

Airports drive me mad. I don't mind the flying; it's all the hassle before you get on the plane and afterwards, including walking five miles through corridors to the point where you queue for ages to check passports and hope your luggage has arrived safely.

June Whitfield



4

DESEMPENHO E
DIMENSIONAMENTO DE REDES



universidade de aveiro
teoria poesisis praxis

Atenção!

Todo o conteúdo deste documento pode conter alguns erros de sintaxe, científicos, entre outros... **Não estudes apenas a partir desta fonte.** Este documento apenas serve de apoio à leitura de outros livros, tendo nele contido o programa completo da disciplina de Desempenho e Dimensionamento de Redes, tal como foi lecionada, no ano letivo de 2017/2018, na Universidade de Aveiro. Este documento foi realizado por Rui Lopes.

mais informações em ruieduardofalopes.wix.com/apontamentos

Toda a nossa formação em termos de redes neste preciso momento foca-se numa abordagem de aplicação dos conceitos teóricos de como é que uma ou mais comunicações se podem efetuar numa rede de pequenas dimensões — como as das nossas casas — até redes maiores — como as redes de operadoras de serviço, encarregues pelos ISPs.

Nesta disciplina iremos abordar as várias características que permitem avaliar tanto a grandeza como o impacto das várias ações e comunicações dentro do contexto de uma rede. Assim, aspetos como o tempo de resposta, a largura de banda de uma ligação ou até mesmo a geração de filas de espera entre vários nós de uma rede, serão alvo de estudo neste documento.

1. Teoria de Probabilidades Revisitada

Uma das várias ferramentas que vamos ter necessidade de rever é a matemática, mais em particular os vários estudos que fizemos em termos de **probabilidades**. Em Métodos Probabilísticos para Engenharia Informática (a2s1) abordámos alguns destes conceitos que agora nos serão vitais.

probabilidades

Experiências aleatórias e probabilidades definidas sobre eventos

No universo da estatística, os seus estudiosos, estão constantemente preocupados em tentar avaliar o contexto de uma tendência dentro de um estudo científico ou de um plano. Por exemplo, uma determinada equipa de investigação poderá estar intrigada sobre o número de suicídios que ocorrem numa determinada cidade de um país, podendo, posteriormente, justificar tais casos como razão de uma população em grande parte deprimida ou não.

Um estudo deste tipo diz-se ser uma **experiência**, sendo que pretende descrever qualquer processo que gera um conjunto de dados. Consideremos, para o efeito, algo mais simples: atirar uma moeda ao ar — este acontecimento já sabemos que só poderão acontecer dois resultados, sendo eles ou sair cara, ou sair coroa.

experiência

Mas como é que podemos obter uma noção de tendência através de experiências? Ora, através da observação, se as repetirmos vezes o suficiente talvez possamos estabelecer relações entre os vários resultados finais — mais precisamente, podemos estabelecer relações entre os vários **acontecimentos ou eventos**.

acontecimentos, eventos
espacoo de resultados

Numa experiência aleatória dizemos também que o **espacoo de resultados**, S , é o conjunto de todos os resultados possíveis de uma experiência, sendo cada um destes denominado de **elemento** (membro ou ponto de amostragem). Por exemplo, se considerarmos a experiência de lançar um dado comum de seis faces podemos ter o seguinte espaço de resultados $S = \{1,2,3,4,5,6\}$.

elemento

Contudo, poderá ser do nosso interesse não apenas relacionar os vários resultados entre si, mas também vários acontecimentos. Para isso, e tirando algum usufruto da lógica matemática, precisamos de saber a que correspondem as operações de **união** e de **interseção** dos acontecimentos E e F , sendo eles $E \cup F$ e EF , respetivamente. Contudo, há que denotar que quando $EF = \emptyset$ (sendo \emptyset o conjunto vazio), os acontecimentos dizem-se **mutuamente exclusivos**. Mais, diz-se também que o **complemento** do acontecimento E , designado por E^c é conjunto de elementos de S que não pertencem a E .

união
interseção

mutuamente exclusivos,
complemento

Talvez tenha mesmo sido o chamamento ao mundo dos jogos que levou o homem ao desenvolvimento da teoria da probabilidade, dado que para aumentar o número de vitórias, os jogadores começaram a ser chamados de matemáticos, por trazerem estratégias ótimas para vários jogos de azar. Alguns destes matemáticos que troxeram tais estratégias foram mesmo Pascal, Leibniz, Fermat ou até mesmo James Bernoulli. Como resultado de tal desenvolvimento da teoria da probabilidade, das inferências estatísticas (com todas as suas previsões e generalizações), foi então criada uma ramificação que pudesse levar com técnicas semelhantes dada a necessidade de avaliar tendências, como a política, negócios, previsões metereológicas, investigação científica ou até mesmo redes de computadores [1].

- ◎ Blaise Pascal
- ◎ Gottfried Leibniz
- ◎ Pierre de Fermat
- ◎ James Bernoulli

3 DESEMPENHO E DIMENSIONAMENTO DE REDES

Para responder à necessidade de avaliar tendências surgiu então o conceito de probabilidades, estas, com base em três **axiomas** fundamentais para o seu estudo, sendo eles: (1) a probabilidade de um acontecimento E está compreendida entre 0 e 1, $0 \leq P(E) \leq 1$; (2) a soma das probabilidades do espaço de resultados é igual a 1, $P(S) = 1$; (3) para qualquer conjunto de acontecimentos mutuamente exclusivos E_1, E_2, \dots, E_n , a probabilidade da sua união é igual à soma das várias probabilidades, conforme podemos ver em (1.1).

axiomas

$$P\left(\bigcup_i E_i\right) = \sum_i P(E_i) \quad (1.1)$$

Consequência da aplicação de tais axiomas temos dois corolários: num primeiro temos que a soma das probabilidades de um acontecimento E e o seu complementar E^c é igual ao acontecimento certo $P(E) + P(E^c) = 1$; num segundo temos que a probabilidade da união de dois acontecimentos E e F são a soma da probabilidade do acontecimento E com a probabilidade do acontecimento F , com a diferença da probabilidade da intersecção EF , sendo $P(E \cup F) = P(E) + P(F) - P(EF)$.

Probabilidades condicionadas e acontecimentos independentes

Por vezes, na relação entre vários acontecimentos, precisamos de saber como julgar eventos que aconteceram com uma determinada ordem, por exemplo, em termos de tempo. Por exemplo, se considerarmos a experiência lançar um dado comum de seis faces, podemos querer saber qual a probabilidade do acontecimento “sair número 4 ou 6”, sendo que já “saiu par”. Para este caso precisamos de saber definir o conceito de **probabilidade condicionada**.

Assim, dados dois acontecimentos E e F , a probabilidade condicionada de E ocorrer dados que F ocorreu designa-se por $P(E | F)$ e é dada por (1.2).

probabilidade condicionada

$$P(E | F) = \frac{P(EF)}{P(F)} \quad (1.2)$$

Contudo, na relação entre dois acontecimentos, nem sempre (1.2) se terá de aplicar de forma tão explícita. Quando temos dois acontecimentos cujas ações não se manifestem entre si, isto é, que não interfiram um com o outro, então dizemos que temos dois acontecimentos **independentes**. Por exemplo, consideremos a experiência “verificar cara ou coroa num lançamento de uma moeda”. Depois de lançadas três vezes, saiu três vezes coroa, mas qual é a probabilidade de, num novo lançamento, sair cara? Ora, como o que aconteceu no passado não altera o acontecimento atual, então a probabilidade continua a ser de 0.5, dado que estes acontecimentos são considerados independentes.

independentes

Em termos matemáticos, dois acontecimentos são independentes quando o seu produto é equivalente à sua intersecção (1.3), por conseguinte, quando a probabilidade condicionada de ambos é igual à probabilidade que se pretende conhecer (1.4), prova de que o acontecimento anterior não interfere.

$$P(EF) = P(E)P(F) \quad (1.3)$$

$$P(E | F) = \frac{P(EF)}{P(F)} \therefore P(E | F) = \frac{P(E)P(F)}{P(F)} = P(E) \quad (1.4)$$

No seguimento da probabilidade condicionada, sejam F_1, F_2, \dots, F_n acontecimentos mutuamente exclusivos tais que a união forma o espaço de resultados S . Então, podemos reescrever que a soma de todas as probabilidades da intersecção de acontecimentos E e F_i (com $1 \leq i \leq n$) é igual à soma dos produtos da probabilidade condicionada $P(E | F_i)$ pela probabilidade $P(F_i)$, como podemos ver em (1.5).

$$P(E) = \sum_{i=1}^n P(EP_i) = \sum_{i=1}^n P(E|F_i)P(F_i) \quad (1.5)$$

Regra de Bayes

Uma das regras mais utilizadas na teoria das probabilidades deve-se a Thomas Bayes e permite-nos inverter a ordem de lógica das probabilidades condicionadas. Enquanto que nas probabilidades condicionadas temos uma forma de calcular uma probabilidade de um acontecimento sabendo uma sua causa, aqui pensamos ao contrário, isto é, sabendo um acontecimento que ocorreu com uma causa, queremos saber a probabilidade desta. A derivação de tal regra é direta, mas as suas implicações são profundas.

Começando pela definição da probabilidade condicionada (1.2), sendo F_1, F_2, \dots, F_n acontecimentos mutuamente exclusivos tais que a sua união forma o espaço de resultados S , então tendo ocorrido o acontecimento E , a probabilidade de F_j com $j = 1, 2, \dots, n$ é representada por $P(F_j | E)$. Sabendo que $P(EF) = P(F_j | E)P(E)$ podemos interpretar que a probabilidade de E e de F ocorrer é igual ao produto das probabilidades dos dois eventos: primeiro, de E acontecer; depois, que a condicionada em E e F ocorra. Relembremo-nos, entretanto, que $P(F_j | E)$ está definida como sendo um evento F_j que se segue de um evento E . Agora, consideremos o seguinte: F é sabido que ocorreu, mas qual é a probabilidade de E ter ocorrido? Fazer esta questão é semelhante ao seguinte cenário: se há fogo, então há fumo; mas se virmos fumo, qual é a probabilidade que tal acontecimento tenha ocorrido pelo fogo? A probabilidade que procuramos é, precisamente, $P(E | F_j)$. Usando a definição de probabilidade condicionada dada anteriormente, temos (1.6).

$$P(F_j | E) = \frac{P(EF_j)}{P(E)} = \frac{P(E|F_j)P(F_j)}{P(E)} = \frac{P(E|F_j)P(F_j)}{\sum_{i=1}^n P(E|F_i)P(F_i)} \quad (1.6)$$

Uma forma de interpretar este resultado é como sendo o cálculo do grau sobre o qual um efeito ou um F_j posterior poderá ser atribuído a uma determinada causa ou posterior E . Este resultado também é denominado de **Lei da Probabilidade Total**.

Lei da Probabilidade Total

Variáveis aleatórias

Até aqui revimos os nossos conhecimentos em termos de eventos ou acontecimentos, estes, coleções de observações ou experiências. Contudo, muitas das vezes estamos interessados em estudar quantidades abstratas ou resultados de experiências que, por si, são derivadas de acontecimentos, mas que não acontecimentos por si só. Embora nos possa parecer algo contraintuitivo, vejamos o seguinte exemplo: se lançarmos um dado comum e quisermos calcular a probabilidade do quadrado da face resultante ser mais pequeno que 10, podemos considerar este cenário como aleatório, associando-lhe uma probabilidade que somente dependerá de outros eventos igualmente aleatórios que aconteçam. E, no entanto, este cenário não é uma observação nem um acontecimento — denomina-se de **variável aleatória**.

Uma variável aleatória X é assim uma função que atribui um número real a cada ponto do espaço de resultados S de uma experiência aleatória. Esta variável aleatória possui uma **função distribuição** $F(x)$ (ou função de distribuição cumulativa) que é definida por (1.7).

variável aleatória

função distribuição

$$F(x) = P(X \leq x), \quad -\infty < x < +\infty \quad (1.7)$$

Uma vez que a função distribuição se encontra definida sobre os axiomas das probabilidades é natural que consigamos atribuir, como uma das suas propriedades, que o seu valor

5

DESEMPENHO E DIMENSIONAMENTO DE REDES

terá de estar compreendido entre 0 e 1, isto é, que $0 \leq F(x) \leq 1$, para todo o x . Também, se $x_1 \leq x_2$, então $F(x_1) \leq F(x_2)$, mostrando-se como uma função não-decrescente. Uma vez que esta função é cumulativa das várias probabilidades, o seu limite quando x tende para $-\infty$ é igual a 0, sendo que, pelo contrário, quando x tende para $+\infty$, o seu limite é 1, como mostra (1.8).

$$\lim_{x \rightarrow -\infty} F(x) = 0 \quad \lim_{x \rightarrow +\infty} F(x) = 1 \quad (1.8)$$

Por fim, para um $a < b$ temos que $P(a < X \leq b) = F(b) - F(a)$.

Uma variável aleatória pode, também assumir um valor dentro de conjuntos de valores discretos, dizendo-se assim uma **variável aleatória discreta**. Neste caso, e como podemos contar e identificar os vários casos de valores de variáveis aleatórias, podemos descrever uma função que devolve o valor de uma determinada probabilidade de um evento ocorrer, denominada de **função probabilidade**. Tal função, também vulgarmente denominada por função massa de probabilidade de uma variável aleatória discreta é designada da seguinte forma, como podemos ver em (1.9).

$$f(x_i) = P(X = x_i), \text{ para todos os valores de } i = 1, 2, 3, \dots \quad (1.9)$$

Claro está que, somadas todas as probabilidades de todos os valores possíveis da variável aleatória, o resultado terá de ser 1, uma vez que caso contrário não estariam a respeitar um dos 3 axiomas-base das probabilidades. Assim sendo, podemos formar a nossa função distribuição da variável aleatória X da seguinte forma em (1.10).

$$F(x) = \sum_{x_i < x} f(x_i) \quad (1.10)$$

Dentro do conjunto das várias variáveis aleatórias discretas, um dos exemplos possíveis de serem estudados e aplicados são as **variáveis de Bernoulli**. Uma variável aleatória discreta de Bernoulli é, assim, uma experiência que pode resultar em sucesso com uma probabilidade de p ou em insucesso com uma probabilidade $1 - p$. Basicamente, com este tipo de variável aleatória, podemos obter um de dois valores, sendo que a probabilidade de obter o primeiro é igual à diferença entre a probabilidade de um acontecimento certo e a probabilidade de sair o segundo. Em termos algébricos, esta definição poderá ser descrita como podemos ver em (1.11).

$$f(i) = p^i (1-p)^{1-i}, \quad i = 0, 1 \quad (1.11)$$

Na Figura 1.1 podemos verificar a aplicação das funções de probabilidade e de distribuição (à esquerda e à direita, respetivamente). No primeiro gráfico temos que para cada valor de X (note-se 0 ou 1), com uma probabilidade de $p = 0.8$, existe um e um só valor de $f(x)$, sendo que a sua soma é igual a 1, como podemos verificar pela figura da direita, onde a função de distribuição vai somando os vários resultados, terminando em 1.

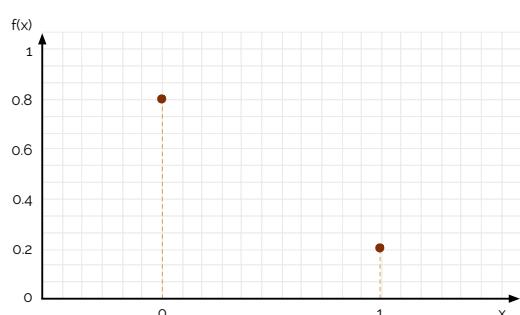


figura 1.1

variável aleatória discreta

função probabilidade

variáveis de Bernoulli

Outro tipo de variável aleatória é a **binomial**. Diz-se que se tem uma variável aleatória discreta binomial quando se admite que se têm n experiências por Bernoulli independentes, em que cada uma resultado num sucesso com probabilidade p ou num insucesso com probabilidade $1 - p$ [2]. Aqui, se X representar o número de insucessos em n experiências, a função probabilidade $f(x)$ equivale a (1.12).

$$f(i) = \binom{n}{i} p^i (1-p)^{n-i}, \quad i = 0, 1, 2, \dots, n, \text{ e onde } \binom{n}{i} = \frac{n!}{i!(n-i)!} \quad (1.12)$$

Por outras palavras, uma distribuição binomial com parâmetros n e p é uma distribuição discreta que descreve o número de sucessos numa sequência de n experiências independentes, cada uma questionando uma pergunta de resposta sim/ não, com um consequente sendo uma resposta booleana: uma variável aleatória contendo somente um bit de informação: sucesso (com probabilidade p) ou insucesso (com probabilidade $1 - p$).

Na Figura 1.2 podemos verificar, de forma análoga à Figura 1.1, a função massa de probabilidade $f(x)$ e a função distribuição $F(x)$ para a distribuição binomial. Como podemos analisar a partir do gráfico da esquerda, numa distribuição binomial temos então uma acentuação do número de valores a serem aleatorizados próximo da probabilidade p que fora designada como sucesso. Por conseguinte, vamos obter um gráfico para a função distribuição que cresce lentamente, seguido de uma alta subida e de uma posterior atenuação até ao valor de probabilidade 1.

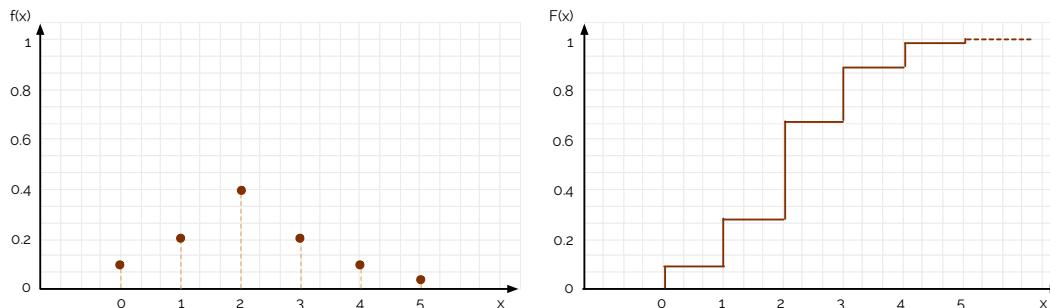


figura 1.2

Por fim, outra variável aleatória discreta que merece a nossa atenção é a variável aleatória **geométrica**, que admite que são realizadas experiências de Bernoulli independentes com parâmetro p (probabilidade de sucesso), até que ocorra um sucesso [2]. Se X representar o número de insucessos antes do primeiro sucesso, a função probabilidade é dada por (1.13).

$$f(i) = (1-p)^i p, \quad i = 0, 1, 2, \dots \quad (1.13)$$

Contudo, esta distribuição não se encontra definida somente desta forma, mas também como a distribuição do número de X experiências de Bernoulli necessárias para se atingir o primeiro sucesso, operação suportada pelo conjunto $\{1, 2, 3, \dots\}$. Neste caso, a nossa função de massa de probabilidade é dada por (1.14).

$$f(i) = (1-p)^{i-1} p, \quad i = 1, 2, \dots \quad (1.14)$$

Mais uma vez, na Figura 1.3 podemos verificar a presença de dois gráficos que representam ambas funções probabilidade e distribuição para a distribuição geométrica. Contudo, desta vez, podemos reparar que temos dois conjuntos de gráficos sendo que temos duas possíveis representações para a mesma distribuição. Na verdade, podemos aqui reparar que um dos gráficos não passa do deslocamento do outro, sendo que teoricamente são ambas distribuições geométricas. Note-se, então, que chamar de “distribuição geométrica” a ambas as representações é meramente uma questão de convenção e de conveniência.

geométrica

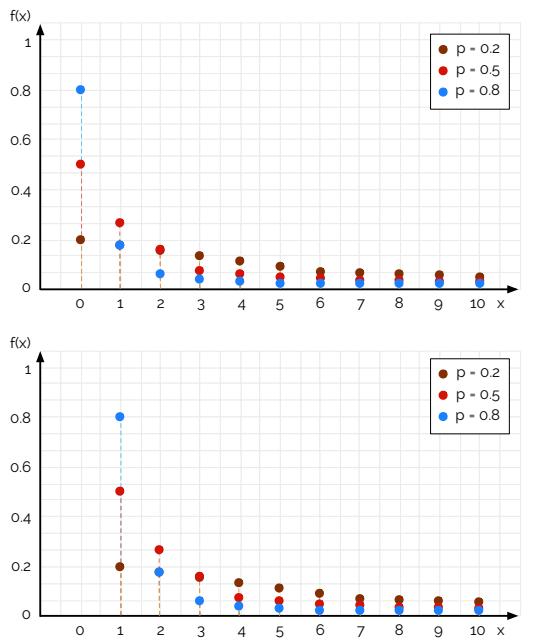


figura 1.3

Da mesma forma que temos variáveis aleatórias que produzem resultados enquanto descritas de forma discreta, também podemos ter variáveis aleatórias **contínuas**. Este tipo de variáveis aleatórias aplicam-se se existir uma função não-negativa $f(x)$ tal que, para qualquer conjunto de números reais R , $f(x)$ em (1.15).

contínuas

$$P(X \in R) = \int_R f(x) dx, \text{ sendo que } \int_{-\infty}^{+\infty} f(x) dx = 1 \quad (1.15)$$

A partir de (1.15) temos que a probabilidade de uma variável X ocorrer, sendo esta compreendida entre um a e um b é igual a (1.16), sendo a sua função distribuição dada por (1.17).

$$P(a \leq X \leq b) = \int_a^b f(x) dx \quad (1.16)$$

$$F(x) = P(X \in [-\infty, x]) = \int_{-\infty}^x f(y) dy \quad (1.17)$$

Vejamos, então, alguns exemplos de variáveis aleatórias contínuas. A primeira que podemos analisar é a variável aleatória com **distribuição uniforme**, dizendo-se tal num intervalo $[a, b]$ se a função densidade de probabilidade possuir a forma de (1.18).

distribuição uniforme

$$f(x) = \begin{cases} \frac{1}{b-a}, & a < x < b \\ 0, & \text{caso contrário} \end{cases} \quad (1.18)$$

Embora tenhamos definido a função densidade de probabilidade, também precisamos de saber como definir qual a sua função de distribuição. Para a obter, e conforme podemos confirmar em (1.17), temos que a mesma é efetuada através da integração do intervalo entre a e b .

Olhando para um gráfico como o da Figura 1.4, onde podemos ver a função densidade de probabilidade da distribuição uniforme, podemos também verificar que a função distribuição poderá ser obtida através do cálculo da área abaixo da linha definida pela função em causa e pelo eixo dos xx , que corresponde aos vários valores atribuídos pela variável aleatória, desta feita, contínuos.

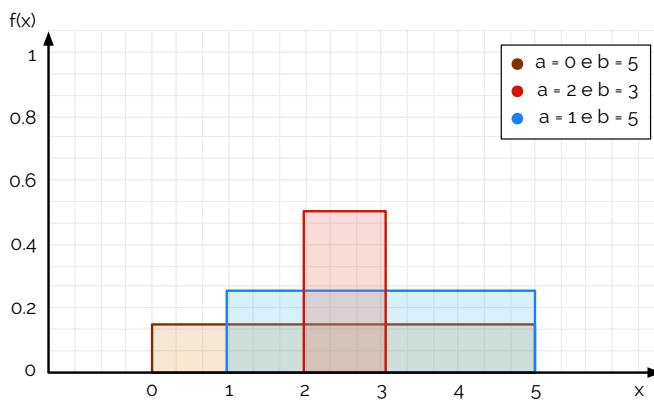


figura 1.4

Também já conhecida, e conforme já fora abordado na disciplina de Métodos Probabilísticos para Engenharia Informática (a2s1), entre outras, também temos a variável aleatória com **distribuição gaussiana** (também denominada de normal). Neste caso, dizemos que uma variável aleatória é definida como sendo com distribuição gaussiana com média μ e desvio padrão σ se a sua função de densidade de probabilidade for a formalizada em (1.19).

distribuição gaussiana
◎ Carl Gauss

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(x-\mu)^2/2\sigma^2} \quad (1.19)$$

De forma análoga às variáveis discretas binomiais, agora temos uma função que nos permite precisar mais os nossos valores aleatórios à probabilidade que pretendemos, desta vez, através de dois parâmetros diferentes: uma média e um desvio padrão. Um caso muito específico desta distribuição, denominado de distribuição gaussiana (ou normal) padrão, possui média $\mu = 0$ e desvio padrão $\sigma = 1$. Na Figura 1.5 podemos visualizar o resultado da função densidade de probabilidade e respetiva função de distribuição para vários valores de média e de desvio padrão.

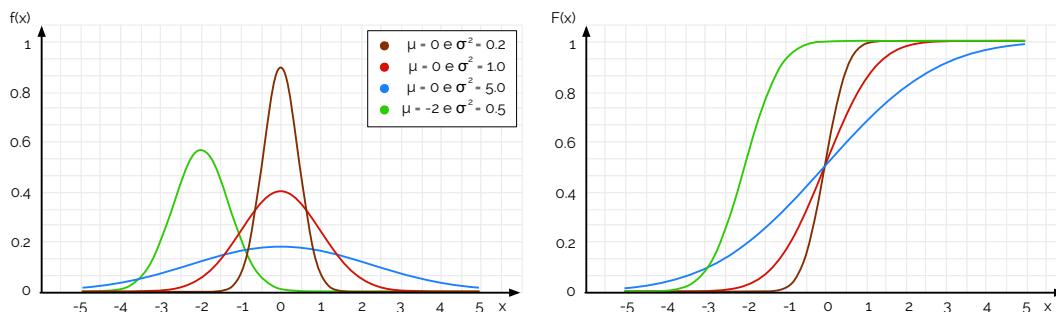


figura 1.5

Uma vez que falamos em média e desvio padrão convém-nos rever os seus conceitos.

Para começar, dizemos que a **média** (ou valor esperado) de uma variável aleatória X é dado por (1.20).

$$E[X] = \begin{cases} \sum_{j=1}^{\infty} x_j f_X(x_j) & \text{se variável aleatória for discreta} \\ \int_{-\infty}^{+\infty} x f_X(x) dx & \text{se variável aleatória for contínua} \end{cases} \quad (1.20)$$

Note-se, que este cálculo possui algumas propriedades: primeiro, a soma das médias é equivalente à média das somas e, segundo, o produto da média por um escalar, por conseguinte, é equivalente à média do produto com a variável aleatória $E[cX] = cE[X]$.

Um segundo aspeto a rever é a **variância**: a variância de uma variável aleatória permite-nos medir o quanto esparsos estão os valores de uma dada amostra aleatória. Esta é então dada pela forma dada em (1.21).

$$\text{Var}[X] = E[(X - E[X])^2] = E[X^2] - E[X]^2 \quad (1.21)$$

variância

Note-se, não obstante, que a variância também possui um conjunto de propriedades associadas, entre as quais o facto de ter sempre um valor positivo ($\text{Var}[X] \geq 0$) e de que a variância de uma variável aleatória cujos valores são sujeitos ao produto de um escalar é equivalente ao produto do quadrado deste escalar pela variância da variável, como podemos ver em (1.22).

$$\text{Var}[cX] = c^2 \text{Var}[X] \quad (1.22)$$

Mais, podemos dizer que a variância da soma de variáveis aleatórias é igual à soma das respetivas variâncias se e só se as várias variáveis X_i forem independentes entre si, permitindo (1.23).

$$\text{Var}\left[\sum_{i=1}^n X_i\right] = \sum_{i=1}^n \text{Var}[X_i] \quad (1.23)$$

Por fim, uma última distribuição que possamos falar, sendo contínua, é a **distribuição exponencial**. Aqui dizemos que uma variável aleatória possui uma distribuição exponencial com um parâmetro λ , com $\lambda > 0$, se a sua função densidade de probabilidade for dada por (1.24).

distribuição exponencial

$$f(x) = \begin{cases} \lambda e^{-\lambda x} & , x \geq 0 \\ 0 & , x < 0 \end{cases} \quad (1.24)$$

Por sua vez, a função de distribuição terá de refletir o acréscimo de probabilidades à medida que os valores da variável aleatória aumentam. Para isso, necessitamos de especificar a função de distribuição conforme (1.25).

$$F(x) = \begin{cases} 1 - e^{-\lambda x} & , x \geq 0 \\ 0 & , x < 0 \end{cases} \quad (1.25)$$

Neste caso de implementação, temos que a média é dada por $1/\lambda$ e que a sua variância é dada por $1/\lambda^2$. Note-se também que este tipo de distribuição possui uma propriedade muito *sui generis*: diz-se que a distribuição exponencial não tem memória. Esta propriedade de **perda de memória** exibe-nos um comportamento muito especial desta distribuição. Consideremos, para melhor perceber este conceito, uma variável aleatória X que se traduz pela vida de uso de um carro, expressa em termos de “número de quilómetros realizados até que o motor quebre”. Está claro, com base na nossa intuição, que um motor que é conduzido mais do que 500 000 quilómetros tenha uma menor variável X que um segundo motor equivalente, conduzido apenas por 1 000 quilómetros. Por conseguinte, tal variável não possuiria a propriedade de perda de memória, uma vez que está condicionada por fatores externos ao longo do tempo.

perda de memória

Consideremos agora, então, um exemplo contrário, que exibe características de perdas de memória. Imaginemos um longo corredor sobre o qual se encontram centenas de cacifos encostados a uma parede. Cada cacifo possui um aloquete com 500 posições e a cada um

foi atribuída uma posição de abertura aleatoriamente. Agora, uma pessoa excêntrica caminha pelo corredor fora, parando a cada um dos cacos e tentando abrir um por um. Neste caso, definimos como variável aleatória X o tempo de procura por um aloque semi-aberto, expresso em termos de “número de tentativas que uma pessoa terá de efetuar até conseguir abrir um caco”. Neste caso, o valor esperado será sempre igual a 500, independentemente do número de tentativas feitas. A cada nova tentativa, a probabilidade de acertar mantém-se sempre sendo $1/500$, pelo que mesmo que um aloque tenha sido experimentado 499 vezes consecutivas sem sucesso, ficamos sempre à espera de mais 500 novas tentativas para observar o próximo sucesso. Por outro lado, se uma pessoa se fixasse num só caco e se “lembresse” dos passos para o abrir, teria a garantia de que necessitaria de, no máximo, 500 tentativas para abrir o próximo.

Em termos algébricos, temos que $P[X > s + t | X > t] = P[X > s]$. Aqui, se ambas X_1 e X_2 forem variáveis aleatórias independentes e exponencialmente distribuídas, com médias $1/\lambda_1$ e $1/\lambda_2$ respectivamente, então temos (1.26).

$$P[X_1 < x_2] = \frac{\lambda_1}{\lambda_1 + \lambda_2} \quad (1.26)$$

Processos estocásticos

Por fim, como vimos em Métodos Probabilísticos para Engenharia Informática (a2s1), um **processo estocástico** $\{X(t), t \in T\}$ é um conjunto de variáveis aleatórias, sendo que para cada t de T , $X(t)$ é uma variável aleatória. Nesta notação lógica, o índice t é frequentemente interpretado como tempo, sendo que $X(t)$, então, considera-se como o estado do processo no instante de tempo t .

Esta evolução lógica permite-nos consolidar um conjunto T , como sendo o conjunto dos índices do processo estocástico. Neste conjunto, se T for um conjunto contável, então podemos dizer que o processo estocástico descrito existe em tempo discreto. Pelo contrário, se T for um intervalo da reta real (um conjunto de valores não contável), então designamos o processo estocástico como sendo definido em tempo contínuo.

Estes estados encontram-se todos definidos sobre um único espaço, denominado como o **espaço de estados**, sendo este o conjunto de todos os valores que as variáveis aleatórias $X(t)$ podem tomar.

Alguns exemplos de processos estocásticos já foram indicados ao longo do tempo e deste documento, contudo, considerando um sistema em que temos uma fila de espera de serviços e um servidor, chegando clientes para serem servidos: “atrasos servidos por cada cliente na fila de espera” é um processo estocástico em tempo discreto, com um estado definido como uma variável contínua; “o número de clientes em espera” é um processo estocástico em tempo contínuo, sendo o estado definido sob a forma de uma variável discreta.

2. Cadeias de Markov e Sistemas de Filas de Espera

Tendo abordado processos estocásticos na secção anterior, um dos tipos possíveis de processos são os **processos de contagem**. Um processo estocástico diz-se ser de contagem se $N(t)$ representar o número total de eventos que ocorreram até a um instante t .

processos de contagem

Este tipo de processo necessita de satisfazer as seguintes condições: primeiro, o número total de eventos até ao instante t tem de ser superior ou igual a 0 ($N(t) \geq 0$); segundo, este número terá de ser apenas constituído de elementos inteiros; terceiro, se um instante s ocorrer antes de um instante t , então o número de eventos em s terá de ser inferior ou igual ao número de eventos em t ($N(s) \leq N(t)$); por fim, se um instante s ocorrer antes de um instante t , então a diferença $N(t) - N(s)$ é igual ao número de eventos que ocorreram entre os instantes s e t .

Tais processos de contagem poderão, ainda, ser incluídos dentro de duas categorias: se o número de eventos em intervalos de tempo disjuntos for independente, então diz-se que o

processo em causa possui **incrementos independentes**; caso contrário, isto é, se a distribuição do número de eventos que ocorre em qualquer intervalo de tempo depender exclusivamente do comprimento do intervalo de tempo, então diz-se que o processo possui **incrementos estacionários**. [3]

incrementos independentes

Processos de Poisson

As experiências que lançam valores numéricos de uma variável aleatória X , ao número de resultados que ocorrem durante um dado intervalo de tempo é denominado de **experiência de Poisson**. O intervalo de tempo em causa poderá ser de qualquer duração, entre um minuto, um dia, entre outros... Por exemplo, uma experiência de Poisson poderá ser a geração de observações para uma variável aleatória X , esta, representando o número de chamadas telefónicas recebidas por hora num escritório ou o número de dias em que uma escola se encontra fechada devido a catástrofes naturais. Estas experiências de Poisson derivam, entretanto, de **processos de Poisson**.

Os processos de Poisson, em si, possuem um conjunto de propriedades que os definem. Uma primeira propriedade define que o número de resultados a ocorrer num intervalo de tempo é independente do número em que ocorre num outro intervalo de tempo disjunto — dizemos que o processo de Poisson não tem memória. Segundo, a probabilidade de que um resultado ocorrerá durante um muito curto intervalo de tempo é proporcional à duração do intervalo de tempo e não depende do número de resultados a ocorrer fora deste. Por último, a probabilidade de que mais do que um resultado ocorrerá num curto intervalo de tempo é negligenciável.

O número X de resultados que ocorrem durante um processo de Poisson é denominado de variável aleatória de Poisson, sendo a sua função distribuição chamada de **distribuição de Poisson**. O número médio de resultados é calculado através de $\mu = \lambda t$, onde t é o “tempo”, “distância”, “área” ou “volume” de interesse — vulgarmente, dentro destas unidades, aplicado a redes usamos a noção de tempo.

experiências de Poisson

◎ **Siméon Poisson**

processos de Poisson

Em suma, a função distribuição de uma variável aleatória de Poisson X , representando o número de resultados que ocorrem num dado intervalo de tempo denotado por t é dado por (2.1), onde $x = 0, 1, 2, 3, \dots$, λ é o número médio de resultados por unidade de tempo (ou distância, ou área, ...) e $e = 2.71828\dots$ (é o número de Neper).

distribuição de Poisson

◎ **John Napier**

(2.1)

$$f(x) = \frac{e^{-\lambda t} (\lambda t)^x}{x!}$$

Na Figura 2.1 podemos ver alguns casos de funções de densidade de probabilidade para as distribuições de Poisson. Tal como muitas outras distribuições, quer contínuas como discretas, a forma da distribuição de Poisson torna-se cada vez mais simétrica, à medida que a média se torna maior. Na Figura 2.1 podemos verificar três gráficos da função de densidade para $\mu = 0.1$, $\mu = 2$ e $\mu = 5$. Note-se que a simetria torna-se muito visível quando μ fica tão grande como 5.

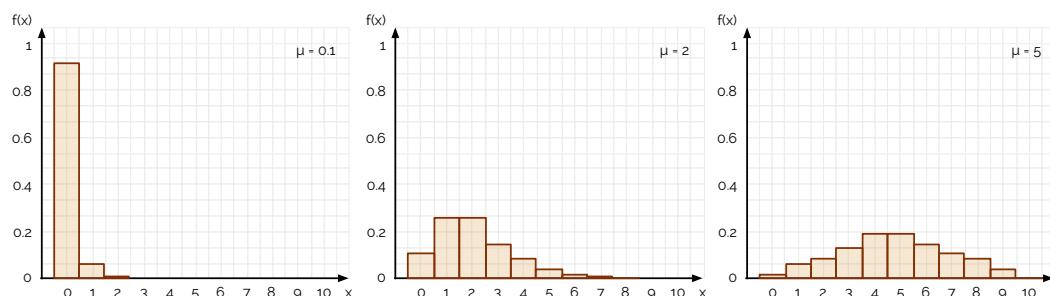


figura 2.1

Como vimos, então, num processo de Poisson com taxa λ temos que podemos considerar um T_1 como o instante de tempo de um primeiro evento. Nesta consideração, tomando os seguintes eventos por n , então T_n (com $n = 1, 2, 3, \dots$) é o intervalo de tempo

entre o $(n-1)$ -ésimo evento e o n -ésimo. Assim sendo podemos afirmar estas variáveis aleatórias como sendo independentes e identicamente com uma distribuição exponencial com média $1/\mu$, conforme vimos na secção anterior. Também com um processo de Poisson $\{N(t), t \geq 0\}$ com taxa λ , cada evento poderá ser classificado de forma independente em evento do tipo 1 com probabilidade p ou evento de tipo 2 com probabilidade $(1-p)$. Assim sendo, $\{N_1(t), t \geq 0\}$ e $\{N_2(t), t \geq 0\}$ são o número de eventos de cada tipo que ocorreram no intervalo $[0,t]$, pelo que $N_1(t)$ e $N_2(t)$ são ambos processos independentes e de Poisson com taxas λp e $\lambda(1-p)$.

Consideremos agora o cenário em que sejam $\{N_1(t), t \geq 0\}$ e $\{N_2(t), t \geq 0\}$ processos de Poisson independentes com taxas λ_1 e λ_2 . Podemos então dizer que o processo soma de ambos $N(t) = N_1(t) + N_2(t)$ é também um processo de Poisson com taxa também soma sendo $\lambda = \lambda_1 + \lambda_2$. Mais, sabendo que num processo de Poisson ocorreram exatamente n eventos até ao instante de tempo t , então os instantes de ocorrência dos eventos são distribuídos independente e uniformemente no intervalo $[0,t]$. Por esta razão diz-se que num processo de Poisson as chegadas são aleatórias [3].

Cadeias de Markov em tempo contínuo

Em Métodos Probabilísticos para Engenharia Informática (a2s1) abordámos o conceito de cadeias de Markov em tempo discreto. Contudo, a definição do tempo no espaço de soluções de um problema não terá de ser, necessariamente, discreto — o tempo poderá estar definido em termos contínuos.

Consideremos assim um processo estocástico em tempo contínuo $\{X(t), t \geq 0\}$ com o espaço de estados definido pelo conjunto dos números inteiros não-negativos. Dizemos que $X(t)$ é uma **cadeia de Markov** se, para todo o s , $t \geq 0$ e inteiros não-negativos i, j , $x(u)$, $0 \leq u < s$, (2.2) se aplicar.

$$\begin{aligned} P[X(s+t) = j | X(s) = i, X(u) = x(u), 0 \leq u < s] &= \\ &= P[X(s+t) = j | X(s) = i] \end{aligned} \tag{2.2}$$

Olhando para (2.2) podemos reparar que existe uma conclusão que lhe conseguimos retirar: a distribuição futura $X(s+t)$ condicionada ao presente $X(s)$ e ao passado $X(u)$, depende apenas do presente e é independente do passado. A esta conclusão damos o nome de **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), definindo-se (2.3).

◎ Andrey Markov

cadeia de Markov

propriedade markoviana
probabilidades de transição
estacionárias

$$P[X(s+t) = j | X(s) = i] = P[X(t) = j | X(0) = i] \tag{2.3}$$

Em suma, numa cadeia de Markov, um processo satisfaz a propriedade markoviana se conseguirmos fazer previsões futurísticas acerca do mesmo, baseando-nos apenas no seu estado atual. Note-se que, vulgarmente, na literatura, costuma-se usar o termo “processo de Markov” para designar aquilo que na verdade é uma cadeia de Markov, mas definida sobre uma linha temporal contínua, contrariamente a uma discreta, vulgarmente denominada por “cadeia de Markov”. Neste documento, sempre que nos referirmos a “cadeia de Markov” pretendemos abordar cadeias de Markov em tempo contínuo, dando seguimento ao título da secção presente.

Uma cadeia de Markov tem então como propriedades, os seguintes tópicos: quando um processo entra num estado i , o tempo de permanência nesse estado, antes de efetuar uma transição para um estado diferente, é exponencialmente distribuído (designamos a média por $1/q_i$)¹; quando o processo deixa o tal estado i , entra de seguida no estado j com uma probabilidade P_{ij} que satisfaz as condições de (2.4).

¹ Equivale tal dizer que quando o processo está no estado i , este transita para outro estado qualquer a uma taxa q_i .

$$P_{ij} = 0 \quad 0 \leq P_{ij} \leq 1, \quad j \neq i \quad \sum_j P_{ij} = 1 \quad (2.4)$$

Note-se ainda, que, numa cadeia de Markov, o tempo de permanência num estado e o próximo estado visitado são variáveis aleatórias independentes.

Para qualquer par de estados i e j seja $q_{ij} = q_i P_{ij}$, onde q_i é 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 e 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**, sendo estas as grandezas habitualmente representadas nos diagramas de transição de estados. Como (2.5) resulta que a especificação das taxas de transição instantâneas determina a cadeia de Markov em tempo contínuo. [3]

$$q_i = \sum_j q_i P_{ij} = \sum_j q_{ij} \quad P_{ij} = \frac{q_{ij}}{q_i} = \frac{q_{ij}}{\sum_j q_{ij}} \quad (2.5)$$

taxas de transição
instantâneas

Probabilidades limite

Consideremos novamente o processo da subsecção anterior, agora para identificarmos a probabilidade de um processo presentemente no estado i estar no estado j após um intervalo de tempo t , sendo $P_{ij}(t) = P[X(s+t) = j | X(s)]$.

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, como podemos ver em (2.6).

$$\pi_j \equiv \lim_{t \rightarrow \infty} P_{ij}(t) \quad (2.6)$$

Para que possa existir a possibilidade de probabilidades limite, a cadeia de Markov em causa necessita de ser irreductível, isto é, começando no estado i existe uma probabilidade positiva de alguma vez se estar no etado j , para todo o par de estados (i, j) . Mais, a cadeia de Markov necessita de ser recorrente positiva, ou seja, começando em qualquer estado, o tempo médio para voltar a esse estado terá de ser finito.

Estas probabilidades limite poderão ser calculadas através da resolução das seguintes equações em (2.7) — tanto (a) como (b).

$$(a) \quad q_j \pi_j = \sum_{k \neq j} q_{kj} \pi_k, \quad \text{para todos os estados } j \quad (2.7)$$

$$(b) \quad \sum_j \pi_j = 1$$

As equações que estão designadas em (2.7) são denominadas de **equações de balanço**, isto porque permitem equivaler a taxa à qual o sistema transita do estado j com a taxa à qual o sistema transita para o estado j . Nestas, a probabilidade π_i pode ser interpretada como a proporção de tempo em que o processo está no estado j . Já as probabilidades π_i são vulgarmente denominadas de **probabilidades estacionárias**. Aqui, se o estado inicial for dado pela distribuição $\{\pi_i\}$, então a probabilidade de se estar no estado j no instante t é π_i , para todo o t .

equações de balanço

probabilidades estacionárias

Sistemas de fila de espera

As **filas de espera**, tal como as conhecemos, baseiam-se meramente no conceito de espera em linhas tais como uma portagem de uma autoestrada, um balcão de uma sucursal de um banco, parar num semáforo, entre outros... Basicamente, uma fila de espera é uma

filas de espera

linha de pessoas ou de algo mais abstrato que esperam pelo atendimento de um serviço, este, servido num centro com um ou mais prestadores de serviço.

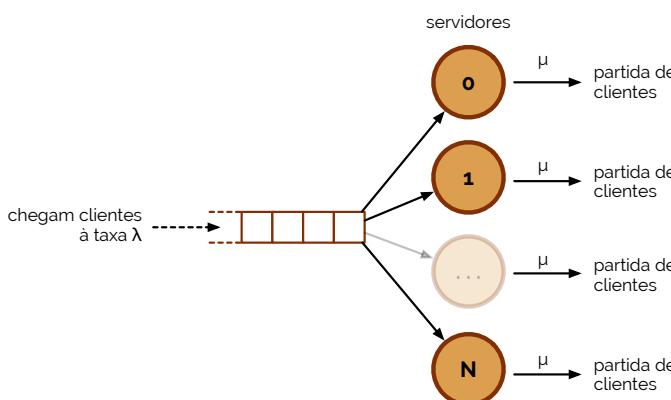
Por exemplo, se houvesse um número infinito de pessoas a atenderem clientes numa sucursal de um banco, então não haveria a necessidade de criar filas de espera, contudo, esta prática teria um custo muito alto a si associado e pode ser considerada como uma prática impossível de se concretizar.

Esta teoria de filas de espera não existe apenas na aplicação de gestão de tempos em bancos ou estradas, mas também dentro da área de redes de computadores. Como já devemos ter experiência, em redes de computadores, por vezes, a velocidade de atendimento de um serviço prestado por um nó da rede poderá levar a congestionamentos na transmissão de pacotes. Neste passo, é importante que existam mecanismos que possam prevenir todo o sistema de perda de pacotes. Estas perdas poderão ocorrer uma vez que cada nó da rede montada necessita de preservar, por vezes, conjuntos de pacotes por questões de ordenação, entre outros... Ao evitar estas perdas, a velocidade de transmissão dentro da rede torna-se diretamente limitada por ela, sendo que não será possível enviar mais tráfego por ligações que se encontram em espera pela transmissão de pacotes anteriores.

Em termos algébricos, um sistema de fila de espera é então caracterizado por um conjunto de c servidores, cada um com capacidade para servir clientes a uma taxa μ e uma fila de espera com uma determinada capacidade, em número de clientes. Como dito, a este sistema chegam clientes a uma taxa dada por μ , sendo que, quando um chega, este começa a ser servido por um servidor se houver algum disponível; caso contrário, este é colocado na fila de espera, se os servidores estiverem todos ocupados. Em suma, os clientes numa fila de espera são atendidos segundo uma disciplina de *first-in-first-out (FIFO)*, como podemos ver representado na Figura 2.2.

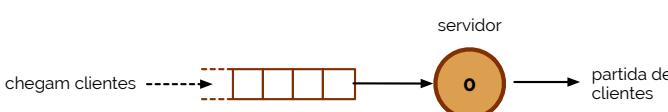
FIFO

figura 2.2



Em termos da complexidade de uma rede de computadores, examinemos, primeiramente, as propriedades de uma fila de espera única e simples, desenvolvendo depois para um sistema constituído por uma rede de filas de espera. Como podemos verificar na Figura 2.3, uma fila de espera simples é constituída por um ou mais servidores e clientes à espera de um serviço, caracterizados por três propriedades: o processo de entrada, o mecanismo de serviço e a disciplina de fila de espera. [3]

figura 2.3



O processo de entrada de uma fila de espera é-nos dado em termos de distribuição de probabilidade dos tempos entre chegadas dos vários clientes. Já o mecanismo de serviço descreve, estatisticamente, as propriedades do processo de serviço. Por fim, a disciplina de fila de espera é a regra que se usar para determinar o quanto é que cada cliente está à espera para ser servido. Para evitar quaisquer ambiguidades na especificação destas características, uma fila de espera é usualmente descrita em termos de uma notação muito conhecida e criada por D. G. Kendall. Na **notação de Kendall**, uma fila de espera é cara-

© David George Kendall
notação de Kendall

terizada por seis parâmetros $A/B/C/K/m/z$, sendo A o processo de chegada (a distribuição de tempo entre chegadas), B o processo de serviço (a distribuição do tempo de serviço), C o número de servidores, K a capacidade máxima da fila de espera (por omissão sendo ∞), m sendo a população de clientes (por omissão é ∞) e z sendo a disciplina de serviço. Contudo, por meios de simplificação, neste documento, iremos considerar que um sistema de fila de espera é apenas representado por quatro parâmetros, sendo eles $A/B/c/d$, sendo A e B iguais, mas c sendo o número de servidores e d sendo a capacidade do sistema em número de clientes (isto é, o número de servidores mais a capacidade da fila de espera) — note-se que quando d é omissivo, diz-se que a fila de espera tem tamanho infinito.

Nesta mesma notação de Kendall, cada um dos parâmetros poderá ser caracterizado segundo um de três valores: D , se se considerar uma lei determinística (constante), isto é, em que as distribuições de tempo entre chegadas e de serviço forem fixas; M , se se considerar uma lei markoviana (ou exponencial), isto é, em que as distribuições de tempo entre chegadas e de serviço são distribuídas exponencialmente; ou G , se se considerar algo genérico, ou seja, se nada é sabido acerca da distribuição de tempo entre chegadas e de serviço. Note-se, não obstante, que existem outros três tipos de valores para estes parâmetros, entre eles GI , E_k e H_k , contudo, porque não os desenvolvemos neste documento, não indicamos o seu valor.

Em questão de uso temos então que, se nos indicarem um sistema de gestão de filas de espera $M/G/4$, isto significa que temos um sistema descrito com um intervalo de tempos de chegada de clientes exponencialmente distribuído, cujos tempos de serviço seguem uma distribuição de probabilidades genérica, com 4 servidores e com um número de clientes infinito.

Para obter o **tempo de espera** ou de fila de espera, geralmente aplica-se um resultado já bastante conhecido em termos de engenharia de filas de espera denominado de **teorema de Little**. O teorema de Little, assim designado depois da sua comprovação pelo seu autor em 1961, relaciona o número médio de clientes numa fila de espera à média da taxa de chegada e ao tempo médio de chegada. Afirma-se assim que, um sistema de fila de espera, com uma taxa de chegada média λ e tempo de espera médio por cliente W , tem um número médio de clientes na fila (ou tamanho médio de fila de espera) L , dado por (2.8).

tempo de espera

teorema de Little

◎ **John Little**

$$L = \lambda W \quad (2.8)$$

Este teorema é bastante generalizado, sendo aplicado a qualquer tipo de sistemas de fila de espera. Assume-se, aqui, que um sistema está num estado de equilíbrio, o que significa que as probabilidades deste estar num estado em particular convergiram e já não deverão mudar com o avançar do tempo.

Com a equação em (2.8) deverá notar-se que permanece válida ainda que as políticas de operação de uma fila de espera se alterem. Por exemplo, o seu valor mantém-se caso estejamos a abordar uma rede de filas de espera e servidores. Também, por analogia, podemos aplicar o conceito deste teorema a conceitos da nossa vida prática. Por exemplo, num dia de chuva, o mesmo tráfego (mesmo λ) é mais lento do que normalmente (W maior) e as ruas estão mais congestionadas (maior L). Outro exemplo, um restaurante de refeições rápidas (W menor) precisa de uma sala menor (L menor) do que um outro normal, para a mesma taxa de chegada de clientes (λ igual).

Consideremos a Figura 2.4 em que temos um gráfico que mostra a evolução dos clientes ao longo do tempo num sistema de fila de espera. Neste mesmo gráfico podemos verificar que temos a impressão do estado dos clientes sob a forma de número de chegadas $A(t)$ e sob a forma do número de partidas $D(t)$. Suponhamos, com ele, que pretendemos manter o registo dos tempos de chegada e de partida dos clientes individuais por um tempo longo t_0 . Se t_0 for grande, então o número de chegadas será aproximadamente igual ao número de partidas. Considerando este número como sendo o nosso $N(t)$, temos (2.9).

$$\text{taxa de chegada} = \lambda_t = \frac{N(t)}{t_0} \quad \lambda = \lim_{t \rightarrow \infty} \lambda_t \quad (2.9)$$

Se subtraímos a curva de partidas à curva de chegadas a cada instante de tempo, podemos obter o número de clientes que estavam presentes no sistema no mesmo momento. Tal área resultante representa, assim, o tempo total perdido dentro do sistema pelo conjunto dos seus clientes. Se denominarmos esta grandeza por W_t temos (2.10).

$$W_t = \frac{\sum_{i=0}^{N(t)} W_i}{N(t)} \quad W = \lim_{t \rightarrow \infty} W_t \quad (2.10)$$

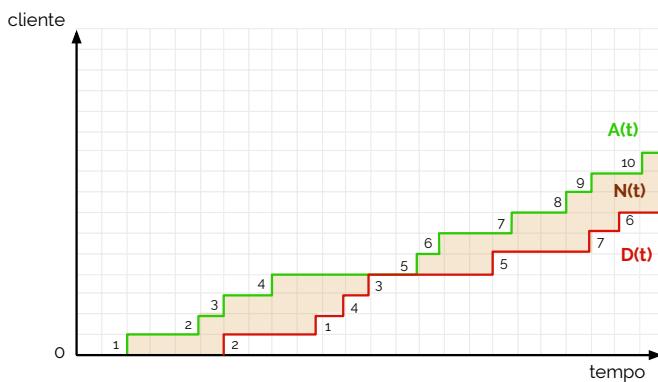


figura 2.4

Já a média temporal do número de clientes observados até ao instante t será atribuída pela área que determinámos anteriormente, conforme podemos verificar em (2.11).

$$L_t = \frac{1}{t} \int_0^t L(\tau) d\tau \quad L = \lim_{t \rightarrow \infty} L_t \quad (2.11)$$

Se considerarmos que temos um sistema de filas de espera em que os clientes chegam um de cada vez e que são servidos, igualmente, um de cada vez, sendo $L(t)$ o número de clientes no sistema no instante t e sendo P_n (com $n \geq 0$) definido por (2.12) 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), com um a_n sendo a proporção de clientes que, ao chegar encontram n clientes no sistema e 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 $a_n = d_n$.

$$P_n = \lim_{t \rightarrow \infty} P[L(t) = n] \quad (2.12)$$

A este resultado damos o nome de **propriedade PASTA**, sendo este acrônimo para *Poisson Arrivals See Time Averages*. Este é, de facto, um dos mais importantes resultados obtidos na teoria de sistemas de filas de espera. Uma aplicação típica é o seguimento de um cliente através do sistema, começando no momento em que este chega até ao momento em que parte de uma fila de espera. Somando todos os atrasos ao longo de toda a rota deste cliente podemos derivar expressões para o tempo de resposta do cliente. Para isto usamos a propriedade PASTA para argumentar que, na chegada, as estatísticas da fila de espera vistas pelo cliente eram idênticas às estatísticas estacionárias da mesma. Em suma, a propriedade PASTA designa que as chegadas de Poisson, em que o tempo de serviço é estatisticamente independente dos instantes de chegada, veem sempre médias temporais $a_n = P_n$.

propriedade PASTA

Tentemos então exemplificar a eficácia desta propriedade apontando alguns contra-exemplos. Primeiro, consideremos 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. Neste caso não podemos aplicar a propriedade PASTA,

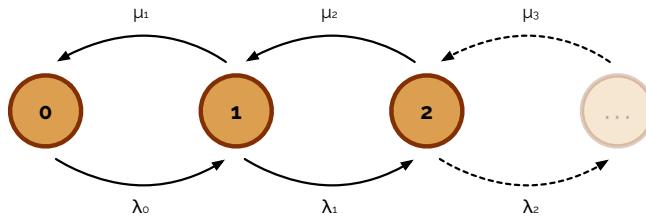
uma vez que o sistema não corrobora o requisito de ser modelado por chegadas de Poisson. Um outro exemplo acontece se considerarmos um sistema em que o processo de chegada de clientes é, de facto, um processo de Poisson e que o tempo de serviço do n -ésimo cliente é igual à metade do intervalo entre a chegada do n -ésimo e do $(n+1)$ -ésimo cliente. Neste caso a propriedade PASTA não terá resultados conclusivos uma vez que o tempo de serviço não é independente das chegadas.

Outro fator que nos é importante destacar quando abordamos a prática e o estudo de sistemas de filas de espera são os **processos de nascimento e morte**. Consideremos assim um sistema cujo estado representa o número de clientes no sistema. Sempre que o sistema tem n clientes tanto chegam novos clientes ao sistema a uma taxa exponencial λ_n , como também partem outros à taxa exponencial μ_n . A este sistema denominamos de processo de nascimento e morte. Aos parâmetros λ_n (com $n = 0, 1, \dots$) e μ_n (com $n = 1, 2, \dots$) são designadas por **taxas de chegada** (ou de nascimento) por **taxas de partida** (ou de morte), respetivamente. Na Figura 2.5 temos assim uma possível representação de um processo de nascimento e morte com as devidas transições do grafo denotadas com as respetivas taxas.

processos de nascimento e morte

taxas de chegada, taxas de partida

figura 2.5



Aplicando as equações de balanço que abordámos anteriormente podemos agora aferir que a taxa de nascimento terá de estar equilibrada com a taxa de morte. Por esta mesma razão, para um estado 0 temos que $\lambda_0\pi_0 = \mu_1\pi_1$, para um estado 1 temos que $(\lambda_1 + \mu_1)\pi_1 = \mu_2\pi_2 + \lambda_0\pi_0$, para um estado 2 temos que $(\lambda_2 + \mu_2)\pi_2 = \mu_3\pi_3 + \lambda_1\pi_1$ e para um estado n temos que $(\lambda_n + \mu_n)\pi_n = \mu_{n+1}\pi_{n+1} + \lambda_{n-1}\pi_{n-1}$.

Através destes resultados podemos obter (2.13) que mostra a distribuição de probabilidade em tempo 0 e n , sendo que em (2.14) temos a condição necessária para a existência de probabilidades-limite.

$$\pi_0 = \frac{1}{1 + \sum_{i=1}^{\infty} \frac{\lambda_0\lambda_1\dots\lambda_{i-1}}{\mu_1\mu_2\dots\mu_i}} \quad (2.13)$$

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

$$\sum_{i=1}^{\infty} \frac{\lambda_0\lambda_1\dots\lambda_{i-1}}{\mu_1\mu_2\dots\mu_i} < \infty \quad (2.14)$$

Sistema de fila de espera M/M/1

Num primeiro sistema de fila de espera a analisar consideremos uma fila de espera modelada como **M/M/1**. Quer M/M/1 dizer que estamos perante um processo de nascimento e morte em que a chegada de clientes é um processo de Poisson com taxa λ , cujo tempo de atendimento de um servidor é exponencialmente distribuído com média $1/\mu$, onde o sistema possui apenas um servidor e acomoda um número infinito de clientes. Na Figura 2.6 podemos ver uma pequena representação deste mesmo sistema.

M/M/1

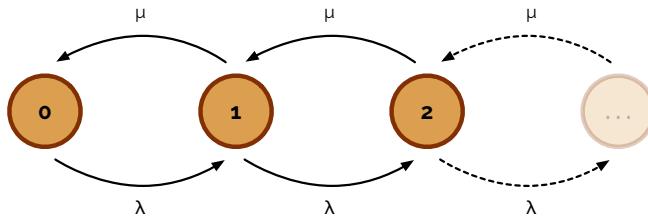


figura 2.6

Neste mesmo sistema já sabemos como é que podemos aferir as probabilidades-limite do estado 0 (conforme vimos em (2.13)) ou estado n (conforme vimos em (2.13) também). Contudo, será que podemos calcular o número médio de clientes no sistema e o atraso médio deste? Ora, para o efetuar precisamos apenas de ter em conta o teorema de Little, tal como vimos anteriormente. Neste teorema vimos que um sistema de fila de espera, com uma taxa de chegada média λ e tempo de espera médio por cliente W , tem um número médio de clientes na fila (ou tamanho médio de fila de espera) L . Neste caso, para o nosso primeiro resultado necessitamos de obter o valor de L , ou seja, o número médio de clientes do sistema, sendo este produto de uma relação entre o estado presente e a sua probabilidade de estado, o que resulta em (2.15).

$$L = \sum_{n=0}^{\infty} n P_n = \frac{\lambda}{\mu - \lambda} \quad (2.15)$$

Já no segundo caso, para obter o atraso médio no sistema precisamos de saber W . Ora, seguindo acordando com o teorema de Little, temos que $W = L/\lambda$, pelo que sabendo W e λ podemos obter o atraso conforme (2.16).

$$W = \frac{L}{\lambda} = \frac{1}{\mu - \lambda} \quad (2.16)$$

Acabámos de definir o número médio de clientes e o atraso médio de um sistema que seja coordenado segundo uma fila de espera do tipo M/M/1, mas e a fila em si, como é que se comporta em termos de número médio de clientes e atraso médio? Ora, para começarmos podemos tentar aferir qual o atraso médio na fila de espera, sendo este um valor obrigatoriamente menor que o atraso do sistema visto como um todo. Assim sendo, definimos atraso médio na fila de espera W_Q como sendo a diferença entre W e o tempo de atendimento $1/\mu$, como podemos ver em (2.17).

$$W_Q = W - \frac{1}{\mu} = \frac{\lambda}{\mu(\mu - \lambda)} \quad (2.17)$$

Por outro lado, no que toca ao número médio de clientes em fila de espera, numa aplicação direta do teorema de Little, temos (2.18).

$$L_Q = \lambda W_Q = \frac{\lambda^2}{\mu(\mu - \lambda)} \quad (2.18)$$

Sistema de fila de espera M/M/m

Com o sistema de fila de espera da secção anterior, se modificarmos o número de servidores em atendimento para m obtemos uma sistema classificado como **M/M/m**. Um sistema de fila de espera classificado como $M/M/m$ é então um processo de nascimento e morte em que a chegada dos clientes é um processo de Poisson com taxa λ , onde o tempo de atendimento de um servidor é exponencialmente distribuído com média $1/\mu$ e o sistema possui m servidores, acomodando um número infinito de clientes. Uma possível represen-

M/M/m

tação para este tipo de sistema de fila de espera poderá ser visualizado na Figura 2.7, que difere da Figura 2.6 no limite de representação para as transições de morte μ .

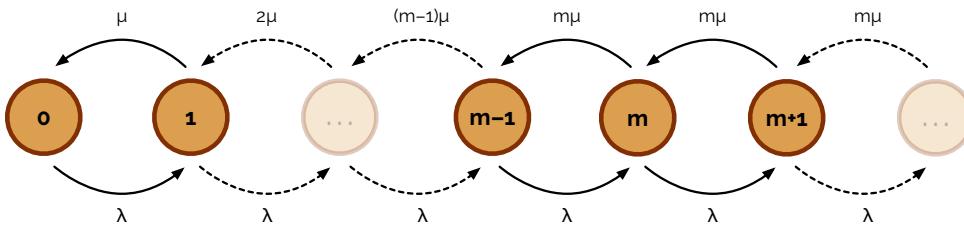


figura 2.7

Neste sistema possuímos, então, novas equações de balanço. Assim sendo, temos que $\lambda P_{n-1} = n\mu P_n$, $n \leq m$ e $\lambda P_{n-1} = m\mu P_n$, para $n > m$. Isto acontece uma vez que até ao estado m poderemos ter sempre mais um servidor em atendimento. Como todos eles atendem os pedidos à taxa μ , então teremos $m\mu$ como sendo a taxa do sistema. Em estado estacionário, temos que a probabilidade de haver n clientes poderá ser obtida da forma explícita em (2.19), onde inicialmente temos a descrição de π_0 e depois do subsequente π_n .

$$\pi_0 = \frac{1}{\sum_{n=0}^{m-1} \frac{(m\rho)^n}{n!} + \frac{(m\rho)^m}{m(1-\rho)}}, \text{ com } \rho = \frac{\lambda}{m\mu} < 1 \quad (2.19)$$

$$\pi_n = \begin{cases} \pi_0 \frac{(m\rho)^n}{n!} & \text{se } n \leq m \\ \pi_0 \frac{m^m \rho^n}{m!} & \text{se } n > m \end{cases}, \text{ com } n \geq 1$$

Se todos os servidores estiverem a ser usados, ou seja, se o sistema estiver num estado tal em que se encontra em m ou superior, então a sua probabilidade de se encontrar neste estado poder-se-á calcular através de (2.20), obtendo a expressão a que damos o nome de **fórmula de Erlang C**.

fórmula de Erlang C

◎ Agner Erlang

(2.20)

$$P_Q = \sum_{n=m}^{\infty} \frac{\pi_0 (m\rho)^m}{m! (1-\rho)}$$

Voltando a caracterizar o sistema e a fila de espera, tal como fizemos na secção anterior para o sistema M/M/1, agora, o número médio de clientes na fila de espera num sistema M/M/ m pode ser obtido através de (2.21).

$$L_Q = \sum_{n=0}^{\infty} n P_0 \frac{m^m \rho^{m+n}}{m!} = P_Q \frac{\rho}{1-\rho} \quad (2.21)$$

O atraso médio na fila de espera pode ser obtido através de uma aplicação direta do teorema de Little, da mesma forma que verificámos na secção anterior. Obtemos assim (2.22).

$$W_Q = \frac{L_Q}{\lambda} = P_Q \frac{\rho}{\lambda (1-\rho)} \quad (2.22)$$

Olhando agora para o sistema em si, temos que para obter o atraso médio no sistema temos de somar ao W_Q que vimos em (2.22) a taxa de atendimento $1/\mu$. Para isso, temos (2.23) onde desenvolvendo o resultante de Q conseguimos uma expressão que se mostra mais legível, em comparação ao caso do sistema M/M/1.

$$W = \frac{1}{\mu} + W_Q = \frac{1}{\mu} + \frac{P_Q}{m\mu - \lambda} \quad (2.23)$$

Por fim, caraterizando o número médio de clientes no sistema podemos aplicar diretamente o teorema de Little no que surgirá (2.24).

$$L = \lambda W = m\rho + \frac{\rho P_Q}{1 - \rho} \quad (2.24)$$

Sistema de fila de espera M/M/1/m

Até agora já caraterizámos sistemas de fila de espera com um ou mais servidores, sendo que nestes considerámos sempre que o número de clientes era infinito. O que é poderá mudar se tornarmos o número de clientes finito? Consideremos, portanto, o sistema de fila de espera **M/M/1/m**, sistema esse caraterizado pela chegada de clientes ser um processo de Poisson com taxa λ , pelo tempo de atendimento de um servidor ser exponencialmente distribuído com média $1/\mu$, por possuir 1 servidor e por acomodar, no máximo, m clientes (tendo capacidade para $m - 1$ clientes).

M/M/1/m

Na Figura 2.8 podemos ver uma simples representação deste tipo de sistema de fila de espera.

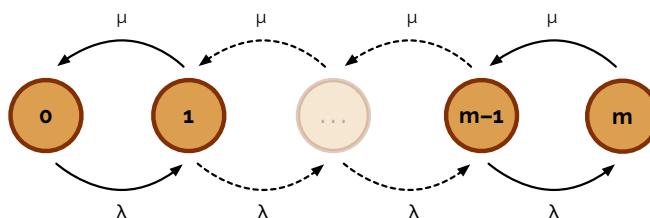


figura 2.8

Neste sistema as nossas equações de balanço são bastante simples, uma vez que de forma semelhante à fila M/M/1, $\lambda P_{n-1} = \mu P_n$, com $n = 1, 2, \dots, m$.

A probabilidade de n clientes no sistema atingirem um estado estacionário é dado por (2.25).

$$\pi_n = \frac{\left(\frac{\lambda}{\mu}\right)^n}{\sum_{i=0}^m \left(\frac{\lambda}{\mu}\right)^i}, \text{ com } n = 0, 1, \dots, m \quad (2.25)$$

Pela propriedade PASTA podemos afirmar que o sistema encontra-se cheio assim que atinge o estado m , isto porque corresponde ao estado que não possui mais servidores disponíveis para efetuar o atendimento.

Sistema de fila de espera M/M/m/m

Num sistema de fila de espera do tipo **M/M/m/m** temos que se carateriza por ser um processo de nascimento e morte em que a chegada de clientes é um processo de Poisson com taxa λ , o tempo de atendimento de um servidor é exponencialmente distribuído com média $1/\mu$, o sistema possui m servidores e acomoda, no máximo, m clientes (não possui qualquer fila de espera). Na Figura 2.9 podemos ver uma representação algo simplificada deste mesmo sistema de fila de espera.

M/M/m/m

Uma vez que agora voltamos a ter um conjunto de m servidores a efetuarem atendimentos aos vários clientes, então as nossas equações de balanço tornam-se bastante simples, sendo $\lambda P_{n-1} = \mu P_n$, com um $n = 1, 2, \dots, m$. Isto acontece mais uma vez porque sendo a taxa de atendimento a mesma para qualquer um dos servidores, então se tivermos m a atender, a taxa de atendimento do sistema será de $m\mu$.

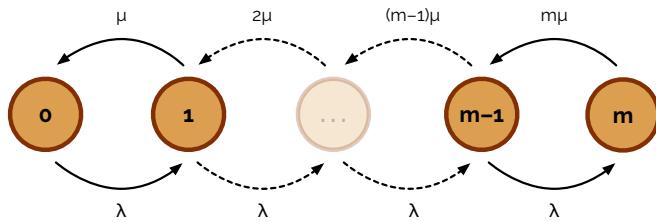


figura 2.9

Neste sistema, a probabilidade de n clientes no sistema estarem em estado estacionário é dada por (2.26).

$$\pi_n = \frac{\left(\frac{\lambda}{\mu}\right)^n}{\frac{n!}{\sum_{i=0}^m \left(\frac{\lambda}{\mu}\right)^i / i!}}, \text{ com } n = 0, 1, \dots, m \quad (2.26)$$

Pela propriedade PASTA, assim que chegarmos ao estado m podemos concluir que o nosso sistema se encontra cheio (neste caso significa tanto que não há mais ninguém para atender, como também que não há mais ninguém a ser atendido, sendo que o número de clientes é o mesmo que o de servidores). Contudo, neste estado, podemos obter a **fórmula de Erlang B**, reconhecida e simplificada em (2.27).

fórmula de Erlang B

$$\pi_m = \frac{\left(\frac{\lambda}{\mu}\right)^m / m!}{\sum_{i=0}^m \left(\frac{\lambda}{\mu}\right)^i / i!} \quad (2.27)$$

Sistema de fila de espera M/G/1

Como vimos na apresentação da notação de Kendall nem todos os sistemas de fila de espera são necessariamente definidas como sendo markovianas. Pelo contrário, um outro tipo com que poderão estar definidas é como sendo genéricas. Neste caso, analisando um sistema do tipo **M/G/1** temos um sistema caracterizado por uma chegada de clientes como sendo um processo de Poisson com taxa λ , pelo tempo de atendimento S do servidor sendo uma distribuição genérica e independente das chegadas dos clientes, por ter apenas um servidor em atendimento e por conseguir acomodar um número infinito de clientes.

M/G/1

Sendo conhecidos o valor do tempo de atendimento esperado $E[S]$ e da sua variância $E[S^2]$ podemos aplicar a **fórmula de Pollaczek-Khintchine** que enuncia que o atraso médio na fila de espera é dado por (2.28).

fórmula de
Pollaczek-Khintchine
(2.28)

- ◎ Felix Pollaczek
- ◎ Aleksandr Khintchine

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

Por outro lado, para obter o atraso médio no sistema podemos verificar o valor resultante de (2.29), sendo este apenas a soma do tempo de atendimento do sistema à fila.

$$W = \frac{\lambda E[S^2]}{2(1 - \lambda E[S])} + E[S] \quad (2.29)$$

Esta aproximação não é, em nada, distante dos casos estudados de sistemas de fila de espera M/M/1 e M/M/1/m, pelo que quando o tempo de serviço é exponencialmente dis-

tribuído (como era o caso de ambos), o sistema terá $E[S] = 1/\mu$ (e variância $2/\mu^2$) pelo que obtemos um valor de W_Q representável por (2.30).

$$W_Q = \frac{\lambda}{\mu(\mu - \lambda)} \quad (2.30)$$

Por outro lado, quando os tempos de serviço são iguais para todos os clientes com valor $1/\mu$, então o sistema diz-se resultar num **M/D/1**, pelo que obtemos um valor de variância $E[S^2] = 1/\mu^2$. Neste caso o atraso da fila é dado por (2.31).

M/D/1

$$W_Q = \frac{\lambda}{2\mu(\mu - \lambda)} \quad (2.31)$$

Diz-se, então, que uma ligação ponto-a-ponto com capacidade de μ pacotes por segundo com uma fila de espera muito grande onde chegam pacotes a uma taxa de λ (de Poisson) pacotes por segundo é um sistema M/G/1, sendo que: se o comprimento dos pacotes for exponencialmente distribuído então este degenera num M/M/1; e se o comprimento dos pacotes for fixo, degenera num sistema M/D/1. [3]

3. Simulações Baseadas em Eventos Discretos

Por vezes poderá acontecer estar perante um cenário em que se desenvolve uma solução para um problema que possui uma escala muito grande para que o seu teste seja realizado num ambiente controlado, mas real. Consideremos o caso de um servidor de uma emissão de vídeo por um conjunto de máquinas espalhadas pelo mundo — testar este conceito ou qualquer possível solução que possa partir deste problema é inaceitável uma vez que dificilmente se iria concluir sobre as garantias de implementação e segurança da mesma (para além do custo que estaria envolvido).

Como solução a este novo problema, as **simulações** poderão aparecer e ser realizadas, como dito, num ambiente controlado, mas virtualizado, olhando apenas para os vários impactos das várias variáveis às quais um determinado sistema está sensível. Note-se que para que isto possa acontecer há que ter o domínio total (ou quase total) das variáveis que possam causar maior interferência no sistema.

simulações

Elementos e estrutura de um simulador baseado em eventos discretos

Ora, estamos a referir simuladores, mas como temos controlo das várias variáveis, então podemos referir simuladores baseados em eventos discretos. Basicamente, queremos efetuar uma 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 se diz ocorrer um **evento**.

evento

Para que isto possa acontecer um conjunto de elementos necessitarão de estar presentes, sendo eles: variáveis de estado, para descrever o estado do sistema num instante de tempo t ; contadores estatísticos, sendo estes variáveis que armazenam informação estatística relativa ao desempenho do sistema; um relógio de simulação, para indicar, a cada instante, o tempo de simulação, sendo este diferente do tempo de computação; eventos, sendo os vários tipos de ocorrências instantâneas que alteram o estado do sistema e/ou os contadores estatísticos; e uma lista de eventos, sendo um conjunto onde são armazenados os instantes de ocorrência dos eventos futuros.

Na Figura 3.1 podemos ver a esquematização de uma estrutura simplificada de um simulador baseado em eventos discretos. Nesta mesma representação primeiramente inicializamos as variáveis de estado, os contadores estatísticos e a lista de eventos com o(s) primeiro(s) evento(s). De seguida, determina-se o próximo evento da lista de eventos.

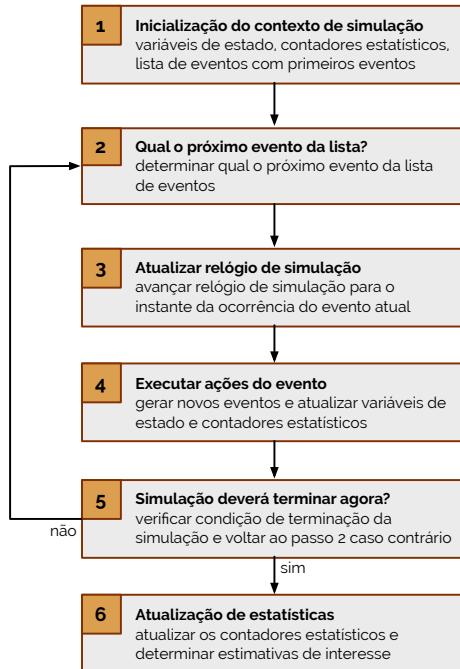


figura 3.1

Tendo o evento E sido determinado como o evento a ser realizado, então no ciclo seguinte atualiza-se o relógio de simulação para o instante de ocorrência de E , sendo, num quarto ciclo, executadas as ações associadas com E , como a geração de novos eventos e atualização das variáveis de estado e dos contadores estatísticos. Terminado isto deverá-se verificar se a simulação atingiu o seu fim ou não. Caso não tenha atingido, então deverá recuperar-se o segundo passo para voltar a iterar sobre os eventos. Caso contrário, a simulação devirá então terminar, mas antes deverá atualizar os contadores estatísticos e determinar as estimativas de interesse.

Consideremos para o caso, um sistema constituído por um servidor de emissão de vídeos que se pretende simular [7]. Como parâmetros de entrada do simulador teremos o tempo entre os pedidos de filmes (como sendo uma variável exponencialmente distribuída com média de 60 minutos), a duração média dos filmes (como sendo uma variável exponencialmente distribuída com média de 90 minutos e a capacidade da ligação de rede do servidor (sendo de 2 filmes). Como critério de paragem de simulação, consideraremos o instante de tempo em que se realiza o quarto pedido de filme (este pedido ainda deverá contar para os contadores estatísticos). Por fim, pretendem-se estimar a probabilidade de bloqueio (percentagem de pedidos de filmes que são recusados porque a ligação de rede está completamente ocupada) e a ocupação média da ligação de rede (em número de filmes).

Dado este nosso cenário de simulação, que eventos, variáveis de estado e contadores estatísticos é que possuímos? Ora, como eventos podemos designar a **CHEGADA** como sendo um pedido de um filme e a **PARTIDA** como sendo a terminação de um filme que estava em transmissão. Como variáveis de estado podemos considerar somente um **ESTADO** que cumprirá a tarefa de guardar o número de filmes em transmissão. E como contadores estatísticos podemos ter o **OCUPAÇÃO** que registará o integral da ocupação da ligação (em número de filmes, conforme pedido), desde o início da simulação até ao instante presente, um **PEDIDOS_RECUSADOS** sendo o número de pedidos de filmes recusados até ao instante presente e o **PEDIDOS**, sendo o número total de filmes pedidos até ao instante presente [7].

Geração de números aleatórios

Numa simulação, uma das questões mais importantes e mais necessárias é a **geração de números aleatórios**. Esta é uma questão muito forte e muito difícil de se responder simplesmente porque a entropia não é algo que seja fácil de reproduzir no contexto de um sistema computacional, como vimos em Métodos Probabilísticos para Engenharia Infor-

geração de números
aleatórios

mática (a2s1). Contudo, dependendo do intervalo de valores que se pretende obter, vários geradores já são considerados bons para a tarefa.

Um dos geradores mais populares, com distribuição uniforme entre 0 e 1, são os geradores lineares congruenciais **LCG** (sigla de *Linear Congruential Generator*). Neste tipo de procedimento inicialmente geram-se os inteiros Z_1, Z_2, \dots de acordo com a fórmula recursiva $Z_i = (aZ_{i-1} + c) \pmod{m}$, onde m, a, c e Z_0 são inteiros não-negativos e faz-se $U_i = Z_i/m$. Este tipo de gerador é usado em máquinas de linguagens como a MATLAB, neste caso, com um valor de m igual a $2^{31}-1$.

Não obstante, é possível gerar valores com outras distribuições, como com variáveis discretas. Consideremos, para o efeito, que a variável aleatória pode assumir os valores X_1, X_2, \dots, X_n . A probabilidade do valor X_i é igual a $P(X = X_i) = f_i$. O método, neste caso, é dividir o intervalo $[0, 1]$ em n intervalos proporcionais a f_i , com $i = 1, \dots, n$. De seguida, gerar um valor aleatório U em $[0, 1]$ com distribuição uniforme. No fim, há que retornar X_i se U estiver no i -ésimo intervalo.

Se a distribuição-base forem variáveis contínuas, então uma possibilidade é usar o método da **transformação inversa**. Consideremos $F(X)$ como sendo a função distribuição de uma variável aleatória contínua X . Aqui, $F^{-1}(U)$ será a sua função inversa. O método a ser aplicado descreve-se em gerar $U \sim U(0,1)$ e retomar $X = F^{-1}(U)$ [7,8].

LCG**transformação inversa**

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 μ e variância σ^2 finitas, sendo resultado de diferentes réplicas da simulação do sistema. A **média amostral** definida por (3.1) é um estimador de μ .

média amostral

$$\bar{X}(n) = \frac{\sum_{i=1}^n X_i}{n} \quad (3.1)$$

Por outro lado, a variância amostral definida por (3.2) é um estimador de σ^2 .

$$S^2(n) = \frac{\sum_{i=1}^n (X_i - \bar{X}(n))^2}{n-1} \quad (3.2)$$

A análise dos resultados de uma simulação é feita, normalmente, com base no teorema do limite central em que, seja Z_n a variável aleatória dada por (3.3) e seja $F_n(z)$ a sua função distribuição para uma amostra de tamanho n , o **teorema do limite central** enuncia que a tendência de n aumentar fará com que a distribuição seja $\Phi(z)$, sendo esta a função distribuição de uma variável aleatória gaussiana (ou normal) padrão, ou seja, em que a média é nula e o desvio-padrão é unitário.

teorema do limite central

$$Z_n = \frac{\bar{X}(n) - \mu}{\sqrt{\sigma^2/n}} \quad (3.3)$$

Uma vez que, quanto maior for o n , mais $S^2(n)$ é σ^2 então a variável aleatória de (3.4) possui uma distribuição aproximadamente gaussiana-padrão.

$$\frac{\bar{X}(n) - \mu}{\sqrt{S^2(n)/n}} \quad (3.4)$$

Para um n suficientemente elevado, então (3.5) ocorre, onde $Z_{1-\alpha/2}$ é o ponto crítico da distribuição gaussiana-padrão, sendo o valor de z tal que $P(x \leq z) = 1 - \alpha/2$ em que x é uma variável aleatória com distribuição gaussiana-padrão.

$$\begin{aligned} P\left(-Z_{1-\alpha/2} \leq \frac{\bar{X}(n) - \mu}{\sqrt{S^2(n)/n}} \leq Z_{1-\alpha/2}\right) &= \\ &= P\left(\bar{X}(n) - Z_{1-\alpha/2}\sqrt{S^2(n)/n} \leq \mu < \bar{X}(n) + Z_{1-\alpha/2}\sqrt{S^2(n)/n}\right) \approx 1 - \alpha \end{aligned} \quad (3.5)$$

Assim, o intervalo de confiança aproximado a $100(1 - \alpha)\%$ para μ é dado por (3.6).

$$\bar{X}(n) \pm Z_{1-\alpha/2}\sqrt{S^2(n)/n} \quad (3.6)$$

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, isto é, que dependem das condições iniciais, antes de atingir o regime estacionário. Para garantir que os parâmetros de desempenho sejam corretamente calculados, é necessário deixar aquecer a simulação (terminologia aproximada de *warm-up*, do termo original), dando tempo para que os regimes transitórios iniciais se extingam. Se o tempo simulado for muito superior ao tempo de *warm-up*, então os contadores estatísticos poderão ser inicializados no início da simulação. Caso contrário, os contadores deverão ser inicializados apenas após o tempo de *warm-up* (que deverá ser previamente estimado) [7,8].

4. Partilha de Recursos de Comunicação

Numa rede vários nós comunicam entre si. Para que essa comunicação possa ser feita, grande parte das vezes vários canais de comunicação são usados em simultâneo, através de processos de multiplexagem. Não podendo (nem havendo grande predisposição dado o custo) de evitar estes procedimentos, esta prática possui algumas “desvantagens”, sendo uma delas não haver uma concordância única por ligação estabelecida sobre a percentagem dos recursos usados ao longo de uma conexão.

Em Arquiteturas de Redes Avançadas (a4s1) tivemos, inclusivamente, a oportunidade de estudar redes MPLS. Uma rede MPLS é um exemplo tal que permite a criação de vários caminhos entre vários nós, permitindo várias conexões em simultâneo.

Contexto, canais de comunicação, recursos e circuitos

Um **recurso de comunicações** digital é um elemento de rede que permite a traçoação de informação em formato digital. Uma das formas possíveis de ver este elemento é como um túnel através do qual podem ser transmitidas/recebidas determinadas quantidades/sequências de bits. Estas sequências/quantidades formam um valor a que damos o nome de **capacidade de recurso**. Por exemplo, uma interface de rede de um servidor de vídeo é um recurso de comunicações para os terminais de vídeo. Um outro exemplo poderá ser uma ligação de um *router* de uma casa ao ISP ser constituído por dois recursos de comunicação, um em cada sentido, cuja capacidade é normalmente diferente em cada um destes.

Um recurso de comunicação poderá ser explicitamente estruturado em **circuitos** (também denominados de canais), que correspondem a partições da capacidade total do recurso. Este circuito é caracterizado pela sua **largura de banda** (um valor em bits por segundo) e a sua estruturação numa ligação ponto-a-ponto poderá ser feita, como já tivemos oportu-

recurso de comunicações

capacidade de recurso

circuitos

largura de banda

tunidade de referenciar, através de variadas formas de multiplexagem, entre elas no tempo (TDM), na frequência (FDM) ou até mesmo nos comprimentos de onda (WDM).

Referimos que um recurso de comunicações poderia ser explicitamente estruturado em circuitos, isto porque também poderá ser implicitamente usado como se um fosse estruturado em circuitos, isto é, teremos circuitos virtuais. Neste contexto, cada comunicação usa uma partição da capacidade do recurso.

Uma **chamada**, quando é estabelecida, corresponde à atribuição temporária de um circuito (ou mais) com uma determinada largura de banda. Disponível e acessível em ambos os sentidos, 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. Deste modo, um recurso de comunicações pode ser partilhado por diferentes **fluxos de chamadas**.

Denominamos também por **classe de serviço** a identificação de um fluxo de chamadas com as mesmas características de tráfego e os mesmos requisitos de largura de banda.

As características de tráfego são determinadas através da descrição estocástica do processo de chegada de chamadas, da descrição estocástica do tempo das chamadas e da largura de banda (em número de circuitos) requerida por cada chamada. Note-se, que neste contexto, o parâmetro de qualidade mais importante é a **probabilidade de bloqueio**, sendo esta a probabilidade de um pedido de chamada não ser aceite pela ligação por falta de recursos disponíveis).

Um recurso de comunicações poderá ser caracterizado como **uni-serviço** se suportar apenas fluxos de chamadas de uma única classe de serviço ou **multi-serviço** se suportar fluxos de chamadas de diferentes classes de serviço.

A diferenciação de serviço faz-se através da função de controlo de fluxos (também designada de controlo de admissão de chamadas). [9]

Comunicações uni-serviço

Durante a segunda secção anterior vimos algumas situações onde se aplicam comunicações uni-serviço. Por exemplo, quando estudámos o sistema de fila de espera $M/M/m/m$ verificámos que os pacotes, aqui, chegavam segundo um processo de Poisson com uma taxa λ e com um atendimento exponencialmente distribuído com média de $1/\mu$.

Aplicando a este exemplo um fluxo de chamadas, isto é, um conjunto de chamadas que irão partilhar o mesmo canal de comunicação, temos então que iremos ter um grupo de m circuitos ao qual é oferecido um fluxo de chamadas de Poisson com a mesma taxa λ e em que a duração das chamadas é exponencialmente distribuído com média $1/\mu$. Mais, vimos também que à relação entre a taxa de chegada e a taxa de atendimento damos o nome de taxa de ocupação, no nosso caso, mais adequadamente, **intensidade de tráfego**, unidade medida em **Erlang**. Foi assim que concluímos e chegámos à expressão de Erlang B, passível de ser vista em (2.27), que nos fornece a capacidade de calcularmos a probabilidade de bloqueio.

Consideremos agora um recurso de comunicações com capacidade para N circuitos e que serve M clientes (respeitando uma relação de $M > N$). Cada um dos M clientes encontra-se inativo durante um período de tempo exponencialmente distribuído com média $1/\lambda$ e gera uma chamada com uma duração de $1/\mu$. A cada chamada pedida é atribuído um dos N circuitos disponíveis, sendo que se não houver qualquer circuito disponível, então a chamada é bloqueada.

Este sistema poderá ser modelado por um processo de nascimento e morte com estados $n = 0, 1, \dots, N$, representando o número de circuitos ocupados e com taxas de nascimento e morte dadas por (4.1).

$$\begin{aligned}\lambda_n &= (M - n)\lambda_1, & 0 \leq n < N - 1 \\ \mu_n &= n\mu, & 1 \leq n < N\end{aligned}\tag{4.1}$$

Deste modo e nestas condições, podemos aplicar a **fórmula de Engset** de (4.2) para obter a probabilidade de n chamadas no sistema.

$$\pi_n = \frac{\binom{M}{n} \left(\frac{\lambda}{\mu}\right)^n}{\sum_{n=0}^N \binom{M}{n} \left(\frac{\lambda}{\mu}\right)^n} \quad (4.2)$$

Em (4.3) podemos ver como aferir a probabilidade de bloqueio. Note-se que a propriedade PASTA já não se verifica neste caso, sendo que a taxa de chegada de chamadas não é estatisticamente independente do estado do sistema. [9]

$$P_b = \frac{\binom{M-1}{N} \left(\frac{\lambda}{\mu}\right)^N}{\sum_{n=0}^N \binom{M-1}{n} \left(\frac{\lambda}{\mu}\right)^n} \quad (4.3)$$

Comunicações multi-circuito

Se ao invés de um único circuito tivermos vários (C circuitos) para servir K classes de serviço e se a cada classe k estiver associada uma taxa de chegada λ_k , um tempo médio de serviço $1/\mu_k$ e uma largura de banda b_k (em número de circuitos), as chamadas das K classes chegam de acordo com processos independentes de Poisson à taxa de λ_k . Aqui, uma chamada de 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 q_k a intensidade de tráfego de cada classe, isto é, $q_k = \lambda_k/\mu_k$ e assuma-se que os tempos de serviço das chamadas são independentes entre si e independentes dos processos de chegadas. Se n_k for o número de chamadas da classe k no sistema e $n = (n_1, \dots, n_k)$ e $b = (b_1, \dots, b_k)$ vetores, então o número total de circuitos ocupados no sistema é dado por (4.4).

$$b \cdot n = \sum_{k=1}^K b_k n_k \quad (4.4)$$

Assim, com (4.4), podemos afirmar que uma chamada de classe k é admitida no sistema se $b_k \leq C - b \cdot n$, caso contrário é bloqueada e perdida.

O espaço de estados do processo de nascimento e morte multidimensional, portanto, é definido como $S = \{n \in I^K : b \cdot n \leq C\}$ (onde I é o conjunto dos inteiros não-negativos). Assim, seja S_k o subconjunto dos estados para os quais uma chamada da classe k é admitida quando chega à rede, isto é, $S_k = \{n \in S : b \cdot n \leq C - b_k\}$. Aqui as probabilidades limite de cada estado são dadas por (4.5).

$$P(n) = \frac{1}{G} \prod_{l=1}^K \frac{\rho_l^{n_l}}{n_l!} \quad , \text{ com } n \in S \text{ e onde } G = \sum_{n \in S} \prod_{l=1}^K \frac{\rho_l^{n_l}}{n_l!} \quad (4.5)$$

Neste mesmo contexto, a probabilidade de bloqueio é dada por (4.6), o que mostra o complemento da probabilidade dos estados para os quais uma chamada da classe k é admitida. Mostra-se, então, a equivalência desta expressão com a probabilidade dos estados $S \setminus S_k$ (estados para os quais uma chamada da classe k é rejeitada). [9]

$$B_k = 1 - \frac{\sum_{n \in S_k} \prod_{l=1}^K \frac{\rho_l^{n_l}}{n_l!}}{\sum_{n \in S} \prod_{l=1}^K \frac{\rho_l^{n_l}}{n_l!}} \Leftrightarrow B_k = \frac{\sum_{n \in S \setminus S_k} \prod_{l=1}^K \frac{\rho_l^{n_l}}{n_l!}}{\sum_{n \in S} \prod_{l=1}^K \frac{\rho_l^{n_l}}{n_l!}} \quad (4.6)$$

Recursos de comunicação com comutação de pacotes

Numa rede com **comutação de pacotes** as coisas são ligeiramente diferentes. Apesar do conceito de classe de serviço se aplicar para **fluxos de pacotes** (repare-se que já não estamos a referir fluxos de chamadas), um dos parâmetros de qualidade de serviço mais importantes é o **atraso médio** por pacote, entre outros como a variância do atraso ou a percentagem de perda de pacotes.

Neste contexto de fluxos de pacotes há, primeiro, que saber distinguir duas situações muito próprias: quando é que estamos a referir multiplexagem estatística e quando é que pretendemos referir multiplexagem determinística.

Vejamos bem: se só tivermos um único circuito inserido numa ligação, mas neste efectuarmos um conjunto de comunicações entre nós distintos, então haverá a necessidade de ser executado um procedimento de multiplexagem entre os vários pacotes provenientes das várias entidades comunicantes. A esta aplicação de multiplexar diferentes fluxos de pacotes num só circuito damos o nome de **multiplexagem estatística**.

Por outro lado, quando temos um conjunto de circuitos pelos quais teremos de distribuir fluxos de pacotes, então estamos a referir **multiplexagem determinística**.

Em ambos os casos, as características de tráfego são determinadas pela descrição estocástica do processo de chegada dos pacotes e pela descrição estocástica do tamanho dos pacotes.

Para que possamos prosseguir, primeiramente analisando o caso da multiplexagem estatística, consideremos um sistema de fila de espera em que as chegadas de pacotes são processos de Poisson, o tamanho dos pacotes é exponencialmente distribuído e a fila de espera é atendida com uma disciplina *First-In-First-Out*, mais vulgarmente denominada de **FIFO**. Se esta fila de espera possuir um tamanho infinito, então o sistema de fila de espera é modelado pelo sistema M/M/1, em que a taxa de chegadas é a soma das várias taxas e a taxa de atendimento dos pacotes é a razão da capacidade C da ligação (largura de banda em bits por segundo) pelo tamanho médio por pacote (em bits).

Na Figura 4.1 podemos ver uma pequena representação deste conceito, com dois fluxos de pacotes.

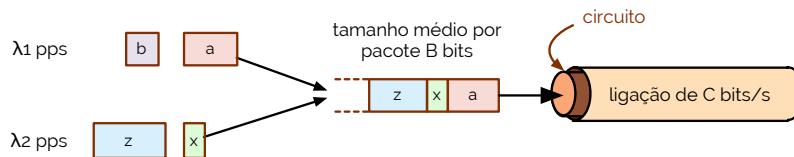


figura 4.1

Uma vez que estamos a referir um sistema modelado por M/M/1, então o seu atraso é dado por vias da expressão $1/(\mu - \lambda)$, o que, substituindo pelas variáveis da nossa representação ficamos com (4.7).

$$W = \frac{1}{\mu - \lambda} \therefore W = \frac{1}{C/B - (\lambda_1 + \lambda_2)} \quad (4.7)$$

Se ao invés da modelação por M/M/1 for realizada uma modelação pelo sistema M/M/1/ m , temos um sistema de fila de espera finita com capacidade para m pacotes, onde a percentagem de perda de pacotes é dada por (4.8).

$$\mu_m = \frac{(\lambda/\mu)^m}{\sum_{i=0}^m (\lambda/\mu)^m} \quad (4.8)$$

Por aplicação do teorema de Little podemos descrever o atraso médio dos pacotes em M/M/1/ m . Assim sendo, veja-se (4.9).

comutação de pacotes
fluxos de pacotes

atraso médio

multiplexagem estatística
multiplexagem determinística

FIFO

$$W = \frac{L}{\lambda(1 - \mu_m)} = \frac{\sum_{i=0}^m i \times \pi_i}{\lambda(1 - \mu_m)} = \frac{1}{\lambda(1 - \mu_m)} \times \frac{\sum_{i=0}^m i \times (\lambda/\mu)^i}{\sum_{i=0}^m (\lambda/\mu)^i} \quad (4.9)$$

Dada a implementação deste tipo de multiplexagem podemos concluir que se a fila de espera for de tamanho infinito, então o sistema poderá ser modelado por M/M/1. Contudo, se a fila de espera tiver capacidade para m pacotes, então o sistema deverá ser modelado por M/M/1/ m .

Se ao invés da multiplexagem estatística tivermos num contexto de multiplexagem determinística, então é porque o nosso conjunto de fluxos de pacotes está para ser distribuída entre um outro conjunto de circuitos (conjunto com mais do que uma unidade de circuitos). A Figura 4.2 tenta criar uma possível representação deste tipo de multiplexagem.

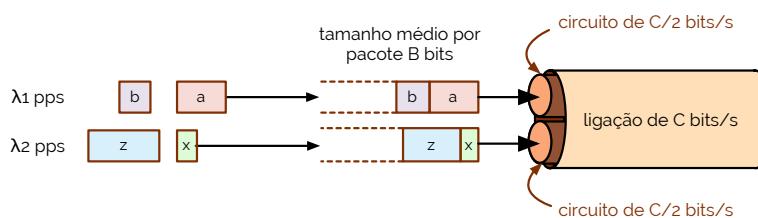


figura 4.2

No esquema de multiplexagem determinística da Figura 4.2 podemos ver cada uma das filas como um sistema à parte, com $C/2$ bits por segundo. Assim sendo, o atraso do sistema 1 e do sistema 2 serão os expressos em (4.10).

$$W_1 = \frac{1}{C/2B - \lambda_1} \quad W_2 = \frac{1}{C/2B - \lambda_2} \quad (4.10)$$

Concluindo, temos que, em geral, a multiplexagem estatística conduz a atrasos médios por pacote inferiores que na multiplexagem determinística, dado que um circuito dedicado a um fluxo de pacotes e que esteja momentaneamente livre não pode ser usado pelos outros fluxos. No entanto, a multiplexagem determinística conduz a variâncias de atraso menores, uma vez que na estatística, o número médio de clientes em espera é maior.

Disciplinas de escalonamento

Nos procedimentos de multiplexagem estatística, os pacotes de cada fluxo podem ser tratados de forma diferente entre eles. Como vimos em Arquitetura de Redes (a3s2), se atribuirmos diferentes prioridades aos diversos pacotes estamos a conseguir categorizar os fluxos de forma a que aqueles que mais prioridade têm, mais rapidamente circulam na rede. A este esquema em particular damos o nome de **prioritização estrita**.

Consideremos, para o efeito, um sistema de fila de espera M/G/1 com prioridades, em que existem n classes de serviço. Neste contexto, a k -ésima classe é determinada por possuir taxa de chegada λ_k , um primeiro e segundo momentos do tempo de serviço definidos por (4.11) e uma prioridade k (sendo que quanto menor for este valor, maior a prioridade).

prioritização estrita

$$E(S_k) = \frac{1}{\mu_k} \quad E(S_k^2) \quad (4.11)$$

O sistema em si irá transmitir primeiro os pacotes que forem descritores das classes com maior prioridade. Os pacotes de uma mesma classe são transmitidos por ordem de chegada, numa disciplina FIFO.

Porque considerámos 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 com maior prioridade (modo **non-preemptive**).

non-preemptive

Neste caso, o atraso médio de espera correspondente aos pacotes da classe k é dado por (4.12).

$$W_{Q_k} = \frac{\sum_{i=1}^n \lambda_i E(S_i^2)}{2(1 - \rho_1 - \dots - \rho_{k-1})(1 - \rho_1 - \dots - \rho_k)}, \text{ com } \rho_k = \lambda_k / \mu_k \text{ e } \rho_1 + \dots + \rho_n < 1 \quad (4.12)$$

Por fim, também temos a disciplina que traz mais justiça a este escalonamento, denominado de **fair queueing**. Na disciplina FIFO, um fluxo de pacotes que decide momentaneamente aumentar a sua taxa de transmissão consegue capturar uma fração arbitrariamente grande dos recursos do sistema.

Já 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.

Por esta mesma razão criou-se o *fair queueing*, onde 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 enviar pacotes a uma taxa demasiado alta, a sua fila de espera irá encher-se sem que se afete os restantes fluxos.

Na Figura 4.3 podemos ver uma representação deste escalonamento [10].

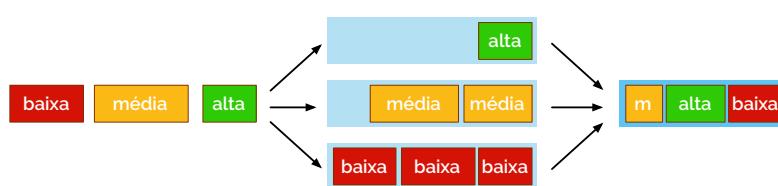


figura 4.3

O algoritmo de *Weighted Fair Queueing* (**WFQ**) é então um algoritmo, tal como abordámos em Arquitetura de Redes (a3s2), que 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.

WFQ

5. Encaminhamento de Pacotes numa Rede de Comutação

Em Arquitetura de Redes Avançadas (a4s1) tivemos oportunidade de abordar, juntamente ao que abordámos inicialmente em Fundamentos de Redes (a3s1), dois tipos de redes de **comutação de pacotes**: redes de datagramas, como é o caso de uma rede IP e redes de circuitos virtuais. Na Figura 5.1 podemos ver um exemplo de uma rede sobre a qual iremos trabalhar para rever os vários conceitos aprendidos.

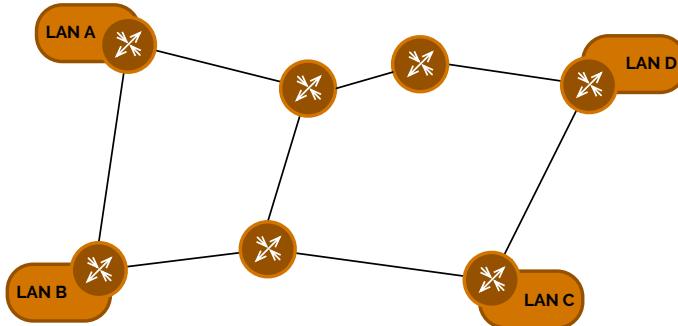
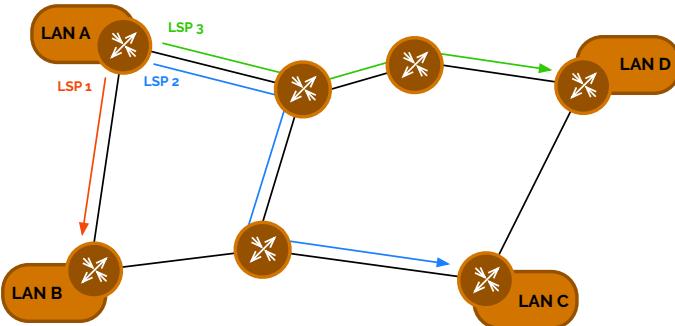
comutação de pacotes

figura 5.1

Conceitos de encaminhamento em redes

Consideremos, portanto, que a rede representada na Figura 5.1 demonstra um caso de aplicação de uma rede gerida por um ISP (*Internet Service Provider*), na qual as várias LANs representadas são várias redes de acesso.

Como vimos em Arquitetura de Redes (a4s1), com a implementação de protocolos como o MPLS, podemos criar o conceito de **círculo virtual**, como caminhos por onde se podem efetuar encaminhamento de pacotes. Estes percursos são definidos na fase de estabelecimento dos circuitos virtuais, depois, sendo os pacotes de cada circuito encaminhados ao longo deste. Nestas redes MPLS os vários caminhos virtuais dão pelo nome de **LSP** (sigla para *Label Switched Path*), sendo exemplos deste, representados na Figura 5.2.



círculo virtual

LSP

figura 5.2

Dentro destes LSPs, alguns deles poderão permitir apenas uma pequena percentagem da largura de banda total para os canais físicos usados. Na Figura 5.3 podemos ver um exemplo em que há dois caminhos que vão para o mesmo destino, sendo que aqui os vários fluxos de pacotes terão de ser repartidos entre ambos os caminhos, de forma a que estes possam prosseguir dentro da redundância estabelecida pelo sistema.

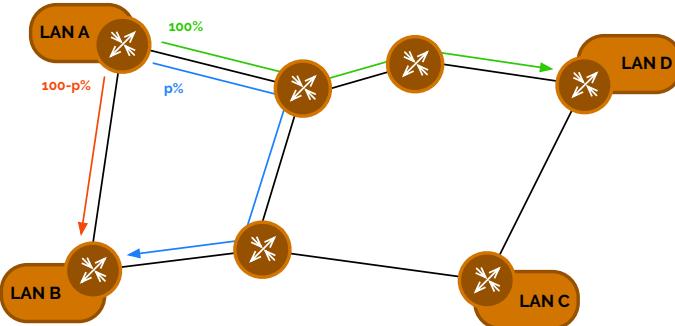


figura 5.3

Por outro lado, em redes de datagramas, como já referimos ser o caso das redes IP, as decisões de encaminhamento (na verdade dir-se-ia roteamento, uma vez que estamos a referir a camada de rede, não de ligação) são efetuadas pacote-a-pacote ou endereço destino a endereço destino. Note-se que, aqui, dois pacotes do mesmo par origem-destino podem seguir percursos distintos na rede.

Para que estes procedimentos funcionem, em redes IP, necessitamos de um conjunto de protocolos denominados de protocolos de roteamento, como o RIP ou o OSPF, para que nos seja possível orientar os vários pacotes pela rede, sempre que um entra num nó de interseção, denominado de *router*.

Neste tipo de rede o roteamento é baseado então em percursos de custo mínimo de cada nó para cada destino. Exemplos desta prática são o RIP, em que o custo é unitário por cada ligação, ou o OSPF, em que o custo é atribuído a cada ligação através de um número positivo. Cada percurso de um *router* para um destino tem um custo igual à soma de todos os custos das ligações que o compõem. Desta forma, cada pacote IP é roteado por um dos percursos de custo mínimo para o destino do pacote.

Entre os vários protocolos de roteamento IP, na busca do custo mínimo para o destino de um pacote, é possível que se cada um possua uma metodologia própria: poderá ser **estática**, se o custo das ligações for sempre fixo, ou **dinâmica**, se o custo das ligações variar ao longo do tempo em função do seu nível de utilização. Em Arquitetura de Redes (a3s2) vimos exemplos de protocolos como os IGRP ou o EIGRP que permite este tipo de ada-

estática, dinâmica

ptabilidade. Basicamente, o funcionamento destes tem por base que o percurso de custo mínimo adapta-se a situações de sobrecarga, obrigando os pacotes a evitarem as ligações mais utilizadas. Contudo, este processo introduz um efeito de realimentação que pode levar a oscilações indesejáveis.

Também vimos em tal disciplina que quando existem múltiplos caminhos de custo mínimo de um nó para um destino, então a técnica de **ECMP** (*Equal Cost Multi-Path*) poderá ser usada, sendo que aqui, em cada nó, o tráfego é bifurcado em igual percentagem por todas as ligações de saída que proporcionam percursos de custo mínimo.

Consideremos, portanto e para o efeito, que aplicamos o protocolo OSPF na rede da Figura 5.3. Na Figura 5.4 podemos ver essa aplicação que resulta, neste caso, da aplicação de um custo unitário OSPF atribuído a cada interface de cada ligação.

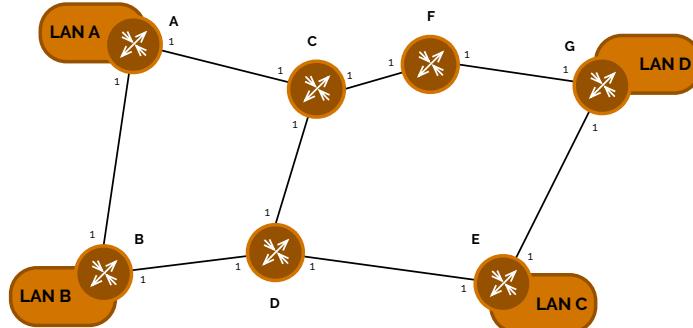


figura 5.4

Pelo uso de ECMP, o *router A* encaminha os pacotes IP com destino para um endereço IP da LAN A em igual percentagem pelos percursos que passam pelo *router B* e pelo *router C*, como podemos ver na Figura 5.5.

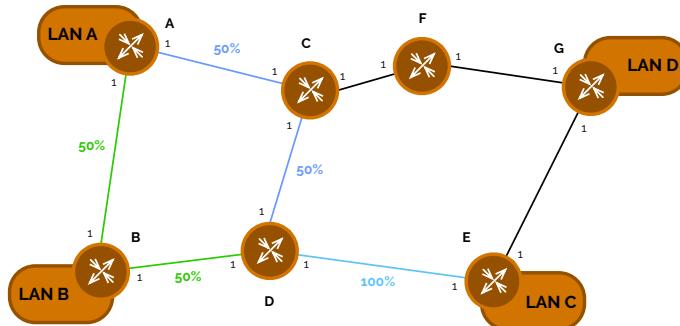


figura 5.5

Mudando o custo da ligação do *router A* para o *router B* de 1 para 3, então agora o *router A* irá encaminhar os pacotes IP com destino para um endereço IP da rede LAN A pelo único caminho de custo mínimo. Na Figura 5.6 podemos ver esta diferença.

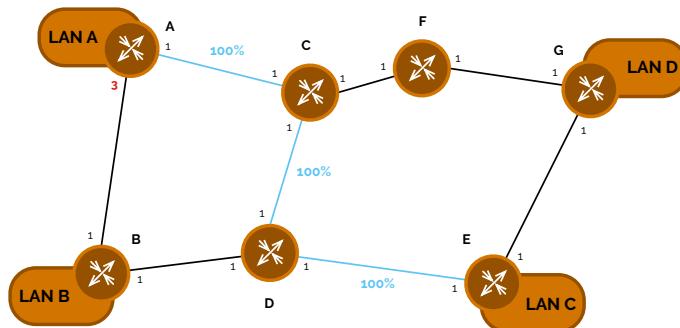


figura 5.6

Em redes de ligações ponto-a-ponto os intervalos entre as várias chegadas dos pacotes tornam-se muito correlacionados com os comprimentos destes, uma vez passados pela

primeira fila de espera do sistema. Este facto tende a dificultar a análise deste problema, tal como podemos verificar na Figura 5.7 onde, considerando duas ligações ponto-a-ponto em cascata, os pacotes chegam à primeira fila de espera a uma taxa de Poisson e o comprimento dos pacotes é exponencialmente distribuído. Aqui, a primeira fila é modelada por um sistema M/M/1, contudo a segunda já não, pelo que o intervalo entre a chegada de dois pacotes consecutivos à segunda fila de espera é sempre superior ou igual ao tempo de transmissão do segundo pacote na primeira fila de espera. Por esta razão, tipicamente, pacotes maiores esperam menos tempo na segunda fila que pacotes mais pequenos.

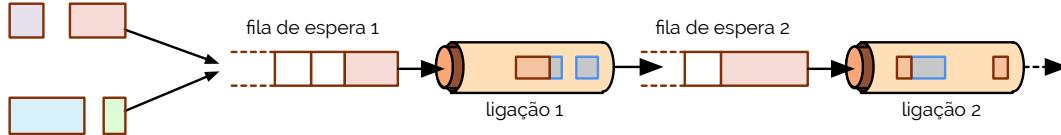


figura 5.7

slow truck effect

Este efeito também é conhecido por **slow truck effect**. Basicamente consideremos que estamos em plena autoestrada e que, ao longo desta, temos de passar por várias estações de portagem. Mesmo que tenhamos um veículo consideravelmente rápido, se não conseguirmos ultrapassar ninguém e tivermos um camião à nossa frente a andar mais lentamente, a nossa velocidade terá de ser regida por este, atrasando a nossa passada na estrada.

Aproximação por independência (aproximação de Kleinrock)

Embora as segundas filas de espera não são modeladas por M/M/1, para que as possamos analisar de forma aproximada, podemos considerá-las como tal. A este procedimento damos o nome de **aproximação de Kleinrock**. Aqui, então, temos de considerar que os fluxos de pacotes são unidireccionais e que as ligações das redes de comutação de pacotes são bidirecionais. Assim, uma ligação de rede entre dois nós de comutação i e j é representada pelos pares orientados (i, j) e (j, i) — estes representam cada um dos seus sentidos.

Se assumirmos que temos um conjunto de fluxos de pacotes $s = 1, \dots, S$ estando, a cada um destes, 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 , então a taxa total de chegada à ligação (i, j) é a apresentada por (5.1), onde λ_s está em pacotes por segundo e corresponde à taxa de chegada do fluxo s .

$$\lambda_{ij} = \sum_{S: (i, j) \in R_s} \lambda_s \quad (5.1)$$

Contudo, como vimos anteriormente, um determinado pacote poderá ter como destino uma entidade que possui um ou mais caminhos alternativos. Neste caso, em termos gerais, temos de contar com a possibilidade de poder haver múltiplos percursos associados a cada fluxo de pacotes.

Por conseguinte, 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 (não há ciclos no roteamento). Neste caso, o conjunto R_s incluirá todas as ligações (i, j) tais que $f_{ij}(s) > 0$. Portanto, a taxa total de chegada à ligação (i, j) é dado por (5.2).

$$\lambda_{ij} = \sum_{S: (i, j) \in R_s} f_{ij}(s) \lambda_s \quad (5.2)$$

Com estas simplificações é natural que a aplicação do teorema de Little também seja clarividente. Basicamente, para uma ligação (i, j) , e considerando que temos um μ_{ij} sendo a capacidade da ligação em número médio de pacotes por segundo, então o número médio de pacotes (o valor de L) em todas as ligações é dado por (5.3).

aproximação de Kleinrock

◎ Leonard Kleinrock

$$L = \sum_{(i,j)} \frac{\lambda_{ij}}{\mu_{ij} - \lambda_{ij}} \quad (5.3)$$

Igualmente simples de perceber, o atraso médio por pacote é dado por (5.4).

$$W = \frac{1}{\gamma} \sum_{(i,j)} \frac{\lambda_{ij}}{\mu_{ij} - \lambda_{ij}}, \text{ onde } \gamma = \sum_s \lambda_s \quad (5.4)$$

Em 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 dado por (5.5), em que d_{ij} é o atraso médio de processamento e propagação associado à ligação (i, j) .

$$W = \frac{1}{\gamma} \sum_{(i,j)} \left(\frac{\lambda_{ij}}{\mu_{ij} - \lambda_{ij}} + \lambda_{ij} d_{ij} \right), \text{ onde } \gamma = \sum_s \lambda_s \quad (5.5)$$

Se só houver um percurso para cada fluxo de pacotes, então o atraso médio por pacote do fluxo s é dado por (5.6). Note-se que aqui não fazemos uma ponderação sobre as várias taxas de chegada, pelo que só quando 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 por (5.5)) em que os pesos são as percentagens do tráfego total que são encaminhados por cada percurso.

$$W_s = \sum_{(i,j)} \left(\frac{1}{\mu_{ij} - \lambda_{ij}} + d_{ij} \right) \quad (5.6)$$

Não nos esqueçamos que estamos a fazer uma aproximação. Neste contexto há que conseguir identificar quais são os nossos erros nestas nossas medidas. No caso de rede 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.

Já no caso de redes com múltiplos percursos por fluxo de pacotes, poderá existir um fator adicional de erro, sendo este consequente da forma como os fluxos são bifurcados nos nós. Por este mesmo motivo consideremos o fluxo de pacotes λ representado na Figura 5.8.

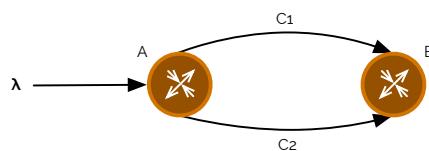


figura 5.8

Na Figura 5.8, portanto, temos uma representação de um fluxo de pacotes λ que é **bifurcado** por duas ligações com capacidades C_1 e C_2 pacotes por segundo. Se considerarmos ambos os fluxos bifurcados por x_1 e x_2 , temos então que o nosso $\lambda = x_1 + x_2$. O número médio de pacotes nesta rede, pela aproximação de Kleinrock é a seguinte representada em (5.7).

bifurcado

$$L = \frac{x_1}{C_1 - x_1} + \frac{x_2}{C_2 - x_2} \quad (5.7)$$

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. Por conseguinte, atendendo à restrição $\lambda = x_1 + x_2$ temos (5.8).

$$L = \frac{x_1}{C_1 - x_1} + \frac{\lambda - x_1}{C_2 - (\lambda - x_1)} \quad (5.8)$$

Agora, a partir de (5.8) podemos derivar e calcular os zeros, de forma a conseguirmos obter o encaminhamento do fluxo de pacotes pelas ligações x_1 e x_2 . [11]

6. Otimização com Base em Programação Inteira Linear

O desenvolvimento da **programação linear** foi avaliado como um dos maiores avanços científicos de meados do século XX. O impacto que esta área teve a partir dos anos '50 tem sido extraordinário, dado que até à data a avaliação de múltiplas variáveis num sistema fechado com interações entre elas era uma tarefa de elevada complexidade, sendo efetuada por humanos.

programação linear

Então mas qual é a natureza desta ferramenta? Basicamente, o tipo de aplicações mais comum desta ferramenta são locais onde se aplica o problema genérico da alocação de recursos limitados entre atividades que competem entre si por estes, resolvendo-o da melhor forma possível, ou seja, otimizando o sistema.

Tendo sido precisamente no tópico de otimização, que ficámos na secção anterior, agora vejamos como é que podemos aplicar programação linear para resolver parte dos nossos problemas, especialmente aqueles que tocam em otimização.

Modelos de programação linear

Num problema de otimização o objetivo é então maximizar (ou minimizar) uma determinada quantidade designada pelo **objetivo** que é dependente de um número finito de variáveis. Estas variáveis poderão estar interligadas entre si, isto é, formando algumas dependências, avaliadas como restrições.

objetivo

Um problema de programação matemática é um problema de otimização tal que o objetivo e as suas restrições estão definidas por funções algébricas e relações funcionais. Neste contexto, um **modelo** é um problema matemático programável.

modelo

Em termos concetuais, temos que para um dado conjunto de n variáveis $X = \{x_1, x_2, \dots, x_n\}$ a norma para definição de um modelo matemático programável está representado no Código 6.1.

Minimizar (ou Maximizar)
 $f(X)$

código 6.1

Sujeito de:
 $g_i(X) \leq k_i$, com $i = 1, 2, \dots, m$
 $(=)$
 (\geq)

Onde:
- m é o número de constantes
- $f(X)$ e todos $g_i(X)$ são funções das variáveis
- k_i são constantes reais

Existem vários modelos de programação linear, sendo os mais usados os seguintes: *linear programming (LP)*, sendo um modelo matemático onde todas as variáveis X são reais não-negativos e $f(X)$ e $g_i(X)$ são funções lineares; *integer linear programming (ILP)*, sendo um modelo com base num LP onde todas as variáveis X são inteiros não-negativos; e *mixed integer linear programming (MILP)*, sendo um modelo com base num LP onde algumas variáveis X são reais não-negativos e outros são inteiros não-negativos.

LP

ILP

MILP

Exemplo de aplicação de programação linear e sua resolução

Consideremos, para melhor percebermos como é que podemos usar programação linear para resolver problemas multi-variável, um exemplo de aplicação. Assim sendo, considere-

mos uma empresa de transportes que foi responsabilizada de efetuar entrega de um conjunto de itens I para um destino em particular. Na Figura 6.1 podemos ver o conjunto de itens, o seu tamanho e o lucro deles obtido.

| item i | 1 | 2 | 3 | 4 | 5 | 6 |
|---------------|-----|-----|-----|-----|-----|-----|
| lucro r_i | 2.3 | 4.5 | 1.5 | 5.4 | 2.9 | 3.2 |
| tamanho s_i | 30 | 70 | 20 | 80 | 35 | 40 |

figura 6.1

Mais dados para o nosso problema, a empresa possui duas carrinhas para fazer as entregas dos itens: uma primeira possui uma capacidade de 100 e uma segunda que possui uma capacidade de 60.

Uma vez que não é possível entregar todos os itens com as duas carrinhas, o objetivo é tentar escolher quais os itens a ser levados em cada carrinha, de forma a que se possa maximizar o lucro. Para resolver esta questão vamos primeiro desenhar a nossa solução através de um modelo ILP e resolvê-lo de seguida.

Então que variáveis é que definem o nosso problema? Ora, para começar, dado que temos 6 itens para entregar, então devemos ter 6 variáveis booleanas que nos possam descrever, em qualquer situação, se o item i será entregue ou não. Para tal, consideremos que possuímos um conjunto de variáveis booleanas x_i (com i sendo qualquer um dos índices dos itens) que alojam o valor ‘1’ caso x_i seja solução deste problema. Por fim, precisamos de variáveis booleanas que nos possam fazer corresponder que item é levado por que carrinha. Para tal criamos variáveis do tipo $y_{i,v}$ (em que i é o item a ser transportado e v a carrinha que o transporta).

Tendo as nossas variáveis bem definidas, então o próximo passo é definir um ficheiro que transporte as definições da nossa solução para que possamos entregar a uma calculadora própria. Para isso vamos usar um formato de ficheiro denominado de **formato LP**. Neste formato primeiramente definimos qual a nossa função objetivo (sendo no nosso caso o lucro total maximizado dos itens entregues), depois definimos que o tamanho total dos itens transportados não poderá exceder o tamanho de cada carrinha, sendo que se um item for levado numa carrinha, então considera-se como entregue e uma lista das variáveis binárias. No Código 6.2 podemos ver um exemplo de aplicação.

formato LP

```

Maximize
+ 2.3 x1 + 4.5 x2 + 1.5 x3 + 5.4 x4 + 2.9 x5 + 3.2 x6
Subject to
+ 30 y1,1 + 70 y2,1 + 20 y3,1 + 80 y4,1 + 35 y5,1 + 40 y6,1 <= 100
+ 30 y1,2 + 70 y2,2 + 20 y3,2 + 80 y4,2 + 35 y5,2 + 40 y6,2 <= 60
+ y1,1 + y1,2 - x1 = 0
+ y2,1 + y2,2 - x2 = 0
+ y3,1 + y3,2 - x3 = 0
+ y4,1 + y4,2 - x4 = 0
+ y5,1 + y5,2 - x5 = 0
+ y6,1 + y6,2 - x6 = 0
Binary
x1 x2 x3 x4 x5 x6
y1,1 y1,2 y2,1 y2,2 y3,1 y3,2 y4,1 y4,2 y5,1 y5,2 y6,1 y6,2
End

```

código 6.2

Para executarmos este ficheiro e obtermos a nossa solução basta enviar o ficheiro para execução numa máquina que o permita. Através de software próprio é possível fornecer o ficheiro LP como *input* para execução. Um exemplo é o CPLEX da IBM, que após execução fornece um resultado como o visível no Código 6.3.

```

CPLEX> read problem.lp
Problem 'problem.lp' read.
Read time = 0.01 sec. (0.00 ticks)
CPLEX> optimize
# ...
MIP - Integer optimal solution: Objective = 1.1500000000e+001
# ...
CPLEX> display solution variables -
Incumbent solution

```

código 6.3

```

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>

```

Pelo Código 6.3 podemos verificar que somente os itens 1, 2, 3 e 6 é que são entregues, sendo os itens 1 e 2 entregues pela primeira carrinha, e os 3 e 6 pela segunda carrinha.

Note-se, não obstante, que os resultados mostrados são uma possível solução, que a máquina considerou como ótimos. Nem todas as máquinas mostram todas as soluções possíveis, sendo que pode dar-se o caso desta solução ter sido a primeira considerada como ótima.

7. Desempenho de Redes com Comutação de Circuitos

Até agora vimos como processar a comutação de pacotes em redes desenhadas para esse efeito. Contudo, ainda antes das redes de comutação de pacotes existirem, outras ainda mais baixo-nível eram usadas. Tais redes, ditas possuírem a **comutação de circuitos**, eram tais que permitiam a introdução de um sistema telefônico com instalação de comutadores (estes que eram inicialmente operados por humanos, mas mudados mais tarde para mecanismos automáticos).

comutação de circuitos

Neste tipo de redes o encaminhamento das chamadas é feito de três formas possíveis: de uma forma fixa — considerando um único percurso para cada fluxo de chamadas suportado pela rede —, de uma forma alternativa — considerando uma sequência ordenada de percursos alternativos para cada fluxo (estabelecendo no n -ésimo percurso se nenhum dos percursos até ao $(n-1)$ -ésimo tiver recurso) — ou de uma forma dinâmica — considerando uma sequência ordenada de percursos alternativos para cada fluxo e essa sequência varia ao longo do tempo.

Uma vez que não estamos a referir um conjunto de nós interligados por onde cada nó terá de decidir a forma como um determinado pacote seguirá até ao seu destino, com chamadas numa rede em malha completa, o percurso considerado como direto é aquele com uma ligação, que interliga o nó origem e o nó destino. É preferível encaminhar uma chamada pelo percurso direto, isto porque os percursos alternativos consomem mais recursos. Contudo, para que isto possa acontecer, e tal como vimos em Arquiteturas de Redes Avançadas (a4s1), é necessário que haja uma reserva de um conjunto de circuitos em cada ligação para chamadas no percurso direto (limita o excesso de encaminhamento alternativo). A este procedimento damos o nome de **trunk reservation**.

trunk reservation

Embora tenhamos as várias formas de encaminhamento possível numa rede de comutação de pacotes, aqui neste documento iremos abordar, essencialmente como é que o encaminhamento fixo poderá ser feito, uma vez que este é base para todas as suas alternativas. Para cumprir assim o nosso objetivo de descrever como fazer o encaminhamento fixo, então abordaremos três métodos para o cálculo das probabilidades de bloqueio de cada fluxo de chamadas.

Cálculo da probabilidade de bloqueio por método exato

Para o efeito, consideremos uma rede com J ligações que serve K fluxos de chamadas. A ligação $j = 1, \dots, J$ tem capacidade C_j número de circuitos. Já o fluxo $k = 1, \dots, K$ tem a si associada uma taxa de chegada λ_k , um tempo médio de serviço $1/\mu_k$ (com uma intensidade de tráfego $Q_k = \lambda_k/\mu_k$), uma largura de banda b_k (em número de circuitos) e um percurso de encaminhamento fixo $R_k \subseteq \{1,2,\dots,J\}$. Neste contexto, as chamadas do fluxo k chegam de acordo com um processo de Poisson de taxa λ_k , sendo que as admitidas pelo sistema ocupam b_k circuitos e têm uma duração exponencialmente distribuída com média

$1/\mu_k$. Mais, a duração das chamadas é independente entre chamadas e independente dos instantes de chegada para todos os fluxos.

Considerando K_j como o conjunto dos fluxos que atravessam a ligação j e sendo n_k o número de chamadas do fluxo k no sistema $n = (n_1, \dots, n_k)$, então uma chamada do fluxo k não é aceite pela rede se, em pelo menos uma das ligações pertencentes ao percurso de k , (7.1) se comprovar, isto é, se a largura de banda para o fluxo k com a soma das larguras respetivas de cada chamada n_k for maior que a capacidade de j em número de circuitos.

$$b_k + \sum_{I \in K_j} b_I n_I > C_j \quad (7.1)$$

O espaço de estados, S , do processo de nascimento e morte multidimensional fica composto por (7.2), onde I é o conjunto dos inteiros não-negativos. Se S_k for considerado como o subconjunto dos estados nos quais uma chamada do fluxo k é admitida quando chega à rede, ficamos com (7.3).

$$S = \left\{ n \in I^K : \sum_{k \in K_j} b_k n_k \leq C_j, \quad j = 1, \dots, J \right\} \quad (7.2)$$

$$S_k = \left\{ n \in S : \sum_{I \in K_j} b_I n_I \leq C_j - b_k, \quad j \in R_k \right\} \quad (7.3)$$

Consideremos, para que possamos ver a aplicação destes conceitos num termo um pouco mais real, um exemplo em que temos uma rede (representada na Figura 7.1) que suporta dois fluxos de chamadas. Um dos fluxos — o fluxo 1 — possui uma taxa de chegada $\lambda_1 = 3$ chamadas por hora, com uma duração média $1/\mu_1 = 2$ minutos ocupando, cada chamada, $b_1 = 64$ Kbps. Já o segundo fluxo — o fluxo 2 — possui uma taxa de chegada $\lambda_2 = 4$ chamadas por hora, com uma duração média $1/\mu_2 = 3$ minutos ocupando, cada chamada, $b_2 = 128$ Kbps.

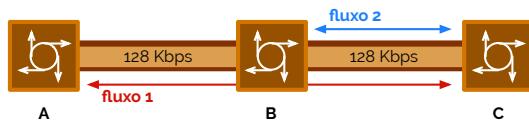


figura 7.1

Ora, tendo como base a rede da Figura 7.1 podemos verificar que para um K de 2 fluxos o primeiro terá uma intensidade de tráfego de $(3/60) \times 2 = 0.1$ Erlang, uma largura de banda b_1 de 1 circuito e um percurso $R_1 = \{AB, BC\}$. Já o segundo terá uma intensidade de tráfego de $(4/60) \times 3 = 0.2$ Erlang, uma largura de banda b_2 de 2 circuitos e um percurso $R_2 = \{BC\}$. Tendo então um J de 2 ligações, podemos então identificar a capacidade de AB, em C_{AB} , com 2 circuitos, da mesma forma que C_{BC} . Para estes, o K_{AB} é composto por {1} e o K_{BC} por {1,2}.

Uma das tarefas mais importantes de serem feitas é gerar um diagrama de estados da cadeia de Markov correspondente ao sistema. Deste modo, para o nosso caso, consideremos que os nossos estados possuem, por legenda e codificação, o estado do número de chamadas ativas no fluxo 1, seguido das do fluxo 2. Na Figura 7.2 podemos então ver a cadeia de Markov resultante para um espaço de resultados $S = \{[0,0], [0,1], [0,2], [1,0]\}$, sendo a chamada do fluxo 1 admitida em $S_1 = \{[0,0], [1,0]\}$ e a chamada do fluxo 2 admitida em $S_2 = \{[0,0]\}$.

Neste método exato de determinar a probabilidade de bloqueio podemos aplicar, novamente os resultados de (4.5) e (4.6) para indicar a probabilidade de bloqueio da classe k .

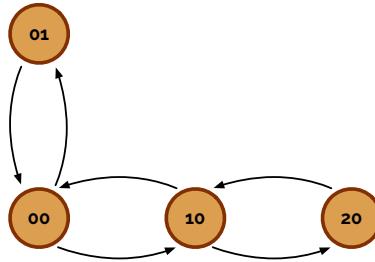


figura 7.2

Para calcular a probabilidade de bloqueio, portanto, podemos aplicar as expressões de (4.6), o que resulta em (7.4).

$$B_k = 1 - \frac{\sum_{n \in S_k} \prod_{l=1}^K \frac{\rho_l^{n_l}}{n_l!}}{\sum_{n \in S} \prod_{l=1}^K \frac{\rho_l^{n_l}}{n_l!}} \quad (7.4)$$

$$B_1 = 1 - \frac{\frac{0.1^0 0.2^0}{0!0!} + \frac{0.1^1 0.2^0}{1!0!}}{\frac{0.1^0 0.2^0}{0!0!} + \frac{0.1^1 0.2^0}{1!0!} + \frac{0.1^2 0.2^0}{2!0!} + \frac{0.1^0 0.2^1}{0!1!}} = 0.157 = 15.7\%$$

$$B_2 = 1 - \frac{\frac{0.1^0 0.2^0}{0!0!}}{\frac{0.1^0 0.2^0}{0!0!} + \frac{0.1^1 0.2^0}{1!0!} + \frac{0.1^2 0.2^0}{2!0!} + \frac{0.1^0 0.2^1}{0!1!}} = 0.234 = 23.4\%$$

Através do método exato para a determinação da probabilidade de bloqueio conseguimos atingir o resultado final mas com algum custo computacional, uma vez que todo o seu procedimento poderá ser considerado como computacionalmente intenso e porque precisa da identificação de todos os estados possíveis da rede.

Cálculo da probabilidade de bloqueio com teorema do limite do produto

Uma alternativa à prática do método exato para a estimativa da probabilidade de bloqueio é o uso do **teorema do limite do produto**.

Neste caso, somente aplicável quando as chamadas de todos os fluxos k requerem a mesma largura de banda b_k (em número de circuitos), com a intensidade de tráfego suportada pela ligação j dada por (7.5), o teorema do limite do produto exprime-se em (7.6).

**teorema do limite do
produto**

$$\bar{\rho}_j = \sum_{k \in K_j} \rho_k \quad (7.5)$$

$$B_k \leq 1 - \prod_{j \in R_k} \left(1 - \text{ErlangB} \left(\bar{\rho}_j, C_j \right) \right) \quad (7.6)$$

Apesar de ser uma aproximação, algebricamente a expressão de (7.6) comprova-se ser um majorante das probabilidades de bloqueio obtidas por um método exato. Contudo, é aplicável quando os fluxos atravessam poucas ligações e quando as probabilidades de bloqueio são pequenas (tipicamente até 1%).

Cálculo da probabilidade de bloqueio na aproximação de carga reduzida

Com a aplicação do teorema do limite do produto pudemos verificar que se obtêm resultados mais rapidamente, uma vez que se trata de um método computacionalmente pouco intensivo, mas que são aproximações. Não obstante, este método só poderá ser apli-

cado quando não há problemas em afirmar que se obterá uma probabilidade de uma ligação j estar totalmente majorada pela expressão ErlangB($\bar{\rho}_j, C_j$), de (7.6).

Uma possibilidade para melhorar a aproximação associada ao teorema do limite do produto está na redução do tráfego oferecido à ligação j , tomando em linha de conta o bloqueio nas outras ligações do percurso de cada fluxo. Através de uma substituição de ρ_k por $\rho_k t_k(j)$, onde $t_k(j)$ 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 (7.7).

$$L_j = \text{ErlangB} \left[\sum_{k \in K_j} \rho_k t_k(j), C_j \right] \quad (7.7)$$

Ora, considerando que o bloqueio é independente de ligação para ligação, então podemos substituir $t_k(j)$ pelo complemento do produto de todas as probabilidades de bloqueio das ligações que não j , como podemos ver em (7.8) — **equações de ponto fixo**.

$$L_j = \text{ErlangB} \left[\sum_{k \in K_j} \rho_k \prod_{i \in R_k - \{j\}} (1 - L_i), C_j \right] \quad (7.8)$$

Por fim, podemos então concluir que a probabilidade de bloqueio através deste método é obtido por (7.9), para as chamadas do fluxo k .

$$B_k \approx 1 - \prod_{j \in R_k} (1 - L_j), \quad \text{com } k = 1, 2, \dots, K \quad (7.9)$$

Para aplicarmos este mecanismo na prática, precisamos primeiro de perceber uma das suas possibilidades de implementação — o algoritmo iterativo. Para tal, seja $L = (L_1, L_2, \dots, L_J)$ um vetor e um operador $T(L) = (T_1(L), T_2(L), \dots, T_J(L))$, onde $T_j(L)$ seria equivalente a (7.8). Assim sendo, as equações de ponto fixo podem ser expressas na forma $L = T(L)$.

Partindo de um vetor inicial $L \in [0,1]^J$ aplica-se sucessivamente o operador T , pelo que ocorre (7.10).

$$L^0 = L \quad L^m = T(L^{m-1}), \quad m = 1, \dots, n \quad (7.10)$$

Olhando para (7.10) temos que, partindo de $L^0 = (1, 1, \dots, 1)$ o método dá origem a $L^1 = (0, 0, \dots, 0)$, L^{2n} converge para um L^+ e L^{2n+1} para um L^- , tal que $L^- \leq L^* < L^+$. As sucessivas m iterações determinam majorantes da solução L^* quando m é ímpar. Termina-se o algoritmo quando os dois limites estão suficientemente próximos.

Encaminhamento dinâmico da rede telefónica

Com o passar dos tempos, em redes de comutação de circuitos, como é o caso da rede telefónica mais antiga — a PSTN, como vimos em Arquiteturas de Redes Avançadas (a4s1) —, tiveram a necessidade de implementar mecanismos de encaminhamento dinâmico, isto é, através de uma conectividade total (há uma ligação direta entre todos os nós de acesso — central que liga uma rede de acesso a uma rede de transporte). [14]

Desta feita, numa rede com N nós temos que existem $N(N - 1)/2$ ligações e o número de percursos com apenas duas ligações entre quaisquer duas centrais é $N - 2$.

Existem vários métodos e consequentes algoritmos para o encaminhamento dinâmico PSTN, contudo, todos eles possuem características comuns. Em todos eles, 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 (isto porque as chamadas por encaminhamento alternativo possuem maior custo para a rede). [14] Uma vez que o percurso direto se encontra indisponível, então a chamada poderá ser encaminhada para um dos percursos alternativos permitidos, sendo que estes possuem sempre duas ligações, ou seja, nunca são estabelecidos em percursos com três ou mais ligações. Basicamente, ao longo de um conjunto de vários algoritmos de implementação, o que mudará será a forma como são definidos, a cada instante de tempo, os conjuntos de percursos alternativos.

O primeiro algoritmo a ser estudado neste documento é o **encaminhamento sequencial**, introduzido pela AT&T nos anos '80 (tecnicamente designado por *Dynamic Non-Hierarchical Routing* ou simplesmente **DNHR**). Esta implementação resulta num roteamento dependente do tempo, ou seja, o seu conjunto de percursos alternativos será diferente em diferentes ocasiões de um dia. Na verdade, no caso deste algoritmo, o período de 24 horas por 7 dias de uma semana foi dividido em 15 partes: 10 para os dias úteis e 5 para fins-de-semana. O número diferente de períodos de carregamento foi determinado com base na compreensão de padrões de tráfego. Por exemplo, um mesmo padrão de tráfego poderá ser usado da meia-noite até às oito da manhã, dado o baixo tráfego. [15]

Para cada período de carregamento, com base na projeção de tráfego, um conjunto de percursos alternativos são calculados previamente. Tipicamente, as projeções de tráfego e os respetivos cálculos são efetuados *offline* com uma semana de avanço e guardadas nos vários comutadores. Quando uma chamada chega a um comutador, então este determina qual a tabela correta para verificar qual o percurso a tomar, com base na janela temporal em atual, tentando as várias rotas, de forma ordenada, conforme apresentadas na tabela. Certamente, neste processo, o tráfego real será diferente que o projetado anteriormente. Assim, as rotas que foram calculadas e a ordem com que foram indicadas poderá não ser uma forma ótima de resolução do problema. Uma forma de corrigir esta solução é permitir que se ative uma opção algorítmica de **crankback** (mecanismo que tenta encontrar novo caminho para o destino, quando um pedido de conexão de serviço está bloqueado porque um nó no caminho não o consegue aceitar). Por exemplo, se já tiverem sido criados três percursos entre uma origem e um destino, uma chamada poderá tentar o primeiro caminho designado (frequentemente sendo o percurso direto). Se nenhuma largura de banda lhe for permitida no primeiro caminho, então a chamada deverá tentar o segundo, e por aí em diante. [14]

Enquanto que os caminhos que são calculados (caminhos pré-criados) poderão fornecer uma classe de serviço aceitável sob condições normais de operação, estes poderão não ser os melhores quando se aplicam em condições de alto-tráfego. Isto acontece essencialmente porque os percursos pré-criados são calculados com base em projeções. Para circundar esta situação o DNHR permite a criação de caminhos adicionais em tempo-real que podem ser adicionados diretamente à lista de percursos pré-criados.

Se um bloqueio entre dois comutadores ocorre fora de um *threshold* aceitável, então uma nova estimativa deverá ser realizada, do tráfego, sempre que uma nova janela temporal de 5 minutos é invocada — aqui são usadas as adições de caminhos calculados em tempo-real. Com base nestes *snapshots* de cinco minutos de tráfego, um cálculo centralizado num centro de computação do gestor de uma rede deverá pesquisar grupos que possuem capacidade para estimar caminhos em tempo-real. O uso destes caminhos pode ser visto numa representação na Figura 7.3, onde pretendemos mostrar um caminho de Nova Iorque para Los Angeles. Note-se que nesta representação os caminhos pré-criados e tempo-real estão representados e particionados com uma barra vertical, sendo que os de tempo-real estão a itálico. [15]

Agora só precisamos de perceber quantos caminhos alternativos conseguimos guardar (em cache) entre os caminhos pré-criados e os de tempo-real. Note-se que para uma rede com N nós que está totalmente ligada, um par de comutadores tem $(N - 1)$ caminhos possíveis de serem realizados com um máximo de duas ligações. Por exemplo, para uma rede

encaminhamento sequencial

DNHR

crankback

de 100 nós, existem 99 possibilidades. Agora, se pretendermos considerar que uma lista de rotas ordenadas para cada um dos 15 períodos é necessária e que tal comutador precisa de ter tais rotas para cada comutador-destino, então a lista de rotas a carregar poderá tomar proporções muito grandes. Este, de facto, é um dos grandes problemas desta abordagem, especialmente no tempo em que este mecanismo fora desenvolvido, quando ainda não havia capacidade de processamento como hoje em dia temos ou memória barata como nos dias que correm. A solução implementada foi permitir, a cada comutador, possuir até 15 rotas na sua lista ordenada, por cada par de comutadores — destes, um máximo de 10 seriam pré-criados e 5 para caminhos de tempo-real.

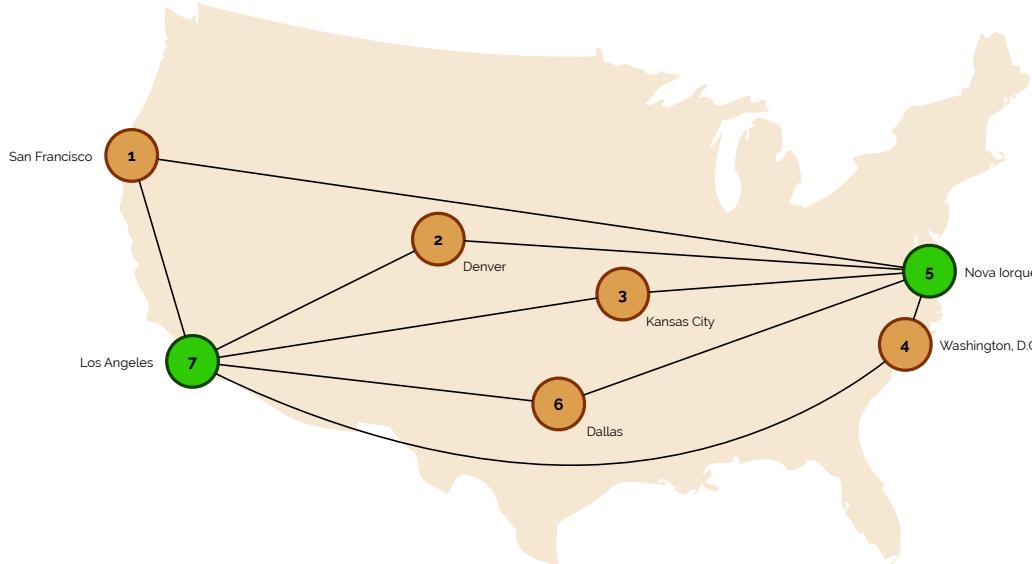


figura 7.3

| período de tempo | sequência de roteamento | | | | | | |
|------------------|-------------------------|---|---|---|---|---|---|
| | 7 | 3 | 6 | 4 | | 2 | 1 |
| tarde (dia útil) | 7 | 6 | | 3 | 4 | 2 | 1 |
| noite (dia útil) | 7 | 4 | 6 | | 3 | 2 | 1 |
| fim-de-semana | 6 | 4 | | 7 | 3 | 2 | 1 |

Nos anos '90 a British Telecom respondeu às necessidades sofridas com o aumento do número de clientes e consequente tráfego com um novo protocolo. Apresentado de uma forma puramente distribuído e com um esquema adaptativo de roteamento, o **encaminhamento aleatório retardado** (tecnicamente designado de *Dynamic Alternative Routing*, ou simplesmente **DAR**), continua limitado a um roteamento de apenas duas ligações, com *trunk reservation* e sem mecanismo de *crankback*. [15]

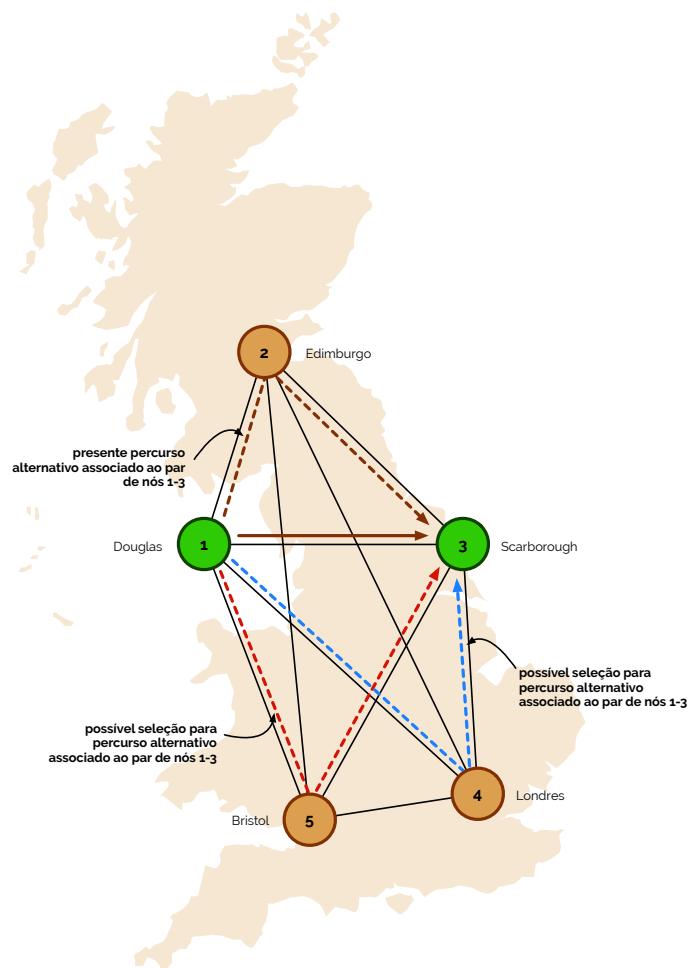
Aqui, para cada comutador-destino j , o comutador-origem i mantém um caminho alternativo k na sua cache. Uma chamada nova que chegue tenta, primeiro, a ligação $i-j$. Se tal ligação não atribuir capacidade à chamada, então tentará a rota presente na cache. Se for atribuída a capacidade do caminho alternativo com sucesso, este permanece em cache. Contudo, se não se conseguir a capacidade no caminho alternativo, então a chamada perde-se, o que significa que o utilizador terá de tentar novamente. Mais, o nó-origem i escolherá um nó à sorte preservando-o como a sua próxima cache para a próxima chamada. Isto é razão para que um caminho alternativo só permaneça na cache enquanto houver pelo menos uma chamada em ação; um novo caminho alternativo é escolhido aleatoriamente assim que uma rota alternativa atual não consiga efetuar uma chamada usando este caminho.

Na verdade, este algoritmo até é bastante elegante, pois vejamos: pensemos num conjunto de chamadas que chegam ao serviço numa janela temporal. Um caminho alternativo que é bom permanece na cache, mas caso contrário, então é porque, provavelmente, as

encaminhamento aleatório retardado, DAR

ligações deste mesmo caminho já se encontram congestionadas. Assim sendo, ao longo do tempo, uma rota menos carregada e alternativa é mais provável de ficar em cache durante uma percentagem maior de tempo. Uma vez que o tráfego de uma rede muda ao longo do tempo, então o roteamento automaticamente se alterará para outro caminho alternativo menos carregado. Na Figura 7.4 podemos ver uma representação deste algoritmo.

figura 7.4



Como resposta ao DAR e sucessor do DNHR, a AT&T criou o protocolo **RTNR** (de *Real-Time Network Routing*) que foi lançado como produto final em 1991 como uma rede de longas distâncias — e que ainda está em uso hoje. Contrariamente ao DNHR, o RTNR é um esquema adaptativo, tal como o DAR, sendo que a tabela de roteamento poderá ser atualizada quase em cada chamada realizada. [17]

Consideremos assim uma rede RTNR com N nós. Quando uma chamada chega ao nó i e é destinada a j no tempo t , o nó i questiona o nó j à procura de informação sobre o estado das ligações para este, especialmente alternativas para k , ao invés de somente por i diretamente. Note-se que o nó i sabe do estado de todas as suas ligações de saída $i-k$. Sendo que um caminho está limitado ao máximo de duas ligações e estas são bidirecionais, então o nó i poderá determinar o estado dos dois caminhos $i-k-j$ para o destino j , através de uma combinação da informação para a ligação $i-k$ e para a ligação $k-j$. Agora, sabendo esta informação, i pode decidir sobre qual rota alternativa realizar.

Agora, que tipo de informação é que surge do nó j ? No caso da aplicação de RTNR, o nó i (na sua forma mais simplificada) pede o estado de disponibilidade de todas as suas ligações de saída como estados binários: 1 se a ligação estiver disponível ou 0 caso contrário. Na sua essência, RTNR não se preocupa muito acerca do quão disponíveis estão as ligações, desde que haja alguma capacidade disponível. Para o conjunto de caminhos para os quais o resultado é 1, o caminho está disponível, pelo que será um destes selecionado, de forma aleatória, pelo nó i para ser um caminho alternativo. Na verdade, este algoritmo

RTNR

ainda faz uma operação AND adicional para verificar preferência de comutadores perante alguma ligação, designada por disponibilidade dos nós, dado que um processador poderá estar congestionado (fazendo com que haja congestão no comutador) ou em manutenção, antes de identificar caminhos úteis. [15] Por estas mesmas razões este algoritmo é mais eficiente que o DAR, embora exija maior complexidade de sinalização.

8. Controlo de Fluxos em Redes de Comutação de Pacotes

Enquanto que o modelo de serviço original do IP é tal que dá o melhor esforço por comportar o melhor possível de qualidade no fornecimento de uma ligação (modelo denominado por **best-effort**), ao longo das décadas a aplicação de critérios por diferenciação de serviços tem sido cada vez mais popular.

A dificuldade que existe com o modelo de *best-effort* prende-se com o facto de que este fornece uma pequena garantia para qualquer tipo de fluxo, a nível de qualidade de serviço (QoS). Já o modelo de diferenciação de serviços permite que haja uma classificação variada do tráfego, aplicável aos vários pacotes que chegam a um determinado nó de uma rede. Através deste processo, um tipo de qualidade de serviço mais leve poderá ser atribuída às diferentes classes de serviço. Por exemplo, tal classificação poderia dar mais ou menos prioridade a clientes cujas comunicações se baseiem em *Service Level Agreements* (SLA). Por outras palavras, e como vimos em Arquitetura de Redes (a3s2), para que se cumpram requisitos de serviço para cada tráfego de cliente, é necessário que exista um mecanismo de qualidade de serviço. Assim sendo, é aqui que entra o modelo *diff-serv*, tal como a necessidade de **controlo de fluxo**.

best-effort

controlo de fluxo

Conceitos introdutórios de controlo de fluxo de pacotes

Para que possamos perceber os vários assuntos que iremos abordar nesta unidade, primeiro, necessitamos de dominar alguns conceitos básicos usados no âmbito de controlo de fluxo de pacotes.

Para começar o que é que significa controlo de fluxo? Ora, o controlo de fluxo é um 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. Por **tráfego efetivo** pretende-se dizer a quantidade de serviço suportada por uma rede com comutação de pacotes e por **atraso médio** a qualidade de serviço proporcionada por esta. Note-se que quando o tráfego oferecido é reduzido é aceite na sua totalidade pelo algoritmo de controlo de fluxo, ficando assim que o tráfego efetivo é igual ao trágido oferecido. Por outro lado, quando o tráfego oferecido é excessivo, então o algoritmo de controlo de fluxo deverá rejeitar parte deste sendo, neste caso, o efetivo, igual à diferença entre o oferecido e o rejeitado. Por conseguinte, à medida que o algoritmo de encaminhamento diminui o atraso médio, o controlo de fluxo aceita a entrada de mais tráfego na rede.

tráfego efetivo

atraso médio

Portanto, é o trabalho de um bom controlo de fluxo, 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 taxa de pacotes perdidos). Isto poderá ser atingido através da garantia de um tratamento equitativo dos diferentes fluxos de pacotes, ao fornecer a qualidade de serviço requerida.

Mas isto leva-nos a uma questão algo pertinente: devemos julgar a qualidade de serviço com base na sua equidade ou com base na utilização dos vários fluxos? Para discriminar ambas as situações consideremos a Figura 8.1, onde temos uma rede de quatro nós e onde se assume que a capacidade de cada ligação é 100.

Olhando para a Figura 8.1 podemos primeiramente tentar perceber qual seria o efeito de uma maximização do tráfego efetivo (ou seja, como teríamos a máxima utilização da rede): uma vez que a ligação do fluxo 1 é mais invasiva que as 2, 3 e 4, damos preferência às 2, 3 e 4, uma vez que temos mais ligações a funcionar, com um tráfego efetivo total de $100 + 100 + 100 = 300$.



figura 8.1

Se, por outro lado, pretendermos uma mesma taxa de transmissão para todos os fluxos (ou seja, máxima equidade), então teremos de ver em que ligações é que mais do que um fluxo ocorre. Como isto acontece sempre com um máximo de dois fluxos por ligação e uma ligação tem capacidade 100, então temos de dividir a capacidade, de forma igual, por cada fluxo, ficando cada um com 50, obtendo um tráfego efetivo total de 200.

Havendo, por fim, uma partilha equitativa dos recursos, então temos de olhar para critérios de justiça vendo o nível de utilização, aproveitando uma divisão de 25 para o fluxo 1 e 75 para todos os outros, atingindo um tráfego efetivo de 250.

Controlo de fluxo através de janelas

Consideremos um fluxo de dados (por vias de pacotes), entre um emissor *A* e um receptor *B*. Por cada unidade de dados recebida, o receptor *B* notifica o emissor *A* do envio para *A* de uma **permissão**.

permissão

Uma permissão pode ser transmitida num pacote de controlo dedicado ou pode ser encavalitada (vulgarmente dito que pode ser *piggybacked*) num pacote de dados enviado no sentido contrário. Quando um dispositivo recebe uma permissão, o emissor *A* fica autorizado a enviar mais uma unidade de dados para o receptor *B*.

Um esquema de controlo de fluxos através de janelas pode ser combinado com um protocolo ARQ de controlo de erros, pelo que, neste caso, as permissões servem também de *acknowledgments*.

Um fluxo entre um emissor *A* e um receptor *B* também se diz 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 por janela). Note-se que neste passo o emissor e o receptor 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.

tamanho da janela

Este tipo de controlo de fluxo, no fundo, já foi abordado anteriormente em disciplinas como Fundamentos de Redes (a3s1), aquando do estudo da camada de transporte, mas em particular do controlo de congestionamento do protocolo TCP, que usa janelas temporais para controlar os seus fluxos. Ainda assim, este protocolo usa uma especificidade de janelas temporais que iremos abordar de seguida.

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, sendo que, 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 receptor poderá 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.

Protocolos como o de transporte TCP usam estratégias para a aplicação de janelas no controlo de fluxos como as **janelas extremo-a-extremo**. Em geral, o tamanho da janela é αW (onde α e W são números positivos). Cada vez que um grupo de α unidades de dados é recebido no nó destino, é gerada uma permissão, autorizando o envio de um novo grupo de α unidades de dados. De seguida, considera-se que $\alpha = 1$ e que cada unidade de dados é um pacote.

janelas extremo-a-extremo

Consideremos, para o efeito, que o atraso de ida-e-volta é dado por d e o tempo de transmissão médio por cada pacote é-nos dado por X . Se $d \leq WX$, então a transmissão de W pacotes demora mais que o atraso de ida-e-volta d ; assim, o emissor poderá transmitir à velocidade máxima de $1/X$ pacotes por segundo. Por outro lado, se $d > WX$, então 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. Assim, a frequência de transmissão é dada por (8.1).

$$r = \min \left\{ \frac{1}{X}, \frac{W}{d} \right\} \quad (8.1)$$

Claro está que tem que existir um *trade-off* 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 lado, as janelas deverão 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 pela expressão em (8.2).

$$W = \left\lceil \frac{d}{X} \right\rceil \quad (8.2)$$

Contudo, esta aproximação de controlo de fluxo realizado por janelas possui as suas desvantagens. Primeiro, não permite assegurar uma taxa mínima de transmissão. Na verdade, quantos mais fluxos forem submetidos na rede, menor é o tráfego efetivo que cada fluxo obtém. Segundo, não fornece um controlo adequado do atraso, sendo que à medida que o número de fluxos aumenta, o tráfego tende para um valor constante — é limitado pela capacidade das ligações. Assim, o atraso médio por pacote aumenta exponencialmente de forma proporcional ao número de fluxos. [16]

Controlo das taxas de transmissão

A função de controlo de fluxos pode atribuir a cada fluxo uma determinada **taxa de transmissão** máxima compatível com as suas necessidades. Este tipo de abordagem dá pelo nome de **traffic shaping** e pode, por exemplo, ter uma taxa definida na fase de estabelecimento de um circuito virtual.

Uma das formas de aplicar taxas específicas de transmissão de pacotes são as **janelas**. Através da manipulação de janelas podemos aplicar um conjunto de regras que nos permitem controlar quantos pacotes podem passar num fluxo em cada $1/r$ segundos. Neste caso em específico dir-se-ia que temos uma taxa de transmissão r . No entanto, este esquema tende a introduzir grandes atrasos quando o tráfego é de rajada, sendo preferível admitir W pacotes a cada W/r segundos.

Atribuindo-se a um fluxo uma taxa de transmissão de r pacotes por segundo e uma janela de W pacotes, então inicialmente o emissor manterá um contador x que indica, em cada instante, o número de pacotes dessa janela que ainda pode ser transmitido (basicamente diríamos que x é inicializado em W). Sempre que um pacote é transmitido, o contador x é decrementado e passados W/r segundos é novamente incrementado (se repararmos bem, temos aqui um mecanismo muito próximo de um semáforo). Neste passo, os pacotes só são enviados para a rede se $x > 0$, onde o número máximo de temporizadores é W .

Note-se, não obstante, que o método do controlo de fluxo por janelas extremo-a-extremo é semelhante a este com a diferença de que o contador é apenas incrementado quando são recebidas as permissões.

Apesar da facilidade de implementação, este método acaba por ser computacionalmente intenso, uma vez que temos de processar os vários valores e devidas alterações de W temporizadores em simultâneo.

Uma alternativa que foi criada dá pelo nome de **leaky bucket**. O *leaky bucket* (estratégia que dá pelo nome de balde com fugas) tenta controlar a taxa de transmissão de pa-

taxa de transmissão
traffic shaping

janelas

leaky bucket

cotes num determinado fluxo, fazendo transparecer que o tráfego é transmitido a uma taxa diferente e constante.

Neste método, o contador de que há pouco referímos é incrementado periodicamente em cada $1/r$ segundos, até um máximo de W pacotes. De forma mais definida, incialmente temos que existe uma fila de espera para pacotes e uma fila de espera de permissões, com capacidade de W permissões. Uma vez que é gerada uma permissão a cada $1/r$ segundos, os pacotes só serão transmitidos quando existe uma permissão disponível na fila de espera respetiva. Na Figura 8.2 podemos ver uma pequena representação desta abordagem.

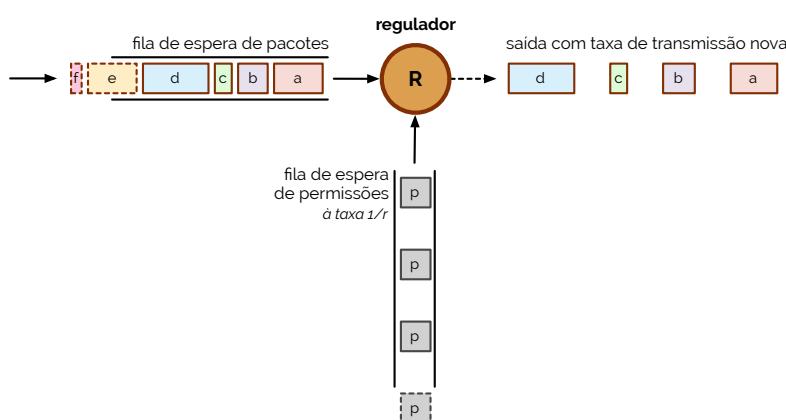


figura 8.2

O uso deste método é, claramente, computacionalmente menos intenso que o controlo por janelas, uma vez que apenas temos um contador para definir os instantes de geração de permissões. [16]

Atribuição de taxas de transmissão

Voltando à nossa questão inicial, como é que podemos aplicar as taxas de transmissão aos vários fluxos, ou seja, garantindo algum critério de justiça?

Consideremos uma rede, representada na Figura 8.3, em que as ligações têm todas capacidade para 120 pacotes por segundo.

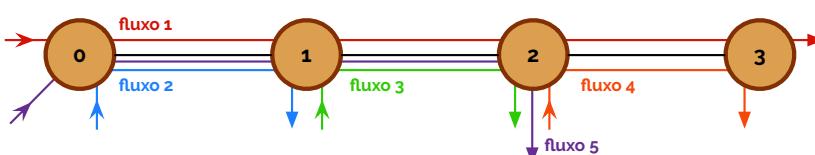


figura 8.3

Uma solução justa seria atribuir a todos os fluxos uma mesma taxa única. Havendo porções da rede em que para uma ligação temos 3 fluxos distintos, poderíamos dividir a capacidade total por 3 fluxos, ficando com capacidade igual de 40 pacotes por segundo. Contudo, se virmos bem, ao aplicarmos 40 pacotes por segundo o nosso fluxo 4 não está a aproveitar toda a capacidade de que dispõe — podia usar 80 pacotes por segundo sem prejudicar os fluxos 1, 2, 3 e 5.

Para resolver este problema surge o princípio de **equidade max-min** (em inglês *max-min fairness*). Segundo este princípio, maximizam-se os recursos atribuídos aos fluxos que podem usar menos recursos.

equidade max-min

Uma forma alternativa de formular este princípio está no seguinte: 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 . [16]

9. Escalonamento em Redes com Comutação de Pacotes

Uma das funções mais importantes dos nós que constituem uma rede (um *router*) é saber manipular os conjuntos de pacotes que lhe chegam, colocando-os num sistema de fila de espera e depois escaloná-los para uma interface de saída. Um requisito importante, com isto, de cumprir, está na necessidade de eficiência, de forma a que, quando um pacote saia de um *router*, o possa fazer da forma mais rápida possível, pelo melhor caminho e dando aso a um conjunto de prioridades que certos pacotes poderão ter perante outros.

Contudo, no caso de cenários em que a intensidade de tráfego é muito alta, um *router* poderá receber muitos pacotes em simultâneo, que serão encarreirados todos num *buffer*. Dado que um *buffer* é uma estrutura finita, é possível que este se torne cheio — neste caso, alguns pacotes deverão ser descartados. Mas aqui voltamos à mesma questão que antes com o escalonamento: que regras ou políticas deverão ser tidas em conta para decidir que pacote pode/deve ser descartado?

Nesta unidade iremos então ver alguns algoritmos de escalonamento e de descarte dos pacotes, para que possamos perceber melhor o contexto e quais são as implementações mais comuns.

Equidade das disciplinas de escalonamento

Já o referimos anteriormente neste documento, mas quando uma ligação está congestionada, o problema mais básico que se coloca à função de escalonamento é a divisão de um recurso escasso, por um conjunto de fluxos com iguais direitos, mas com diferentes necessidades de utilização desse mesmo recurso.

Idealmente, a atribuição deveria ser feita de acordo com o princípio de equidade max-min, isto é, os recursos deveriam ser dados aos fluxos por ordem crescente de necessidade, sendo que a nenhum fluxo é atribuída uma quantidade maior que a sua necessidade e aos fluxos cuja necessidade não tenha sido satisfeita é atribuída uma igual quantidade de recursos.

Consideremos assim um conjunto de fluxos $1, 2, \dots, n$ com necessidades x_1, x_2, \dots, x_n e ordenados pelas suas necessidades ($x_1 \leq x_2 \leq \dots \leq x_n$) e uma ligação com capacidade C . A atribuição dos recursos da ligação é efetuada do seguinte modo: inicialmente todos os fluxos têm direito a $d = C/n$; iterativamente, por cada necessidade, verifica-se se d é menor que esta — caso seja, então atribui-se d aos fluxos $1, 2, \dots, n$; caso contrário, atribui-se x_1 ao fluxo da capacidade em questão e o seguinte terá direito a $d = d + (d - x_{\text{fluxo}})/(n - 1)$.

Olhando para o fator de equidade, também o podemos ter com pesos atribuídos aos fluxos. Neste caso os recursos são atribuídos aos fluxos por ordem crescente de necessidade, estado esta normalizada em relação ao peso. A nenhum fluxo é atribuída uma quantidade de recursos maior do que a sua necessidade e a fluxos cuja necessidade não tenha sido satisfeita é atribuída uma quantidade de recursos proporcional ao seu peso.

Idealmente, a função de escalonamento deverá procurar proteger sempre os fluxos bem comportados, dos fluxos mal comportados. Considera-se, para tal, que 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.

Já de seguida iremos abordar várias **disciplinas de escalonamento**, que se podem classificar em disciplinas com e sem conservação de trabalho. Diz-se que uma disciplina conserva trabalho quando a ligação está inativa apenas se não houver qualquer pacote à espera de ser transmitido. Por outro lado, diz-se uma disciplina não conserva trabalho quando a ligação pode estar inativa mesmo quando há pacotes em fila de espera. [15]

disciplinas de escalonamento

Escalonamento por ordem de chegada com FIFO

A ideia de aplicação de uma disciplina de **First-In-First-Out** (FIFO) é bastante simples. O primeiro pacote que chega a um *router* é o primeiro a ser transmitido. Note-se que

First-In-First-Out

o sistema de fila FIFO também poderá ser denotado de **First-Come, First-Served** (FCFS). FIFO, como sendo a disciplina mais simples de todas, é fácil de implementar e apresenta-se como uma boa alternativa para sistemas que não permitam que haja um grande *overhead* de processamento. Contudo, para qualquer caso, a vantagem do uso de uma disciplina FIFO está no fornecimento de um atraso previsível que os pacotes irão sofrer enquanto passam pelo *router*.

Considere-se, para o efeito, C como a rapidez da ligação e B como o tamanho máximo do *buffer* e teremos sempre que D sendo o intervalo de atraso será sempre menor ou igual que o quociente de B/C .

A maior limitação das filas FIFO está na sua incapacidade de discriminar pacotes com base na sua classe de serviço, pelo que não implementa estratégias de avaliação de parâmetros de qualidade de serviço (QoS). Por exemplo, um fluxo único que possa ter uma chegada de pacotes sucessivos e em *burst* poderá facilmente monopolizar o espaço de *buffer* completo antes de todos os outros, fazendo com que estes sofram recusas de serviço até que o *burst* seja tratado. Em suma, 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 estão protegidos. Por esta mesma razão este é o algoritmo de escalonamento por defeito dos equipamentos, na ausência de configuração de outros.

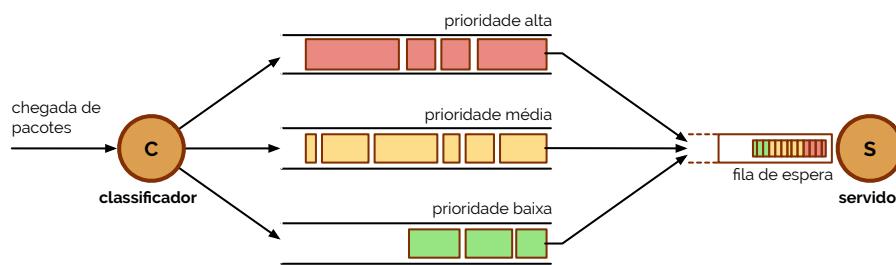
Escalonamento com base em prioridade estrita

A ideia-base por detrás do uso de sistemas de fila de espera com prioridades é que os pacotes possam ser classificados em várias emissões de tráfego à chegada do *router* em uma ou mais classes de serviço, como podemos ver na Figura 9.1. Esta funcionalidade é necessária, por exemplo, para a arquitetura de diferenciação de serviços, em termos de qualidade de serviço (QoS). Estas emissões, também denominadas de fluxos, possuem um conjunto de parâmetros de QoS que os permitem ser juntados por serem comuns e, todos eles, definirem classes de serviço. Por exemplo, é comum que tais parâmetros sejam o endereço IP de destino, endereço IP de origem, número de porto, entre outros... Tal informação de prioridade é então comunicada ao longo do caminho que o fluxo faz, por exemplo, marcando diferentes **DSCPs** (sigla para *Differentiated Services Code Bits*) nos pacotes, valor que é usado para distinguir se um determinado pacote merece ou não ser prioritizado. Caso seja, então o *router* deverá guiá-lo até uma fila de espera para tais pacotes. [15]

First-Come, First-Served

DSCP

figura 9.1



Portanto, olhando para a Figura 9.1 podemos verificar como é que o algoritmo trabalha. Essencialmente, os pacotes, quando chegam a um *router* são passados por um elemento classificador, que tem o trabalho de verificar qual é a classe de serviço a que correspondem. Feito isso, cada classe entra para uma fila própria interna, ordenadas por prioridade. Dentro destas filas os pacotes possuem a mesma ordenação que na disciplina FIFO, sendo que o algoritmo, entretanto, guia para o servidor os pacotes da prioridade mais alta, seguidos dos da prioridade imediatamente mais baixos, e por aí em diante...

A grande vantagem da aplicação de sistemas de fila com prioridade está na segregação do tráfego em diferentes classes de serviço, sendo cada uma servida de forma diferente das outras. Esta segregação é ótima para casos de tráfego em tempo-real como VoIP ou transmissões de vídeo (com uma prioridade máxima), sobre tráfego de não-tempo-real como tráfego HTTP. Contudo, uma dificuldade que se tem é não haver garantias de

movimentação dos pacotes com prioridades mais baixas, dado que o algoritmo só se preocupa em deixar avançar quem tem mais prioridade. Consideremos que o tráfego de um fluxo de alta prioridade é incessante — se assim for os pacotes de prioridades mais baixas irão ser atrasados no seu envio até que sejam descartados por estarem demasiado tempo parados e a ocupar espaço de *buffer*.

Apesar destas situações, há casos em que é do nosso melhor interesse aplicar o algoritmo de prioridade estrita. Consideremos o caso em que um fornecedor de um serviço, de forma a transportar tráfego VoIP, terá de seguir determinados regulamentos. Por exemplo, um destes regulamentos poderá ser “nenhum tráfego VoIP deverá ser descartado, independentemente do congestionamento da rede”. Tal regulamento poderá ser suportada por esta disciplina, uma vez que estaríamos a falar de tráfego de alta prioridade que não teria qualquer limitação a nível de largura de banda consumida.

Um outro cenário possível é a proteção e a prioritização de pacotes que transportam informações/atualizações acerca de protocolos de roteamento, durante períodos de congestionamento. Esta prioritização torna-se importante, uma vez que é bom que as tabelas de roteamento estejam estáveis para que haja boa resposta, por exemplo, numa mudança de topologia.

Escalonamento com rotação (round-robin, WRR e DRR)

Um algoritmo algo semelhante à disciplina de prioridade estrita é o **round-robin** (também conhecido como RR). De facto, se olharmos novamente para a Figura 9.1 podemos alterar ligeiramente o esquema para atingir a solução com *round-robin*. Veja-se assim a Figura 9.2 onde temos uma representação do que acontece com RR. [15]

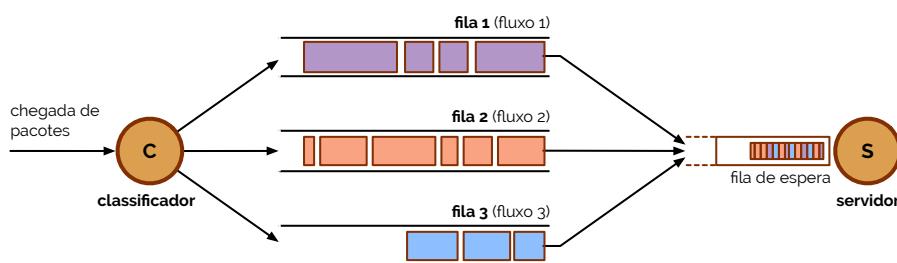


figura 9.2

Como podemos verificar pela Figura 9.2 temos que o RR cria tantas filas como fluxos. Neste processo aplica uma política de justiça entre eles, de forma a que estes sejam todos servidos rotativamente e um por cada fila, desde que não vazia. Este processo não permite que haja diferenciação de serviços embora, em comparação com a disciplina FIFO, o RR sirva o mesmo número de pacotes de todos os fluxos ativos. Por consequência desta forma de execução, há um benefício dos pacotes de maiores dimensões.

Mas olhando para a forma como o *round-robin* funciona é fácil perceber qual a alteração que se deve executar, de forma a que se possa ter o melhor de dois mundos: haver uma diferenciação de serviços e ainda fazer ciclos de leitura rotativa sobre os fluxos — aplicar um conjunto de pesos às várias filas. A esta disciplina damos o nome de *Weighted Round Robin*, **WRR**. É então atribuído um peso p_i a cada fila de espera, peso este proporcional à taxa de serviço a proporcionar a cada fluxo. [15]

Assim, sob uma disciplina WRR, em cada ciclo, serve-se um número de pacotes de cada fila tal que a soma dos seus tamanhos seja proporcional ao seu peso. Para que isto possa acontecer é necessário que se tenha noção *a priori* do comprimento médio dos pacotes. Note-se que com WRR a ligação poderá ficar demasiado tempo a servir cada fluxo de pacotes, tendo um impacto negativo no *jitter* introduzido pela ligação.

Se considerarmos os seguintes pesos $p_1 = 0.5$, $p_2 = 0.75$ e $p_3 = 1.0$ e os seguintes comprimentos de pacotes médios $L_1 = 50$ bytes, $L_2 = 500$ bytes e $L_3 = 1500$ bytes. Os pesos normalizados são $p_1 = 0.5/50 = 60/6000$, $p_2 = 0.75/500 = 9/6000$ e $p_3 = 1.0/1500 = 4/6000$. Assim sendo o número de pacotes por ciclo é 60 para o 1, 9 para o 2 e 4 para o 3.

Por fim, uma pequena alteração que podemos fazer é a aplicação de um serviço até um determinado limiar de cada uma das filas, de forma a que se corte com o benefício de pacotes que possuem maiores dimensões e com a necessidade de saber qual o tamanho dos pacotes.

Aqui, numa disciplina que dá pelo nome de *Deficit Round Robin*, **DRR**, temos que 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, então o crédito disponível é zero.

Se se considerarem limiares diferentes, então a taxa de serviço de cada fluxo será proporcional ao limiar da sua fila de espera.

Uma representação desta disciplina pode ser vista na Figura 9.3.

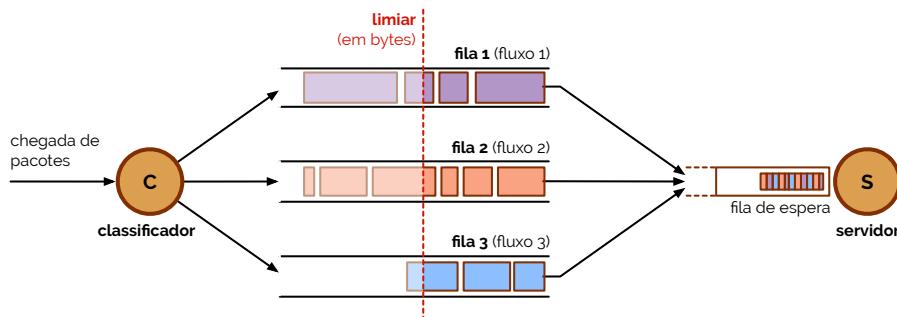


figura 9.3

Consideremos, para o efeito, um limiar de 1000 bytes em todos os fluxos para o exemplo dado há pouco. Num primeiro ciclo a primeira fila não será servida, pelo que obtém um crédito de 1000, o 2 é servido (ficando com crédito de 200) e o 3 não é servido (ficando com um crédito de 1000). Já num segundo ciclo teríamos que o 1 é servido, obtendo um crédito de 500, o 2 fica com crédito 0 e o 3 é servido, ficando com crédito de 800. [18]

Escalonamento por aproximação ao sistema GPS (WFQ e SCFQ)

Um algoritmo que resume todas estas ideias teorizando-as num só modelo é o *Generalized Processor Sharing*, **GPS**. Este algoritmo é considerado como ideal, uma vez que proporciona equidade perfeita, com base num modelo de fluidos, em que o tráfego é considerado infinitamente divisível.

Neste processo existe uma fila de espera por fluxo e é atribuído um peso p_i a cada fila. Assim que uma fila de espera recebe tráfego, então este começa imediatamente a ser servido, em paralelo com o restante tráfego, a uma taxa proporcional ao seu peso.

Num sistema com N filas e em que a ligação tem capacidade C a taxa de serviço garantida é dada por (9.1).

$$r_i = \frac{p_i}{\sum_{j=1}^N p_j} C \quad (9.1)$$

Dado que é um sistema que conserva trabalho, 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, dada pela expressão (9.1) para um conjunto de j ativos.

Um fluxo diz-se ativo (ou *backlogged*) quando tem pacotes no sistema, quer em transmissão, em fila de espera, ou ambos. Um fluxo ativo no intervalo $[\tau, t]$ com serviço nesse mesmo intervalo $S_i(\tau, t)$ obedece a (9.2).

$$\frac{S_i(\tau, t)}{S_j(\tau, t)} \geq \frac{p_i}{p_j}, \quad j = 1, 2, \dots, N \quad (9.2)$$

Contudo, como já avançámos no primeiro parágrafo, este algoritmo é meramente teórico, sendo impossível de ser implementado na prática. Ainda assim, a sua importância é bastante alta, sendo que é usado como base para outros algoritmos.

Um dos exemplos de algoritmos que tentam fazer uma aproximação ao sistema GPS é o *Weighted Fair Queueing*, **WFQ**, em que se tenta servir os pacotes pela ordem em que terminariam o serviço no sistema GPS.

Então, 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. Já um *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, temos que o RN é dado por (9.3).

$$RN(\tau_i + t) = RN(\tau_i) + \frac{1}{\sum_j \text{ativos} p_j} t \quad (9.3)$$

O RN é processado sempre que o número de fluxos ativos se altera quando um pacote de um fluxo que não tem pacotes no sistema ou quando um pacote de um fluxo termina de ser transmitido e o fluxo não tem nenhum outro pacote na fila de espera. Assim, o $FN_{i,k}$ do pacote k com comprimento L_k pertencente à fila i é dado por (9.4).

$$FN_{i,k} = \max(FN_{i,k-1}, RN) + \frac{L_k/C}{p_i} \quad (9.4)$$

Por forma a evitar o cálculo do RN do WFQ, criou-se uma nova estratégia denominada de *Self Clocking Fair Queueing*, **SCFQ**, que substitui este parâmetro pelo valor de FN do pacote em serviço, FN_s , qualquer que seja o fluxo a que este pertence. Assim, o $FN_{i,k}$ do pacote k com comprimento L_k pertencente à fila i é dado por (9.5).

$$FN_{i,k} = \max(FN_{i,k-1}, FN_s) + \frac{L_k}{p_i} \quad (9.5)$$

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.

Como também vimos anteriormente, na unidade anterior, o *leaky bucket* é um formador de tráfego que permite impor um majorante ao tráfego gerado por um dado fluxo. Se A representar a quantidade de tráfego do fluxo i que entrou na rede no intervalo de tempo $[\tau, t]$, então $A(\tau, t) \leq \sigma_i + q_i(t - \tau)$, como podemos ver na Figura 9.4.

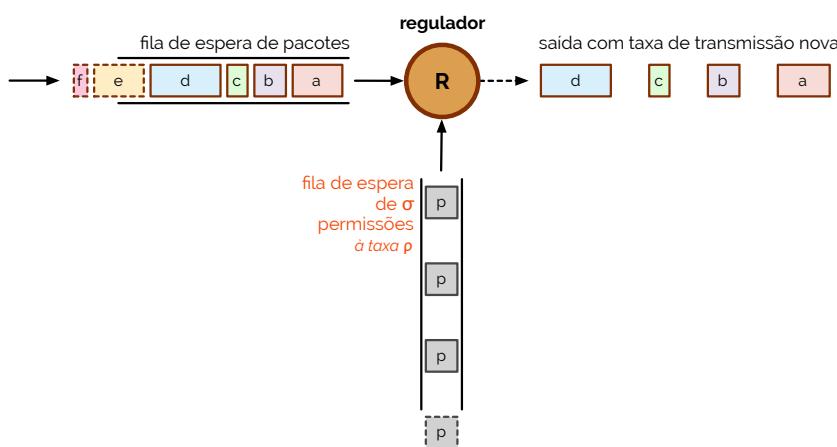


figura 9.4

Numa disciplina GPS, se designarmos $S_i(\tau, t)$ o tráfego de um fluxo i servido num intervalo de tempo $[\tau, t]$, temos (9.6).

$$S_i(\tau, t) \geq r_i(t - \tau) \quad \text{em que } r_i = \frac{p_i}{\sum_j p_j} C \quad (9.6)$$

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 dado por (9.7).

$$\begin{aligned} Q_{i,\max}(t) &= A_{i,\max}(0,t) - S_{i,\min}(0,t) \\ &= \sigma_i + \rho_i t - r_i t \\ &\leq \sigma_i \quad \Leftrightarrow r_i \geq \rho_i \end{aligned} \quad (9.7)$$

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, temos (9.8).

$$D_i = \frac{\sigma_i}{r_i} \quad (9.8)$$

Numa disciplina WFQ, o atraso máximo é maior que no GPS porque a informação é transmitida sob a forma de pacotes. Consideremos assim um fluxo i formatado por um *leaky bucket* com parâmetros σ_i e q_i que atravessa n ligações ponto-a-ponto, uma capacidade da ligação j (C_j), uma largura de banda reservada para o fluxo i em todas as ligações ($r_i \geq q_i$), um tamanho máximo dos pacotes do fluxo i , L_i e um tamanho máximo dos pacotes de todos os fluxos, L_{\max} . Prova-se que o atraso mínimo (D_i) que os pacotes do fluxo i sofrem é dado por (9.9), onde Γ é o atraso total de propagação de todas as ligações. [18]

$$D_i = \frac{\sigma_i + (n - 1)L_i}{r_i} + \sum_{j=1}^n \frac{L_{\max}}{C_j} + \Gamma \quad (9.9)$$

Métodos de descarte de pacotes

Quando os pacotes não têm direito a espaço no *buffer* dos nós de uma rede, então estes terão de sofrer a consequência do **descarte**. Esta ação, contudo, precisa de ser avaliada em quatro parâmetros de forma a que haja justiça e rigor neste processo: posição de descarte, prioridade de descarte, grau de agregação e descarte antecipado.

Por **posição de descarte**, um pacote poderá ser descartado se estiver na cauda de uma fila — normalmente usado por omissão, sendo a mais simples de implementar (o pacote nem chega a entrar na fila), acontecendo essencialmente quando uma fila tem muitos pacotes pertencentes a poucos fluxos, onde se o pacote não pertencer a nenhum desses fluxos, então a estratégia não pode ser considerada justa —, se estiver numa posição aleatória — onde se escolhe aleatoriamente um pacote para ser eliminado, sendo que aqui os fluxos com mais pacotes são os mais penalizados (é uma estratégia mais justa) — ou se estiver na cabeça da fila — aqui retira-se o pacote mais antigo da fila e aceita-se o que chegou (o que é computacionalmente leve), sendo uma estratégia tão boa como a posição aleatória em termos de justiça e útil quando o controlo de fluxo é baseado em perdas de pacotes.

Outro parâmetro é a **prioridade de descarte**. Aqui o emissor ou a rede 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 também ser descartados, pois deixam de ter qualquer utilidade. Já um método de descarte possível consiste em dar maior prioridade de descarte aos pacotes que passaram por menos ligações, ou seja, que usaram menos recursos.

descarte

posição de descarte

prioridade de descarte

Um terceiro parâmetro é o **grau de agregação**. O grau poderá ser um de dois possíveis: haver uma agregação de fluxos ou uma agregação da memória dedicada às filas de espera. No primeiro caso, 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, sendo que quanto mais fluxos forem agregados, menor a proteção entre os fluxos pertencentes ao mesmo. Já do lado de uma agregação da memória, se existe uma fila de espera por fluxo de pacotes e a memória é partilhada por todas as filas, então consegue-se uma atribuição de memória max-min *fair* quando se descarta o último pacote da fila mais longa. Este paradigma com WFQ corresponde a descartar o pacote com maior FN.

Por fim, temos o **descarte antecipado**. Aqui podemos descartar pacotes em dois momentos próprios, sendo que em ambos, a fila está cheia e não cabe mais nenhum pacote. Assim sendo, usam-se alguns algoritmos próprios de descarte antecipado como é o caso do **RED** (*Random Early Discard*) ou do **WRED** (*Weighted Random Early Discard*): no primeiro, quando cada pacote chega à fila, este é descartado com uma probabilidade diretamente proporcional à ocupação da fila, não proporcionando diferenciação de qualidade de serviço; já no segundo, atribuem-se diferentes probabilidades a pacotes pertencentes a diferentes classes de serviço. Mais informação acerca destes tópicos poder-se-ão consultar nos apontamentos de Arquitetura de Redes (a3s2). [18]

grau de agregação

descarte antecipado

RED, WRED

10. Engenharia de Tráfego em Redes de Comutação de Pacotes

Quando estamos perante uma rede que possui um conjunto de fluxos sobre ela há que saber determinar com que taxas cada um destes deverá lançar os pacotes. Surge assim a necessidade de aplicar **engenharia de tráfego**.

A engenharia de tráfego é assim a tarefa de escolher para cada fluxo do conjunto dos presentes numa rede, qual a percentagem da sua taxa de chegada que deverá ser roteada por cada um dos caminhos possíveis para destinos.

Uma das formas de efetuar engenharia de tráfego é através da criação de regras de roteamento para encontrar um único caminho até um determinado destino, sendo este processamento aplicável a fluxos de uma rede. Enquanto que neste esquema não é permitido qualquer tipo de bifurcação, também será necessário garantir condições de simetria, ou seja, que o caminho por onde se leva um pacote de uma origem para um destino será o mesmo, mas num sentido inverso, do destino para uma origem — usa as mesmas ligações.

Consideremos assim uma variável binária x_{sr} associada a cada fluxo s de uma rede (de um conjunto de fluxos S). Mais, cada rota r (do conjunto de rotas possíveis R_s), quando possuir o valor de '1', então significa que o fluxo s é roteado por r . Qualquer solução de engenharia de tráfego com roteamento por caminho único deverá respeitar as seguintes regras, com as seguintes restrições: para cada fluxo s uma das suas variáveis x_{sr} associadas deverá possuir o valor '1', sendo que todas as outras deverão ter o valor '0'; para cada ligação (i, j) do conjunto A de ligações de uma rede, a soma de todas as taxas de chegada dos fluxos λ_s roteados através destas não poderão exceder a capacidade da ligação c_{ij} .

Como objetivo da engenharia de tráfego é então natural que se pretenda otimizar um ou mais parâmetros relacionados tanto com o desempenho de rede e/ou com a qualidade de serviço (QoS) fornecida pela rede. Opcionalmente, também se poderá pretender garantir limites máximos ou mínimos sobre outros parâmetros relacionados com os mesmos tópicos. Alguns exemplos destas práticas são a otimização do atraso médio global dos pacotes (usando a aproximação de Kleinrock, por exemplo), a otimização do pior caso de atraso médio de pacotes (para maximizar os parâmetros de justiça ao longo dos vários fluxos) ou até mesmo a otimização dos piores casos das taxas de carregamento das ligações (de forma a aumentar a robustez das redes em casos de crescimento de tráfego imprevisíveis).

Em suma, a melhor solução de engenharia de tráfego depende exclusivamente da função objetivo definida pelo operador de rede, sem que haja conflitos entre as várias funções criadas. [19]

engenharia de tráfego

Criação de algoritmos para aplicação em funções objetivo

Em Introdução à Inteligência Artificial (a3s1) estudámos algumas formas de tentar atingir funções objetivo que eram definidas por um conjunto de variáveis muito extenso ou simplesmente cujo processamento seria computacionalmente intensivo. Nessa mesma disciplina, e como possível aproximação à solução ideal — pelo menos implementável — concluímos que a utilização exclusiva de algoritmos exatos não seria solução para qualquer problema com estas restrições, mas antes uma conjugação entre um algoritmo exato e um conjunto de **heurísticas** já poderíamos ter uma redução do espaço de resultados.

Um **algoritmo exato** é uma formulação específica de um conjunto finito de tarefas com base em modelos matemáticos, por norma, envolvendo alguma computação mais exigente. Embora teoricamente obtenham sempre a solução ideal, como vimos na disciplina de Algoritmos e Complexidade (a2s2) são muito inefficientes para instâncias de problemas maiores, dado que ou não dão resultados conclusivos em tempo útil ou terminam por falta de memória.

Por contraste, os **algoritmos heurísticos** acabam por ser um componente bastante interessante a adicionar. Sendo muito simples, pelo que, por conseguinte, muito fáceis de programar e de encontrar soluções, possuem um problema: não dão garantias sobre as soluções que encontra serem as ótimas. Contudo, encontram sempre, se bem desenhadas, um conjunto de boas soluções, o que é tarefa bastante eficiente para problemas com uma complexidade elevada, fornecendo resultados em tempo útil.

A lógica do uso de ambos algoritmos exatos e heurísticos é análogo à seguinte situação: imaginemos que estamos à procura de um objeto com que estávamos há umas horas atrás, mas que, do nada, parece que desapareceu. Sabemos que foi algures em casa, mas encontrar esse objeto será uma tarefa semelhante, como o ditado popular o diz, a “encontrar uma agulha num palheiro”. Poderíamos começar por varrer a casa toda de uma ponta a outra, mas para encontrar um objeto tão pequeno demorariamos uma eternidade — não iríamos encontrar uma solução em tempo útil. Contudo, a nossa intuição (heurística) diz-nos que uns dias atrás estivemos com esse objeto nas mãos num determinado compartimento da casa, pelo que podemos começar por investigar esse sítio. Basicamente o que isto significa é que, mesmo que a solução não esteja lá, poderá estar perto, o que decresceu em muito o nosso espaço de pesquisa, acelerando o procedimento. Ainda assim, embora no primeiro caso chegássemos a uma solução ótima porque encontrariamos precisamente o que queríamos, confiando no nosso instinto poderá levar a muita entropia e poderemos encontrar soluções próximas da nossa, como objetos semelhantes ou em estados semelhantes ao que estamos à procura.

Note-se que quando referimos algoritmos heurísticos não estamos a confundir com **métodos heurísticos**, sendo que uma coisa é distinta da outra. Um método heurístico é uma aproximação genérica para encontrar boas soluções num espaço de pesquisa, sendo aplicável a qualquer problema de otimização. Por outro lado, um algoritmo heurístico é um algoritmo específico de otimização que resultou da aplicação de um método heurístico a um problema específico de otimização.

Alguns métodos heurísticos são aplicados com base em duas estratégias muito próprias: a criação de soluções de raiz, como é o caso de algoritmos aleatórios, gulosos (*greedy*), aleatórios e gulosos (*greedy randomized*), entre outros...; ou a obtenção de melhores soluções através de outras soluções já conhecidas, como é o caso da pesquisa local, *simulated annealing*, entre outros... [19]

Nas secções à frente iremos abordar de ambos métodos heurísticos.

Criação de uma solução de raiz

Não tendo um espaço de resultados possíveis, num problema de otimização pode ser boa estratégia aplicar um método heurístico que nos permita criar uma solução de raiz.

heurísticas

algoritmo exato

algoritmos heurísticos

métodos heurísticos

Uma boa forma de começarmos a analisar esta abordagem talvez seja pelo estudo do paradigma mais simples — a **estratégia aleatória** (*random strategy*). Consideremos uma rede onde pretendemos calcular caminhos entre nós, tal como a representada na Figura 10.1, que usaremos até ao fim desta unidade.

estratégia aleatória

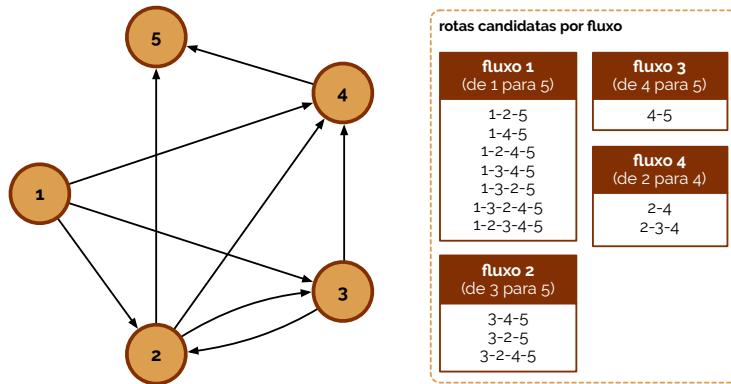


figura 10.1

Aplicando a estratégia aleatória teríamos que escolher, aleatoriamente para cada conjunto de rotas candidatas por fluxo, uma rota. Na Figura 10.2 podemos ver dois possíveis resultados da aplicação deste algoritmo.

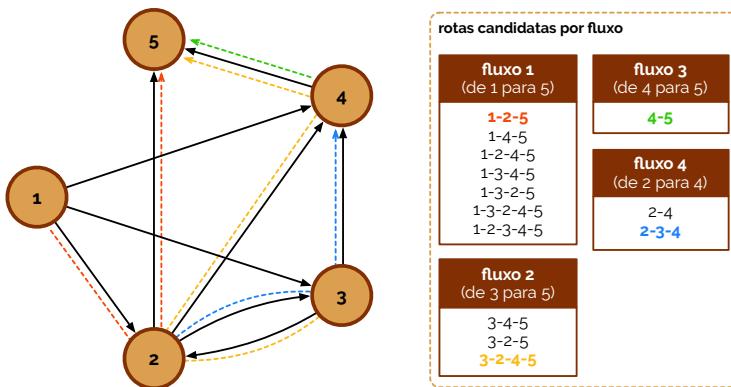
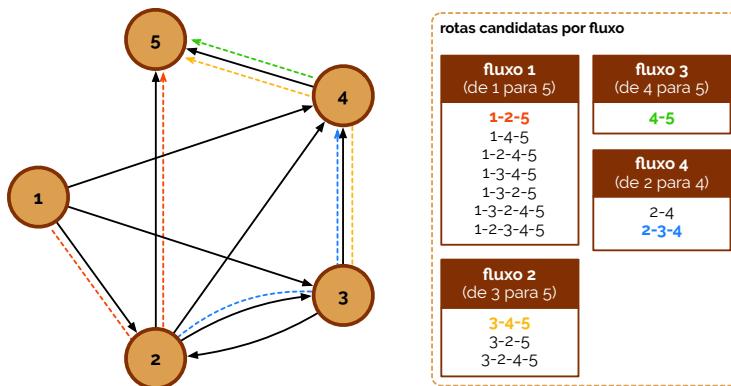


figura 10.2



Com esta estratégia estamos cegos a muitas variáveis do nosso sistema, contudo, se considerarmos apenas caminhos com melhores características como menor número de ligações, mais capacidade de ligação, entre outros, já seremos capazes de filtrar mais soluções.

Ainda assim, uma outra estratégia é passível de ser implementada, esta, denominada **estratégia gulosa** (*greedy strategy*). Aqui começamos por considerar uma rede sem qualquer rota escolhida, sendo que, depois, para cada fluxo s iremos atribuir um caminho que,

estratégia gulosa

juntamente com a rota anteriormente escolhida, fornece o melhor valor para a função objetivo. Na Figura 10.3 podemos ver a segunda fase desta estratégia, que corresponde à verificação do valor de atraso médio para cada uma das escolhas, tendo em conta um caminho já escolhido (o de 1 para 5, marcado com um traço mais largo).

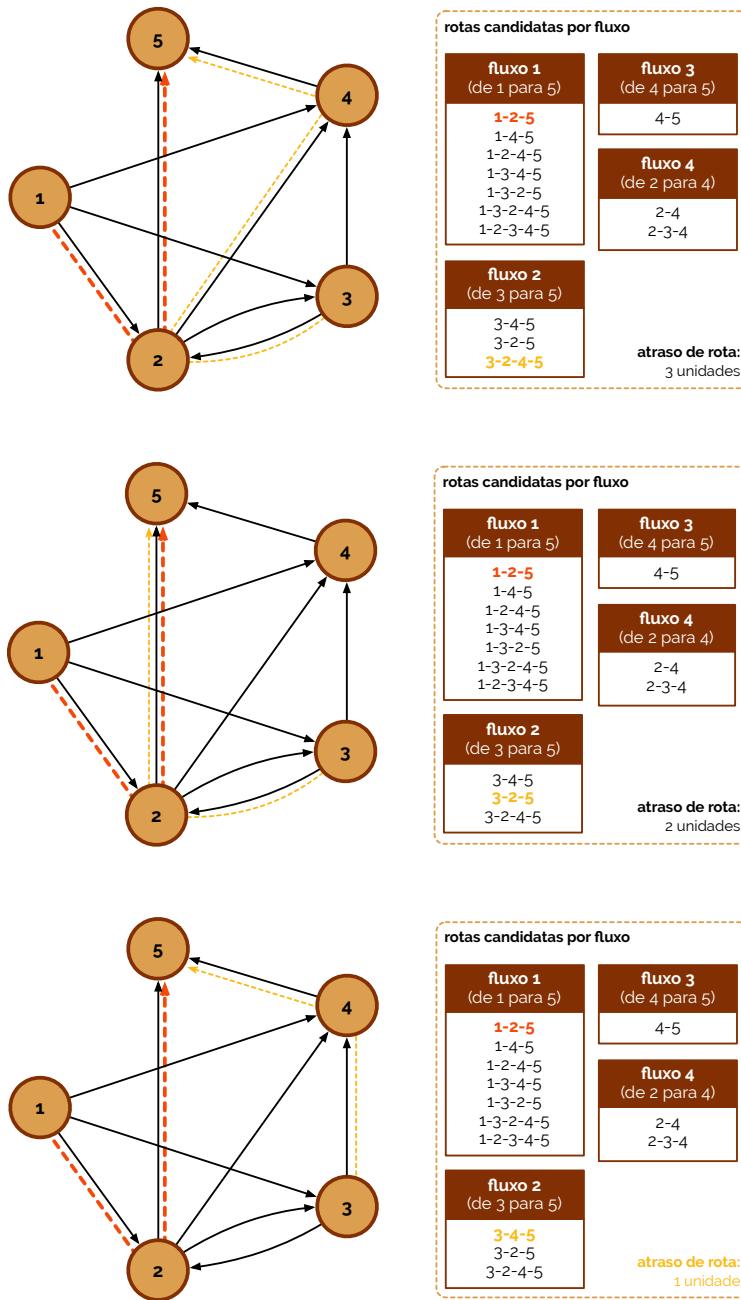


figura 10.3

Como podemos ver na Figura 10.3 a rota escolhida na segunda iteração foi a 3-4-5 para o fluxo 2. Esta escolha foi feita através do cálculo do atraso, sendo escolhido o que tiver menor atraso na soma com o atraso dos caminhos já escolhidos, para esta rota, como podemos ver em (10.1).

$$W_{3-4-5} = \frac{L_{12} + L_{25} + L_{34} + L_{45}}{\lambda_1 + \lambda_2} \quad (10.1)$$

Numa iteração seguinte, com a rota 3-4-5 escolhida, temos agora de escolher para os fluxos seguintes. Como o fluxo 3 só tem a rota 4-5 escolhe-se essa e depois, no fluxo 4, calcula-se o que tiver menor custo, como podemos ver na Figura 10.4.

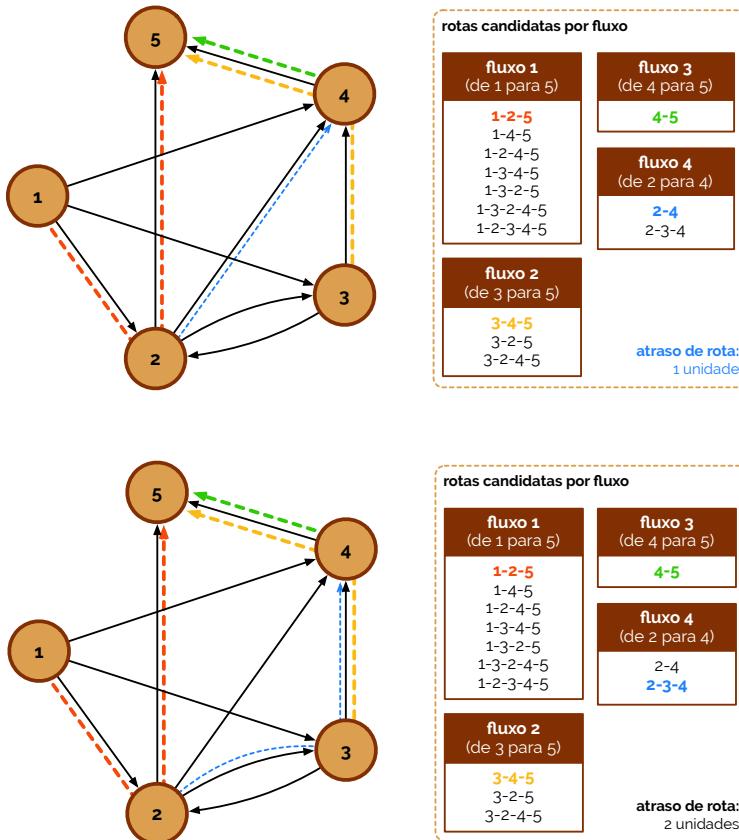


figura 10.4

Por fim, podemos juntar ambas estratégias e desenvolver uma unificada, que dá pelo nome de **estratégia gulosa aleatória** (em inglês *greedy randomized strategy*). Aqui o objetivo será obter uma solução do tipo da gulosa, mas diferente por cada vez que esta for executada. Uma primeira abordagem está em aleatorizar a ordem de fluxos a calcular os atrasos. Se não o pretendemos fazer assim podemos, para cada fluxo s , selecionar aleatoriamente uma rota entre todas as outras que dão valores ótimos para a função objetivo (não são todas as rotas por si só!). Caso contrário, se não quisermos fazer nenhuma destas abordagens por si só, podemos combinar ambas numa terceira abordagem. [19]

estratégia gulosa aleatória

Criar uma solução melhorada por vias de uma conhecida

Como referimos logo no início desta unidade, por vezes é-nos interessante melhorar resultados que já são conhecidos, uma vez que quando os problemas possuem uma complexidade relativamente elevada, de nada nos vale repetir processos de obtenção de novos resultados. Por esta mesma razão vale a pena estudar estratégias que se aproveitam de resultados já conhecidos para obter deles melhores e mais em conta com a nossa função objetivo.

Uma das estratégias mais importantes é a de **pesquisa local**. Contudo, esta estratégia possui um conjunto de variantes interessante e que iremos abordar com algum detalhe.

pesquisa local

A primeira variante da pesquisa local a abordar é a de movimento pelo **melhor vizinho**. Nesta estratégia começamos por uma solução inicial e tentamos movermo-nos para uma melhor através de variações locais. Estas variações deverão ser repetidas tantas vezes necessárias até que mais nenhuma consiga produzir um valor mais próximo do alvo da nossa função objetivo.

melhor vizinho

Em termos descritivos temos então o seguinte conjunto de procedimentos: primeiro, para uma dada solução já conhecida, calculamos todas as soluções vizinhas à atual e seleccionamos a melhor; segundo, se a melhor solução escolhida for melhor que a solução atual, então movemo-nos para esta e voltamos a repetir o primeiro passo; terceiro, caso contrário, paramos e a solução atual é a melhor encontrada, sendo o nosso resultado final.

Uma segunda variante é a de movimento pelo **primeiro vizinho**. No procedimento anterior, se o cálculo de todos os vizinhos for relativamente complexo, então podemos não estar a considerar uma solução muito eficiente. Para responder a isto a variante de movimento pelo primeiro vizinho poderá ser uma alternativa.

primeiro vizinho

Em termos descritivos, temos então o seguinte conjunto de procedimentos: primeiro, para uma dada solução já conhecida, calculamos todas as soluções vizinhas à atual e selecionamos a melhor; segundo, se uma solução de um vizinho for melhor que uma que já exista, então marcamos essa como solução atual e voltamos ao primeiro passo; terceiro, caso contrário, paramos e o resultado final é a solução em que ficámos.

Normalmente é mais eficiente usar a primeira variante, embora em alguns casos seja importante rever e reconstruir o espaço de vizinhos para cálculo, pelos motivos já indicados anteriormente.

O conjunto de soluções vizinhas é um atributo intrínseco ao problema que temos em mãos, contudo deverá ser definido cautelosamente de forma a que o algoritmo possa calcular todas as soluções em tempo útil. Em engenharia de tráfego o conjunto de vizinhança define-se da seguinte forma: para uma solução já conhecida, uma solução vizinha é tal que difere da atual em um caminho de um fluxo único, sendo que, quando o conjunto de rotas de cada fluxo é determinado, o número de soluções vizinhas é dada por (10.2).

$$\sum_{s \in S} (|P_s| - 1), \text{ sendo } P_s \text{ o conjunto de rotas do fluxo } s \quad (10.2)$$

Na Figura 10.5 podemos ver um processo de escolha entre vizinhos.

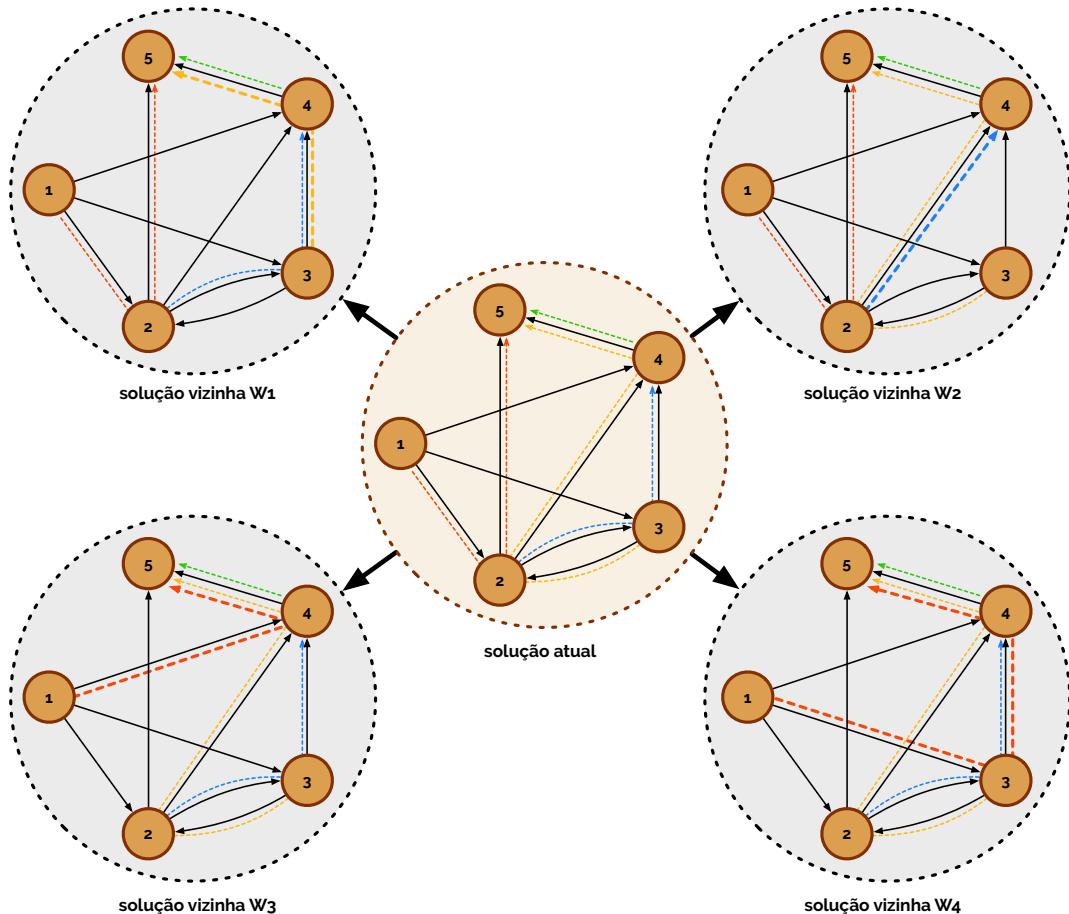


figura 10.5

Já na Figura 10.6 podemos verificar qual o impacto da utilização de heurísticas e a razão do nome do algoritmo de pesquisa local. Basicamente, como podemos ver na figura, temos um gráfico de uma função que apresenta uma monotonia inconstante, mas que pos-

sui um conjunto de mínimos relativos e máximos relativos. Esta função é a nossa função objetivo, sendo que é objetivo do algoritmo encontrar uma solução de minimização (ou maximização), ou seja, um mínimo relativo (ou máximo relativo). Se executássemos um algoritmo exato o resultado seria o mínimo (máximo) absoluto, contudo, como estamos a executar um algoritmo com base em heurísticas, então não há garantias de que estamos a obter um mínimo (máximo) absoluto, podendo apenas dizer que estamos a obter um (ou um conjunto de) mínimo(s) (máximo(s)) relativos. [19]

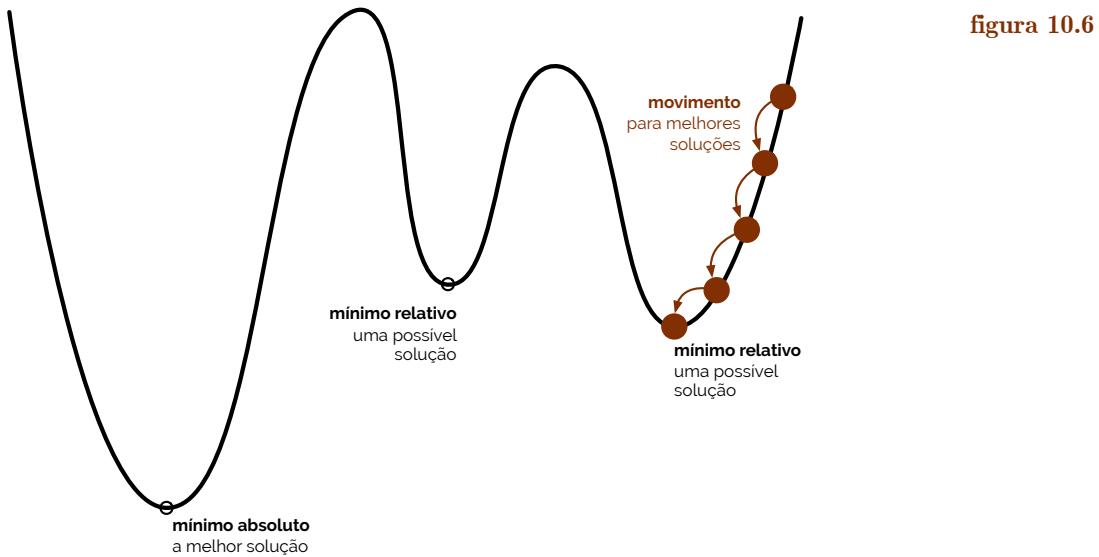


figura 10.6

Heurística Multi Start Local Search (MSLS)

Se olhamos bem para a Figura 10.6 podemos ver que o algoritmo de pesquisa local trata de encontrar uma boa solução em termos locais, ou seja, dependendo do local por onde este começa, o algoritmo irá procurar nas suas vizinhanças, soluções melhores.

Acontece que, como também já vimos, a solução encontrada pelo algoritmo de pesquisa local não terá, necessariamente, de ser a solução ótima — pelo que o sistema não garante este tipo de solução.

Uma forma de tentar atingir melhores soluções passa por reposicionar a origem do primeiro movimento que se faz no algoritmo. Uma vez que o primeiro movimento é realizado sobre uma solução já conhecida, o que se pode fazer é criar uma solução própria de raiz, executando o algoritmo de pesquisa local a partir desta. Depois, repetimos o processo até que uma determinada condição de paragem é atingida. Esta condição de paragem poderá ser a execução um dado número de vezes ou correr até que o valor final não seja melhor num determinado número de iterações.

Este algoritmo é denominado de *Multi Start Local Search (MSLS)* e a sua execução segue um padrão muito semelhante ao representado na Figura 10.7, onde podemos então ver que é procurada a solução a partir de vários locais de partida, por acaso, encontrando a solução ótima. Note-se, que, o “por acaso” aplica-se, uma vez que continuamos na presença de uma algoritmo heurístico e, como tal não temos forma de garantir encontrar a solução ideal.

Consideremos agora que o nosso conjunto de rotas para um fluxo é relativamente grande, por exemplo, tendo mais do que 200 rotas possíveis. A execução de tal algoritmo tornar-se-á bastante lenta, uma vez que o número de iterações e respetivas comparações é demasiado alto. Para isso, algo a fazer é reconstruir o conjunto de rotas para um determinado fluxo. Para tal, podemos usar o algoritmo de Dijkstra para calcular caminhos de menor custo sendo que, quando uma rota está para ser adicionada a um fluxo, atribuímos um custo de Dijkstra a cada ligação e corremos o seu algoritmo para determinar o camin-

MSLS

◎ Edsger Dijkstra

ho de menor custo. Estes custos de Dijkstra deverão ser bem selecionados, devendo depender somente na função objetivo a ser otimizada.

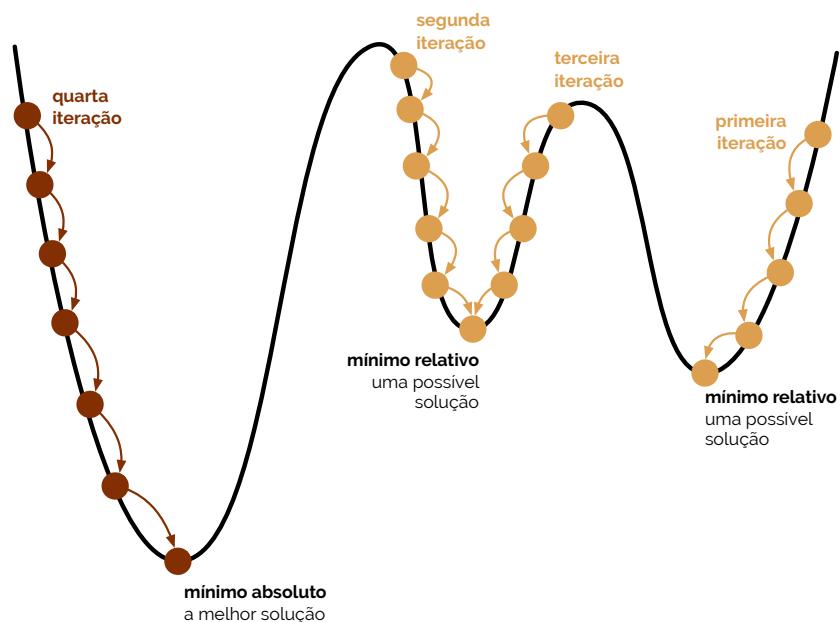


figura 10.7

Uma estratégia de implementação desta lógica é a *greedy randomized* que pode ser descrita da seguinte forma: primeiro começamos por considerar uma rede sem qualquer tipo de rota; segundo, determinamos uma ordem aleatória para o cálculo das rotas de um fluxo s ; terceiro, para cada fluxo s , e pela ordem designada, atribuímos um custo a cada ligação com base nos fluxos atualmente roteados, executamos o algoritmo de Dijkstra para calcular a rota do fluxo s e atualizamos a rede roteando a taxa média de pacotes de um fluxo s através da sua rota atribuída.

Ao invés da estratégia *greedy randomized* podemos usar a pesquisa local, por si, para reconstruir as rotas. Para isso, para cada fluxo s , determina-se a solução global com todos os fluxo exceto s , atribui-se um custo a cada ligação com base nos fluxos roteados e executa-se o algoritmo de Dijkstra para calcular o caminho de menor custo. Note-se que neste caso, a solução vizinha poderá ser igual à solução atual. [19]

E assim terminam os apontamentos da disciplina de Desempenho e Dimensionamento de Redes (a4s2).

1. Teoria de Probabilidades Revisitada

| | |
|---|----|
| Experiências aleatórias e probabilidades definidas sobre eventos..... | 2 |
| Probabilidades condicionadas e acontecimentos independentes | 3 |
| Regra de Bayes..... | 4 |
| Variáveis aleatórias..... | 4 |
| Processos estocásticos..... | 10 |

2. Cadeias de Markov e Sistemas de Filas de Espera

| | |
|--|----|
| Processos de Poisson..... | 11 |
| Cadeias de Markov em tempo contínuo..... | 12 |
| Probabilidades limite | 13 |
| Sistemas de fila de espera..... | 13 |
| Sistema de fila de espera M/M/1..... | 17 |
| Sistema de fila de espera M/M/m..... | 18 |
| Sistema de fila de espera M/M/1/m..... | 20 |
| Sistema de fila de espera M/M/m/m..... | 20 |
| Sistema de fila de espera M/G/1..... | 21 |

3. Simulações Baseadas em Eventos Discretos

| | |
|--|----|
| Elementos e estrutura de um simulador baseado em eventos discretos | 22 |
| Geração de números aleatórios | 23 |
| Análise dos resultados de uma simulação | 24 |

4. Partilha de Recursos de Comunicação

| | |
|--|----|
| Contexto, canais de comunicação, recursos e circuitos..... | 25 |
| Comunicações uni-serviço | 26 |
| Comunicações multi-circuito | 27 |
| Recursos de comunicação com comutação de pacotes | 28 |
| Disciplinas de escalonamento | 29 |

5. Encaminhamento de Pacotes numa Rede de Comutação

| | |
|--|----|
| Conceitos de encaminhamento em redes | 30 |
| Aproximação por independência (aproximação de Kleinrock) | 33 |

6. Otimização com Base em Programação Inteira Linear

| | |
|--|----|
| Modelos de programação linear | 35 |
| Exemplo de aplicação de programação linear e sua resolução | 35 |

7. Desempenho de Redes com Comutação de Circuitos

| | |
|---|----|
| Cálculo da probabilidade de bloqueio por método exato | 37 |
| Cálculo da probabilidade de bloqueio com teorema do limite do produto | 39 |
| Cálculo da probabilidade de bloqueio na aproximação de carga reduzida..... | 39 |
| Encaminhamento dinâmico da rede telefónica | 40 |

8. Controlo de Fluxos em Redes de Comutação de Pacotes

| | |
|---|----|
| Conceitos introdutórios de controlo de fluxo de pacotes | 44 |
| Controlo de fluxo através de janelas | 45 |
| Controlo das taxas de transmissão..... | 46 |
| Atribuição de taxas de transmissão..... | 47 |

9. Escalonamento em Redes com Comutação de Pacotes

| | |
|---|----|
| Equidade das disciplinas de escalonamento | 48 |
| Escalonamento por ordem de chegada com FIFO | 48 |
| Escalonamento com base em prioridade estrita | 49 |
| Escalonamento com rotação (round-robin, WRR e DRR)..... | 50 |
| Escalonamento por aproximação ao sistema GPS (WFQ e SCFQ) | 51 |
| Métodos de descarte de pacotes | 53 |

10. Engenharia de Tráfego em Redes de Comutação de Pacotes

| | |
|---|----|
| Criação de algoritmos para aplicação em funções objetivo..... | 55 |
| Criação de uma solução de raiz | 55 |
| Criar uma solução melhorada por vias de uma conhecida | 58 |
| Heurística Multi Start Local Search (MSLS) | 60 |

As referências abaixo correspondem às várias citações (quer diretas, indiretas ou de citação) presentes ao longo destes apontamentos. Tais referências encontram-se dispostas segundo a norma IEEE (as páginas Web estão dispostas de forma análoga à de referências para livros segundo a mesma norma).

- [1] R. E. Walpole, R. H. Myers, S. L. Myers, and K. Ye, *Probability & Statistics for Engineers & Scientists*, 9th ed. Boston, MA: Prentice Hall, 2012.
- [2] Sousa, Amaro de, *Revisões sobre Probabilidades, Variáveis Aleatórias e Processos Estocásticos — Diapositivos das Aulas de Desempenho e Dimensionamento de Redes*, Universidade de Aveiro, 2018.
- [3] Sousa, Amaro de, *Processos de Poisson, Cadeias de Markov em Tempo Contínuo e Sistemas de Filas de Espera — Diapositivos das Aulas de Desempenho e Dimensionamento de Redes*, Universidade de Aveiro, 2018.
- [4] Nelson, Randolph, *Probability, Stochastic Processes, and Queueing Theory: The Mathematics of Computer Performance Modeling*, Springer Science, 1 ed., 1995.
- [5] M. N. O. Sadiku and S. M. Musa, *Performance analysis of computer networks*, vol. 9783319016. 2013.
- [6] D. Gross, J. Shortle, F. Thompson, and C. Harris, *Fundamentals of Queueing Theory*. 2008.
- [7] Sousa, Amaro de, *Introdução à Simulação baseada em Eventos Discretos — Diapositivos das Aulas de Desempenho e Dimensionamento de Redes*, Universidade de Aveiro, 2018.
- [8] E. T. Jaynes, *Probability Theory: The Logic of Science.*, Math. Intell., vol. 27, no. 2, 2003.
- [9] Sousa, Amaro de, *Partilha de um Recurso de Comunicações — Diapositivos das Aulas de Desempenho e Dimensionamento de Redes*, Universidade de Aveiro, 2018.
- [10] Lopes, Rui, *Apontamentos de Arquitetura de Redes*, Aveiro, 2017.
- [11] Sousa, Amaro de, *Encaminhamento em Redes com Comutação de Pacotes — Diapositivos das Aulas de Desempenho e Dimensionamento de Redes*, Universidade de Aveiro, 2018.
- [12] Sousa, Amaro de, *Optimization based on Integer Linear Programming — Diapositivos das Aulas de Desempenho e Dimensionamento de Redes*, Universidade de Aveiro, 2018.
- [13] F. Hilier and G. Lieberman, *Introduction to Operational Research*, no. i. 2015.
- [14] Sousa, Amaro de, *Desempenho de Redes com Comutação de Pacotes — Diapositivos das Aulas de Desempenho e Dimensionamento de Redes*, Universidade de Aveiro, 2018.
- [15] K. Medhi, Deepankar, and Ramasamy, *Network Routing : Algorithms, Protocols, and Architectures.*, in Network Routing : Algorithms, Protocols, and Architectures., no. September, 2007.
- [16] Sousa, Amaro de, *Controlo de Fluxos em Redes com Comutação de Pacotes — Diapositivos das Aulas de Desempenho e Dimensionamento de Redes*, Universidade de Aveiro, 2018.
- [17] G. R. Ash and E. Oberer, *Dynamic routing in the AT&T network-improved service quality at lower cost*, in Global Telecommunications Conference, 1989, and Exhibition. Communications Technology for the 1990s and Beyond. GLOBECOM '89., IEEE, 1989, pp. 303–308 vol.1.
- [18] Sousa, Amaro de, *Escalonamento em Redes com Comutação de Pacotes — Diapositivos das Aulas de Desempenho e Dimensionamento de Redes*, Universidade de Aveiro, 2018.
- [19] Sousa, Amaro de, *Traffic Engineering in Packet Switched Networks — Diapositivos das Aulas de Desempenho e Dimensionamento de Redes*, Universidade de Aveiro, 2018.

Apontamentos de Desempenho e Dimensionamento de Redes

1^a edição - junho de 2018



Autor: Rui Lopes

Agradecimentos: professor Amaro de Sousa

Todas as ilustrações gráficas são obra de Rui Lopes e as imagens são provenientes das fontes bibliográficas divulgadas.



apontamentos

© Rui Lopes 2018 Copyright: Pela Creative Commons, não é permitida a cópia e a venda deste documento. Qualquer fraude será punida. Respeite os autores e as suas marcas. Original - This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-nd/4.0/deed.en_US.