begin{cases} \frac{\partial L}{\partial x_1} = \frac{\partial L}{\partial x_2} \\ x_1 + x_2 = 48 \end{cases} = \begin{cases} \frac{64}{(32 - x_1)^2} = \frac{64}{(52 - x_2)^2} \\ x_1 + x_2 = 48 \end{cases} = \begin{cases} \frac{8}{32 - x_1} = \frac{8}{52 - x_2} \\ x_2 = 48 - x_1 \end{cases} = \begin{cases} x_1 = 14 \text{ pps} \\ x_2 = 34 \text{ pps} \end{cases}$$

$$W = \frac{L}{\lambda} = \frac{\frac{14}{32 - 14} + \frac{14}{32 - 24} + \frac{34 + 12}{64 - (34 + 12)}}{48 + 12} = 0.069 \text{ seg.}$$



# **Optimization based on Integer Linear Programming**

Desempenho e Dimensionamento de Redes

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

DETI-UA, 2017/2018

# Mathematical programming model

- In an *optimization problem*, the aim is to maximize (or minimize) a given quantity designated by the *objective* that depends on a finite number of variables.
- The variables might be independent or might be related between them through one or more *constraints*.
- A *mathematical programming problem* is an optimization problem such that the objective and the constraints are defined by mathematical functions and functional relations.
- A *mathematical programming model* describes a mathematical programming problem.

# Mathematical programming model

For a given set of  $n$  variables  $X = \{x_1, x_2, \dots, x_n\}$ , the standard way of defining a Mathematical Programming Model is:

Minimize (or Maximize)

$$f(X)$$

Subject to:

$$g_i(X) \leq k_i \quad , i = 1, 2, \dots, m$$

(=)

( $\geq$ )

where:

- $m$  is the number of constraints
- $f(X)$  and all  $g_i(X)$  are functions of the variables
- $k_i$  are real constants

# (Mixed Integer) Linear Programming model

- A *Linear Programming (LP) model* is a mathematical programming model where all variables  $X = \{x_1, x_2, \dots, x_n\}$  are non-negative reals and  $f(X)$  and  $g_i(X)$  are linear functions:
  - functions in the form  $a_1x_1 + a_2x_2 + \dots + a_nx_n$  where all  $a_i$  are real constants.
- An *Integer Linear Programming (ILP) model* is an LP model where all variables  $X = \{x_1, x_2, \dots, x_n\}$  are non-negative integers.
- A *Mixed Integer Linear Programming (MILP) model* is an LP model where some variables  $X = \{x_1, x_2, \dots, x_n\}$  are non-negative integers and others are non-negative reals.

## Illustrative example

Consider a transportation company that has been requested to deliver the following items to a particular destination:

Item $i$ :	1	2	3	4	5	6
Revenue ( $r_i$ ):	2.3	4.5	1.5	5.4	2.9	3.2
Size ( $s_i$ ):	30	70	20	80	35	40

The company has 2 vans for item delivery:

- the first van has a capacity of 100
- the second van has a capacity of 60.

Since it is not possible to deliver all items with the 2 vans, the aim is to choose the items to be carried on each van to maximize the revenue.

Solving steps:

- 1<sup>st</sup> - define the ILP model of the optimization problem
- 2<sup>nd</sup> – solve the ILP model (using an available solver)

# Illustrative example

Item $i$ :	1	2	3	4	5	6
Revenue ( $r_i$ ):	2.3	4.5	1.5	5.4	2.9	3.2
Size ( $s_i$ ):	30	70	20	80	35	40

## VARIABLES DEFINING THE PROBLEM:

- $x_1$  – Binary variable that, if is 1 in the solution, indicates that item 1 is delivered
- $x_2$  – Binary variable that, if is 1 in the solution, indicates that item 2 is delivered
- ...
- $x_6$  – Binary variable that, if is 1 in the solution, indicates that item 6 is delivered

- $y_{1,1}$  – Binary variable that, if is 1 in the solution, indicates that item 1 is carried by first van
- $y_{1,2}$  – Binary variable that, if is 1 in the solution, indicates that item 1 is carried by second van
- ...
- $y_{6,1}$  – Binary variable that, if is 1 in the solution, indicates that item 6 is carried by first van
- $y_{6,2}$  – Binary variable that, if is 1 in the solution, indicates that item 6 is carried by second van

# Illustrative example

Item $i$ :	1	2	3	4	5	6
Revenue ( $r_i$ ):	2.3	4.5	1.5	5.4	2.9	3.2
Size ( $s_i$ ):	30	70	20	80	35	40

INTEGER LINEAR PROGRAMMING (ILP) MODEL (in LP format):

The objective function is the total revenue  
of the delivered items

Maximize

$$+ 2.3 x_1 + 4.5 x_2 + 1.5 x_3 + 5.4 x_4 + 2.9 x_5 + 3.2 x_6$$

Subject To

$$+ 30 y_{1,1} + 70 y_{2,1} + 20 y_{3,1} + 80 y_{4,1} + 35 y_{5,1} + 40 y_{6,1} \leq 100$$

$$+ 30 y_{1,2} + 70 y_{2,2} + 20 y_{3,2} + 80 y_{4,2} + 35 y_{5,2} + 40 y_{6,2} \leq 60$$

$$+ y_{1,1} + y_{1,2} - x_1 = 0$$

$$+ y_{2,1} + y_{2,2} - x_2 = 0$$

$$+ y_{3,1} + y_{3,2} - x_3 = 0$$

$$+ y_{4,1} + y_{4,2} - x_4 = 0$$

$$+ y_{5,1} + y_{5,2} - x_5 = 0$$

$$+ y_{6,1} + y_{6,2} - x_6 = 0$$

The total size of the items carried on each van must be within the van capacity

If an item is carried in one van, then, the item is delivered

Binary

$$x_1 \ x_2 \ x_3 \ x_4 \ x_5 \ x_6$$

$$y_{1,1} \ y_{1,2} \ y_{2,1} \ y_{2,2} \ y_{3,1} \ y_{3,2} \ y_{4,1} \ y_{4,2} \ y_{5,1} \ y_{5,2} \ y_{6,1} \ y_{6,2}$$

End

List of binary variables

# **Illustrative example – using CPLEX (1)**

## **Starting CPLEX:**

```
Welcome to IBM(R) ILOG(R) CPLEX(R) Interactive Optimizer 12.6.1.0
with Simplex, Mixed Integer & Barrier Optimizers
5725-A06 5725-A29 5724-Y48 5724-Y49 5724-Y54 5724-Y55 5655-Y21
Copyright IBM Corp. 1988, 2014. All Rights Reserved.
```

```
Type 'help' for a list of available commands.
Type 'help' followed by a command name for more
information on commands.
```

```
CPLEX>
```

## **Reading file ‘exemplo.lp’ on CPLEX:**

```
CPLEX> read exemplo.lp
Problem 'exemplo.lp' read.
Read time = 0.01 sec. (0.00 ticks)
CPLEX>
```

## Illustrative example – using CPLEX (2)

Solving the problem on CPLEX:

```
CPLEX> optimize
```

	Nodes				Cuts /		
Node	Left	Objective	IInf	Best Integer	Best Bound	ItCnt	Gap
*	0+	0		1.5000	19.8000		---
*	0+	0		11.2000	19.8000		76.79%
	0	0	11.8286	1	11.2000	11.8286	4 5.61%
*	0+	0		11.5000	11.8286		2.86%
	0	0	cutoff	11.5000		4	0.00%

Elapsed time = 0.14 sec. (1.12 ticks, tree = 0.00 MB, solutions = 3)

Root node processing (before b&c):

Real time = 0.14 sec. (1.12 ticks)

Parallel b&c, 4 threads:

Real time = 0.00 sec. (0.00 ticks)

Sync time (average) = 0.00 sec.

Wait time (average) = 0.00 sec.

-----  
Total (root+branch&cut) = 0.14 sec. (1.12 ticks)

Solution pool: 4 solutions saved.

MIP - Integer optimal solution: Objective = 1.1500000000e+001  
Solution time = 0.16 sec. Iterations = 4 Nodes = 0  
Deterministic time = 1.12 ticks (7.17 ticks/sec)

Optimal solution value

## Illustrative example – using CPLEX (3)

Displaying the values of the optimal solution:

Items 1, 2, 3 and 6  
are selected to be  
delivered

Items 1 and 2 are  
carried by first van

Items 3 and 6 are  
carried by second van

```
CPLEX> display solution variables -  
Incumbent solution  
Variable Name           Solution Value  
x1                      1.000000  
x2                      1.000000  
x3                      1.000000  
x6                      1.000000  
y1,1                     1.000000  
y2,1                     1.000000  
y3,2                     1.000000  
y6,2                     1.000000  
All other variables in the range 1-18 are 0.  
CPLEX>
```

Item $i$ :	1	2	3	4	5	6
Revenue ( $r_i$ ):	2.3	4.5	1.5	5.4	2.9	3.2
Size ( $s_i$ ):	30	70	20	80	35	40

## Illustrative example – mathematical notation

Parameters:

$n$  – number of items

$r_i$  – revenue of delivering item  $i$ , with  $i = 1, \dots, n$

$v$  – number of vans

$s_i$  – size of item  $i$ , with  $i = 1, \dots, n$

$c_j$  – capacity of van  $j$ , with  $j = 1, \dots, v$

Variables:

$x_i$  – binary variable that is 1 if item  $i$  is delivered,  $i = 1, \dots, n$

$y_{ij}$  – binary variable that is 1 if item  $i$  is carried on van  $j$ ,  $i = 1, \dots, n$  and  $j = 1, \dots, v$

ILP model:      Maximize  $\sum_{i=1}^n r_i x_i$

Subject to:

$$\sum_{i=1}^n s_i y_{ij} \leq c_j \quad , j = 1 \dots v$$

$$\sum_{j=1}^v y_{ij} = x_i \quad , i = 1 \dots n$$

$$x_i \in \{0,1\} \quad , i = 1 \dots n$$

$$y_{ij} \in \{0,1\} \quad , i = 1 \dots n \quad , j = 1, \dots, v$$

# Illustrative example – generating LP file with MATLAB

$$\text{Maximize } \sum_{i=1}^n r_i x_i$$

$$\sum_{i=1}^n s_i y_{ij} \leq c_j , j = 1 \dots v$$

$$\sum_{j=1}^v y_{ij} = x_i , i = 1 \dots n$$

$$x_i \in \{0,1\}, i = 1 \dots n$$

$$y_{ij} \in \{0,1\}, i = 1 \dots n , j = 1, \dots v$$

```
r= [2.3 4.5 1.5 5.4 2.9 3.2];
s= [30 70 20 80 35 40];
c= [100 60];
n= length(r);
v= length(c);
fid = fopen('exemplo.lp', 'wt');
fprintf(fid, 'Maximize\n');
for i=1:n
    fprintf(fid, ' + %f x%d', r(i), i);
end
fprintf(fid, '\nSubject To\n');
for j=1:v
    for i=1:n
        fprintf(fid, ' + %f y%d,%d', s(i), i, j);
    end
    fprintf(fid, ' <= %f\n', c(j));
end
for i=1:n
    for j=1:v
        fprintf(fid, ' + y%d,%d', i, j);
    end
    fprintf(fid, ' - x%d = 0\n', i);
end
fprintf(fid, 'Binary\n');
for i=1:n
    fprintf(fid, ' x%d\n', i);
    for j=1:v
        fprintf(fid, ' y%d,%d\n', i, j);
    end
end
fprintf(fid, 'End\n');
fclose(fid);
```

# Illustrative example - using Gurobi on Internet (1)

- Prepare an ASCII file with the problem defined in LP format and compress it with Zip:  
for example: exemplo.zip
- Go to <https://neos-server.org/neos/solvers/index.html>
- Select Mixed Integer Linear Programming tools
- Select Gurobi [[LP Input](#)]

## Mixed Integer Linear Programming

- Cbc [[AMPL Input](#)][[GAMS Input](#)][[MPS Input](#)]
- CPLEX [[AMPL Input](#)][[GAMS Input](#)][[LP Input](#)][[MPS Input](#)]
- feaslp [[AMPL Input](#)][[CPLEX Input](#)][[MPS Input](#)]
- FICO-Xpress [[AMPL Input](#)][[GAMS Input](#)][[MOSEL Input](#)][[MPS Input](#)]
- Gurobi [[AMPL Input](#)][[GAMS Input](#)][[LP Input](#)][[MPS Input](#)]
- MINTO [[AMPL Input](#)]
- MOSEK [[AMPL Input](#)][[GAMS Input](#)][[LP Input](#)][[MPS Input](#)]
- proxy [[CPLEX Input](#)][[MPS Input](#)]
- qsopt\_ex [[AMPL Input](#)][[LP Input](#)][[MPS Input](#)]
- scip [[AMPL Input](#)][[CPLEX Input](#)][[GAMS Input](#)][[MPS Input](#)][[OSIL Input](#)][[ZIMPL Input](#)]
- SYMPHONY [[MPS Input](#)]

## Illustrative example - using Gurobi on Internet (2)

1 . Upload exemplo.zip

2. Check the box

3. Insert a valid  
email address

4. Submit your  
ILP problem

The screenshot shows a web-based form for submitting an ILP problem. The form includes fields for LP file, Parameter file, and Comments, along with checkboxes for Dry run and Short Priority, and an E-Mail address field. At the bottom, there is a note about not clicking the 'Submit to NEOS' button more than once, and buttons for 'Submit to NEOS' and 'Clear this Form'.

LP file  
Enter the path to the LP file  
 No file chosen

Parameter file  
Enter the path to the parameter file  
 No file chosen

Return .sol file  
Check the box to include the solution file as part of the results

Comments

Additional Settings

Dry run: generate job XML instead of submitting it to NEOS

Short Priority: submit to higher priority queue with maximum CPU time of 5 minutes

E-Mail address:

Please do not click the 'Submit to NEOS' button more than once.

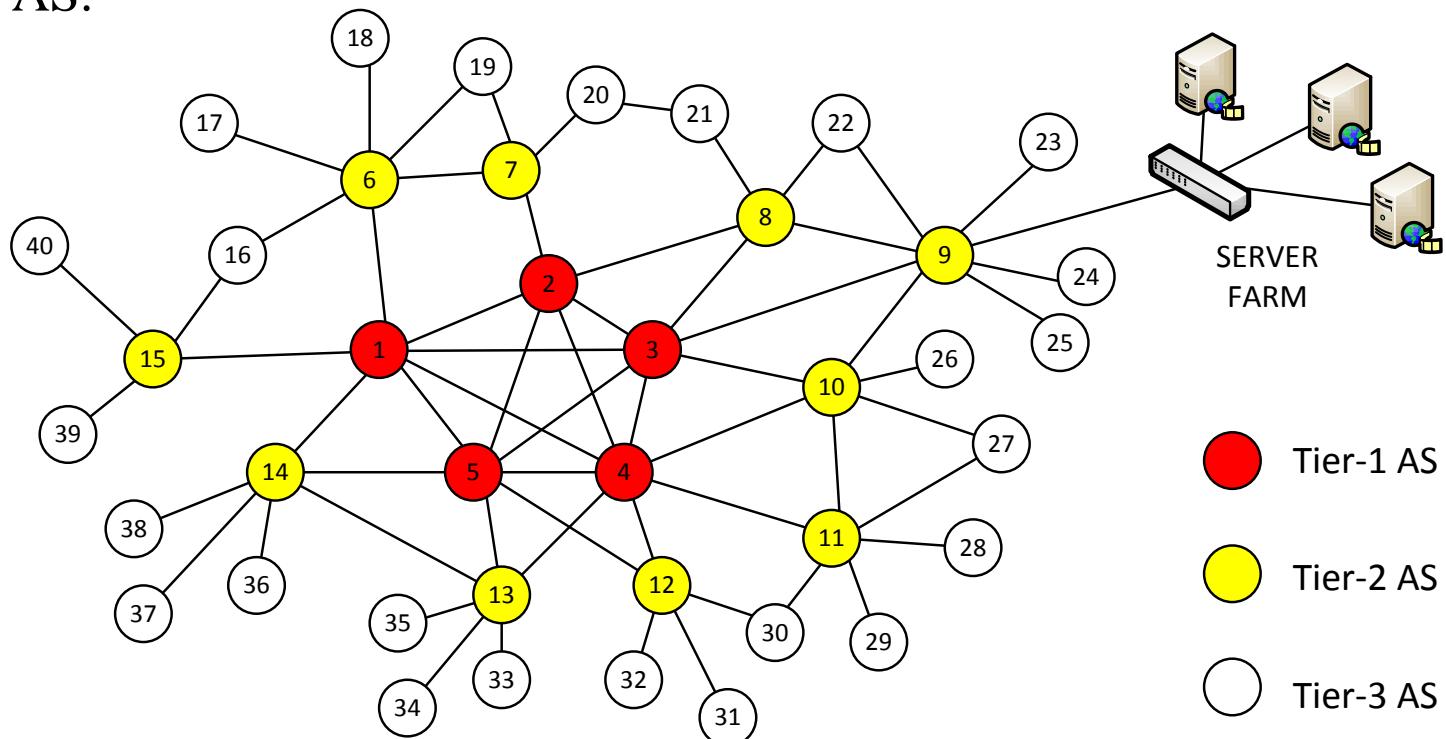
## Illustrative example - using Gurobi on Internet (3)

- After the problem is solved, the solution is displayed (and also sent to the email address):

```
Optimal solution found (tolerance 1.00e-04)
Best objective 1.150000000000e+01, best bound 1.150000000000e+01, gap 0.0000%
Optimal objective: 11.5
***** Begin .sol file *****
# Objective value = 11.5
x1 1
x2 1
x3 1
x4 0
x5 0
x6 1
y1,1 1
y2,1 1
y3,1 0
y4,1 0
y5,1 0
y6,1 0
y1,2 0
y2,2 0
y3,2 1
y4,2 0
y5,2 0
y6,2 1
***** End .sol file *****
```

# Solving the server farm location problem with ILP

- We have a set of Autonomous Systems (ASs) and we aim to select a subset of ASs to connect one server farm on each selected AS.
- Only Tier-2 of Tier-3 ASs provide the Internet access service.
- The solution must guarantee that there is a path between each Tier-2 and Tier-3 AS and at least one server farm with no more than one intermediate AS.



# **Server farm location problem: Notation and Variables**

## **NOTATION:**

$n$  – number of Tier-2 and Tier-3 ASs where server farms can be connected to;

$c_i$  – OPEX cost of connecting a server farm to AS  $i$ , with  $1 \leq i \leq n$ ;

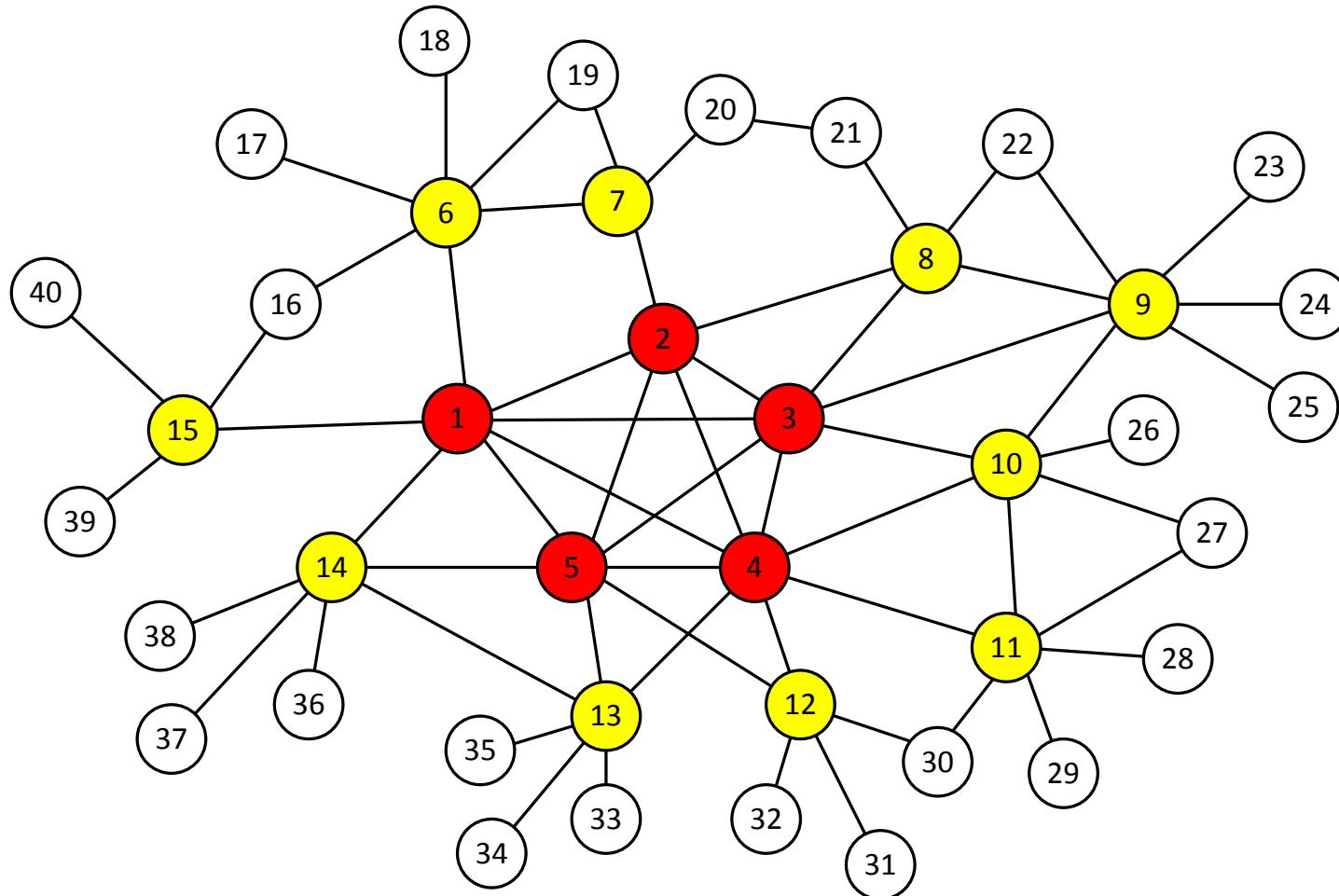
$I(j)$  – set of Tier-2 and Tier-3 ASs such that there is a shortest path between AS  $j$  and each AS  $i \in I(j)$  with at most one intermediate AS.

## **VARIABLES:**

$x_i$  – binary variable, with  $1 \leq i \leq n$ , that when is equal to 1 means that AS  $i$  must be connected to one server farm;

$y_{ji}$  – binary variable, with  $1 \leq j \leq n$  and  $i \in I(j)$ , that when is equal to 1 means that AS  $j$  is associated with AS  $i$ .

## Server farm location problem: examples of sets $I(j)$



Set  $I(j)$  for  $j = 6$  is: {6,7,14,15,16,17,18,19,20}

for  $j = 16$  is: {6,7,15,16,17,18,19,39,40}

# Server farm location problem: ILP Model

$$\text{Minimize } \sum_{i=1}^n c_i x_i \quad (1)$$

Subject to:

$$\sum_{i \in I(j)} y_{ji} = 1 \quad , j = 1 \dots n \quad (2)$$

$$y_{ji} \leq x_i \quad , j = 1 \dots n, i \in I(j) \quad (3)$$

$$x_i \in \{0,1\} \quad , i = 1 \dots n \quad (4)$$

$$y_{ji} \in \{0,1\} \quad , j = 1 \dots n, i \in I(j) \quad (5)$$

- The objective (1) is the minimization of the OPEX costs of the selected server farms.
- Constraints (2) guarantee that each AS  $j$  is associated with one AS  $i \in I(j)$  while constraints (3) guarantee that an associated AS  $i \in I(j)$  must have one server farm connected. So, constraints (2–3) guarantee that each AS  $j$  has always one server farm whose shortest path has at most one intermediate AS.
- Constraints (4–5) define all variables as binary variables.



## **Desempenho de Redes com Comutação de Circuitos**

Desempenho e Dimensionamento de Redes

Prof. Amaro de Sousa ([asou@ua.pt](mailto:asou@ua.pt))

DETI-UA, 2017/2018

# Encaminhamento em redes com comutação de circuitos

1. Encaminhamento *fixo*: considera um único percurso para cada fluxo de chamadas suportado pela rede.
2. Encaminhamento *alternativo*: considera uma sequência ordenada de percursos alternativos para cada fluxo (estabelece no  $n$ -ésimo percurso se nenhum dos percursos até ao  $(n-1)$ -ésimo tiver recursos).
3. Encaminhamento *dinâmico*: considera uma sequência ordenada de percursos alternativos para cada fluxo e essa sequência varia ao longo do tempo.
  - Numa rede em malha completa, o percurso *direto* é o percurso com uma ligação, que interliga o nó origem e o nó destino.
  - É preferível encaminhar uma chamada pelo percurso direto (os percursos alternativos consomem mais recursos).
  - *Trunk reservation*: Reserva de um conjunto de circuitos em cada ligação para chamadas no percurso direto (limita o excesso de encaminhamento alternativo).

# Encaminhamento fixo

Vamos abordar 3 métodos para o cálculo das probabilidades de bloqueio de cada fluxo de chamadas

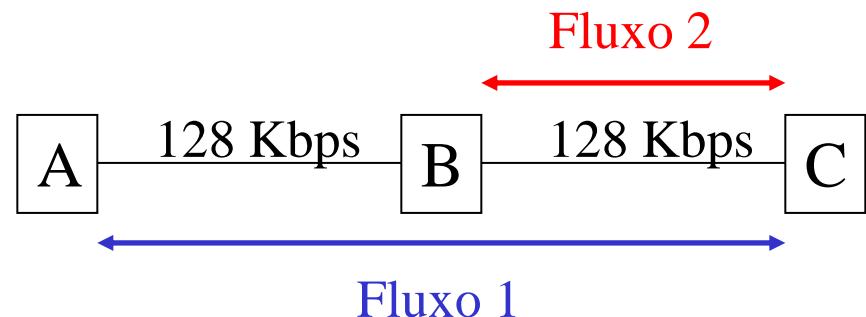
- Método exato
  - Computacionalmente pesado
  - Exige identificação de todos os estados possíveis da rede
- Teorema do limite do produto
  - Computacionalmente leve
  - É um majorante da probabilidade de bloqueio exata
- Aproximação de carga reduzida
  - Uma aproximação (normalmente boa) dos valores exatos
  - Matematicamente complexo
  - Existem algoritmos iterativos de cálculo

# Encaminhamento fixo – Método Exato

- Considere-se uma rede com  $J$  ligações que serve  $K$  fluxos de chamadas.
- A ligação  $j = 1, \dots, J$  tem capacidade  $C_j$  (em número de circuitos).
- Ao fluxo  $k = 1, \dots, K$  está associada uma taxa de chegada,  $\lambda_k$ , um tempo médio de serviço  $1/\mu_k$  (a intensidade de tráfego é  $\rho_k = \lambda_k / \mu_k$ ), uma largura de banda  $b_k$  (em número de circuitos) e um percurso de encaminhamento fixo  $R_k \subseteq \{1, 2, \dots, J\}$ :
  - (1) As chamadas do fluxo  $k$  chegam de acordo com um processo de Poisson à taxa  $\lambda_k$ .
  - (2) As chamadas do fluxo  $k$  admitidas pelo sistema ocupam  $b_k$  circuitos e têm uma duração exponencialmente distribuída com média  $1/\mu_k$ .
  - (3) A duração das chamadas é independente entre chamadas e independente dos instantes de chegada para todos os fluxos.
- O conjunto dos fluxos que atravessam a ligação  $j$  é dado por  $K_j$ .

# Encaminhamento fixo (Exemplo 1)

Considere a rede de figura que suporta 2 fluxos de chamadas: fluxo 1 com taxa de chegada  $\lambda_1 = 3$  chamadas/hora, duração média das chamadas  $1/\mu_1 = 2$  minutos e cada chamada ocupa  $b_1 = 64$  Kb/s; fluxo 2 com taxa de chegada  $\lambda_2 = 4$  chamadas/hora, duração média das chamadas  $1/\mu_2 = 3$  minutos e cada chamada ocupa  $b_2 = 128$  Kb/s.



$K = 2$  fluxos:

1:  $\rho_1 = \lambda_1/\mu_1 = 3/60 \times 2 = 0.1$  Erlangs,  $b_1 = 1$  circuito,  $R_1 = \{AB, BC\}$

2:  $\rho_2 = \lambda_2/\mu_2 = 4/60 \times 3 = 0.2$  Erlangs,  $b_2 = 2$  circuitos,  $R_2 = \{BC\}$

$J = 2$  ligações:

AB:  $C_{AB} = 2$  circuitos,  $K_{AB} = \{1\}$

BC:  $C_{BC} = 2$  circuitos,  $K_{BC} = \{1,2\}$

# Encaminhamento fixo – Método Exato

Seja  $n_k$  o número de chamadas do fluxo  $k$  no sistema,  $\mathbf{n} = (n_1, \dots, n_k)$

Uma chamada do fluxo  $k$  não é aceite pela rede se em pelo menos uma das ligações pertencentes ao percurso de  $k$ :

$$b_k + \sum_{l \in K_j} b_l n_l > C_j$$

O espaço de estados do processo de nascimento e morte multidimensional é

$$S = \left\{ \mathbf{n} \in I^K : \sum_{k \in K_j} b_k n_k \leq C_j, \quad j = 1, \dots, J \right\}$$

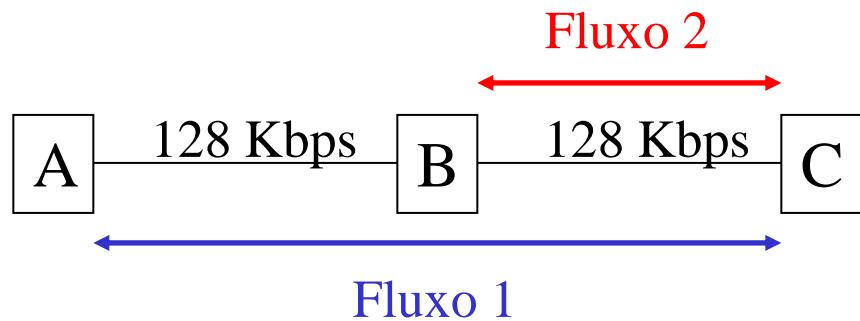
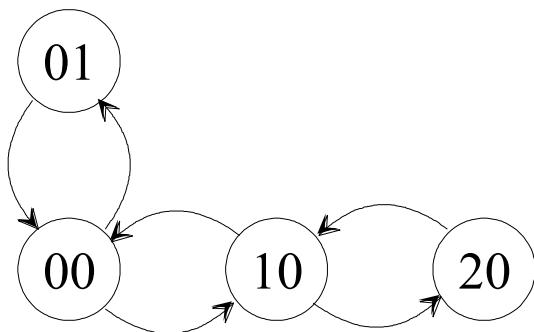
onde  $I$  é o conjunto dos inteiros não negativos.

Seja  $S_k$  o subconjunto dos estados nos quais uma chamada do fluxo  $k$  é admitida quando chega à rede, isto é,

$$S_k = \left\{ \mathbf{n} \in S : \sum_{l \in K_j} b_l n_l \leq C_j - b_k, \quad j \in R_k \right\}$$

# Encaminhamento fixo (Exemplo 1)

Considere a rede de figura que suporta 2 fluxos de chamadas: fluxo 1 com taxa de chegada  $\lambda_1 = 3$  chamadas/hora, duração média das chamadas  $1/\mu_1 = 2$  minutos e cada chamada ocupa  $b_1 = 64$  Kb/s; fluxo 2 com taxa de chegada  $\lambda_2 = 4$  chamadas/hora, duração média das chamadas  $1/\mu_2 = 3$  minutos e cada chamada ocupa  $b_2 = 128$  Kb/s.



Espaço de estados:  $S = \{[0,0], [1,0], [2,0], [0,1]\}$

Estados nos quais uma chamada do fluxo 1 é admitida:

$$S_1 = \{[0,0], [1,0]\}$$

Estados nos quais uma chamada do fluxo 2 é admitida:

$$S_2 = \{[0,0]\}$$

# Encaminhamento fixo – Método Exato

A probabilidade limite de cada estado é dada por

$$P(\mathbf{n}) = \frac{1}{G} \prod_{k=1}^K \frac{\rho_k^{n_k}}{n_k!} \quad \mathbf{n} \in S$$

onde  $G = \sum_{\mathbf{n} \in S} \prod_{k=1}^K \frac{\rho_k^{n_k}}{n_k!}$

e a probabilidade de bloqueio da classe  $k$  é dada por

$$B_k = 1 - \frac{\sum_{\mathbf{n} \in S_k} \prod_{l=1}^K \frac{\rho_l^{n_l}}{n_l!}}{\sum_{\mathbf{n} \in S} \prod_{l=1}^K \frac{\rho_l^{n_l}}{n_l!}} \Leftrightarrow B_k = \frac{\sum_{\mathbf{n} \in S \setminus S_k} \prod_{l=1}^K \frac{\rho_l^{n_l}}{n_l!}}{\sum_{\mathbf{n} \in S} \prod_{l=1}^K \frac{\rho_l^{n_l}}{n_l!}}$$

# Encaminhamento fixo (Exemplo 1)

Considere a rede de figura que suporta 2 fluxos de chamadas: fluxo 1 com taxa de chegada  $\lambda_1 = 3$  chamadas/hora, duração média das chamadas  $1/\mu_1 = 2$  minutos e cada chamada ocupa  $b_1 = 64$  Kb/s; fluxo 2 com taxa de chegada  $\lambda_2 = 4$  chamadas/hora, duração média das chamadas  $1/\mu_2 = 3$  minutos e cada chamada ocupa  $b_2 = 128$  Kb/s.

$$1: \rho_1 = 0.1 \text{ Erlangs}$$

$$2: \rho_2 = 0.2 \text{ Erlangs}$$

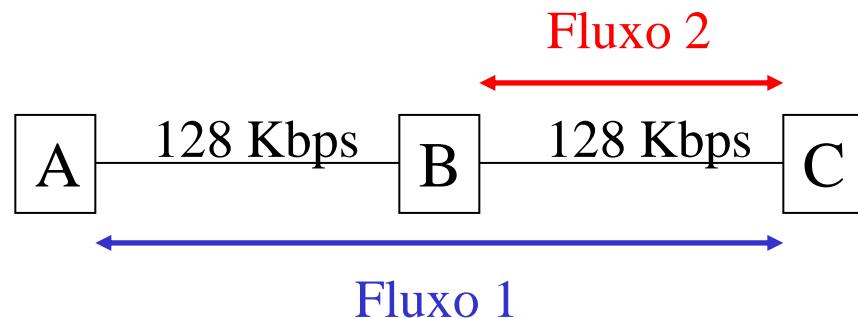
$$S = \{[0,0], [1,0], [2,0], [0,1]\}$$

$$S_1 = \{[0,0], [1,0]\}$$

$$S_2 = \{[0,0]\}$$

$$B_k = 1 - \frac{\sum_{n \in S_k} \prod_{l=1}^K \frac{\rho_l^{n_l}}{n_l!}}{\sum_{n \in S} \prod_{l=1}^K \frac{\rho_l^{n_l}}{n_l!}}$$

$$\begin{aligned} B_1 &= 1 - \frac{\frac{0.1^0 0.2^0}{0! 0!} + \frac{0.1^1 0.2^0}{1! 0!}}{\frac{0.1^0 0.2^0}{0! 0!} + \frac{0.1^1 0.2^0}{1! 0!} + \frac{0.1^2 0.2^0}{2! 0!} + \frac{0.1^0 0.2^1}{0! 1!}} \\ &= 1 - \frac{1 + 0.1}{1 + 0.1 + \frac{0.01}{2} + 0.2} = 0.157 = 15.7\% \end{aligned}$$



# Encaminhamento fixo (Exemplo 1)

Considere a rede de figura que suporta 2 fluxos de chamadas: fluxo 1 com taxa de chegada  $\lambda_1 = 3$  chamadas/hora, duração média das chamadas  $1/\mu_1 = 2$  minutos e cada chamada ocupa  $b_1 = 64$  Kb/s; fluxo 2 com taxa de chegada  $\lambda_2 = 4$  chamadas/hora, duração média das chamadas  $1/\mu_2 = 3$  minutos e cada chamada ocupa  $b_2 = 128$  Kb/s.

$$1: \rho_1 = 0.1 \text{ Erlangs}$$

$$2: \rho_2 = 0.2 \text{ Erlangs}$$

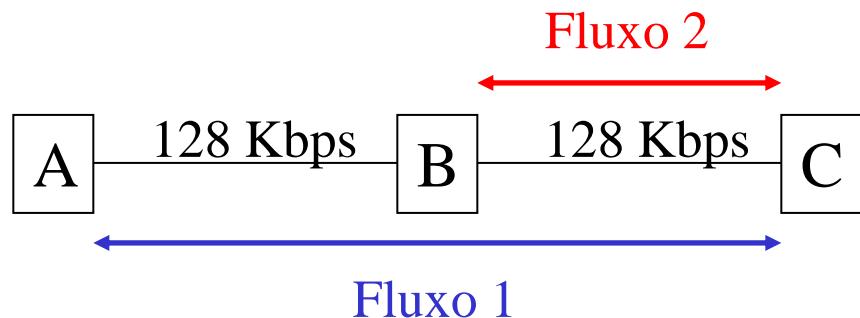
$$S = \{[0,0], [1,0], [2,0], [0,1]\}$$

$$S_1 = \{[0,0], [1,0]\}$$

$$S_2 = \{[0,0]\}$$

$$B_k = 1 - \frac{\sum_{n \in S_k} \prod_{l=1}^K \frac{\rho_l^{n_l}}{n_l!}}{\sum_{n \in S} \prod_{l=1}^K \frac{\rho_l^{n_l}}{n_l!}}$$

$$\begin{aligned} B_2 &= 1 - \frac{\frac{0.1^0 0.2^0}{0! 0!}}{\frac{0.1^0 0.2^0}{0! 0!} + \frac{0.1^1 0.2^0}{1! 0!} + \frac{0.1^2 0.2^0}{2! 0!} + \frac{0.1^0 0.2^1}{0! 1!}} \\ &= 1 - \frac{1}{1 + 0.1 + \frac{0.01}{2} + 0.2} = 0.234 = 23.4\% \end{aligned}$$



# Teorema do Limite do Produto

- Aplica-se apenas quando as chamadas de todos os fluxos  $k$  requerem a mesma largura de banda  $b_k$  (em número de circuitos).
- Seja a intensidade de tráfego suportada pela ligação  $j$  dada por:

$$\bar{\rho}_j = \sum_{k \in K_j} \rho_k$$

- O teorema do limite do produto declara que

$$B_k \leq 1 - \prod_{j \in R_k} \left( 1 - ER[\bar{\rho}_j, C_j] \right) \quad C_j - \text{capacidade da ligação } j \text{ (em número de circuitos)}$$

em que  $ER[\rho, C]$  representa a fórmula de ErlangB.

- Prova-se matematicamente que este valor é um majorante das probabilidades de bloqueio exatas. É uma boa aproximação quando:
  - (1) os fluxos atravessam poucas ligações
  - (2) as probabilidades de bloqueio são pequenas (menores que 1%)

# Aproximação de carga reduzida

- Uma possibilidade para melhorar a aproximação associada ao teorema do limite do produto é reduzir o tráfego oferecido à ligação  $j$ , tomando em linha de conta o bloqueio nas restantes ligações do percurso de cada fluxo.
- O teorema do limite do produto implica que a probabilidade de uma ligação  $j$  estar totalmente ocupada é majorada por

$$ER \left[ \sum_{k \in K_j} \rho_k, C_j \right]$$

- Fazendo a substituição de  $\rho_k$  por  $\rho_k t_k(j)$ , em que  $t_k(j)$  corresponde à probabilidade de existir pelo menos uma unidade de capacidade disponível em cada ligação pertencente a  $R_k - \{j\}$  temos que a probabilidade de bloqueio (aproximada) da ligação  $j$  vem dada por

$$L_j = ER \left[ \sum_{k \in K_j} \rho_k t_k(j), C_j \right]$$

# Aproximação de carga reduzida

- Tomando como aproximação adicional que o bloqueio é independente de ligação para ligação, resulta

$$t_k(j) = \prod_{i \in R_k - \{j\}} (1 - L_i)$$

e, finalmente, combinando as duas aproximações anteriores resultam as seguintes equações de ponto fixo (uma por cada ligação da rede)

$$L_j = ER \left[ \sum_{k \in K_j} \rho_k \prod_{i \in R_k - \{j\}} (1 - L_i), C_j \right], \quad j = 1, 2, \dots, J$$

- Admitindo novamente que o bloqueio é independente de ligação para ligação resulta a probabilidade de bloqueio das chamadas do fluxo  $k$  é

$$B_k \approx 1 - \prod_{j \in R_k} (1 - L_j) \quad k = 1, 2, \dots, K$$

# Algoritmo iterativo de cálculo da aproximação de carga reduzida

Seja  $\mathbf{L} = (L_1, L_2, \dots, L_J)$  e o operador  $\mathbf{T}(\mathbf{L}) = (T_1(\mathbf{L}), T_2(\mathbf{L}), \dots, T_J(\mathbf{L}))$  onde

$$T_j(\mathbf{L}) = ER \left[ \sum_{k \in K_j} \rho_k \prod_{i \in R_k - \{j\}} (1 - L_i), C_j \right]$$

As equações de ponto fixo podem ser expressas na forma  $\mathbf{L} = \mathbf{T}(\mathbf{L})$ .

Método iterativo – partindo de um vetor inicial  $\mathbf{L} \in [0,1]^J$  aplica-se sucessivamente o operador  $\mathbf{T}$ :

$$\mathbf{L}^0 = \mathbf{L}$$

$$\mathbf{L}^m = \mathbf{T}(\mathbf{L}^{m-1}), \quad m = 1, \dots, n$$

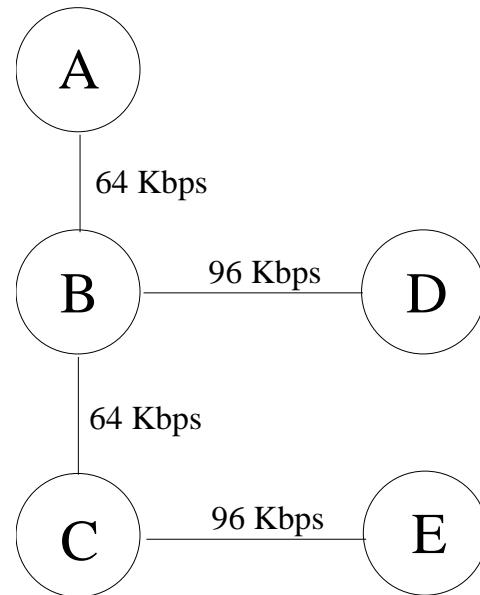
Partindo de  $\mathbf{L}^0 = (1, 1, \dots, 1)$ , o método dá origem a  $\mathbf{L}^1 = (0, 0, \dots, 0)$ ,  $\mathbf{L}^{2n}$  converge para um  $\mathbf{L}^+$  e  $\mathbf{L}^{2n+1}$  converge para um  $\mathbf{L}^-$  tal que  $\mathbf{L}^- \leq \mathbf{L}^* \leq \mathbf{L}^+$ .

As sucessivas iterações  $m$  determinam majorantes da solução  $\mathbf{L}^*$  quando  $m$  é par e minorantes da solução  $\mathbf{L}^*$  quando  $m$  é ímpar. Termina-se o algoritmo quando os dois limites estão suficientemente próximos.

## Exemplo 2

Considere a rede da figura. A rede suporta 3 fluxos de chamadas: fluxo 1 entre A e D, fluxo 2 entre C e D e fluxo 3 entre E e B. As chamadas chegam de acordo com processos de Poisson com taxa  $\lambda_1 = 20$  chamadas/hora,  $\lambda_2 = 60$  chamadas/hora e  $\lambda_3 = 20$  chamadas/hora. Em todos os fluxos, a duração de cada chamada é exponencialmente distribuída com média  $1/\mu = 3$  minutos e cada chamada requer uma largura de banda de 32 Kbps.

- (1) Calcule um limite superior para a probabilidade de bloqueio de cada fluxo usando o teorema do limite do produto.
- (2) Escreva as equações que permitem calcular as probabilidades de bloqueio de cada fluxo através da aproximação de carga reduzida.



# Encaminhamento dinâmico da rede telefónica

Os métodos de encaminhamento dinâmico são usados nas redes de transporte dos operadores telefónicos.

Estas redes têm conectividade total, ou seja, incluem uma ligação direta entre todos os pares de nós de acesso.

Um nó de acesso é uma central que liga uma rede de acesso à rede de transporte.

Assim, numa rede com  $N$  nós:

- existem  $N(N - 1) / 2$  ligações
- o número de percursos com apenas duas ligações entre quaisquer duas centrais é  $N - 2$ .

# **Encaminhamento dinâmico da rede telefónica**

Os métodos de encaminhamento dinâmico têm as seguintes características em comum:

1. quando é pedido o estabelecimento de uma chamada entre duas centrais, a chamada é encaminhada no percurso direto, se houver pelo menos um circuito disponível;
2. quando o percurso direto estiver indisponível, a chamada pode ser encaminhada num dos percursos alternativos permitidos;
3. os percursos alternativos têm sempre apenas duas ligações, ou seja, as chamadas nunca são estabelecidas em percursos com três ou mais ligações.

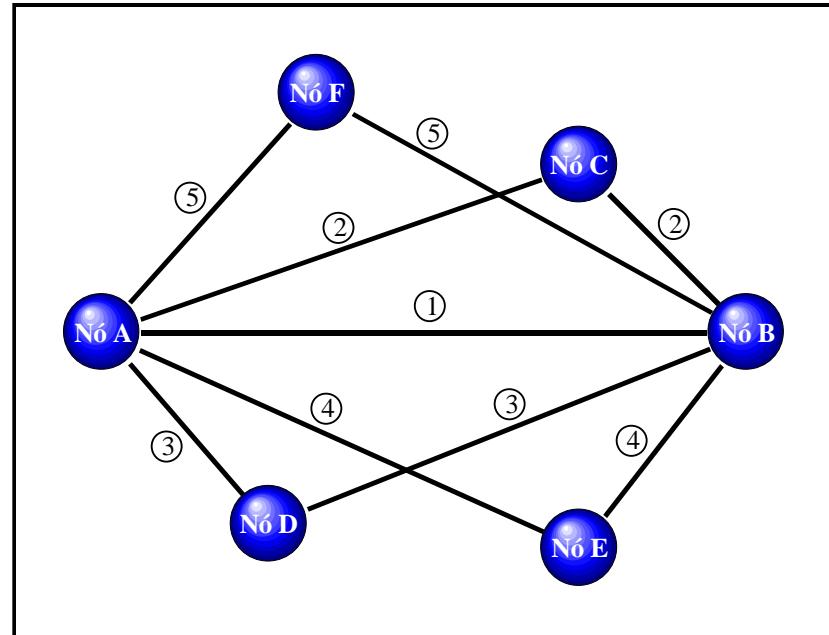
Os diferentes métodos de encaminhamento diferem na forma como é definido, em cada instante, o conjunto de percursos alternativos.

# Encaminhamento sequencial

- É a base do método introduzido pela AT&T nos anos 80 designado por DNHR - Dynamic Non-Hierarchical Routing.
- A cada par de centrais origem-destino associa-se uma lista ordenada de percursos alternativos. Se uma chamada que chega encontra o percurso direto indisponível:
  - 1) a chamada é estabelecida no primeiro percurso alternativo disponível da lista ordenada;
  - 2) se todos os percursos alternativos estiverem indisponíveis a chamada é bloqueada.
- Na implementação do DNHR, os diferentes parâmetros da rede (a lista de percursos alternativos, a ordenação dos percursos na lista, a percentagem de circuitos reservados em cada ligação, etc...) variam no tempo, sendo considerados até 10 períodos distintos em cada dia.
- Quando a segunda ligação de um percurso alternativo estiver indisponível, a central intermédia tem de sinalizar a central origem dessa ocorrência para que esta possa tentar outro percurso alternativo (função designada por crankback).

# DNHR

Nó A → Nó B



Lista #N	Percursos recomendados presentes na lista	Período de tempo
Listá #1	1→3→2→4	10:00– 12:00
Listá #2	1→4→2	12:00 – 17:00
Listá #3	1→3→5→2	17:00– 19:00
Listá #4	1→5→4	19:00– 23:00
Listá #5	1→2→3	23:00 – 10:00

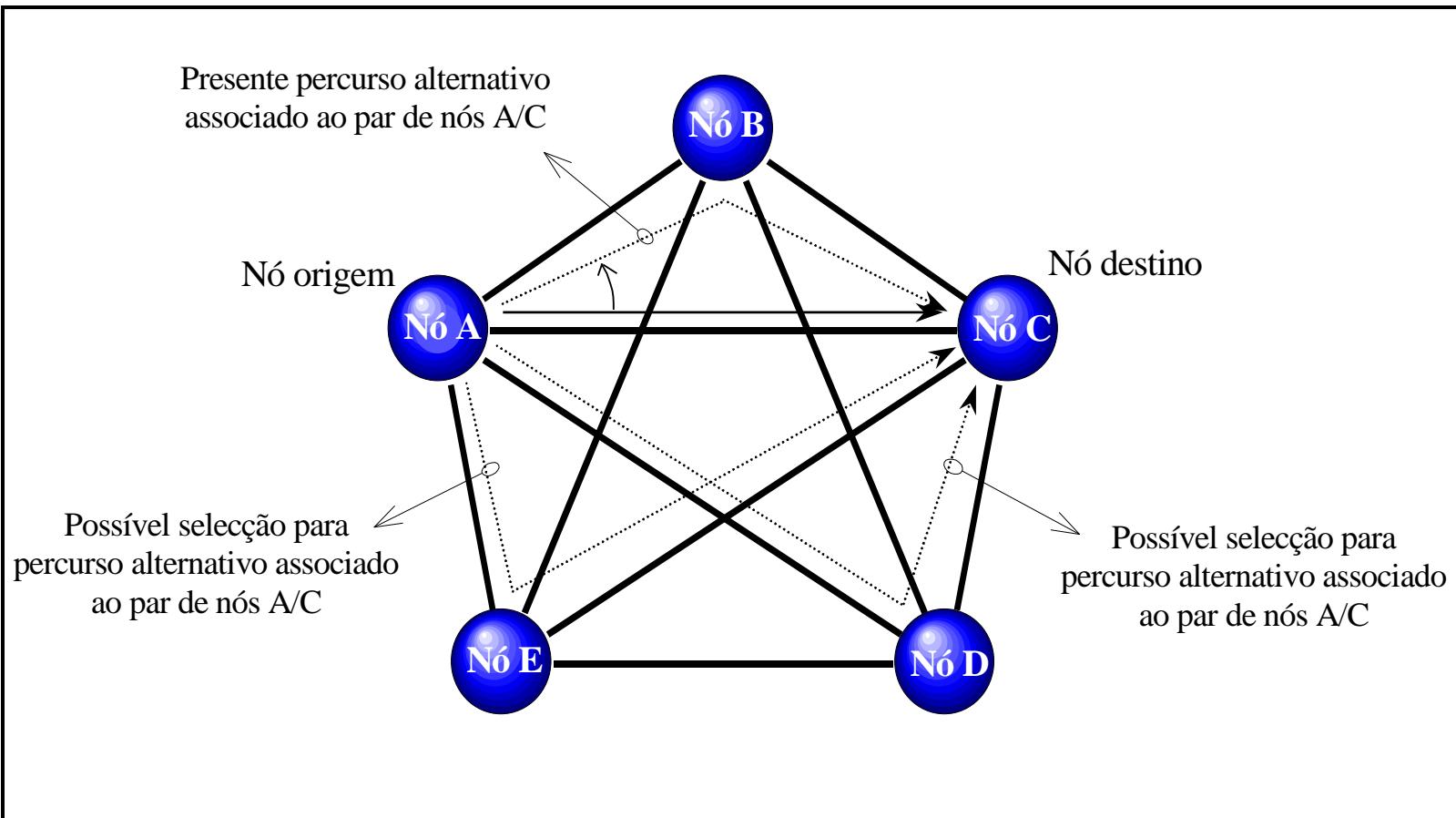
# Encaminhamento aleatório retardado

- Este método é a base do protocolo DAR - Dynamic Alternative Routing introduzido pelos British Telecom nos anos 90.
- Associa-se a cada par de centrais origem-destino um percurso alternativo (e apenas um):
  - 1) se quando uma chamada chega o percurso direto está indisponível e o percurso alternativo está disponível a chamada é estabelecida no percurso alternativo;
  - 2) caso contrário, a chamada é bloqueada e é escolhido um novo percurso alternativo para as chamadas subsequentes;
  - 3) o novo percurso alternativo é escolhido aleatoriamente de entre os  $N - 3$  percursos alternativos restantes.

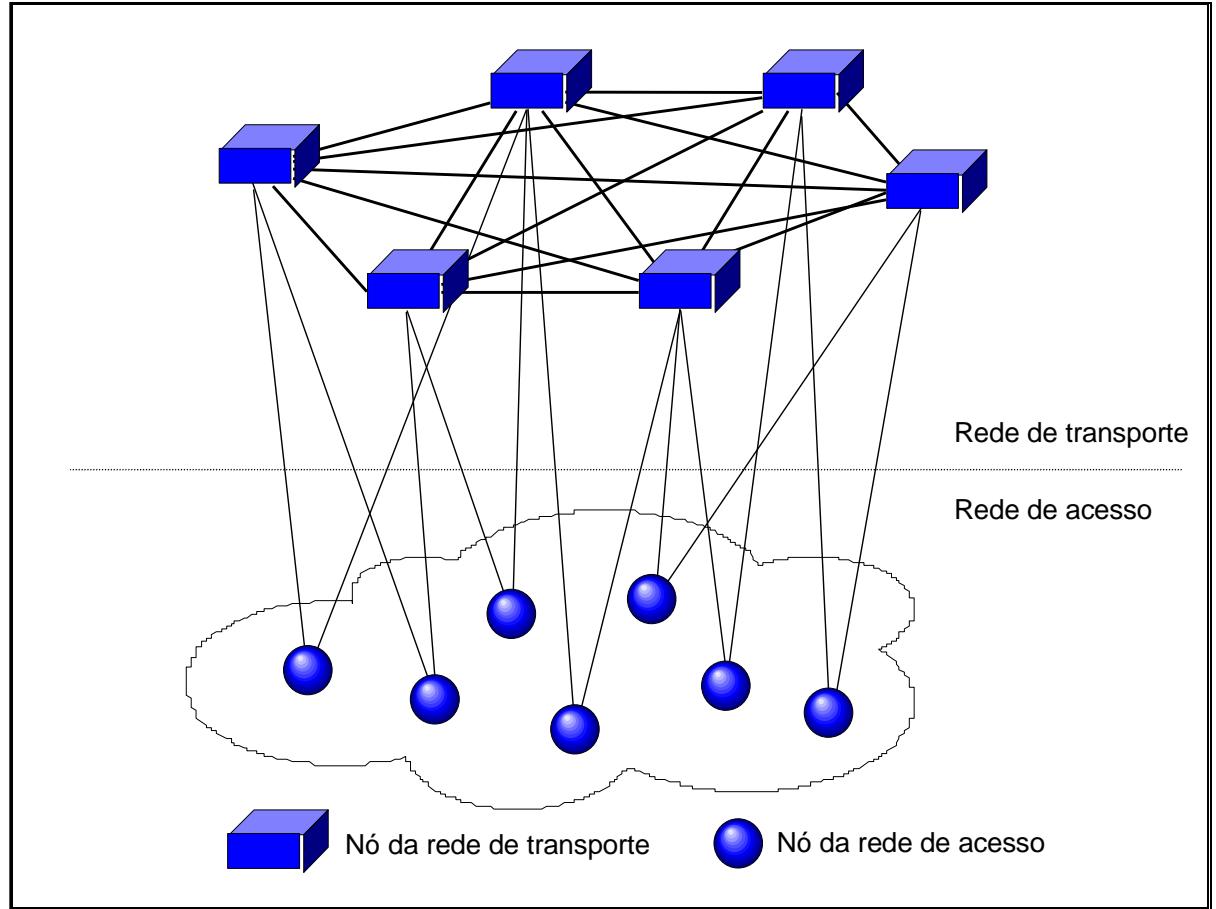
Relativamente ao DHNR:

- não é necessário implementar a função de *crankback*, porque apenas é tentado um percurso alternativo (os mecanismos de sinalização são mais simples)
- adapta-se dinamicamente ao tráfego sem necessidade de gestão centralizada dos percursos alternativos

# Operação do protocolo DAR

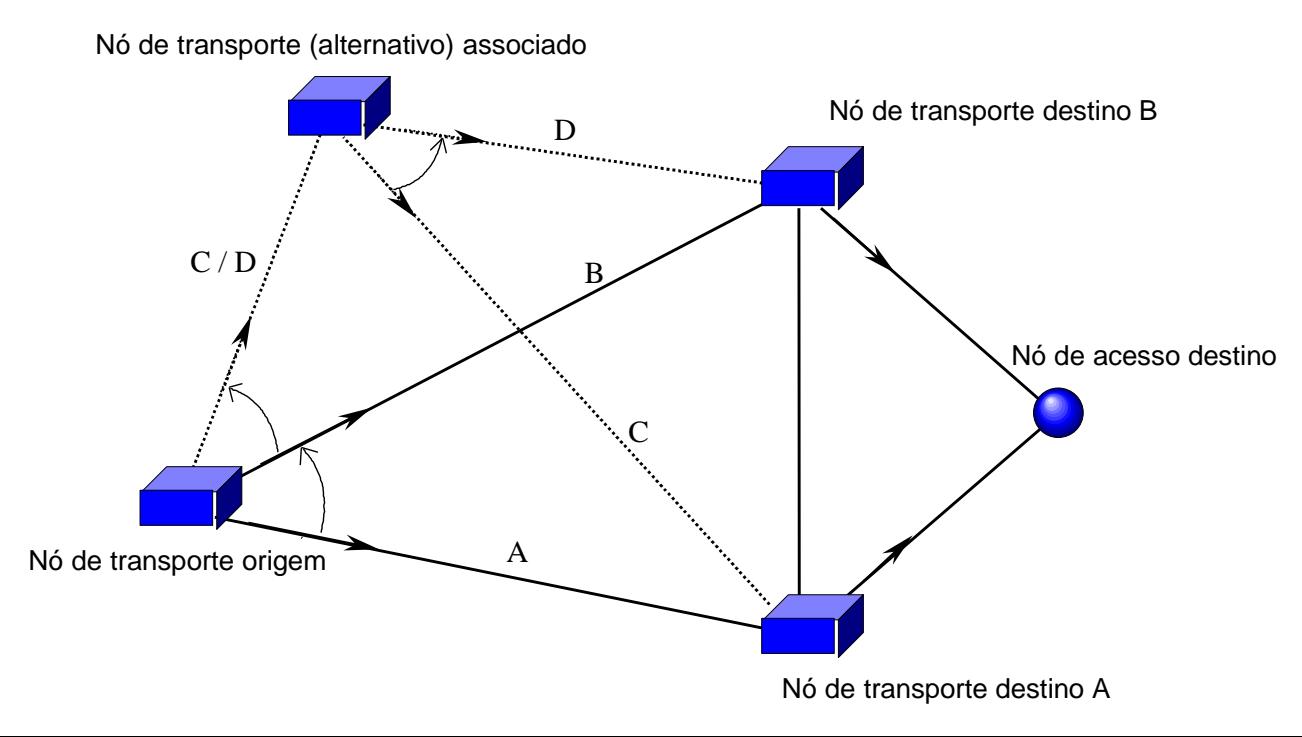


# DAR na rede da British Telecom



- Rede organizada em rede de transporte e rede de acesso.
- Rede de transporte com conectividade física total.
- Cada nó de acesso com ligação física a dois nós de transporte (para proteção de falha individual de ligação).

# DAR na rede da British Telecom



- No nó de transporte onde chega o pedido de chamada existem 2 percursos diretos e um nó de transporte alternativo.
- Primeiro são tentados os dois percursos diretos.
- Depois são tentados os dois percursos via nó alternativo.
- Se a chamada bloquear, outro nó alternativo é escolhido para o próximo pedido de chamada para o mesmo destino.

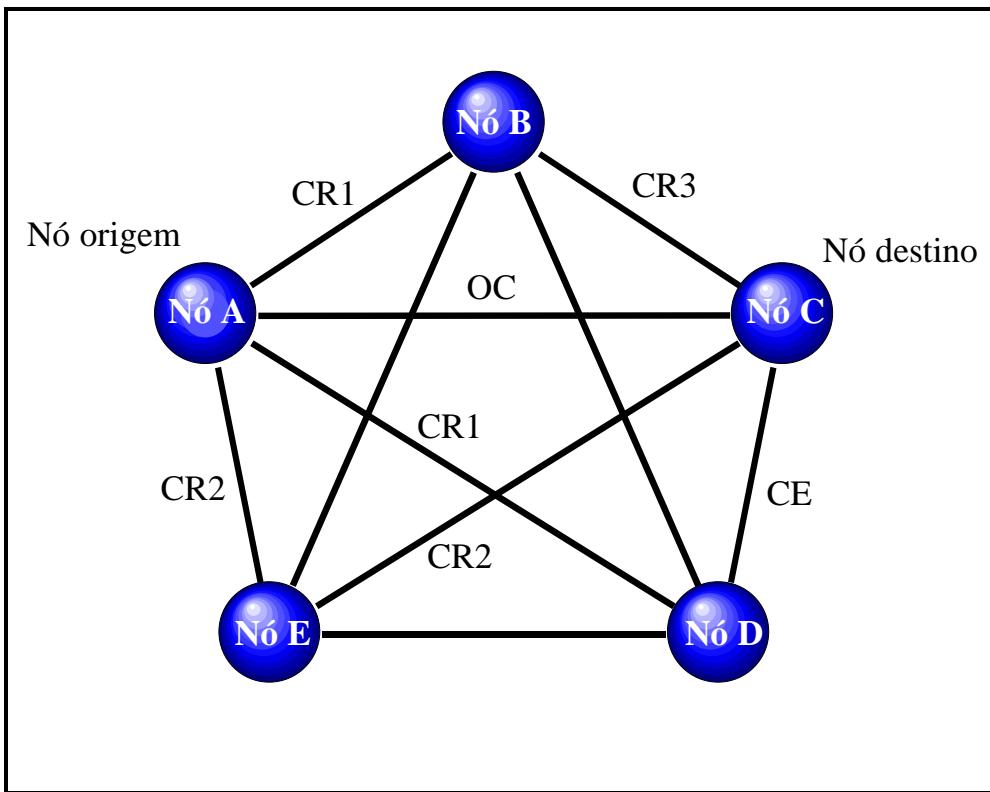
## Encaminhamento de menor carga

- Este método mantém um registo da capacidade não utilizada em cada percurso alternativo (corresponde ao número de circuitos não ocupados, para lá dos circuitos reservados).
- Quando o percurso direto está indisponível, escolhe o percurso alternativo com maior capacidade não utilizada, de entre o conjunto de percursos alternativos permitidos; se todos os percursos alternativos estiverem indisponíveis a chamada é bloqueada.

Este método está na base do método RTNR - Real-Time Network Routing introduzido pela AT&T no início dos anos 90:

- 1) quando o percurso direto está indisponível, a central origem pergunta à central destino qual a capacidade não utilizada de todas as suas ligações;
  - 2) após receber essa informação, a central origem determina qual o percurso alternativo com menor capacidade utilizada, com base na informação de que dispõe sobre a capacidade não utilizada das suas próprias ligações.
- Este método é mais eficiente que o DAR mas exibe uma maior complexidade nos mecanismos de sinalização.

# RTNR



## Níveis de carga considerados:

CR1 - Carga Reduzida de nível 1

CR2 - Carga Reduzida de nível 2

CR3 - Carga Reduzida de nível 3

CE - Carga Elevada

RS – Reservado

OC - Ocupado

$\text{Nó A} \rightarrow \text{Nó C}$

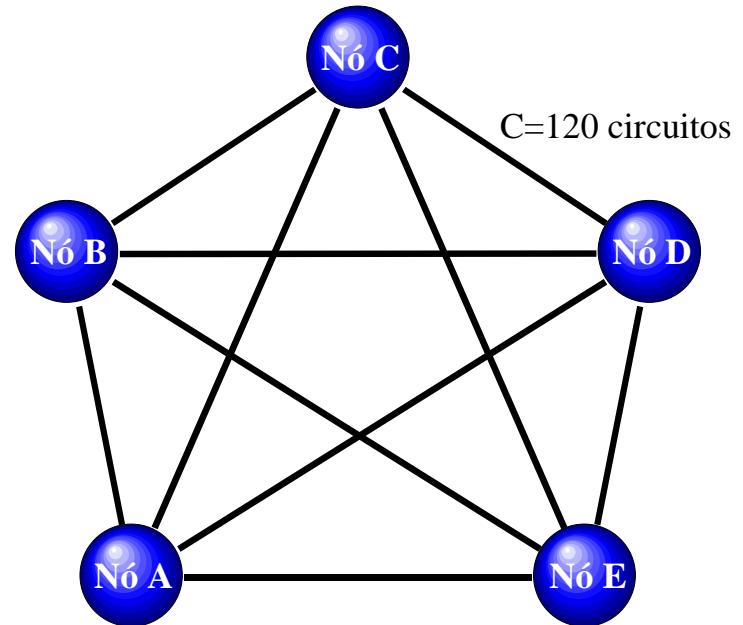
A chamada é estabelecida pelo  
nó intermédio E

# Metaestabilidade e reserva de recursos

Exemplo:

- rede com conectividade total;
- encaminhamento aleatório global;
- tráfego oferecido igual para todos os pares origem-destino;
- mesma reserva de recursos  $r$  para todos os percursos diretos.

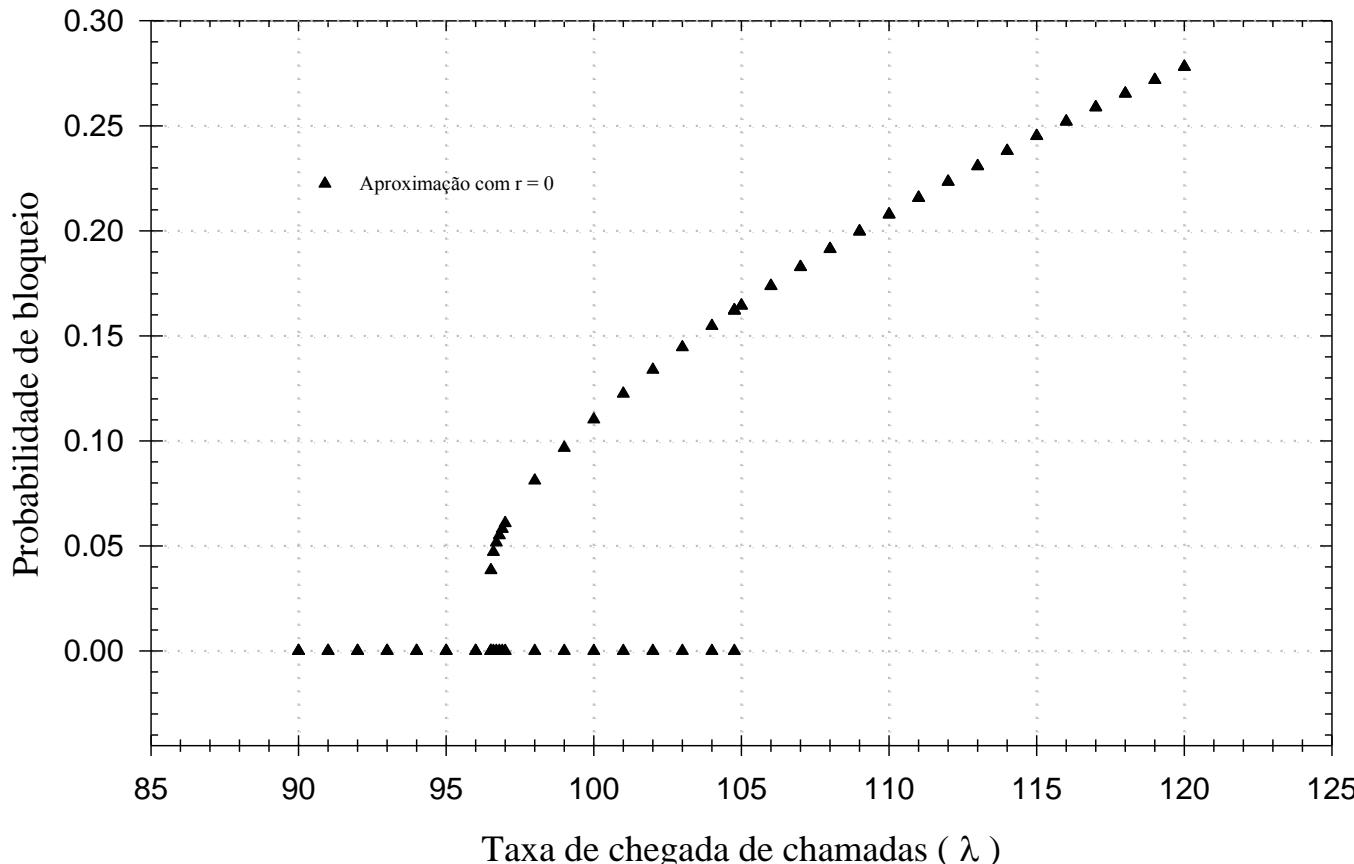
O desempenho desta rede pode ser calculado de forma aproximada por processos analíticos.



# Exemplo sem reserva de recursos ( $r=0$ )

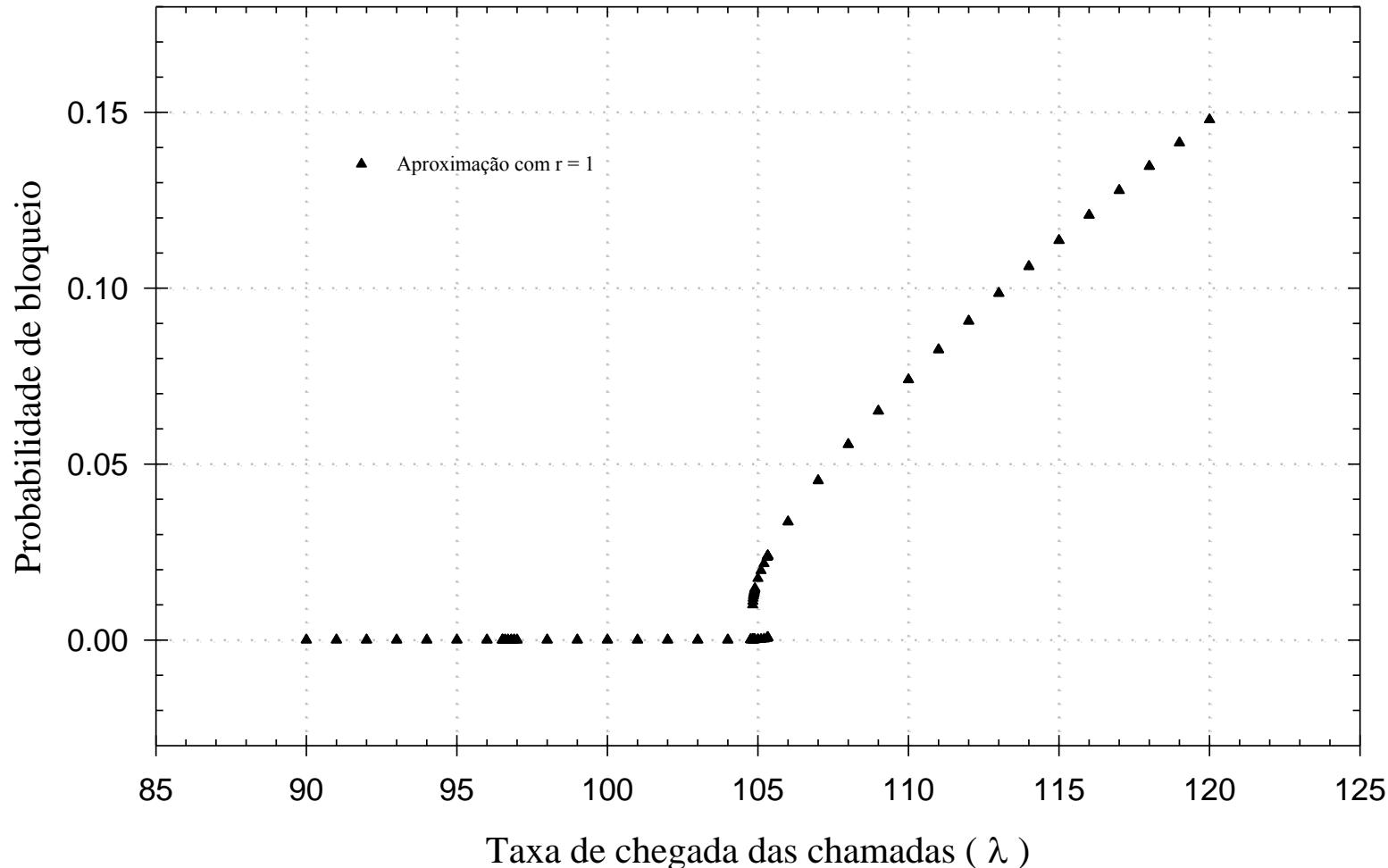
Instabilidade para valores de  $\lambda$  entre 96 e 105:

- Existem intervalos de tempo em que a maior parte do tráfego é encaminhada pelo percurso direto (probabilidade de bloqueio desprezável).
- Existem intervalos de tempo em que a maior parte do tráfego é encaminhada pelos percursos alternativos (probabilidade de bloqueio elevada).



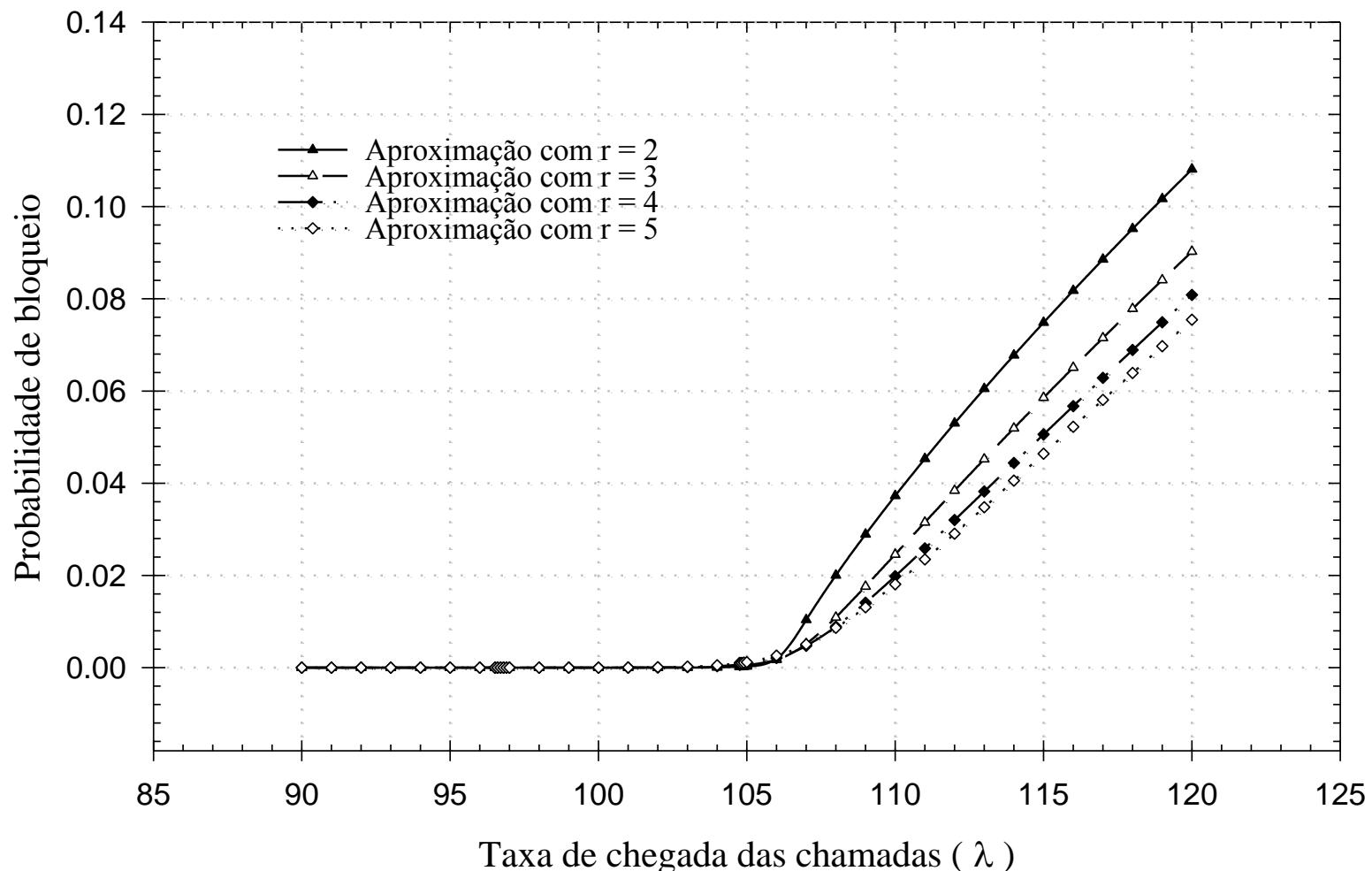
# Exemplo com reserva de um circuito ( $r=1$ )

A instabilidade é reduzida com uma reserva de 1 em 120 circuitos



# Exemplo com reserva de 2 ou mais circuitos

Valores crescentes de reserva anulam a instabilidade e aumentam o desempenho da rede (diminuem a percentagem de tráfego que usa percursos alternativos)





# **Controlo de Fluxos em Redes com Comutação de Pacotes**

Desempenho e Dimensionamento de Redes

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

DETI-UA, 2017/2018

# Controlo de fluxo - introdução

O tráfego efetivo reflete a quantidade de serviço suportada por uma rede com comutação de pacotes.

O atraso médio reflete a qualidade de serviço proporcionada por uma rede com comutação de pacotes.

Controlo de fluxo: mecanismo de realimentação que estabelece um compromisso entre o tráfego efetivo e o atraso médio por forma a manter o atraso médio dentro de limites aceitáveis:

- Quando o tráfego oferecido é reduzido, é aceite na sua totalidade pelo algoritmo de controlo de fluxo e, neste caso,

$$\text{tráfego efetivo} = \text{tráfego oferecido}$$

- Quando o tráfego oferecido é excessivo, o algoritmo de controlo de fluxo rejeita parte dele e, neste caso,

$$\text{tráfego efetivo} = \text{tráfego oferecido} - \text{tráfego rejeitado}$$

- À medida que o algoritmo de encaminhamento diminui o atraso médio, o controlo de fluxo aceita a entrada de mais tráfego na rede.

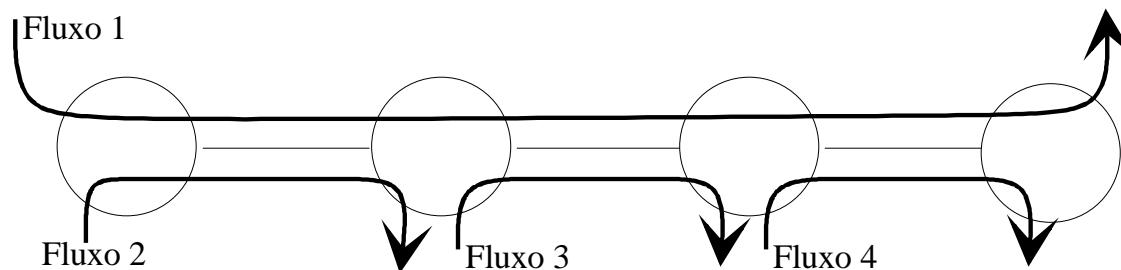
# **Controlo de fluxo - introdução**

Os algoritmos de controlo de fluxo devem idealmente observar os seguintes requisitos:

- Estabelecer um bom compromisso entre:
  - a quantidade de serviço (o tráfego efetivo, sujeito eventualmente à garantia de uma taxa de transmissão mínima) e
  - a qualidade de serviço (medida, por exemplo, a partir do atraso médio e da taxa de pacotes perdidos)
- Garantir um tratamento equitativo dos diferentes fluxos de pacotes, ao fornecer a qualidade de serviço requerida.

# Gestão de recursos: equidade vs. utilização

Considere-se o exemplo da figura assumindo que a capacidade de cada ligação é 100.



Maximização do tráfego efetivo (máxima utilização):

$$\text{Fluxo 1} = 0, \text{ Fluxos 2,3,4} = 100$$

$$\text{Tráfego efetivo total} = 0+100+100+100 = 300$$

Mesma taxa de transmissão a todos os fluxos (máxima equidade):

$$\text{Fluxos 1,2,3,4} = 50$$

$$\text{Tráfego efetivo total} = 50+50+50+50 = 200$$

Partilha equitativa dos recursos:

$$\text{Fluxo 1} = 25, \text{ Fluxos 2,3,4} = 75$$

$$\text{Tráfego efetivo total} = 25+75+75+75 = 250$$

# Controlo de fluxo através de janelas (I)

- Considere um fluxo de dados de um emissor A para um recetor B.
- Por cada unidade de dados recebida, o recetor B notifica o emissor A através do envio para A de uma permissão:
  - Uma permissão pode ser transmitida num pacote de controlo dedicado ou pode ser encavalitada (*piggybacked*) num pacote de dados enviado no sentido contrário.
- Quando recebe uma permissão, o emissor A fica autorizado a enviar mais uma unidade de dados para o recetor B.
- Um esquema de controlo de fluxos através de janelas pode ser combinado com um protocolo ARQ de controlo de erros
  - neste caso, as permissões servem também de *acknowledgments*

## Controlo de fluxo através de janelas (II)

- Um fluxo entre o emissor A e o recetor B diz-se controlada através de janelas se existir um limite máximo para o número de unidades de dados que, tendo sido transmitidas por A, não foram ainda notificadas como tendo sido recebidas por B.
- O limite máximo é designado por tamanho da janela, ou simplesmente, *janela*.
- O emissor e o recetor podem ser dois nós da rede, um terminal e o nó de entrada da rede ou os dois terminais que estão nos extremos do fluxo.
- As unidades de dados podem ser mensagens, pacotes ou bytes.

## Controlo de fluxo através de janelas (III)

- No controlo de fluxos através de janelas, a taxa de transmissão do emissor é reduzida à medida que as permissões demoram mais tempo a regressar.
- Assim, se o percurso utilizado pelo fluxo estiver congestionado as permissões sofrem grandes atrasos o que obriga o emissor a reduzir a sua taxa de transmissão.
- Além disso, o recetor pode atrasar intencionalmente o envio de permissões para reduzir a taxa de transmissão do fluxo com o objetivo de, por exemplo, evitar a sobrecarga do seu *buffer* de receção.

De seguida, considera-se a estratégia de **janelas extremo-a-extremo** (*end-to-end*):

- para cada fluxo de pacotes, o controlo de fluxos é implementado entre o seu emissor e o seu recetor
- estratégia usada pelo TCP nas redes TCP/IP

## Janelas extremo-a-extremo

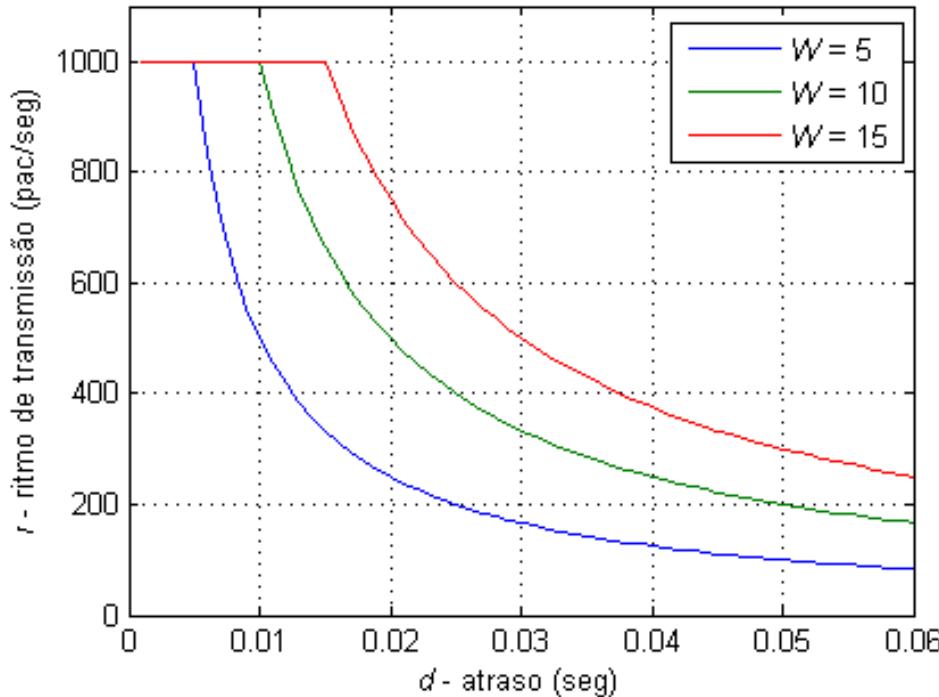
- Em geral, o tamanho da janela é  $\alpha W$  ( $\alpha$  e  $W$  são números positivos). Cada vez que um grupo de  $\alpha$  unidades de dados é recebido no nó destino é gerada uma permissão autorizando o envio de um novo grupo de  $\alpha$  unidades de dados. De seguida, considera-se que  $\alpha = 1$  e que cada unidade de dados é um pacote.
- Considere-se o atraso de ida-e-volta dado por  $d$  e o tempo de transmissão médio de cada pacote dado por  $X$ :
  - ✓ Se  $d \leq WX$ , a transmissão de  $W$  pacotes demora mais que o atraso de ida-e-volta; assim, o emissor pode transmitir à velocidade máxima de  $1/X$  pacotes/segundo.
  - ✓ Se  $d > WX$ , o controlo de fluxos está ativo pois o atraso de ida-e-volta é tão elevado que  $W$  pacotes são transmitidos antes da receção da permissão relativa ao primeiro dos pacotes.

Então, o ritmo de transmissão é dado por:

$$r = \min\left\{\frac{1}{X}, \frac{W}{d}\right\}$$

## Janelas extremo-a-extremo

- Exemplo:  $X = 1$  mseg. e janela  $W = 5, 10$  e  $15$  pacotes.



$$r = \min\left\{\frac{1}{X}, \frac{W}{d}\right\}$$

- ✓ Para valores  $d \leq WX$ , o emissor transmite à velocidade máxima  $r = 1/10^{-3} = 1000$  (em pacotes/segundo)
- ✓ Para valores  $d > WX$ , o controlo de fluxos está ativo e o emissor transmite à velocidade máxima  $r = W/d$  (em pacotes/segundo)

# Dimensionamento do tamanho das janelas

Existe um compromisso entre tráfego efetivo e atraso:

- por um lado as janelas devem ser pequenas para limitar o número de pacotes na rede, evitando assim grandes atrasos e congestão;
- por outro, as janelas devem ser grandes para permitir transmissão à velocidade máxima e tráfego efetivo máximo em condições de tráfego oferecido moderado e leve.

De qualquer modo, é sempre desejável que cada fluxo possa transmitir à velocidade máxima, quando não existe nenhum outro fluxo ativo.

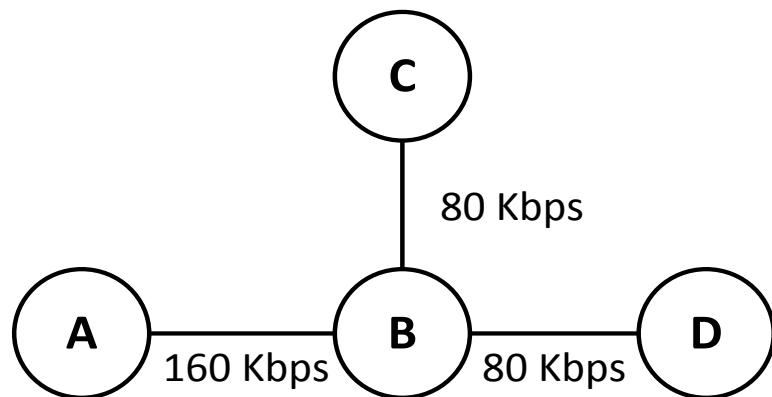
Esta condição impõe um limite inferior ao tamanho das janelas. Se  $d \leq WX$  então o fluxo pode transmitir à velocidade máxima pelo que o tamanho da janela (em número de pacotes) deverá ser dado por

$$W = \left\lceil \frac{d}{X} \right\rceil$$

onde  $\lceil x \rceil$  representa o menor inteiro não inferior a  $x$  e  $d$  deverá ser o menor atraso de ida-e-volta proporcionado pela rede.

## Exemplo 1

Considere a rede com comutação de pacotes da figura. O atraso de propagação de cada ligação é 100 mseg em cada sentido. A rede suporta dois fluxos: A→D com pacotes de tamanho médio 1000 bytes e C→D com pacotes de tamanho médio 500 bytes. A ambos os fluxos é aplicado um mecanismo de controle de fluxos baseado no método das janelas extremo-a-extremo e em ambos os casos, as permissões têm um tamanho fixo de 20 Bytes. Determine o tamanho mínimo (em número de pacotes) das janelas de emissão garantindo que cada fluxo pode emitir ao ritmo máximo quando o outro não está a emitir pacotes.

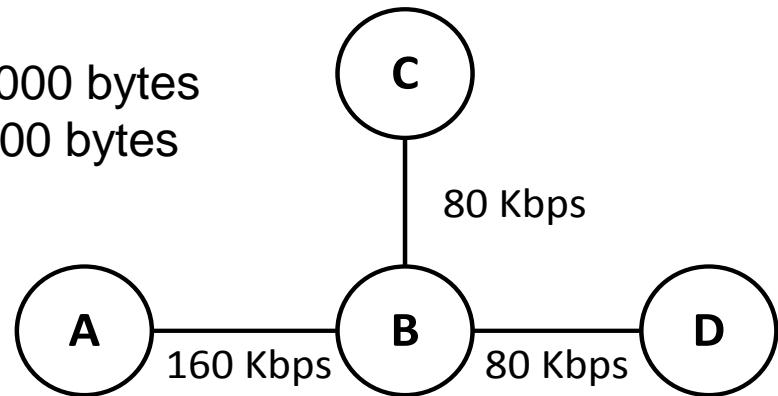


## Exemplo 1 - resolução

A → D com pacotes de tamanho médio 1000 bytes

C → D com pacotes de tamanho médio 500 bytes

$$W \geq \left\lceil \frac{d}{X} \right\rceil$$



$$W_{AD} \geq \left\lceil \frac{\frac{8 \times 1000}{160000} + 0.1 + \frac{8 \times 1000}{80000} + 0.1 + \frac{8 \times 20}{80000} + 0.1 + \frac{8 \times 20}{160000} + 0.1}{\frac{8 \times 1000}{80000}} \right\rceil = \left\lceil \frac{0.553}{0.1} \right\rceil = 6 \text{ pacotes}$$

$$W_{CD} \geq \left\lceil \frac{\frac{8 \times 500}{80000} + 0.1 + \frac{8 \times 500}{80000} + 0.1 + \frac{8 \times 20}{80000} + 0.1 + \frac{8 \times 20}{80000} + 0.1}{\frac{8 \times 500}{80000}} \right\rceil = \left\lceil \frac{0.504}{0.05} \right\rceil = 11 \text{ pacotes}$$

# Limitações do controlo de fluxo baseado em janelas

1. Não permite assegurar uma taxa mínima de transmissão. Quantos mais fluxos forem submetidos na rede, menor é o tráfego efetivo que cada fluxo obtém.
2. Não fornece um controlo adequado do atraso. Considerem-se  $n$  fluxos com controlo de fluxos ativo através de janelas com tamanho fixo  $W_1, \dots, W_n$ . O número total de pacotes e permissões é  $\sum_{i=1}^n W_i$  e o número de pacotes é  $\sum_{i=1}^n \beta_i W_i$  onde  $\beta_i$  é um valor entre 0 e 1.

Pelo teorema de Little, o atraso médio por pacote é

$$T = \frac{\sum_{i=1}^n \beta_i W_i}{\lambda}$$

onde  $\lambda$  é o tráfego efetivo de todos os fluxos. À medida que o número de fluxos aumenta, o tráfego efetivo tende para um valor constante (é limitado pela capacidade das ligações). Assim, o atraso médio por pacote aumenta aproximadamente de forma proporcional ao número de fluxos.

# Controlo de taxas de transmissão

- A função de controlo de fluxos pode atribuir a cada fluxo uma taxa de transmissão máxima compatível com as suas necessidades.
- Essa taxa pode, por exemplo, ser definida na fase de estabelecimento de um circuito virtual.
- De seguida, consideram-se dois métodos para controlar a taxa de transmissão:
  - por janelas
  - através de *leaky bucket* (usado pela arquitetura *Integrated Services* (IntServ) nas redes IP)

## Controlo de taxas de transmissão por janelas (I)

- Considere-se que foi atribuída uma taxa de transmissão de  $r$  pacotes por segundo a um determinado fluxo.
- Uma possibilidade para garantir esta taxa poderia ser aceitar, quando muito, um pacote em cada  $1/r$  segundos.
- No entanto, este esquema tende a introduzir grandes atrasos quando o tráfego é de rajada.
- Neste caso, é preferível admitir  $W$  pacotes em cada  $W/r$  segundos.

## Controlo de taxas de transmissão por janelas (II)

Se foi atribuído a um determinado fluxo: (i) uma taxa de transmissão de  $r$  pacotes/segundo e (ii) uma janela de  $W$  pacotes, então:

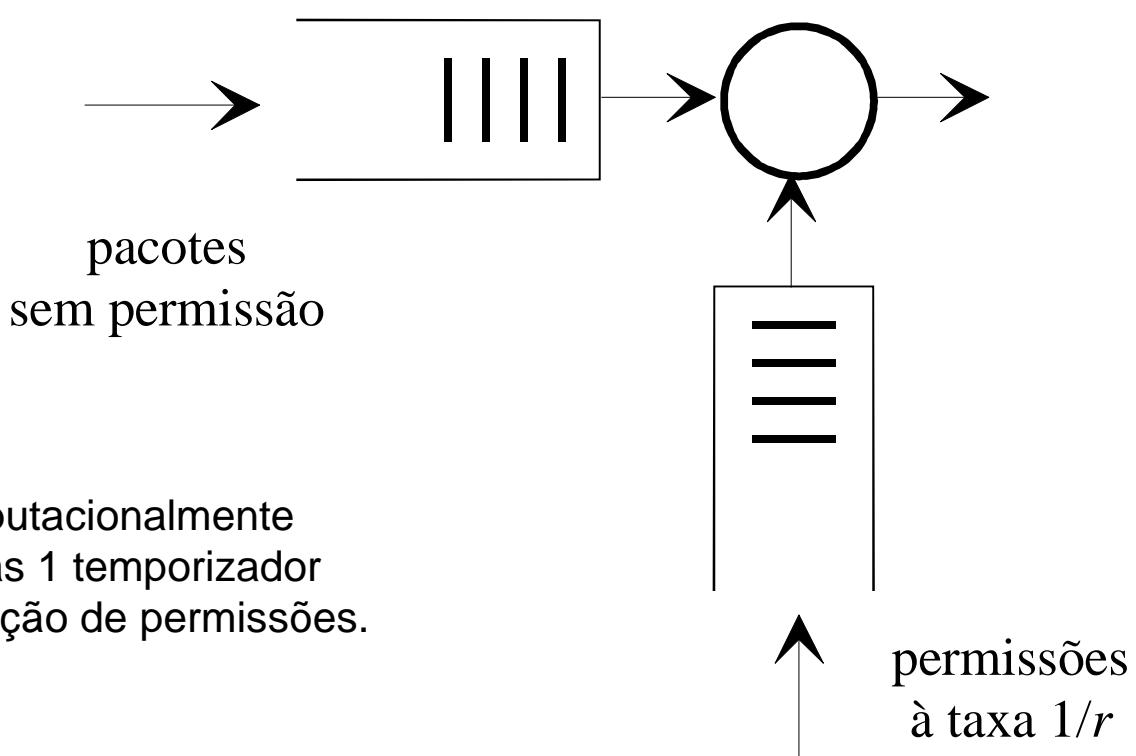
1. O emissor mantém um contador  $x$  que indica, em cada instante, o número de pacotes dessa janela que ainda pode ser transmitido ( $x$  é inicializado a  $W$ ).
2. Sempre que um pacote é transmitido, o contador  $x$  é decrementado e passados  $W/r$  segundos é novamente incrementado (exige um temporizador por cada pacote transmitido).
3. Os pacotes só são enviados para a rede se  $x > 0$  (o número máximo de temporizadores é  $W$ ).

**Nota:** O método do controlo de fluxo por janelas extremo-a-extremo é semelhante a este com a diferença apenas de que o contador é incrementado quando são recebidas as permissões.

**Desvantagem:** este método é computacionalmente pesado pois exige  $W$  temporizadores simultâneos

# Controlo de taxas de transmissão por *leaky bucket*

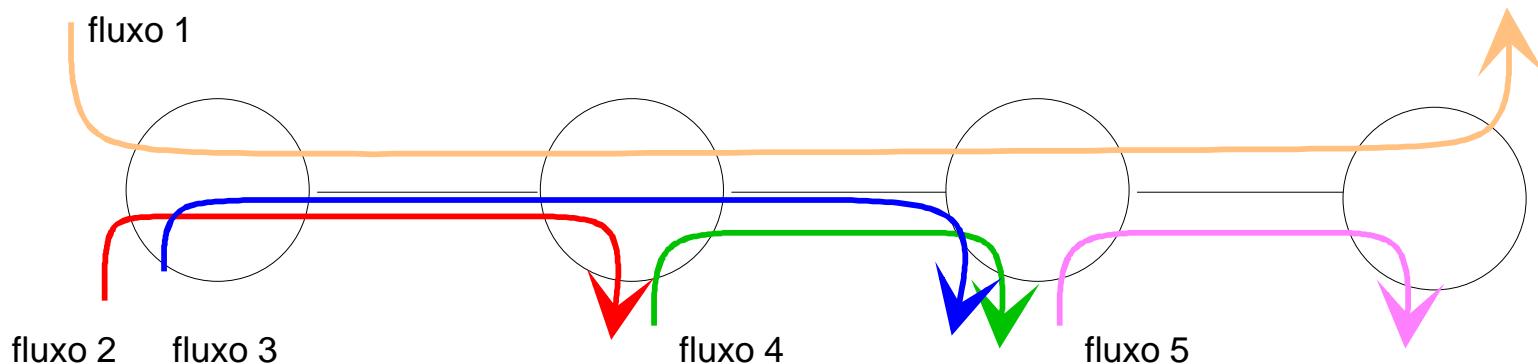
- Neste método, o contador é incrementado periodicamente em cada  $1/r$  segundos, até um máximo de  $W$  pacotes.
- O método pode ser visto da seguinte forma (modelo *leaky bucket*):
  - existe uma fila de espera de pacotes e uma fila de espera de permissões, com capacidade para  $W$  permissões;
  - é gerada uma nova permissão em cada  $1/r$  segundos;
  - os pacotes só são transmitidos quando existe uma permissão disponível na fila de espera respetiva.



**Vantagem:** este método é computacionalmente menos pesado pois exige apenas 1 temporizador para definir os instantes de geração de permissões.

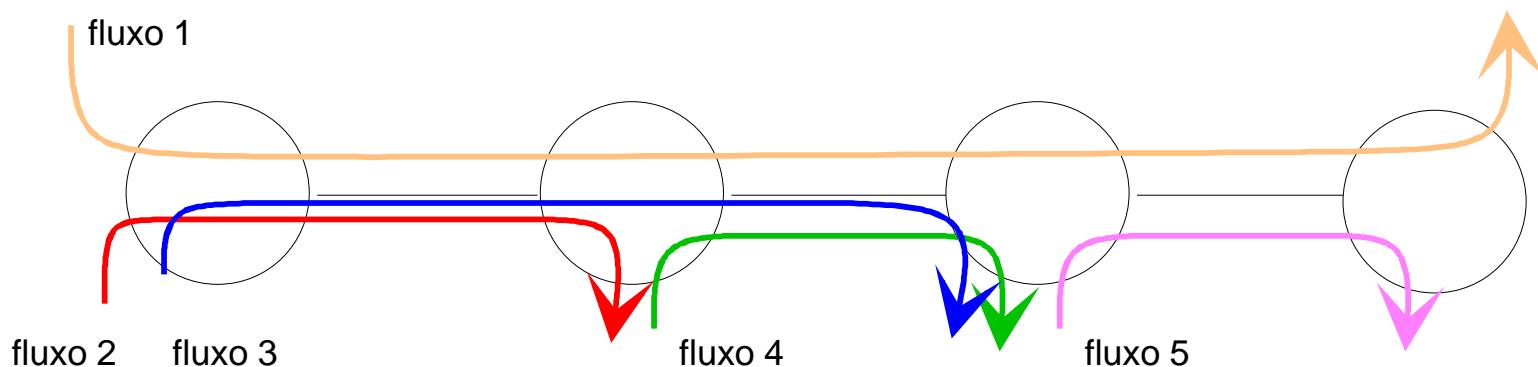
# Atribuição de taxas de transmissão

- Considere a rede da figura em que as ligações têm todas capacidade para 120 pacotes/s.
- Uma solução equilibrada (*fair*) seria atribuir a todos os fluxos uma taxa de  $1/3 \times 120 = 40$  pacotes/s.
- No entanto, não faz sentido restringir a taxa do fluxo 5 a 40 pacotes/s, pois este fluxo pode usar 80 pacotes/s sem prejudicar os fluxos 1, 2, 3 e 4.



# Equidade do tipo *max-min*

- Surge assim o conceito de equidade do tipo max-min (*max-min fairness*).
- Segundo este princípio, maximizam-se os recursos atribuídos aos fluxos que podem usar menos recursos.
- Uma forma alternativa de formular este princípio:
  - Maximizam-se as taxas atribuídas a cada fluxo, respeitando a restrição segundo a qual um incremento na atribuição ao fluxo  $i$  não conduz a uma diminuição da taxa atribuída a qualquer outro fluxo cuja taxa seja menor ou igual que a de  $i$ .



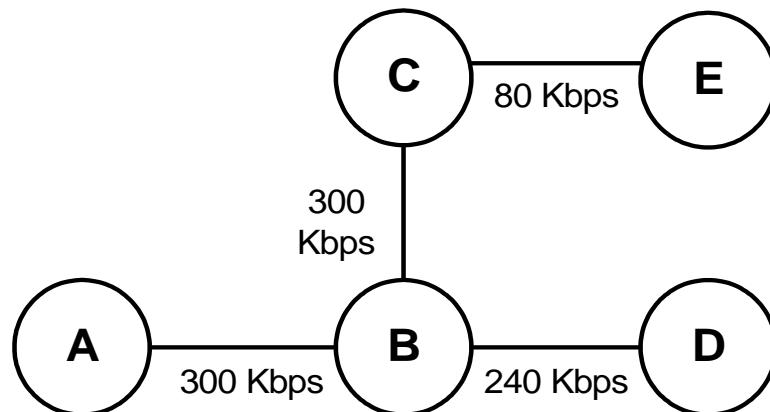
## Exemplo 2

Considere a rede com comutação de pacotes da figura.

A rede suporta 5 fluxos de pacotes: de A para B, de A para C, de A para D, de B para D e de B para E.

A rede permite controlar a taxa de transmissão máxima de cada fluxo através de um qualquer mecanismo adequado.

Calcular que taxas de transmissão máxima se devem atribuir a cada fluxo segundo o princípio de equidade do tipo *max-min*.



## Exemplo 2 - resolução

5 fluxos de pacotes:

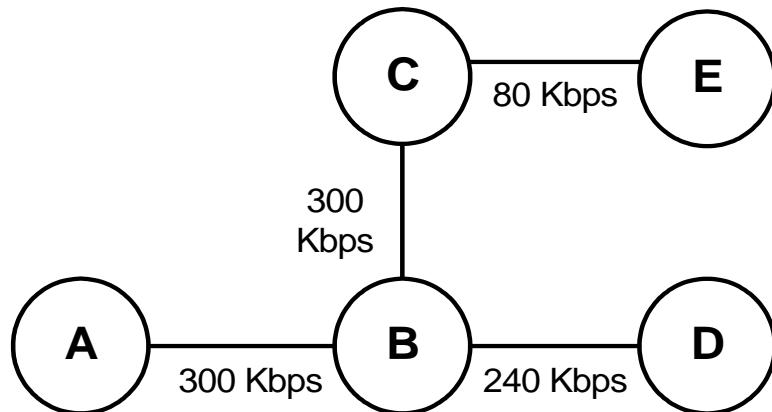
de A para B

de A para C

de A para D

de B para D

de B para E



1<sup>a</sup> iteração:

- a ligação AB atribui  $300/3 = 100$  Kbps por fluxo
- a ligação BC atribui  $300/2 = 150$  Kbps por fluxo
- a ligação BD atribui  $240/2 = 120$  Kbps por fluxo
- a ligação CE atribui  $80/1 = 80$  Kbps por fluxo

O menor valor é o da ligação CE e é atribuído 80 Kbps ao fluxo B-E.

2<sup>a</sup> iteração:

- a ligação AB atribui  $300/3 = 100$  Kbps por fluxo
- a ligação BC atribui  $(300-80)/1 = 220$  Kbps por fluxo
- a ligação BD atribui  $240/2 = 120$  Kbps por fluxo

O menor valor é o da ligação AB e é atribuído 100 Kbps aos fluxos A-B, A-C e A-D.

3<sup>a</sup> iteração:

- a ligação BD atribui  $(240-100)/1 = 140$  Kbps por fluxo

É atribuído 140 Kbps ao fluxo B-D.



# **Escalonamento em Redes com Comutação de Pacotes**

Desempenho e Dimensionamento de Redes

Prof. Amaro de Sousa ([asou@ua.pt](mailto: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  $1, 2, \dots, n$  com necessidades  $x_1, x_2, \dots, x_n$  e ordenados pelas suas necessidades ( $x_1 \leq x_2 \dots \leq x_n$ );
- 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  $d$  ao fluxos  $1, 2, \dots, n$ ;
  - 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  $2, 3, \dots, n$ ;
  - 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

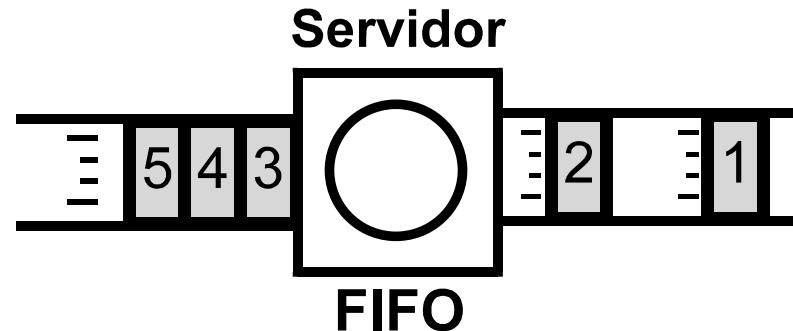
As disciplinas de escalonamento podem classificar-se em disciplinas com e sem conservação de trabalho (*work conserving*):

- 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:

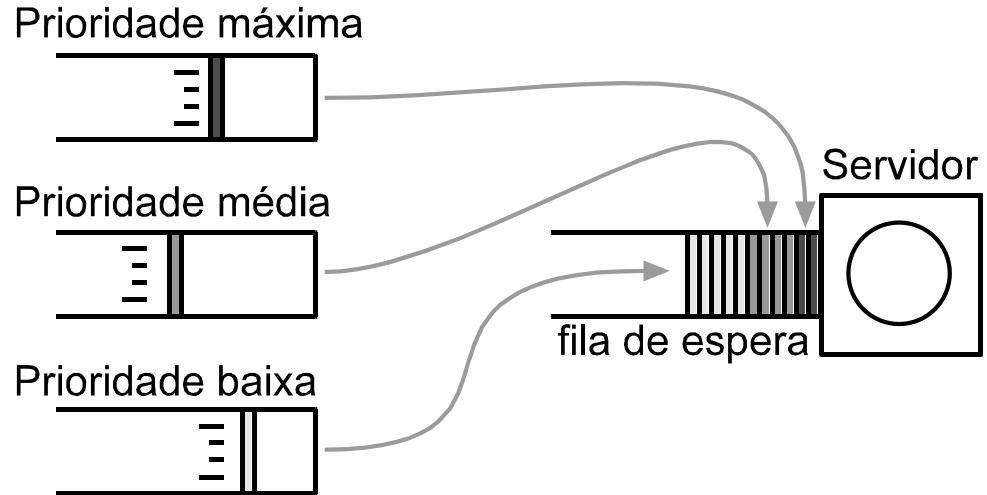
- (1) por ordem de chegada (FIFO),
- (2) com base em prioridade estrita,
- (3) de uma forma rotativa (RR, WRR, DRR),
- (4) por aproximação ao sistema GPS (WFQ, SCFQ).

# First-In-First-Out (FIFO)



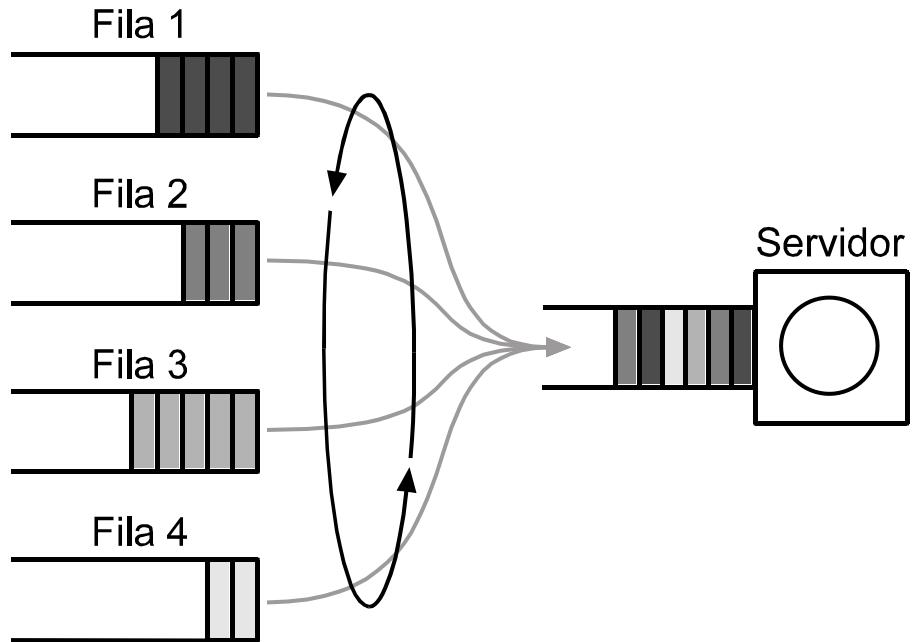
- Os pacotes de todos os fluxos são transmitidos por ordem de chegada.
- Não envolve processamento de ordenação nem de classificação de pacotes.
- Não permite diferenciação de qualidade de serviço (o atraso médio em fila de espera é igual para todos os fluxos).
- Quando a fila de espera não está vazia,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

## Prioridade Estrita



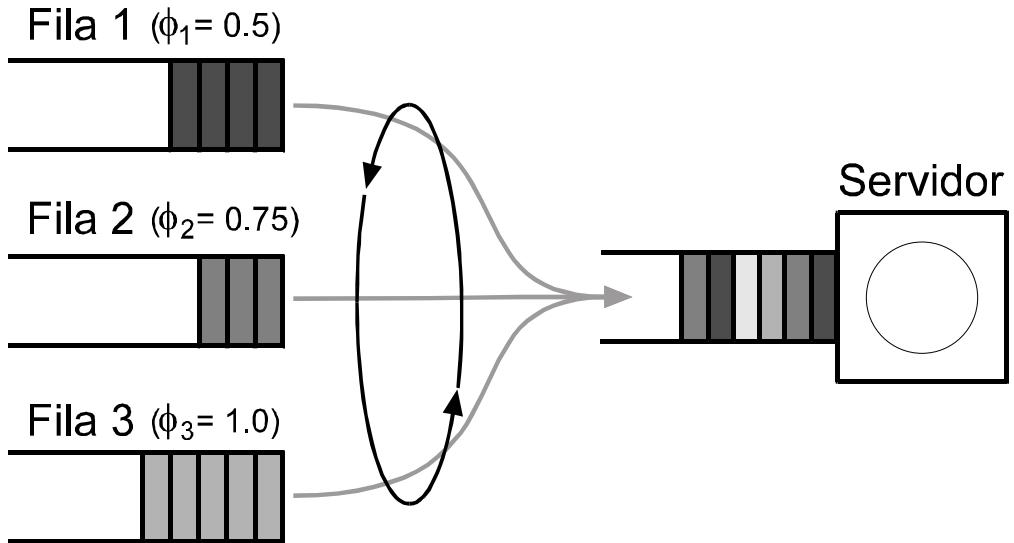
- Os pacotes classificados como de maior prioridade são sempre transmitidos antes dos pacotes de menor prioridade (os pacotes com a mesma prioridade são transmitidos com a disciplina FIFO).
- Não envolve processamento de ordenação.
- Envolve classificação dos pacotes de acordo com a prioridade.
- Permite 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)



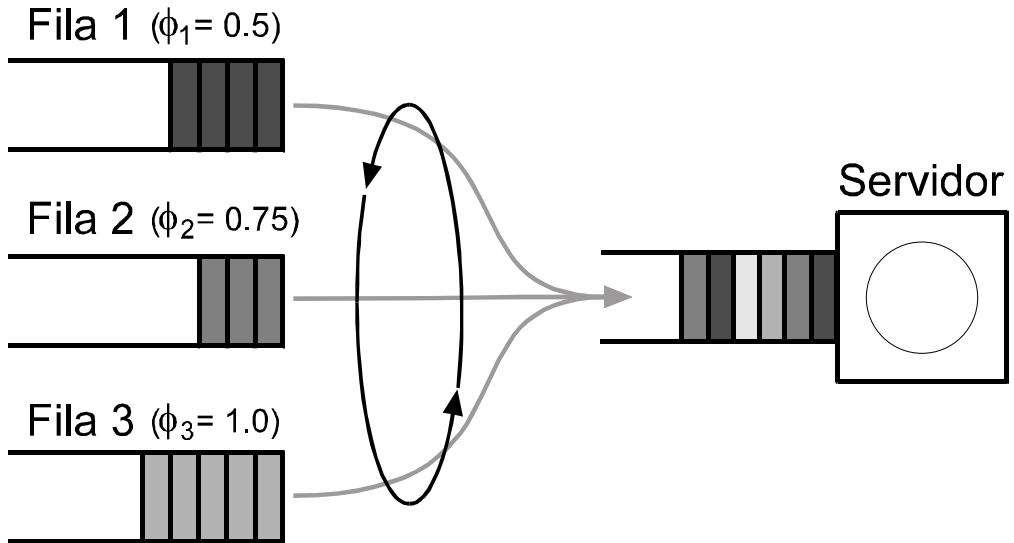
- Existe uma fila por fluxo de pacotes e o algoritmo seleciona um pacote de cada fila (não vazia) de forma rotativa.
- Não permite diferenciação de qualidade de serviço.
- Ao contrário do FIFO, serve o mesmo número de pacotes de todos os fluxos ativos.
- Beneficia os fluxos de pacotes maiores.
- Protege 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  $\phi_i$  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:

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

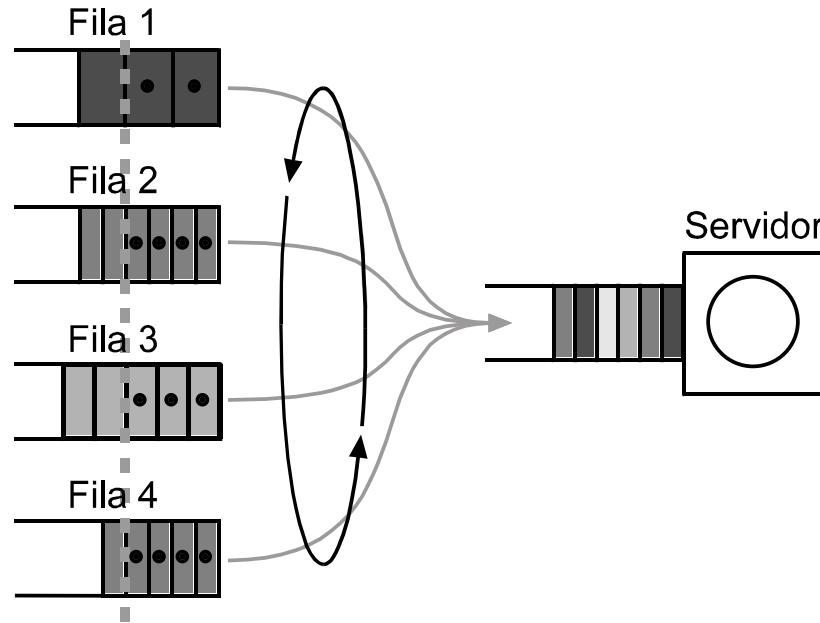
$$\varphi_2 = 0.75/500 = 3/2000 = 9/6000$$

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

Número de pacotes por ciclo:

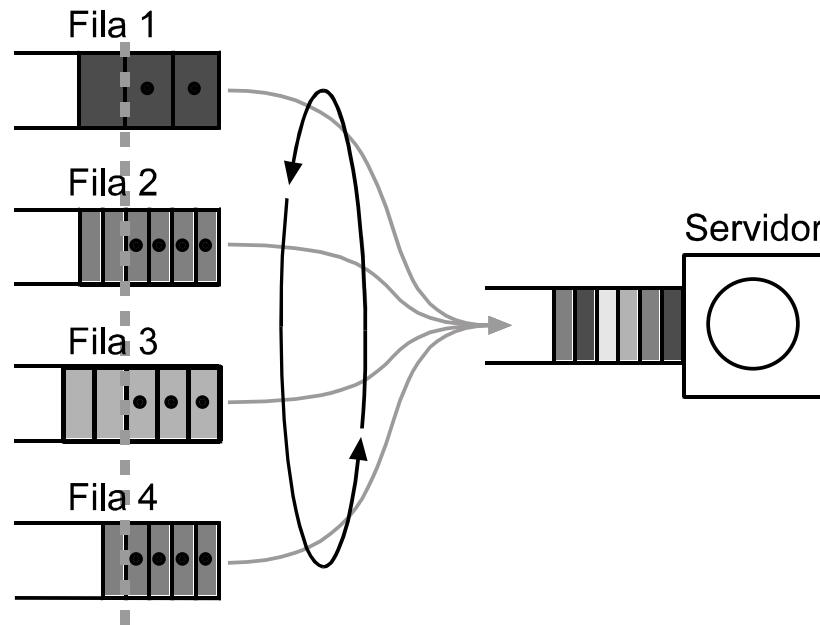
$$\Phi_1 = 60, \Phi_2 = 9, \Phi_3 = 4$$

## Deficit Round Robin (DRR)

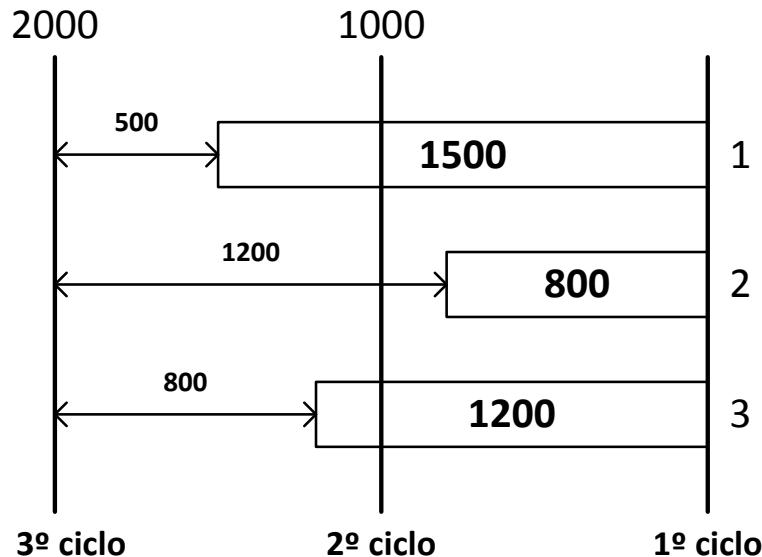


- Em cada ciclo, o algoritmo serve uma quantidade de bytes até um valor máximo designado por limiar.
- A diferença entre a quantidade servida e o limiar é contabilizada em forma de crédito para o ciclo seguinte.
- Quando uma fila está vazia, o crédito respetivo é colocado a zero.
- Se se considerarem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

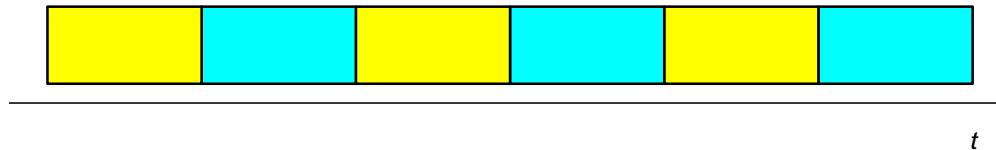
- a) 1 é servido, obtém crédito de 500
- b) 2 fica com crédito a 0
- c) 3 é servido, obtém crédito de 800

# Generalized Processor Sharing (GPS)

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 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  $\phi_i$  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:

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

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

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

# 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:

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

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.

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

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

## 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:

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

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 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:

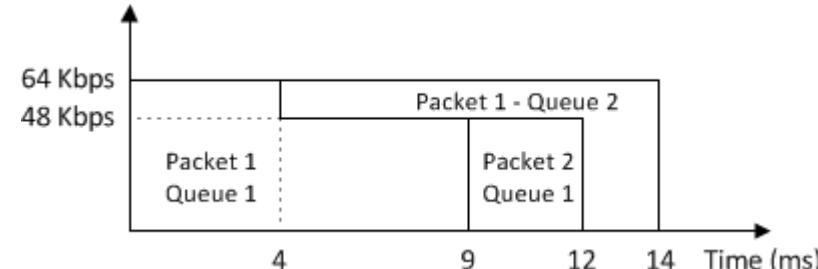
- pacote 1 à fila 1 com 62 Bytes em  $t = 0$ ,
- pacote 1 à fila 2 com 32 Bytes em  $t = 4$  ms e
- pacote 2 à fila 1 com 18 Bytes em  $t = 6$  ms.

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.

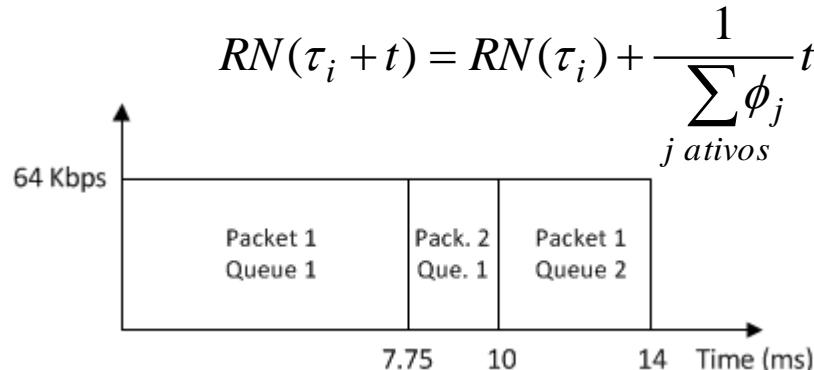


- O pacote 1 da fila 1 é servido inicialmente a 64 Kb/s. Em  $t = 4$  ms, foram servidos  $(64\text{Kb/s}) \times (4\text{ms}) = 256$  bits = 32 Bytes do pacote 1 da fila 1. A partir daqui, a fila 1 é servida a  $(3/4) \times 64$  Kb/s = 48 Kb/s e a fila 2 a  $(1/4) \times 64$  Kb/s = 16 Kb/s.
- Com estas taxas, o pacote 1 da fila 1 demora  $(30 \times 8)/(48\text{Kb/s}) = 5$  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.

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

## 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.

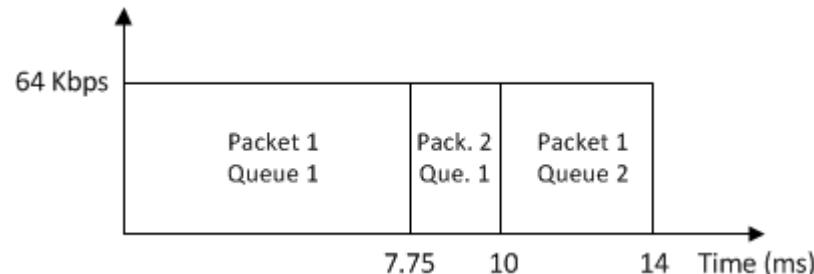


- Em  $t = 0$  ms,  $RN = 0$  e  $FN_{1,1} = 0 + (62 \times 8)/64000/3 = 2.58 \times 10^{-3}$ . O pacote 1 da fila 1 é transmitido em  $(62 \times 8)/(64 \text{Kb/s}) = 7.75$  ms. Assim, o pacote 1 da fila 1 termina a sua transmissão em  $t = 0 + 7.75 = 7.75$  ms.
- Em  $t = 4$  ms :  $RN = 0 + (4 \times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

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

## Exemplo 3 – resolução de (c)

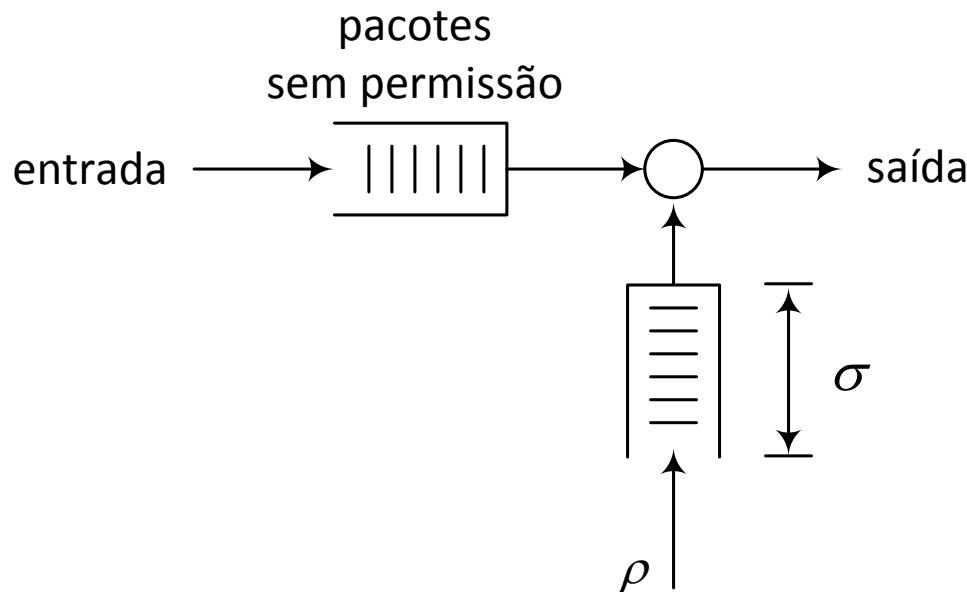
2 filas de espera com pesos  $\phi_1 = 3$  e  $\phi_2 = 1$  com uma ligação de 64 Kbps. Chegam: pacote 1 à fila 1 com 62 Bytes ( $t = 0$ ), pacote 1 à fila 2 com 32 Bytes ( $t = 4$  ms) e pacote 2 à fila 1 com 18 Bytes ( $t = 6$  ms). Escalonamento SCFQ.



- Em  $t = 0$  ms,  $FN_{1,1} = 0 + (62 \times 8)/3 = 165.3$ . 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 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:

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

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

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

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  $\sigma_i$  e  $\rho_i$  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:

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

$\Gamma$  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  $\sigma = 1000$  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 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!) 30

# 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*

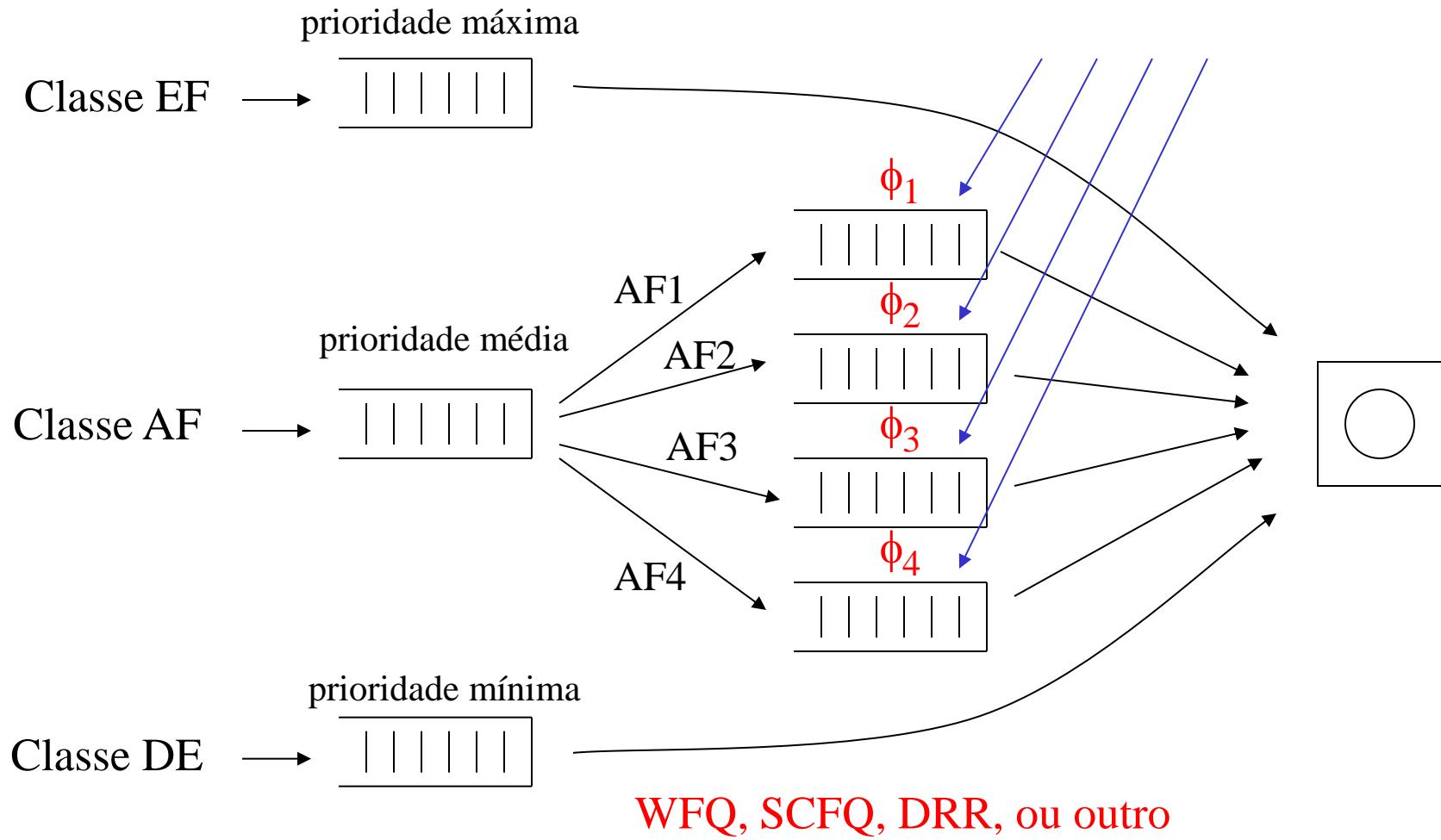
## *Classes de Serviço*

- *Default (DE)* → 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
  - em cada classe há 3 níveis de precedência para eliminação de pacotes em caso de congestionamento

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

# Exemplo de Implementação

WRED (Weighted Random Early Discard)





# **Traffic Engineering in Packet Switched Networks**

Desempenho e Dimensionamento de Redes

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

DETI-UA, 2017/2018

# Traffic engineering

- Consider a network composed by a set of point-to-point links and supporting a set of flows  $S$ , such that all flows have the same average packet size.
  - The network is modelled by a graph  $G=(N,A)$  where  $N$  is the set of network nodes and  $A$  is the set of network links. The element  $(i,j) \in A$  represents the link between nodes  $i \in N$  and  $j \in N$  from  $i$  to  $j$  whose capacity is  $c_{ij}$  (in packets/second).
  - Each flow  $s \in S$  is defined by its origin node  $o_s$ , destination node  $d_s$  and average packet arrival rate  $\lambda_s$ .
  - For each flow  $s \in S$ ,  $P_s$  is the set of the routing paths existent in graph  $G$  from its origin node  $o_s$  to its destination node  $d_s$ .

The traffic engineering task is the task of choosing for each flow  $s \in S$  the percentage of its average arriving rate  $\lambda_s$  that must be routed through each of the routing paths of  $P_s$ .

# Traffic engineering with single path routing

- In the single path routing, each flow must be routed through one single path (no flow bifurcation is allowed).
- Moreover, it is also usual to require symmetrical routing, *i.e.*, the routing path from a node  $j \in N$  to a node  $i \in N$  must use the same links as the routing path from node  $i \in N$  to node  $j \in N$ .

Consider a binary variable  $x_{sp}$  associated to each flow  $s \in S$  and each routing path  $p \in P_s$  that, when is 1, indicates that flow  $s$  is routed through path  $p$ .

Any traffic engineering solution with single path routing must be compliant with the following constraints:

- For each flow  $s \in S$ , one of its associated variables  $x_{sp}$  must be 1 and all other associated variables must be 0.
- For each link  $(i,j) \in A$ , the sum of the average arriving rates  $\lambda_s$  of all flows routed through it cannot be higher than its capacity  $c_{ij}$ .

# Traffic engineering objectives

The traffic engineering task aims to:

- optimize one (or more) parameter(s) related with either the network performance and/or the quality of service provided by the network;
- optionally, guarantee (maximum or minimum) bounds for other parameters related with either the network performance and/or the quality of service provided by the network.

Examples of optimization parameters:

- global average packet delay (using, for example, the Kleinrock approximation)
- worst case average packet delay (to maximize the fairness among all flows)
- worst link load (to maximize the robustness of the network to unpredictable traffic growth)

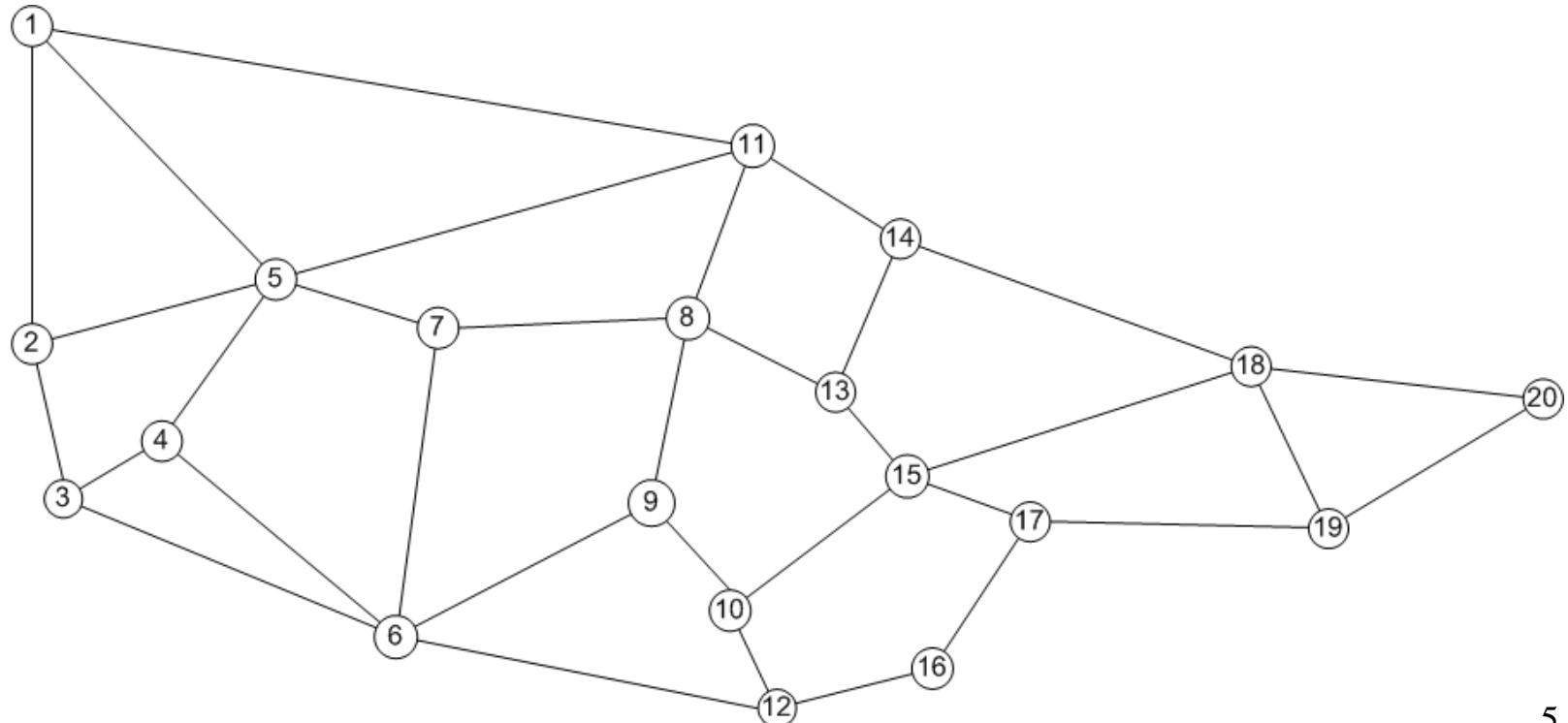
The best traffic engineering solution depends on the objective function defined by the network operator. Different objective functions might be conflicting

- for example, the solution that minimizes the global average delay can have a bad worst link load and vice-versa.

## Example - network

Consider the following network with 20 routers and 33 links where all links have a capacity of 1 Gbps.

The length of the links varies between 88 km (between nodes 10 and 12) and 759 km (between nodes 1 and 11) and the link propagation delay is determined by the light speed ( $3 \times 10^8$  meters/s).



# Example – flow matrix

Consider the following flow matrix (values in Mbps) where all flows have an average packet size of 1000 Bytes:

0.0	47.7	64.4	13.6	10.6	45.5	10.6	12.9	9.8	9.8	13.6	11.4	11.4	41.7	12.9	10.6	9.1	12.9	11.4	22.0
47.0	0.0	68.2	32.6	33.3	189.4	59.8	47.0	49.2	38.6	59.1	53.0	38.6	212.1	31.8	48.5	40.9	43.9	54.5	74.2
65.9	69.7	0.0	8.3	13.6	53.0	13.6	14.4	18.9	14.4	11.4	36.4	12.9	63.6	13.6	14.4	11.4	13.6	15.9	22.7
13.6	46.2	13.6	0.0	12.9	31.1	14.4	12.1	11.4	12.9	31.1	12.1	9.1	31.8	11.4	10.6	14.4	12.9	17.4	24.2
7.6	31.8	10.6	14.4	0.0	55.3	11.4	9.1	9.8	12.9	36.4	11.4	12.9	46.2	12.9	7.6	12.1	15.9	16.7	28.0
52.3	174.2	53.8	55.3	38.6	0.0	39.4	47.0	41.7	40.9	53.8	44.7	42.4	212.1	40.9	71.2	62.9	46.2	56.8	72.0
13.6	32.6	11.4	11.4	12.9	54.5	0.0	7.6	12.1	12.9	12.1	14.4	56.8	55.3	9.8	9.1	13.6	15.9	9.8	21.2
13.6	47.0	14.4	11.4	9.8	37.9	10.6	0.0	12.1	9.1	11.4	13.6	12.1	57.6	9.8	10.6	9.8	17.4	12.9	34.1
9.8	33.3	16.7	12.1	14.4	38.6	12.9	9.8	0.0	11.4	13.6	9.1	13.6	56.1	11.4	13.6	12.1	15.2	18.9	18.9
12.9	53.0	10.6	9.8	11.4	46.2	13.6	10.6	10.6	0.0	9.1	9.8	11.4	42.4	10.6	11.4	10.6	15.9	10.6	25.8
9.1	36.4	9.8	31.1	33.3	40.9	9.8	10.6	12.1	14.4	0.0	9.8	10.6	47.7	9.1	11.4	8.3	9.8	12.9	32.6
10.6	61.4	35.6	9.8	9.8	59.8	14.4	8.3	8.3	10.6	12.1	0.0	9.8	33.3	9.8	28.0	9.8	9.1	14.4	25.0
10.6	32.6	9.1	12.1	9.1	35.6	59.8	10.6	9.8	14.4	9.8	13.6	0.0	41.7	11.4	12.9	13.6	15.9	16.7	40.9
40.9	181.8	49.2	56.1	42.4	189.4	55.3	64.4	57.6	31.8	31.8	33.3	46.2	0.0	40.9	57.6	40.2	48.5	51.5	69.7
12.1	37.1	10.6	9.8	10.6	37.1	8.3	14.4	8.3	10.6	9.8	12.1	11.4	47.7	0.0	11.4	10.6	10.6	9.1	28.8
10.6	47.7	9.8	11.4	11.4	44.7	9.8	11.4	10.6	9.1	9.1	12.9	9.1	56.8	14.4	0.0	11.4	9.1	12.9	30.3
13.6	34.1	10.6	10.6	13.6	55.3	12.1	12.9	9.8	11.4	10.6	12.9	9.1	44.7	10.6	9.1	0.0	10.6	9.8	20.5
13.6	40.9	11.4	9.8	18.9	40.9	11.4	18.2	13.6	18.9	12.9	10.6	17.4	40.9	11.4	12.9	9.8	0.0	34.1	24.2
7.6	49.2	18.9	15.9	12.9	53.0	12.1	9.8	15.9	13.6	15.2	10.6	18.9	47.0	12.9	9.8	9.8	30.3	0.0	18.9
23.5	68.2	26.5	28.8	34.1	65.9	25.8	30.3	15.9	22.7	30.3	28.0	34.1	65.2	34.1	25.8	24.2	21.2	15.9	0.0

## Example – one possible solution

One possible solution is to route each flow  $s \in S$  by the routing path with the shortest length (minimizing, in this way, the propagation delay of each flow).

Using the Kleinrock approximation, we obtain the following performance parameters:

Global average packet delay = 2.45 ms

Worst case average packet delay = 6.06 ms

Worst link load = 99.3%

However, it is possible to obtain better traffic engineering solutions through appropriate optimization algorithms.

# Example – optimal solutions

Minimization of the global average delay:

Global average delay = 2.34 ms

Worst case average delay = 5.23 ms

Worst link load = 89.0%

Minimization of the worst case average delay:

Global average delay = 2.34 ms

Worst case average delay = 5.21 ms

Worst link load = 93.6%

Minimization of the worst link load:

Global average delay = 2.75 ms

Worst case average delay = 8.63 ms

Worst link load = 69.9%

Results analysis:

- The 1st and 2nd solutions are similar and exhibit a high worst link load.
- The 3rd solution is much more robust to unpredictable traffic growth but it exhibits more delay unfairness between flows.

# Example – minimization of the energy consumption

Recently, energy aware routing has become a concern.

The technologies are evolving such that network links (and nodes) turn into a sleeping mode (with a very low energy consumption) if they have no traffic to route.

In the previous example, if we consider a link energy consumption proportional to the link capacity, the optimal solution is:

Global average delay = 3.12 ms

Worst case average delay = 10.54 ms

Worst link load = 82.4%

Number of active links = 26 out of 33

## Result analysis:

- We obtain an energy consumption reduction of 21.2% ( $=(33-26)/33$ ) at the cost of worst values on delay performance parameters.
- Worst link load is not a problem since the sleeping links can turn into active if traffic grows.

# Optimization algorithms

## Exact algorithms

- Based on mathematical models (for example, Integer Linear Programming)
- In the general case, computationally hard
- Theoretically, they are able to compute the optimal solutions
- Inefficient for large problem instances (they either take too long to even compute feasible solutions or finish due to out-of-memory)

## Heuristic algorithms

- Based on simple programming algorithms
- Easy to implement and quick to find solutions
- Do not guarantee optimality
- For larger runtimes, they find better solutions
- Efficient for large problem instances

# **Heuristic Algorithms**

# Heuristic method versus heuristic algorithm

**Heuristic method:** a generic approach to search for good solutions that can be applied to any optimization problem.

**Heuristic algorithm:** an optimization algorithm that has resulted from applying an heuristic method to a particular optimization problem.

Many heuristic methods (usually, also the simplest ones) are based on two algorithmic strategies:

1. To build a solution from the scratch.

- Examples: *random, greedy, greedy randomized, etc...*

2. To get a better solution from a known solution.

- Examples: *local search, tabu search, simulated annealing, etc...*  
(we will address only the local search strategy).

**Building a solution from the scratch**

# Building a solution from the scratch (I)

## Random strategy:

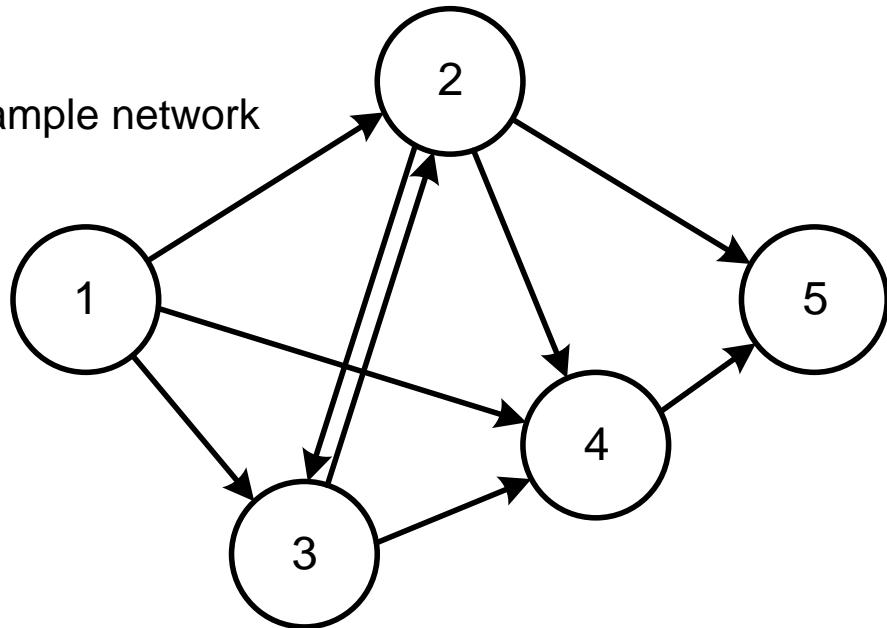
- We assign a random routing path  $p \in P_s$  to each flow  $s \in S$
- We obtain a slightly better performance if we consider higher probabilities to routing paths  $p \in P_s$  with “better characteristics”
  - For example, paths with a smaller number of links, paths containing links of larger capacity, etc...

## Greedy strategy:

- We start by considering the network without any routing path
- For each flow  $s \in S$ :
  - We assign the routing path  $p \in P_s$  that, together with the previous assigned routing paths, gives the best objective function value

# Building a solution from the scratch (II)

Example network



**From 1 to 5:**

- 1-2-5
- 1-4-5
- 1-2-4-5
- 1-3-4-5
- 1-3-2-5
- 1-3-2-4-5
- 1-2-3-4-5

**From 3 to 5:**

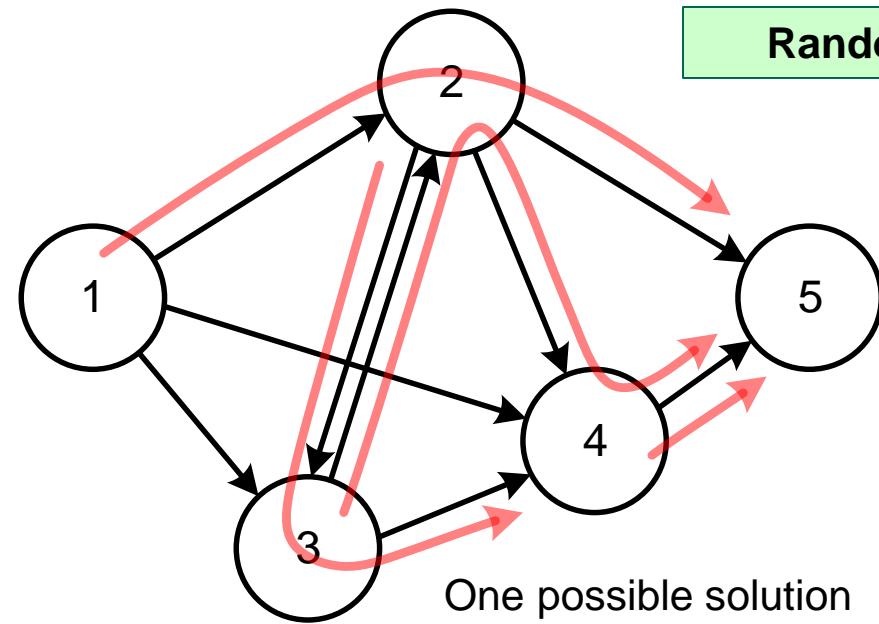
- 3-4-5
- 3-2-5
- 3-2-4-5

**From 2 to 4:**

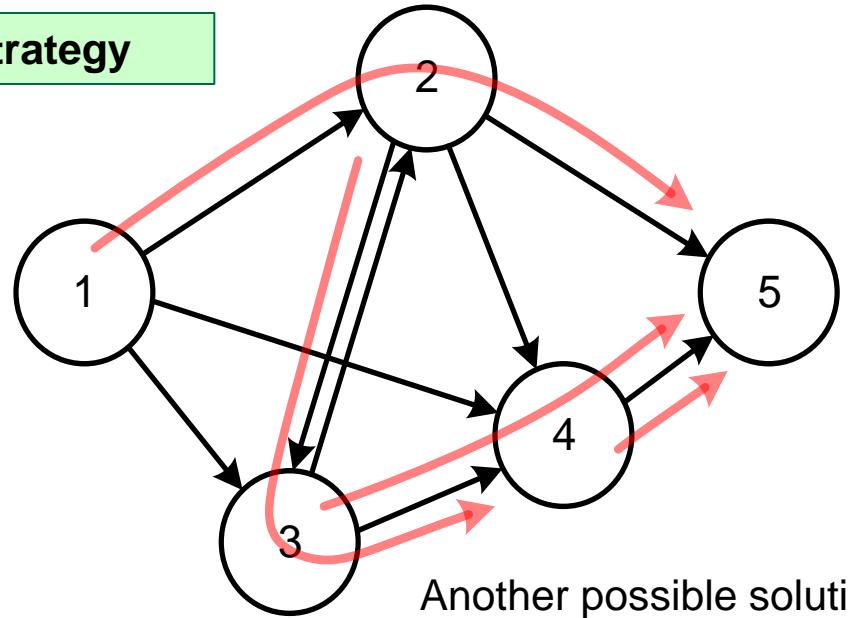
- 2-4
- 2-3-4

Candidate paths for each flow

**Random strategy**

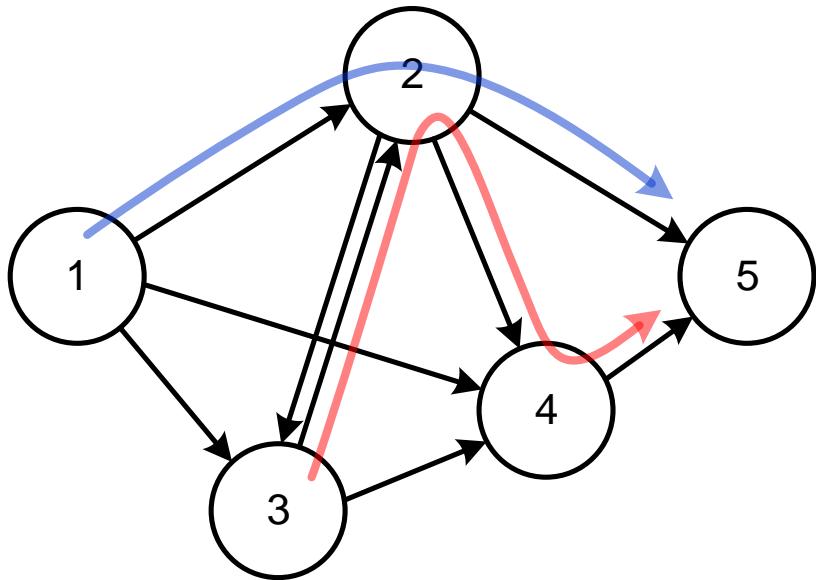


One possible solution

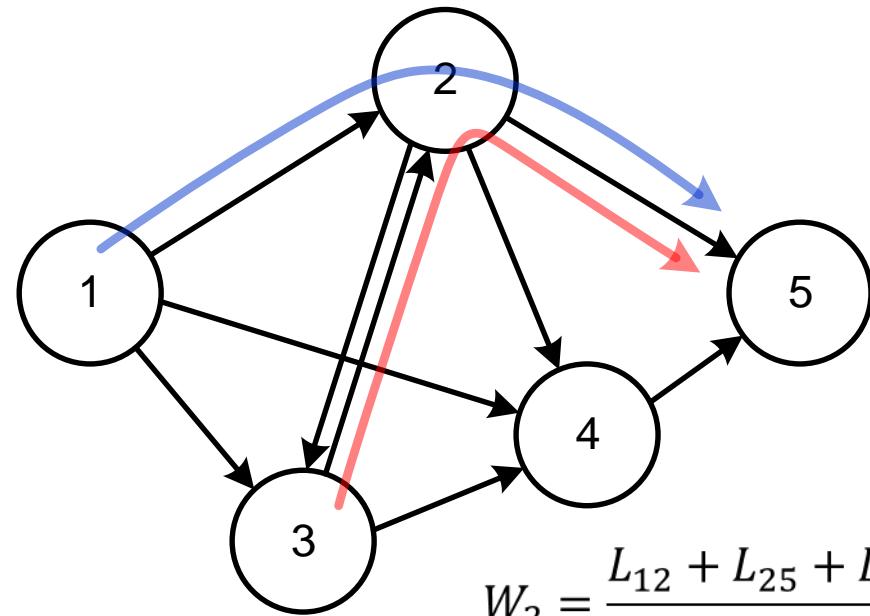


Another possible solution

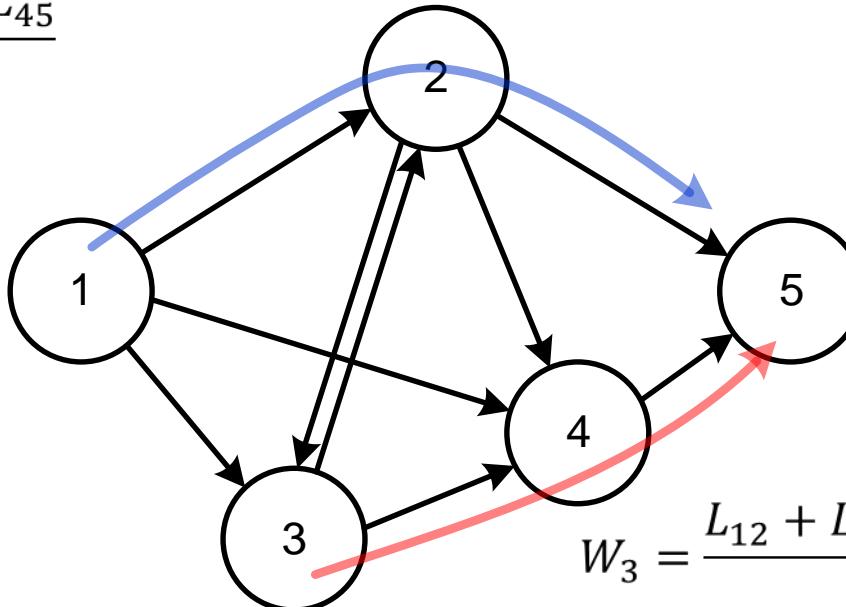
# Building a solution from the scratch (III)



$$W_1 = \frac{L_{12} + L_{25} + L_{32} + L_{24} + L_{45}}{\lambda_1 + \lambda_2}$$



$$W_2 = \frac{L_{12} + L_{25} + L_{32}}{\lambda_1 + \lambda_2}$$

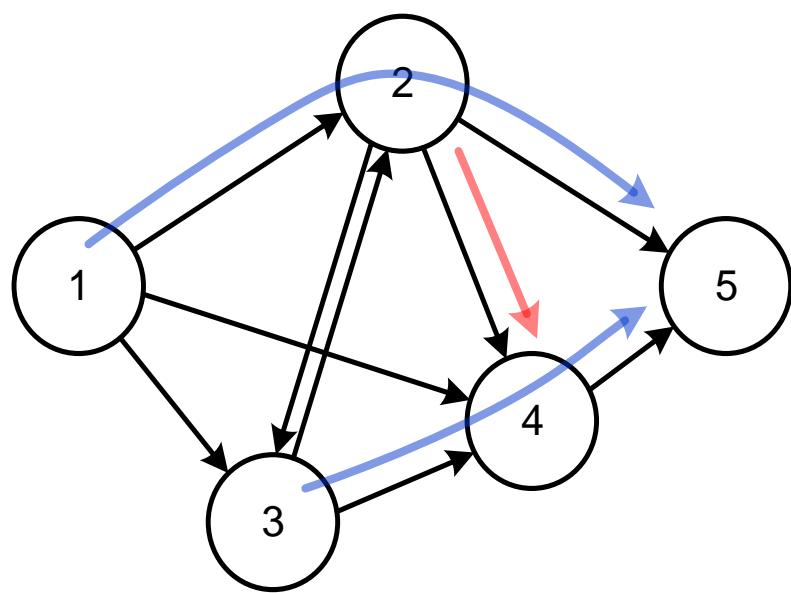


$$W_3 = \frac{L_{12} + L_{25} + L_{34} + L_{45}}{\lambda_1 + \lambda_2}$$

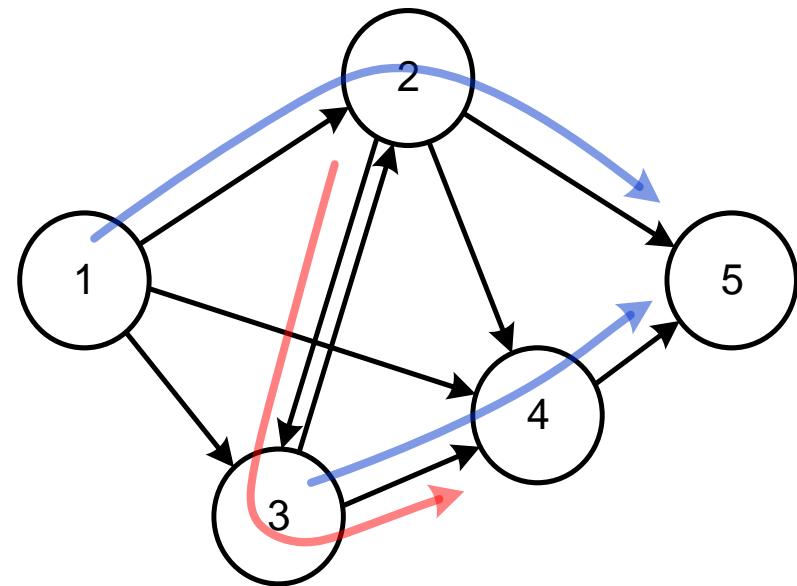
**Greedy strategy:**

- 1<sup>st</sup> flow (blue) already installed
- Try all candidate paths for 2<sup>nd</sup> flow (red) and select the solution with lowest global average delay

## Building a solution from the scratch (IV)



$$W_1 = \frac{L_{12} + L_{25} + L_{24} + L_{34} + L_{45}}{\lambda_1 + \lambda_2 + \lambda_3}$$



$$W_2 = \frac{L_{12} + L_{25} + L_{23} + L_{34} + L_{45}}{\lambda_1 + \lambda_2 + \lambda_3}$$

### Greedy strategy:

- 1<sup>st</sup> and 2<sup>nd</sup> flow (blue) already installed
- Try all candidate paths for 3<sup>rd</sup> flow (red) and select the solution with lowest global average delay

# Building a solution from the scratch (V)

## Greedy randomized strategy:

The aim is to obtain a different solution on different runs.

### First alternative:

- Choose a random order to compute the routing paths of the flows  $s \in S$

### Second alternative:

- For each flow  $s \in S$ , to select randomly one routing path among the best  $\alpha$  routing paths (*i.e.*, the  $\alpha$  paths that, together with the previous assigned routing paths, give the best objective function values)
  - $\alpha$  is a parameter of the algorithm

### Third alternative:

- To have a combination of the 2 previous alternatives

**Getting a better solution from a known solution**

# Getting a better solution from a known solution (I)

## Local search strategy – best neighbour move variant

In this strategy, we start by an initial solution and try to move to a better solution by making local changes. The moves are repeated until no possible local change produces any better solution.

This strategy works with the following steps:

1. For a given current solution (in the first iteration, the current solution is the initial known solution), we compute all neighbour solutions and select the best one.
2. If the best neighbour solution is better than the current solution, we move to this solution (*i.e.*, we set the current solution with the best neighbour solution) and go to step 1.
3. If not, we stop and the current solution is the final result (we say it is a local optimum solution).

# Getting a better solution from a known solution (II)

## Local search strategy – first neighbour move variant

If the evaluation of all neighbours of a current solution is too heady (either because a solution is hard to compute or because the neighbour set is very large), the previous variant might not be efficient.

This variant works with the following steps:

1. For a given current solution (in the first iteration, the current solution is the initial solution), we compute the neighbour solutions until we find a solution better than the current one.
2. If a neighbour solution better than the current one exists, we set the current solution with the computed neighbour solution and go to step 1.
3. If not, we stop and the current solution is the final result.

Usually, it is more efficient to use the best neighbour move variant, although in some problems it requires a careful definition of the neighbour set.

# Getting a better solution from a known solution (III)

## Local search strategy – defining the set of neighbour solutions

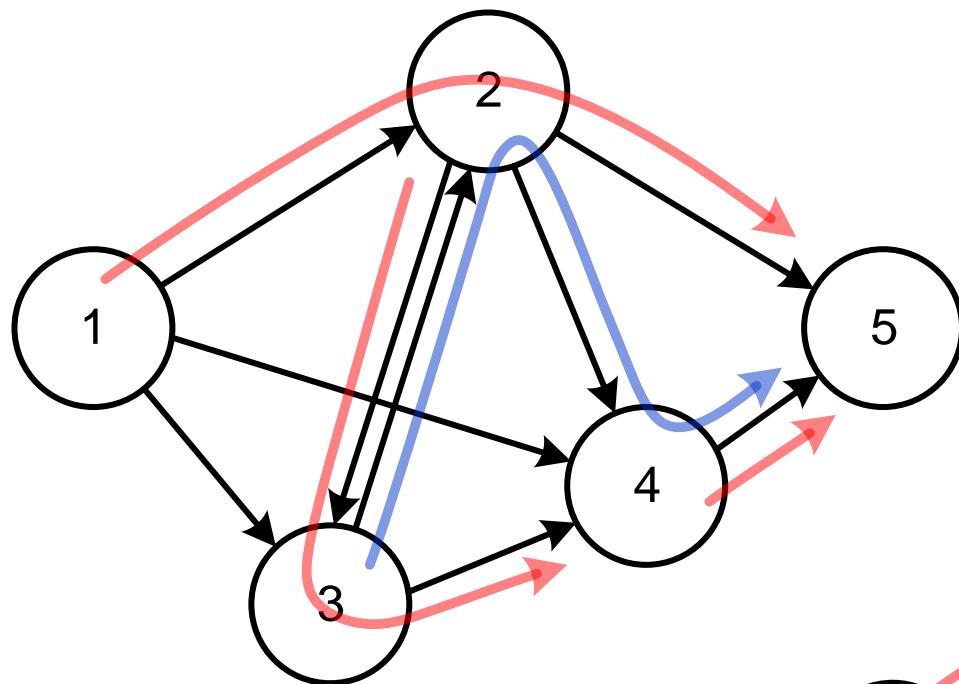
- The set of neighbour solutions (of a given solution) is problem dependent.
- The neighbour set must be carefully defined in order to allow the algorithm to compute all neighbour solutions in reasonable runtime.

In traffic engineering of telecommunication networks, the neighbour set is usually defined as follows:

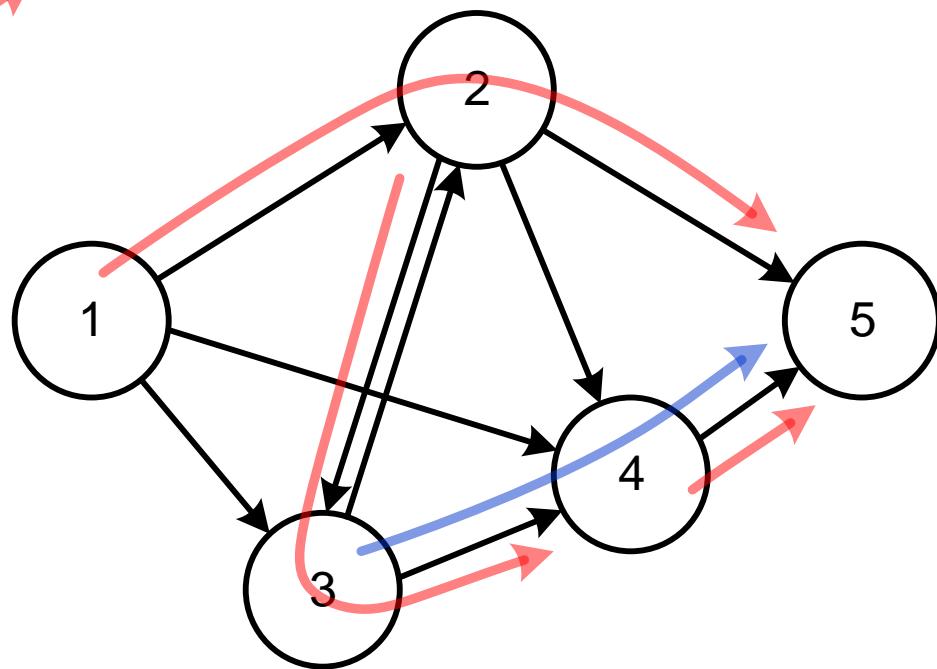
- For a current solution (that defines a routing path  $p \in P_s$  for each flow  $s \in S$ ), a neighbour solution is a solution that differs from the current one in the routing path of a single flow.
- When the set of routing paths  $P_s$  of each flow  $s \in S$  is given, the number of neighbour solutions is:

$$\sum_{s \in S} (|P_s| - 1)$$

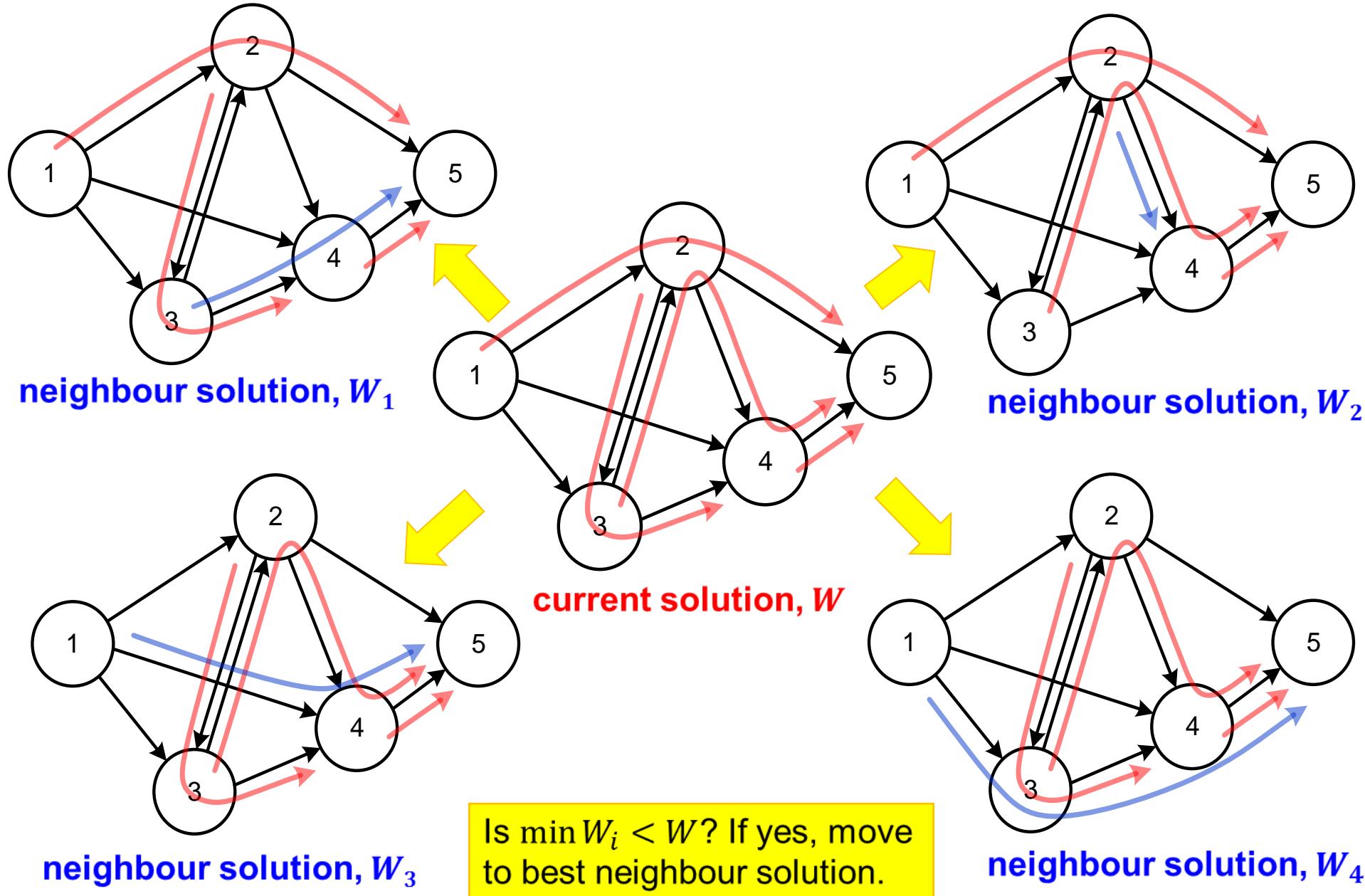
# Getting a better solution from a known solution (IV)



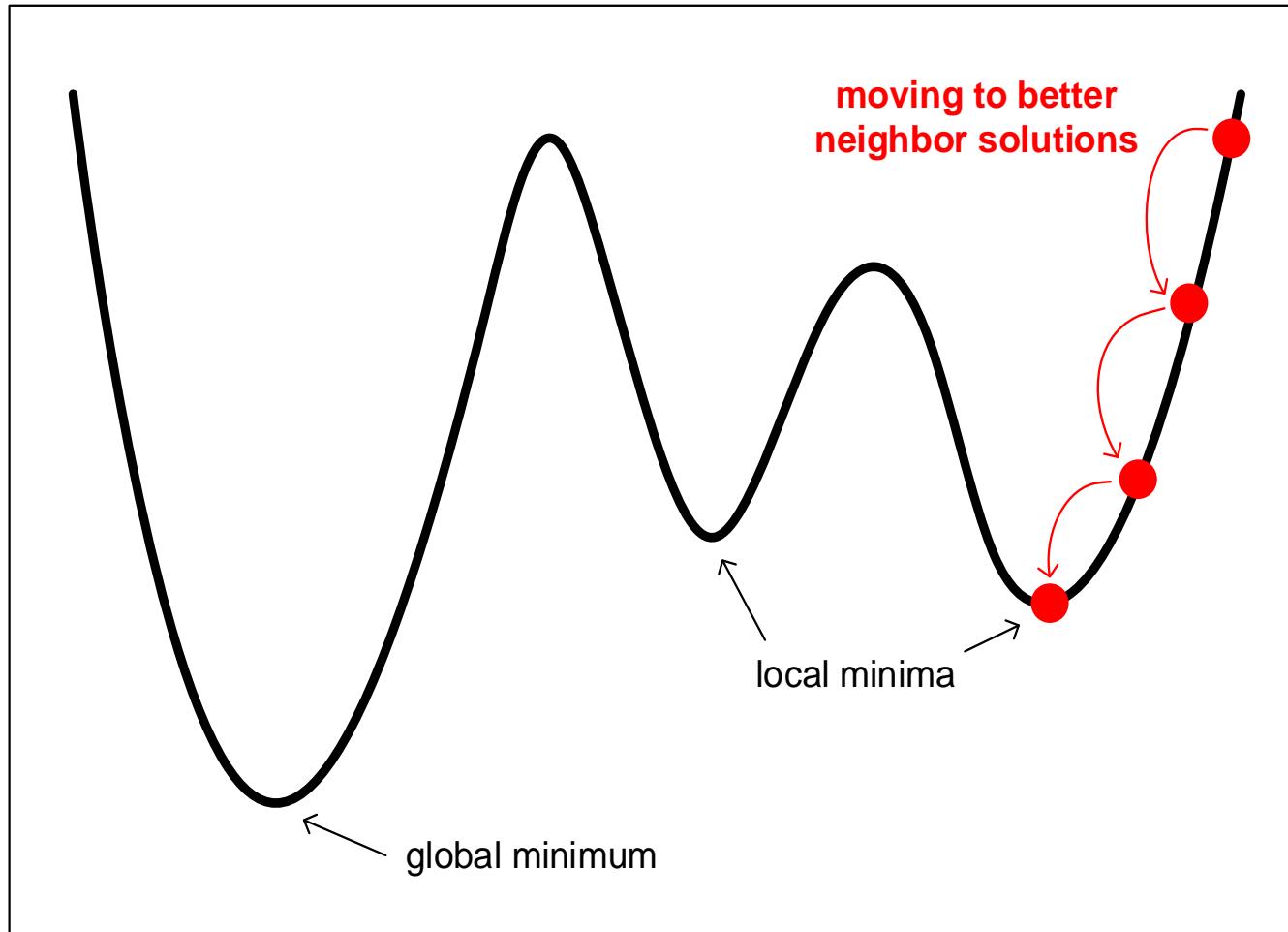
Neighbour solutions



# Getting a better solution from a known solution ( $V$ )



# Getting a better solution from a known solution (VI)



# **Multi Start Local Search Heuristic**

# Multi Start Local Search Heuristic (I)

- This heuristic combines the two algorithmic strategies:
  1. to build a solution from the scratch
  2. to get a better solution from a known solution
- In a problem aiming to minimize function  $F(z)$ , it works as follows:

$F_{best} = +\infty$

**repeat**

$x = BuildSolution()$

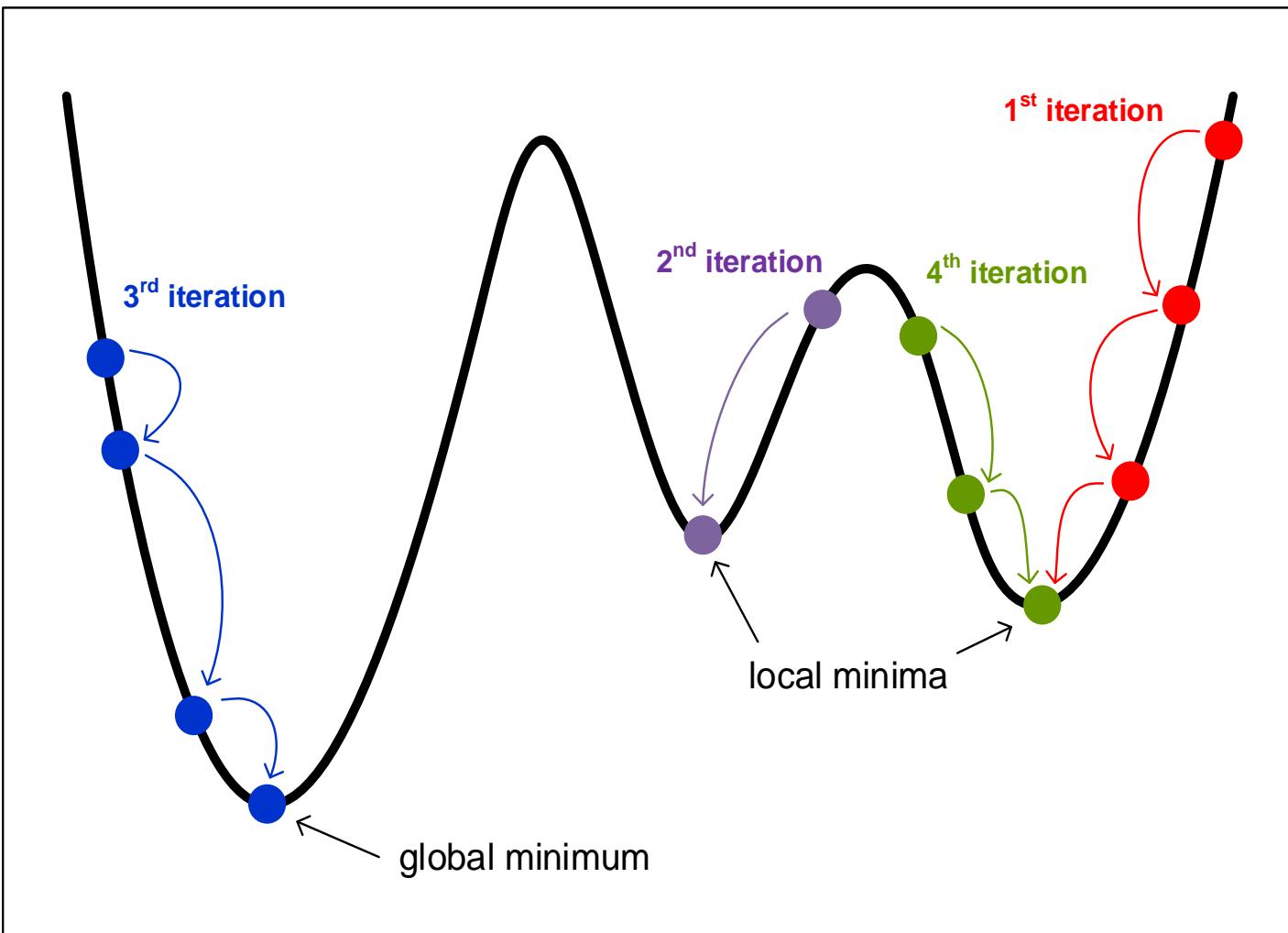
$z = LocalSearch(x)$

**if**  $F(z) < F_{best}$  **then**  $x_{best} = z$  **e**  $F_{best} = F(z)$

**until** Stopping Criteria is met

- Examples of Stopping Criteria:
  - Run a predefined number of iterations
  - Run until  $F_{best}$  not improving a predefined number of iterations

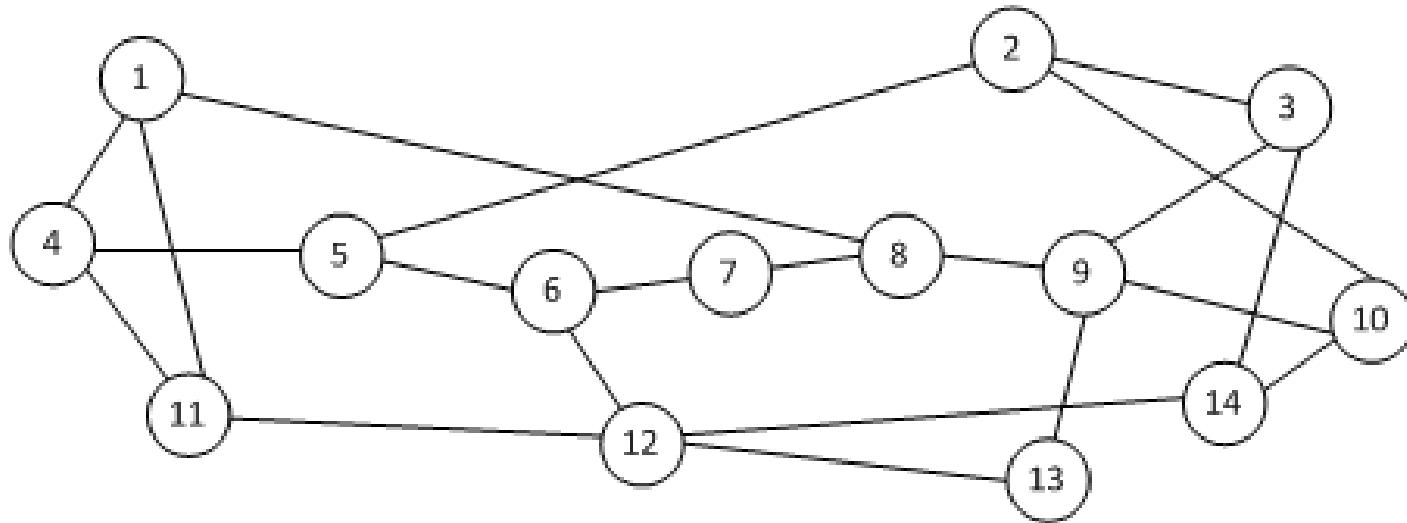
# Multi Start Local Search Heuristic (II)



# **Multi Start Local Search Heuristic without Sets of Routing Paths (*i.e.*, non defined $P_s$ sets)**

## Non defined $P_s$ sets (I)

- The sets  $P_s$  (with the available routing paths for each flow  $s \in S$ ) might be too large.
- For example, in this network, there are more than 200 possible routing paths for any pair of network nodes.



- If  $|P_s|$  is large, the neighbour set of a local search algorithm becomes too large and the performance of the algorithm is penalized.

## **Non defined $P_s$ sets (II)**

- In these cases, an alternative approach is to use a minimum cost path algorithm (like the Dijkstra's algorithm).
- When a routing path is to be assigned to a flow:
  1. We first assign an “appropriate cost value” to each link.
  2. We run Dijkstra's algorithm to determine the minimum cost path.
- The “appropriate cost values” must be carefully selected and they should depend on the objective to be optimized.

Examples:

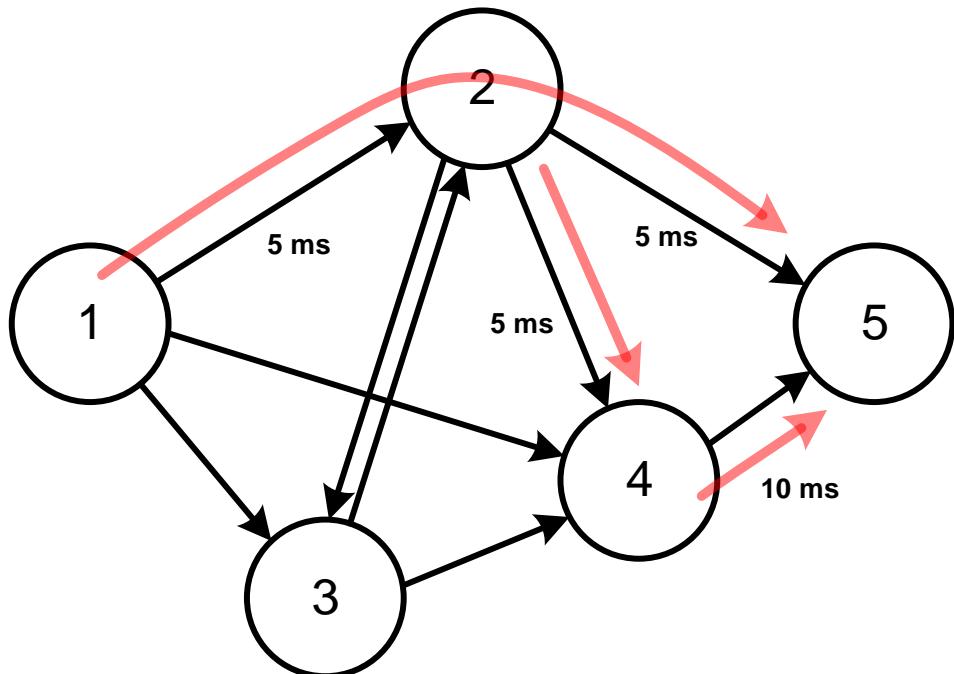
- If the objective function is related with flow packet delays, the cost values must be based on the link delays.
- If the objective function is related with link loads, the cost values must be based on the link loads.

## Non defined $P_s$ sets (III)

### Greedy randomized strategy:

- We start by considering the network without any routing path
- We determine a random order to compute the routing paths of the flows  $s \in S$
- For each flow  $s \in S$ , and by the previous determined order:
  - we assign a cost to each link based on the current routed flows
  - we execute Dijkstra's algorithm to compute the routing path  $p$  of flow  $s$
  - we update the network by routing the average packet rate  $\lambda_s$  of flow  $s$  through its assigned path  $p$

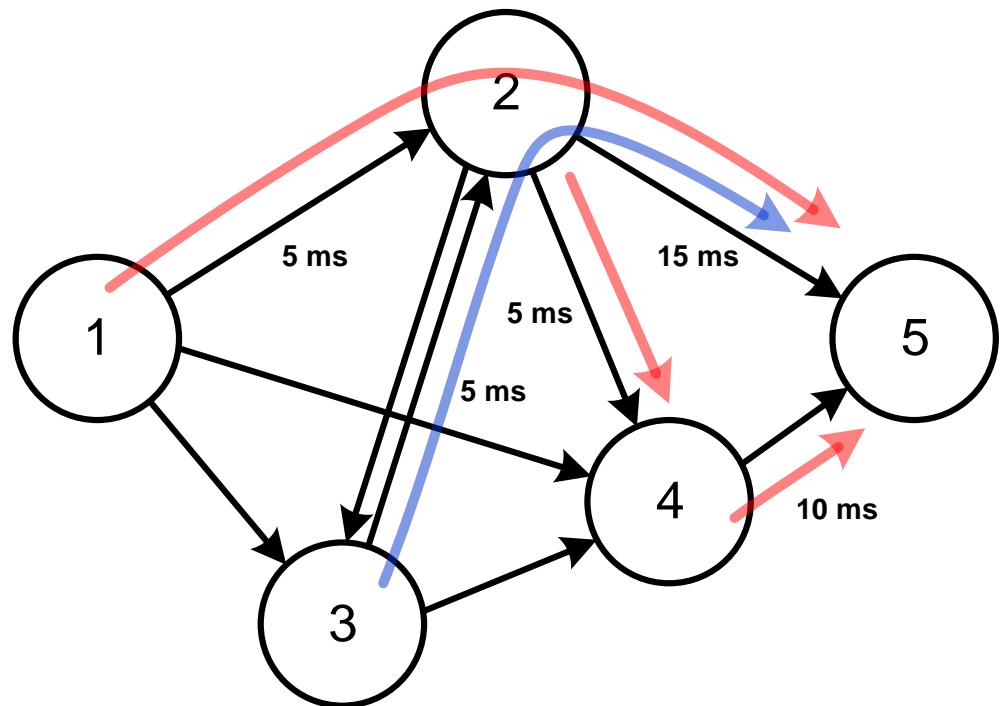
## Non defined $P_s$ sets (IV)



- Previous solution, link delays calculated based on installed flows

$$W_{ij} = \frac{1}{\mu_{ij} - \lambda_{ij}} + d_{ij}$$

Next solution, shortest path based on link delays of previous solution



## Non defined $P_s$ sets (V)

### Local search strategy:

(Remember that) the neighbour set is defined as:

- For a current solution (that defines a routing path  $p$  for each flow  $s \in S$ ), a neighbour solution is a solution that differs from the current one in the routing path of a single flow.

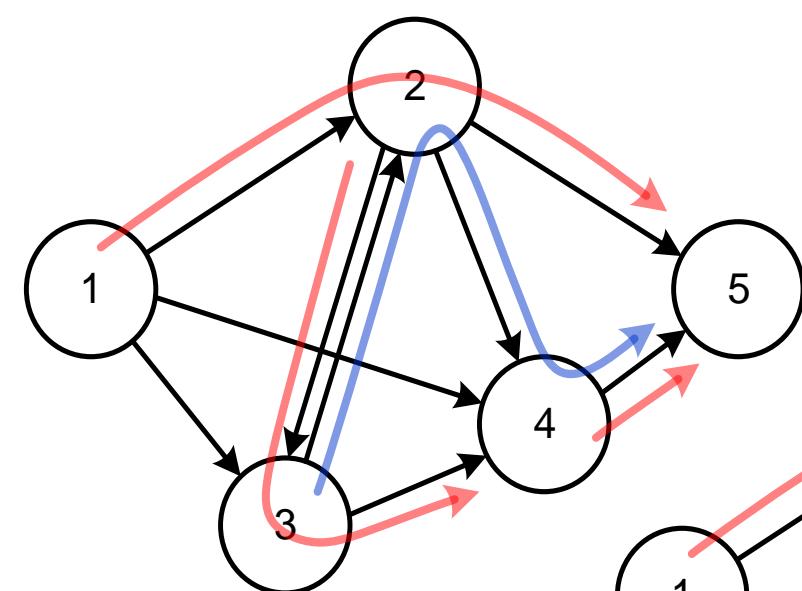
The neighbours of a current solution are computed as follows:

➤ For each flow  $s \in S$ :

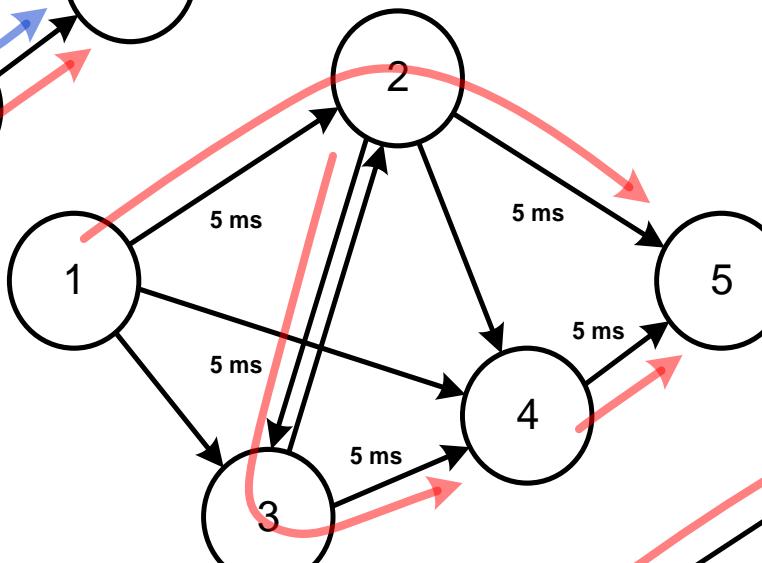
- we determine the network solution with all flows except  $s$
- we assign a cost to each link based on the routed flows
- we execute Dijkstra's algorithm to compute the routing path  $p$  of flow  $s$

Note that, in this case, a neighbour solution might be equal to the current solution.

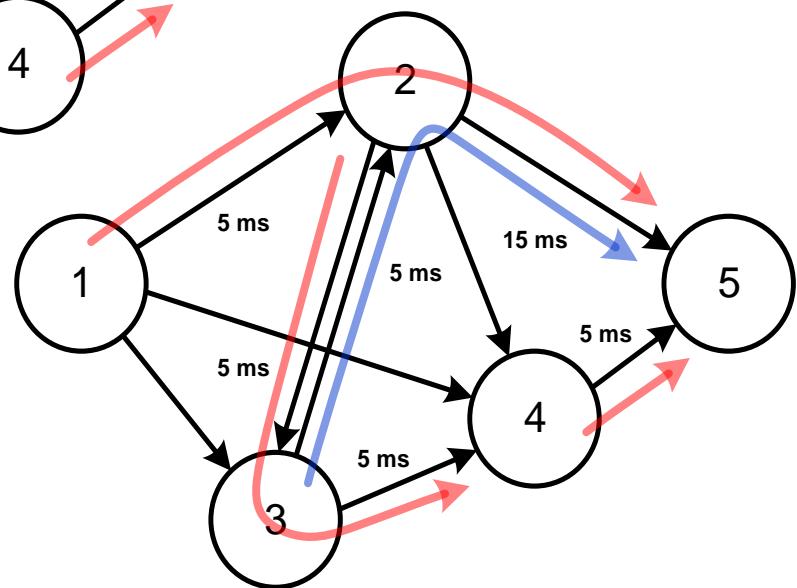
## Non defined $P_s$ sets (VI)



1. Remove path



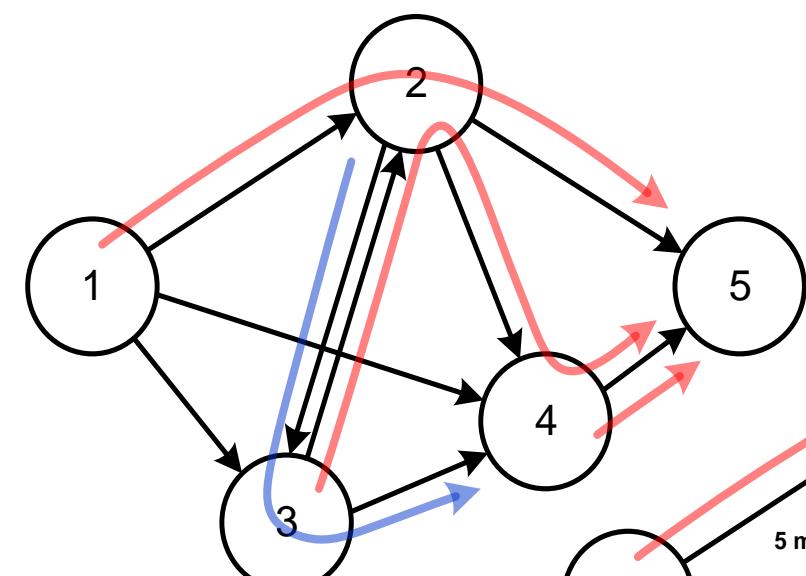
2. Assign costs to links



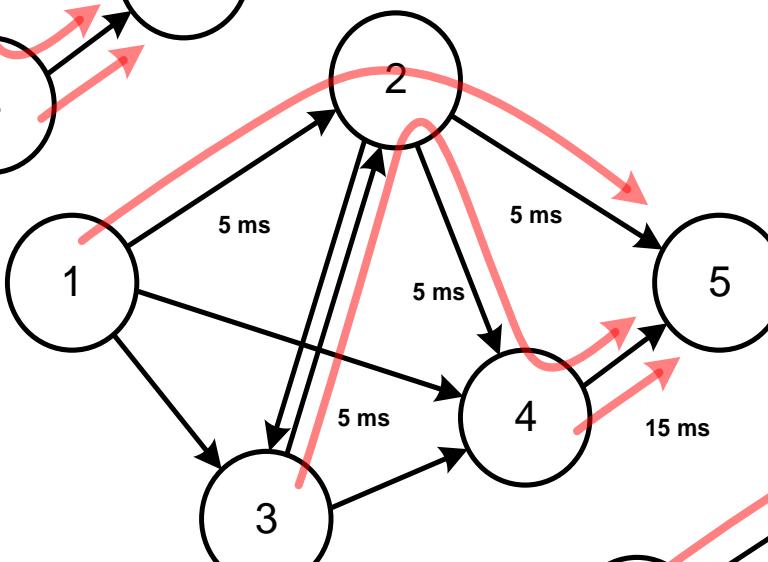
Steps in building neighbor solution

3. Determine shortest path of removed flow (using Dijkstra)

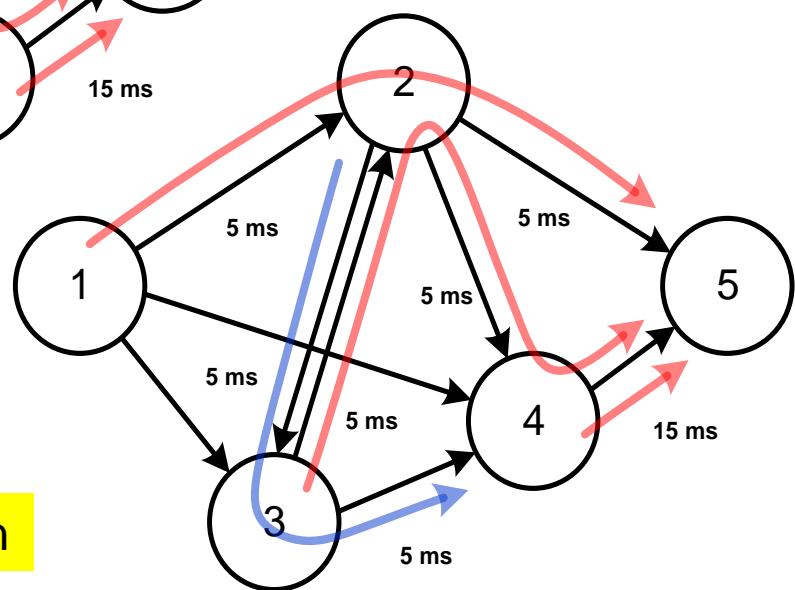
## Non defined $P_s$ sets (VII)



1. Remove path



2. Assign costs to links



Neighbor solution may be equal to current solution

3. Determine shortest path of removed flow (using Dijkstra)

# Multi Start Local Search with non defined $P_s$ sets

Example of a MATLAB code:

```
GlobalBest= Inf;
for iter=1:Iterations
    CurrentSolution= GreedyRandomized();           ← Computes a solution with a Greedy Randomized algorithm
    CurrentObjective= Evaluate(CurrentSolution);    ← Computes the objective function value of a solution
    repeat= true;
    while repeat
        NeighbourBest= Inf;                      ← Computes a neighbour solution
        for i=1:size(CurrentSolution,1)
            NeighbourSolution= BuildNeighbour(CurrentSolution,i);   ↓
            NeighbourObjective= Evaluate(NeighbourSolution);          ← Computes the objective function value of a solution
            if NeighbourObjective < NeighbourBest
                NeighbourBest= NeighbourObjective;
                NeighbourBestSolution= NeighbourSolution;
            end
        end
        if NeighbourBest < CurrentObjective
            CurrentObjective= NeighbourBest;
            CurrentSolution= NeighbourBestSolution;
        else
            repeat= false;
        end
    end
    if CurrentObjective < GlobalBest
        GlobalBestSolution= CurrentSolution;
        GlobalBest= CurrentObjective;
    end
end
```

# Multi Start Local Search with non defined $P_s$ sets

Example of a MATLAB code:

```

GlobalBest= Inf;
for iter=1:Iterations
    CurrentSolution= GreedyRandomized();
    CurrentObjective= Evaluate(CurrentSolution);
    repeat= true;
    while repeat
        NeighbourBest= Inf;
        for i=1:size(CurrentSolution,1)
            NeighbourSolution= BuildNeighbour(CurrentSolution,i);
            NeighbourObjective= Evaluate(NeighbourSolution);
            if NeighbourObjective < NeighbourBest
                NeighbourBest= NeighbourObjective;
                NeighbourBestSolution= NeighbourSolution;
            end
        end
        if NeighbourBest < CurrentObjective
            CurrentObjective= NeighbourBest;
            CurrentSolution= NeighbourBestSolution;
        else
            repeat= false;
        end
    end
    if CurrentObjective < GlobalBest
        GlobalBestSolution= CurrentSolution;
        GlobalBest= CurrentObjective;
    end
end

```

**repeat cycle**

$x = \text{BuildSolution}()$

$z = \text{LocalSearch}(x)$

**Calculation of Best Neighbour Solution**

**Move to Neighbour Solution when Better**

**Exit Local Search when Neighbour Solution is not Better**

**Update Global Best Solution**