Universidade do Minho
Escola de Engenharia

André Filipe dos Santos Ferreira

Controlo de Admissão Adaptativo em Sistemas de Tempo Real

Outubro de 2009

Controlo de Admissão Adaptativo em Sistemas de Tempo Real
André Filipe dos Santos Ferreira

UMinho | 2009

Universidade do Minho
Escola de Engenharia

André Filipe dos Santos Ferreira

Controlo de Admissão Adaptativo em Sistemas de Tempo Real

Mestrado de Engenharia Informática

Trabalho efectuado sob a orientação do
Professora Doutora Solange Rito Lima
e do
Engenheiro Francisco Eugénio Oliveira Ferreira da Costa

Outubro de 2009

Agradecimentos

À Professora Solange Rito Lima, pela atenção, conselhos e orientação que sempre me disponibilizou no decorrer do desenvolvimento desta dissertação.

Ao meu colega e amigo Carlos Lopes, por todo o apoio técnico prestado.

A todos os restantes colegas colaboradores da PT Inovação SA que sempre que necessitei, directa ou indirectamente, me ajudaram na adaptação às novas matérias e tecnologias.

Aos meus melhores amigos André Gentil e Teodoro di Giorgio, pela amizade dedicada.

À Sílvia por acreditar em mim e me dar força.

À minha mãe por todo o afecto e carinho que me dá.

Muito obrigado pela contribuição.

Resumo

Nas plataformas de prestação de serviços a necessidade de controlar a taxa de aceitação de pedidos de requisição de serviço é uma realidade.

A plataforma NGIN (*Next Generation Intelligent Network*), produto da PT INOVAÇÃO S.A., agrega múltiplos serviços de telecomunicações em tempo real. O módulo UIF (*Unified InterFace*) consiste numa interface entre entidades externas e a plataforma NGIN que proporciona a satisfação de pedidos de requisição de serviços. Este módulo carece de controlo do ritmo com que os pedidos podem ser efectuados, sendo estes indiscriminadamente admitidos na plataforma, sem qualquer tipo de diferenciação.

Para colmatar tal necessidade, esta dissertação apresenta um mecanismo que permite o controlo e modulação do ritmo de admissão dos pedidos, de uma forma flexível, diferenciada e configurável. É permitida a atribuição de prioridades distintas entre os vários serviços, limitando indirectamente o nível de carga imposta à plataforma. São assim proporcionadas garantias de qualidade de serviço em cenários de elevada carga, onde o ritmo máximo de admissão é por vezes superado.

É ainda efectuado o estudo de métodos de controlo de admissão e das mais relevantes disciplinas de escalonamento desenvolvidas até à actualidade, sendo analisados diversos conceitos associados às políticas de escalonamento e alocação de recursos. Neste estudo comparam-se várias vertentes que a solução pode adoptar, fundamentando as opções realizadas.

A solução implementada foi sujeita a testes unitários e de carga, cujos resultados revelam a eficácia e robustez da solução, cumprindo todos os requisitos definidos e proporcionando qualidade de serviço imprescindível para enriquecer a fiabilidade do processo de controlo de admissão dos pedidos de requisição de serviço no módulo UIF.

Abstract

In service delivery platforms, the need to control the admission rate of service provisioning requests is a reality.

The NGIN (Next Generation Intelligent Network) platform, a product from PT INOVAÇÃO S.A., deploys several real-time telecommunication services, being the UIF (Unified InterFace) module an interface that performs the admission of service requests from external entities to the NGIN platform.

This module lacks of a rate control scheme by which service requests are served. As consequence, those requests are indiscriminately admitted to the platform, without any kind of differentiation.

To address this need, this dissertation proposes a mechanism that allows to control and modulate the admission rate of service requests, in a flexible, differentiated and configurable way. The solution supports the allocation of distinct priorities among several services, limiting indirectly the load imposed to the platform. Thus, quality of service guarantees can be assured in high load scenarios, where the maximum admission rate is sometimes exceeded.

A study of relevant admission control methods, scheduling algorithms and resource allocation policies is presented and debated.

This debate allows to sustain the design options of the proposed admission control solution.

The implemented solution has been submitted to unitary and load tests. The results showed that the proposal is effective and robust, and provides the quality of service required to enhance the reliability of the admission control of service provisioning requests in UIF module.

Índice

Índice de Figuras

Índice de Tabelas

Índice de Equações

Lista de Acrónimos

CBFQ	Credit Based Fair Queuing
DAO	Data Access Object
DRR	Deficit Round Robin
EJB	Enterprise JavaBeans
FCFS	First-Come First-Served
FQ	Fair Queueing
GPS	Generalized Processor Sharing
HTTP	HyperText Transfer Protocol
IN	Intelligent Network
IP	Internet Protocol
J2EE	Java 2 Enterprise Edition
MCF	Most Credit First
NGIN	Next Generation Intelligent Network
POTS	Plain Old Telephone Service
RR	Round Robin
SCFQ	Self Clock Fair Queueing
UIF	Unified InterFace
URL	Uniform Resource Locator
W2FQ	Worst-case Fair Weighted Fair Queueing
WFQ	Weighted Fair Queueing
WRR	Weighted Round Robin
XML	eXtensible Markup Language

Capítulo 1

Introdução

As redes de telecomunicações evoluíram desde o serviço telefónico POTS (*Plain Old Telephone Service*), tendo sido introduzidos os conceitos de lógica de serviço programáveis, configuração de chamadas, sinalização de controlo, culminando na criação do conceito de rede inteligente (IN). Este conceito proporcionou às operadoras meios para desenvolver e controlar serviços de telecomunicações com maior eficiência, possibilitando ainda o suporte à prestação de serviços de valor acrescentado e serviços pré pagos, em redes de telecomunicações móveis e fixas.

A plataforma NGIN (*Next Generation Intelligent Network*) é uma concretização do conceito IN, tendo sida desenvolvida no seio da PT INOVAÇÃO SA. Esta plataforma agrega múltiplos serviços de telecomunicações em tempo real e permite o suporte a mais de um milhão de clientes, numa única base de dados, traduzindo-se num paradigma de computação distribuída, em tempo real e de alta disponibilidade.

A nível aplicacional a comunicação com a plataforma é efectuada com recurso a várias interfaces. Uma dessas interfaces, designada UIF (*Unified InterFace*), efectua a admissão de pedidos de requisição de serviço URL (*Uniform Resource Locator*) *Encoded* emitidos por entidades externas à plataforma NGIN. Esses pedidos possuem um perfil online, invocando determinados serviços cujo reconhecimento se encontra pré-configurado na interface UIF e que devem ser satisfeitos num curto espaço de tempo.

No seu processo de admissão o módulo UIF aceita os pedidos de requisição de serviço sem limitar o ritmo a que estes podem ser efectuados, sendo estes indiscriminadamente encaminhados para execução.

O processo de execução dos pedidos consome inevitavelmente recursos de memória e processamento, dependendo do tipo de operações necessárias para satisfazer os serviços neles invocados, podendo passar pela execução de procedimentos, funções e métodos.

Também a taxa a que os pedidos ingressam nas interfaces é indeterminado, podendo existir períodos em que são recebidas rajadas de pedidos consecutivos, assim como momentos em que a taxa de ingresso dos pedidos se encontra abaixo do nível limite.

Estes factos podem conduzir a que a carga induzida pelo processo de satisfação dos pedidos seja por vezes demasiado elevada, provocando a congestão do sistema e a retenção de demasiados recursos. Em cenários mais críticos, de períodos de elevada afluência, onde um número excessivo e descontrolado de pedidos necessitam de ser satisfeitos, pode mesmo ser atingido um ponto de ruptura, em que a prestação dos serviços se torna indisponível de todo. Nestas alturas, os recursos disponíveis não são suficientes para satisfazer a quantidade de pedidos a processar.

Outra carência do módulo UIF consiste na admissão indiscriminada dos pedidos de requisição de serviço, o que levanta a necessidade de realizar alguma espécie de diferenciação no processo de admissão. É relevante que, especialmente em cenários de congestão, o modo como a prestação de um tipo de serviços é efectuada, possa adquirir determinada prioridade em relação aos restantes serviços, tornando o processo de admissão prioritário em relação a pedidos que invoquem serviços considerados menos importantes.

Neste âmbito, o conceito de perfis de serviço, onde uma gama de serviços pode ser agrupada numa classe de serviço, revela-se útil para de uma forma simples permitir a atribuição diferenciada de garantias de satisfação dos pedidos, fornecendo consequentemente um determinado nível de qualidade de serviço a cada classe.

1.1. Motivação

A motivação deste trabalho surge da necessidade de colmatar a inexistência de mecanismos de controlo da taxa de admissão dos pedidos de requisição de serviço no módulo UIF, proporcionando ainda um escalonamento diferenciado conforme a classe de serviço a que cada pedido se encontra associado. Tais propósitos deverão ser cumpridos através da integração de um mecanismo configurável de controlo e modulação da taxa de admissão no módulo UIF.

A aplicação desta inovação no módulo, irá permitir a regulação indirecta do nível de utilização global de recursos necessários para dar vazão aos pedidos efectuados, pois

através da limitação da quantidade de pedidos injectados na plataforma também é limitado o nível de carga imposta ao sistema.

Outro motivo reflecte-se no facto do escalonamento diferenciado dos pedidos permitir gerar uma taxa de expedição cuja distribuição é moldada segundo a prioridade de atendimento associada a cada classe de serviço. Este facto é atractivo na óptica de negócio, optimizando a prestação de determinados serviços em relação a outros.

Neste contexto, este trabalho tem a finalidade de proporcionar garantias de qualidade de serviço no processo de admissão dos pedidos de requisição de serviço no módulo UIF, consideradas imprescindíveis para enriquecer a fiabilidade deste módulo.

1.2. Objectivos

O principal objectivo deste trabalho consiste no desenvolvimento de uma solução que permita o controlo e modulação da taxa de admissão dos pedidos de requisição de serviço, provenientes de entidades externas, que são submetidos à interface UIF. A solução deve efectuar o processo de admissão de um modo diferenciado que proporcione garantias de qualidade de serviço em cenários de elevada carga, onde a taxa máxima de admissão é por vezes superada.

Para concretização deste objectivo será inicialmente efectuado o levantamento do estado da arte na área, de forma a permitir a comparação das várias vertentes que a solução pode adoptar, servindo como fundamentação das escolhas realizadas.

Seguidamente deverá ser concebido e implementado um modelo que permita a avaliação detalhada do funcionamento dos mecanismos mais adequados aos requisitos impostos.

Tal modelo deverá cumprir os seguintes requisitos:

- controlo eficiente e configurável da taxa de admissão para obtenção de um perfil de rigidez adaptável. Pretende-se que o nível de granularidade com que o controlo da taxa de admissão é efectuado possa ser ajustado consoante desejado;

- admissão diferenciada dos pedidos de requisição de serviço em função da prioridade definida para cada classe de serviço. Pretende-se proporcionar deste

modo um perfil de admissão dinâmico e justo, a curto e longo prazo, numa perspectiva global da solução;

- implementação modular da solução, de um modo o mais independente possível do ambiente em que possa ser adoptada. Tal permite que a solução desenvolvida possa ser facilmente integrada noutros ambientes;

- opção de diferenciação adaptativa segundo a carga associada a cada classe de serviço, através da atribuição flexível de largura de banda ao longo do tempo.

1.1. Organização da dissertação

Esta dissertação encontra-se dividida em cinco capítulos.

A organização dos capítulos procura estabelecer uma sequência lógica que permita a exposição progressiva e fluida de todas as ideias e conceitos considerados neste trabalho.

A dissertação começa por introduzir os conceitos de prestação de qualidade de serviço no processo de controlo de admissão diferenciado, passando para uma análise abrangente do seu estado da arte. Seguidamente, face à problemática de ausência de controlo de admissão, será apresentada uma solução que visa cumprir os objectivos especificados. Será descrita a sua integração no módulo UIF e focados alguns detalhes da sua implementação.

Finalmente serão analisados resultados de testes que comprovam o correcto funcionamento da solução. Serão realizadas conclusões comparativas entre os vários algoritmos analisados, assim como incluídas referências a trabalho futuro.

Seguindo a sequência apresentada, os vários capítulos encontram-se organizados do seguinte modo:

- **Capítulo 2**

Neste capítulo é apresentado um estudo dos principais métodos utilizados num processo de controlo de admissão, desde métodos de controlo de ritmo aos mais relevantes algoritmos de escalonamento desenvolvidos até à actualidade. É realizada a

comparação das várias vertentes que a solução pode adoptar, sendo fundamentadas as opções tomadas.

Face à relevância do processo de escalonamento no desempenho diferenciado da solução é efectuado um estudo bastante abrangente a este nível, onde são caracterizados os vários tipos de políticas de escalonamento, organizando-os e distinguindo-os consoante o seu perfil de funcionamento.

- **Capítulo 3**

Neste capítulo é apresentada uma solução que permite um processo de controlo de admissão diferenciado e adaptativo. A solução visa colmatar a carência de controlo de admissão na interface UIF, segundo os objectivos especificados.

Começam por ser apresentados os requisitos que a solução deve obedecer assim como o ambiente onde se deverá integrar. Seguidamente é apresentado o decurso que os pedidos de requisição de serviço percorrem no seu processo de admissão, assim como a arquitectura da solução.

É descrito o funcionamento da solução uma vez integrada no módulo UIF. É apresentada uma visão global da arquitectura final do UIF, e são apresentados alguns detalhes de implementação que tiveram lugar no processo de integração.

- **Capítulo 4**

Neste capítulo são apresentados os vários testes efectuados à solução implementada, antes e após da sua integração. Estes testes permitem demonstrar o correcto funcionamento da solução. Os resultados obtidos em cada teste são analisados e validados, sendo ainda efectuada a comparação de funcionamento dos vários algoritmos de escalonamento implementados.

- **Capítulo 5**

Neste capítulo são apresentadas as conclusões relativas às várias fases deste trabalho. São apresentadas as principais contribuições resultantes, assim como os vários aspectos que podem ser abordados para trabalho futuro. São ainda incluídas algumas observações finais relativas aos benefícios deste trabalho.

Capítulo 2

Estado da Arte

Nas plataformas de prestação de serviços o aprovisionamento dos clientes é normalmente efectuado através de pedidos que possuem um perfil online, necessitando de ser atendidos num período de tempo adequado a cada tipo de serviço e com um determinado nível de garantias de qualidade de serviço, mesmo em cenários de congestão. Estes cenários ocorrem quando é submetido ao sistema um número excessivo de pedidos, tal que os recursos disponíveis para satisfazer os serviços invocados são inferiores aos requisitados.

Em cenários de congestão as taxas de prestação de serviço diminuem devido ao aumento de carga do sistema. Também, as entidades que invocam os serviços têm a percepção do estado de congestão quando a sua experiência de serviço se degrada. Deste modo, o sistema prestador deve ser responsável por manter determinados níveis de qualidade de serviço que permitam fornecer bons níveis de utilização aos utilizadores.

Por vezes não é possível prever a quantidade de pedidos efectuados pelas entidades externas. Nestes casos, é necessária a limitação da carga admitida no sistema, sendo fundamental a presença de mecanismos de controlo de admissão, que assegurem aos utilizadores determinado nível de qualidade de serviço.

É vantajoso que a qualidade de serviço prestada possa ser diferenciada segundo os perfis dos vários serviços, consoante os seus níveis de prioridade. Pedidos considerados mais urgentes em relação aos restantes devem ser atendidos com maior prioridade. Revela-se assim útil a agregação, em classes de serviço, dos serviços que requerem níveis idênticos de qualidade de serviço.

O processo de diferenciação no atendimento dos pedidos deve ser executado de um modo justo, tentando contudo optimizar os níveis de utilização de cada classe de serviço. Deve-se evitar que uma classe de serviço consuma mais recursos do que os que lhe são reservados, sendo necessária uma alocação eficiente de largura de banda ao longo do tempo.

A alocação da largura de banda resulta da percentagem de serviço prestada por uma disciplina de escalonamento às várias classes de serviço. A disciplina de escalonamento fornece diferenciação segundo uma partilha hierárquica e justa dos recursos, de modo a garantir um determinado nível de desempenho independentemente do cenário de carga existente para cada classe de serviço.

O presente capítulo apresenta um estudo referente a métodos e abordagens vulgarmente utilizados num processo de controlo de admissão. Muitas dessas metodologias recorrem a filas de espera, gestão e escalonamento das mesmas, sendo vulgarmente aplicadas nos reguladores e escalonadores de tráfego utilizados nas redes IP. São focados alguns dos mais importantes algoritmos de escalonamento, sendo estes analisados e relacionados entre si.

2.1 Controlo de Ritmo

Na óptica do controlo de admissão, um mecanismo de controlo de ritmo deve controlar a distribuição de emissão dos pedidos, moldando-a de forma que o ritmo seja limitado segundo os perfis pretendidos, e assim mantido ao longo do tempo, com a granularidade desejável. Para este fim podem ser utilizados alguns dos métodos e princípios descritos nas subsecções seguintes.

2.1.1 Modulação de Tráfego

Os mecanismos de controlo de ritmo recorrem normalmente à utilização de filas de espera para efectuarem o armazenamento temporário da informação em cenários onde o ritmo limite é ultrapassado, evitando que a informação residual seja instantaneamente descartada. É permitido posteriormente o atendimento dessa informação, no instante em que o ritmo diminui e sem que o ritmo limite seja superado. Deste modo pode ser flexibilizado o processo de admissão dos pedidos de requisição de serviço, permitindo moldar a maneira como estes são encaminhados, optimizando os níveis de utilização da largura de banda disponibilizada.

A modulação do ritmo de serviço de um fluxo de informação pode ser ilustrada pela Figura 1, onde se representa o débito de expedição em função do tempo.

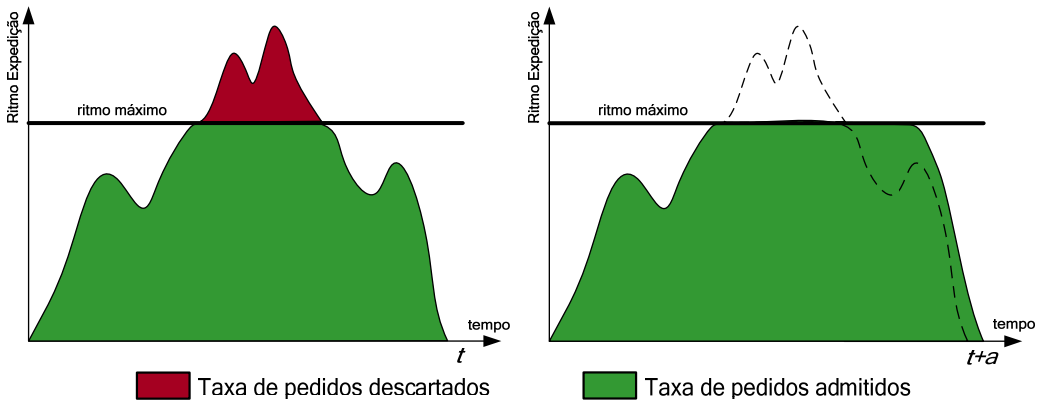

Figura 1. Gráfico figurativo de modulação de ritmo

Na figura esquerda observa-se um processo de limitação de ritmo rígido e intolerante, que não fornece armazenamento provisório, descartando toda a informação que ingressa logo após o ritmo máximo ser atingido. Todavia, na figura direita observa-se que a informação que não é encaminhada quando o ritmo máximo é ultrapassado não é descartada. Este comportamento ocorre graças à inserção da informação residual numa fila de espera, sendo esta encaminhada nos períodos onde o ritmo de expedição relaxa para taxas inferiores ao limite. Porém, ao evitar o descarte da informação é induzido um atraso na sua expedição.

Num cenário onde o ritmo de expedição não é limitado, toda a informação seria encaminha num período de tempo *t*, contudo ao retardar o atendimento dos pedidos residuais, para garantir que o ritmo máximo nunca é ultrapassado, a totalidade dos pedidos passa a ser servida num período de tempo *t+a*, onde *a* representa o atraso total sofrido nas filas de espera.

Idealmente, todos os pedidos de requisição de serviço válidos emitidos a uma plataforma de prestação de serviços devem ser satisfeitos, revelando-se útil a utilização de mecanismos de fila de espera para obtenção da flexibilidade necessária para que o descarte de pedidos seja minimizado.

2.1.2 Gestão de Filas de Espera

Nas plataformas de prestação de serviços os elementos a encaminhar consistem em pedidos que invocam serviços, não devendo ser descartados antes do tempo limite

tolerado para a sua satisfação. Deste modo, os pedidos recém-chegados numa situação de lotação das filas de espera devem ser simplesmente negados.

À medida que o escalonador atende as várias filas de espera, ou que os pedidos excedem o tempo de espera tolerável, a fila deve permitir a inserção de novos pedidos. O nível de congestão diminui se o ritmo médio de ingresso dos pedidos no sistema for inferior à capacidade de atendimento do escalonador. Este método é conhecido como *Tail Drop* [1].

Nestas circunstâncias, o tamanho da fila de espera deve ser cuidadosamente definido, de modo que o tempo de espera para admissão seja superior ao menor tempo de admissão possível para o último elemento da fila, e tendo em consideração o débito máximo de expedição permitido.

Em cenários mais flexíveis, onde os elementos podem ser reenviados, o processo de descarte de pedidos pode passar pelo descarte primário dos elementos de menor prioridade, dentro da mesma fila de espera, recorrendo a técnicas de marcação dos elementos. Deste modo em situações de congestão os pedidos não prioritários seriam descartados, possibilitando a inserção de novos elementos, que seriam descartados à partida sem a aplicação deste método, independentemente da sua prioridade.

O processo de classificação da prioridade dos pedidos pode ser ambíguo, tornando-se algo problemático na medida em que pode prestar uma avaliação injusta do nível de prioridade merecido.

Outro método consiste em descartar os pedidos situados em níveis considerados próximos do ponto de congestão. Um método comum é designado de *Random Early Detection* (RED) [2], que consiste no descarte aleatório dos elementos existentes num determinado intervalo de fila de espera. Este intervalo é determinado entre o limite máximo da fila de espera e uma referência média, definida como próxima do limite máximo. Deste modo, quando a carga de uma fila atinge o indicador médio é sinal que esta se encontra vulnerável a uma situação de congestão.

A probabilidade de descarte é obtida em função do tamanho da fila de espera, o que permite um aumento suave da quantidade de elementos descartados à medida que a lotação da fila é mais evidente [3] [4].

Em alguns cenários é possível a notificação das fontes de informação para redução do seu débito de expedição, para se evitar um cenário de congestão. Contudo torna-se necessária a utilização de protocolos de comunicação com a fonte de informação, o que nem sempre é possível.

2.1.3 Algoritmo *Leaky Bucket*

Um método simples para regular o ritmo de serviço dos pedidos de requisição de serviço pode ser baseado no algoritmo *Leaky Bucket* [5]. Tal método consiste na regulação dos pedidos servidos através de uma fila de espera, designada *bucket*, servida a um ritmo constante.

A designação *Leaky Bucket* provém da analogia a um contentor com um furo no seu fundo e contendo um fluído no seu interior. O contentor vai pingando gotas do fluido a determinado ritmo, independentemente da quantidade existente no mesmo. As gotas abandonam o contentor a um ritmo constante.

O *bucket* recebe pedidos a um ritmo indeterminado, sendo posteriormente servidos de um modo normalizado para obtenção de um ritmo regular efectivo, como se encontra representado na Figura 2.

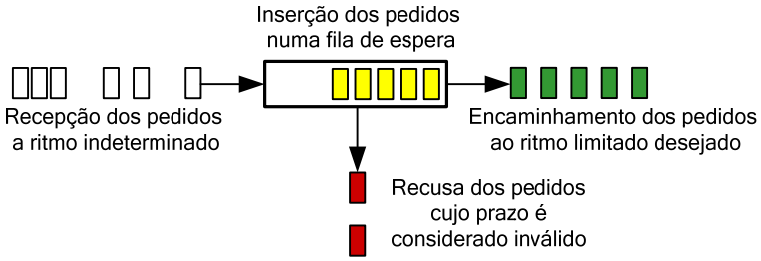

Figura 2. Esquema do método *Leaky Bucket*

O processamento da fila é efectuado periodicamente, por ordem de chegada (FIFO), sendo o período de atendimento equivalente ao ritmo desejado. No caso da fila de espera se encontrar lotada, os pedidos que chegam seriam rejeitados.

Apesar de este algoritmo limitar eficazmente o ritmo a que os pedidos são servidos, é ineficiente em casos onde o ritmo limite raramente é atingido, porque possui uma complexidade induzida desnecessariamente no processo de inserção e remoção de cada pedido na fila de espera.

Se não for recebida informação durante um certo período de tempo, a largura de banda que não é utilizada nesse período não pode ser reutilizada para informação inserida futuramente. [5]

2.1.4 Algoritmo *Token Bucket*

O método *Token Bucket* [6] possui a filosofia do método *Leaky Bucket*, mas permite a regulação do ritmo de um modo menos rígido, possibilitando a admissão de uma rajada de pedidos, i.e. de uma determinada sequência de pedidos num curto espaço de tempo. O algoritmo é representado na Figura 3.

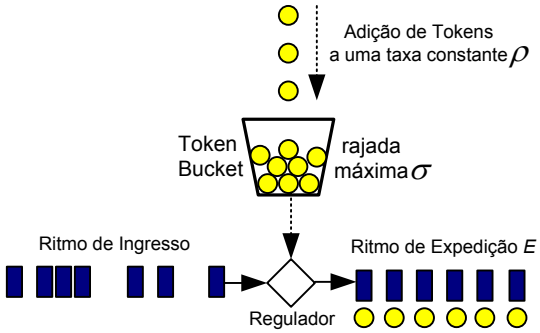

Figura 3. Esquema do método *Token Bucket*

O algoritmo consiste na admissão de pedidos através do consumo de unidades de crédito, designadas *tokens*. Em norma, cada pedido consome um *token* na sua admissão. A taxa de admissão é mantida à custa da adição periódica de *tokens* no *bucket*, visto que se o *bucket* se encontrar vazio os pedidos têm de aguardar a adição de novos *tokens* no *bucket* para que possam ser admitidos. A vantagem consiste na possibilidade de retenção de *tokens* no *bucket*, o que permite a admissão de rajadas pedidos. A dimensão máxima das rajadas depende do tamanho do *bucket*, i.e. da quantidade máxima de *tokens* que podem estar contidos no *bucket*.

No caso de um pedido ingressar no regulador e não dispor de *tokens* no *bucket*, várias acções podem ser tomadas, dependendo do modo como o algoritmo se encontrar configurado. O pedido pode ser descartado, marcado, inserido em filas de espera e tratado de modo diferenciado, etc.

Seja σ o limite do *bucket*, e ρ a taxa com que os *tokens* são adicionados ao *bucket*, que se reflecte na taxa média de admissão, temos que a taxa de expedição *E* pode ser expressa pela seguinte equação:

$$E(t_1,t_2) \leq \sigma + \rho(t_2 - t_1) \quad , \quad \forall\, t_2 \geq t_1 \geq 0$$

Equação 1. Equação da taxa de expedição *Token Bucket* [6]

A expressão demonstra que a taxa de expedição é limitada, nunca ultrapassando a taxa equivalente à soma da rajada de pedidos σ com a taxa ρ com que os *tokens* são adicionados ao *bucket*, para qualquer período de tempo $[t_1, t_2]$.

Este algoritmo pode proporcionar vários perfis de ritmos de expedição, podendo mesmo assumir o comportamento do algoritmo *Leaky Bucket,* expedindo os pedidos a um ritmo perfeito, bastando que o tamanho máximo do *bucket* seja de um *token*. Deste modo nunca seria admitido mais do que um pedido consecutivamente. No *Leaky Buket*, se uma sequência de ritmo muito elevado ingressa na fila de espera, acaba por aguarda na fila, sendo transmitida ao ritmo definindo, enquanto que no algoritmo *Token* tal sequência poderia ser encaminhada directamente, dependendo da dimensão da rajada de pedidos permitida [5].

2.2 Distribuição da Capacidade de Linha

O escalonamento de pedidos resulta normalmente em contenção. Uma disciplina de escalonamento lida com a contenção decidindo a ordem de atendimento prestada às várias filas de espera activas, segundo uma atribuição justa dos recursos e sem deixar de garantir um determinado nível de desempenho.

No âmbito das redes IP, existem várias políticas que um algoritmo de escalonamento pode seguir consoante a finalidade pretendida. Uma propriedade ambicionada consiste em garantir que cada fila de espera não permanece sem ser atendida durante um período indefinido de tempo, e deste modo garantir justiça no processo de admissão. Deste modo, fontes de informação mal comportadas, cujo fluxo de informação é baseado em rajadas consecutivas de pacotes que originam débitos elevados em relação à capacidade disponível, não impedem o serviço das fontes bem comportadas, cujo fluxo de informação é relaxado em relação ao esperado.

Este conceito é conhecido por *fairness*, traduzido nesta dissertação como equidade.

Num modelo com um nível de equidade perfeito o algoritmo de escalonamento deve servir as filas de espera cumprindo a relação entre as prioridades de cada fila e assegurando a largura de banda a que cada fila tem direito.

Contudo é comum que quando uma ou mais filas de espera se encontrem inactivas, a largura de banda que não é consumida por estas, possa ser consumida pelas filas

restantes, de modo que a largura de banda reservada para todas as filas activas seja igual à totalidade de largura de banda existente.

Numa perspectiva idealista, dois fluxos f_1 e f_2 são atendidos com equidade perfeita se a diferença entre a relação do peso da fila *W* e o seu ritmo *r* for nula, independentemente do período de tempo considerado [7], isto é, se a taxa de admissão de cada fila de espera for proporcional ao seu peso, distribuindo a taxa de atendimento de uma forma equitativa entres as filas activas.

Assim, para dois fluxos f_1 e f_2 são atendidos com equidade perfeita se e só se:

$$\frac{Wf_1}{rf_1} - \frac{Wf_2}{rf_2} = 0$$

Equação 2. Equidade entre dois fluxos [7]

Esta condição é idealista, pois só é possível se as unidades de atendimento forem infinitesimais, logo para se atingir o máximo de equidade deve-se ter presente o princípio de aproximar a condição o máximo possível de zero [7]. Assim, a condição apenas permite comparar o nível de equidade entre os vários algoritmos.

Uma métrica utilizada para medir o desempenho dos algoritmos de escalonamento consiste na diferença entre o nível de largura de banda que deve ser atribuído a uma fila de espera e a efectivamente utilizada pela mesma.

2.2.1 Política de Equidade *Max-Min*

Segundo o princípio *Max-Min*, uma alocação de largura de banda justa deve distribuir os recursos existentes por todas as ligações. Se existir algum nível de recursos excedentes das ligações satisfeitas, estes são repartidos pelas ligações insatisfeitas, cuja atribuição de largura de banda inicial não se revelou suficiente. O processo repete-se enquanto existir capacidade de largura de banda excedente e ligações insatisfeitas.

Assim, a largura de banda a consumir pelas fontes de informação é atribuída aos seus fluxos por ordem crescente, i.e., os fluxos de menor ritmo possuem maior prioridade que os fluxos de ritmo superior, garantindo que fontes com maior emissão de rajadas de informação, que excedem a largura de banda que lhe é atribuída, não prejudiquem outras

fontes de informação com prioridades inferiores. Deste modo é garantida equidade entre todas as fontes de informação durante o processo de escalonamento. [8]

Esta política demonstra-se apenas em cenários onde pelo menos uma das fontes consome toda a largura de banda que lhe é destinada.

Segundo o critério de equidade *Max-Min*, seja:

r_i - Taxa de débito de um fluxo i

L_i - Peso do fluxo i

C - Débito máximo

Numa alocação estática de largura de banda, o ritmo de serviço de cada fluxo activo deve ser proporcional ao seu peso, como apresenta a expressão:

$$r_i = L_i \frac{C}{\sum L_i}$$

Equação 3. Ritmo de serviço de um fluxo activo [8]

Para uma alocação dinâmica de largura de banda, o débito que cada sessão obtém é calculado segundo as expressões seguintes:

Seja,

x_n - Recursos requisitados pelo fluxo n, tal que $x_1 \le x_2 \le \ldots \le x_N$

M_n - Recursos disponíveis para o fluxo n

m_n - Recursos alocados ao fluxo n

Então,

$$m_n = \min(x_n, M_n) \qquad 1 \le n \le N$$

$$M_n = \frac{C - \sum_{i=1}^{n-1} m_i}{N - n + 1}$$

Equação 4. Recursos disponíveis para um fluxo[8]

Ou seja, os recursos efectivamente alocados a um fluxo (m_n) serão os requisitados pelo mesmo (x_n) se existirem recursos excedentes suficientes, ou serão apenas os recursos disponíveis (M_n), no caso de ser impossível satisfazer os requisitos de largura de banda respectivos.

A Figura 4 apresenta um exemplo de alocação dinâmica de recursos, efectuada a três fluxos de informação, segundo o princípio de equidade *Max-Min*.

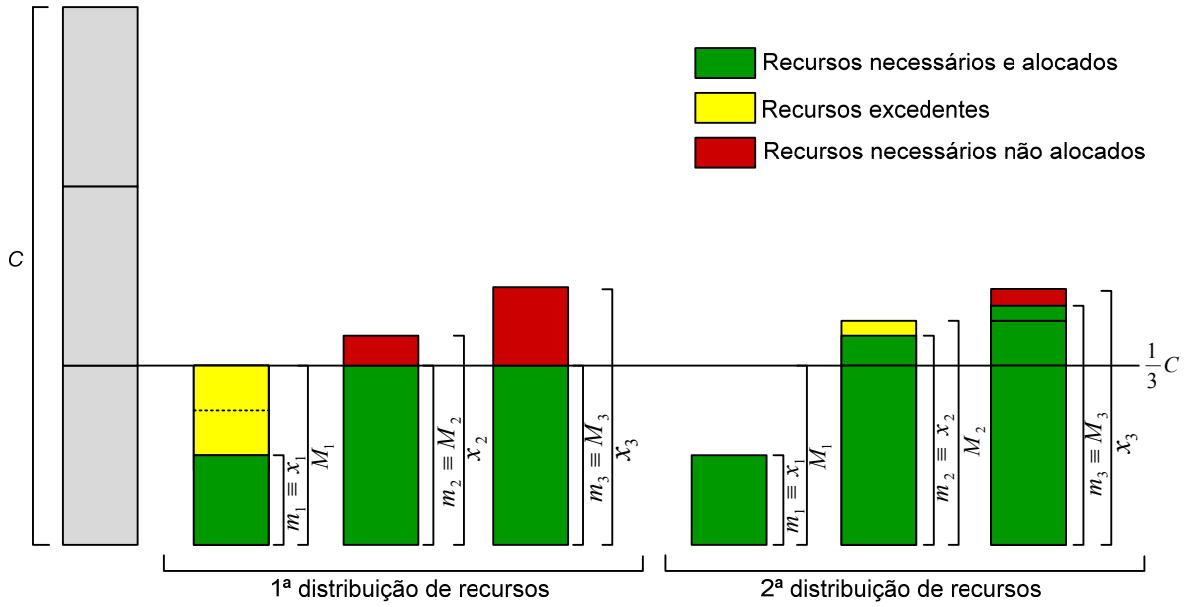

Figura 4. Esquema de alocação dinâmica de recursos (baseado em [3])

A política *Max-Min* subsiste quando os recursos são consumidos na sua totalidade, de modo que o aumento da largura de banda alocada a um fluxo provoca obrigatoriamente o decréscimo da largura de banda atribuída aos restantes.

Um tipo de algoritmo que permite obtenção de uma alocação *Max-Min* é designado *progressive filling*. Este tipo de algoritmos é bastante utilizado para maximizar as taxas de serviço nas redes sem fios mantendo garantias de alocação justa de recursos entre os vários utilizadores. Nestes algoritmos, o serviço das fontes de informação é progressivo, sendo atendidos a taxas semelhantes, até que um deles atinja a capacidade máxima reservada de largura de banda. As taxas de serviço das fontes que atingem o limite não evoluem mais, sendo as restantes incrementadas. Todas as fontes que estabilizam representam ligações *bootleneck*, estando o seu espaço saturado. O algoritmo prossegue o serviço até que seja impossível o incremento da taxa de serviço de qualquer uma das fontes restantes. O algoritmo termina quando todas as fontes tiverem saturado o seu limite de largura de banda. [9]

Assim, a política *Max-Min* é desejável em cenários onde as fontes de informação emitem tráfego elástico cuja taxa de tráfego possa ser ajustada de acordo com as condições da rede. [10]

A filosofia *Max-Min* é aplicada na íntegra nas disciplinas nos algoritmos *Fair Queueing*, *Wheighted Fair Queueing* e *Round-Robin* para unidades de informação de tamanho fixo. Nestes algoritmos é evidente que é garantido o atendimento das fontes de informação que possuem menor fluxo de informação durante a sua actividade.

2.2.2 Política de Maximização da Taxa de Serviço

Uma outra política de escalonamento consiste na optimização da taxa de serviço a que o escalonador encaminha a informação de determinadas fontes. Nesta política é assumido que a taxa média de ingresso dos pacotes nas filas de espera adquire valores inferiores à capacidade total de largura de banda.

O escalonador conhece o nível de conteúdo das filas de espera assim como as condições do canal, mas não necessita de conhecer a capacidade total do canal nem as taxas de ingresso de cada fluxo. [10]

É demonstrado em [11] que a alocação de recursos com vista a maximizar as taxas de serviço das filas de espera pode consistir numa política estável.

Esta política contrasta com a política de equidade na medida em que para obter tais optimizações os princípios de equidade são sacrificados. Em casos onde as fontes possuem prioridades distintas, os recursos podem ser alocados a fontes de prioridade superior, podendo este comportamento induzir negação de recursos em filas de espera com prioridade inferior.

Uma política de maximização da taxa de serviço das fontes é tentadora para maximizar a qualidade de atendimento de uma fila, mas o custo que esta política acresce ao prejudicar as restantes fontes é inaceitável na maioria dos cenários de utilização, degradando a qualidade de serviço numa perspectiva geral.

2.2.3 Política de Equidade Proporcional

O modelo de alocação de recursos designado *proportional fairness* , reúne os princípios da política de equidade *Max-Min* e da política de maximização da taxa de serviço. Nesta dissertação traduzimos *proportional fairness* como equidade proporcional.

Apesar de a política *Max-Min* ser a mais rígida em relação aos níveis de equidade, e a política de maximização de ritmo a que mais carece de equidade, a política de equidade

proporcional efectua a optimização das taxas de serviço, sem deixar de fornecer garantias de equidade [12].

Neste modelo os recursos são divididos com o objectivo de atribuir o mesmo custo a todas as fontes de informação, ou minimizar o custo máximo atribuído às mesmas. Fontes mal comportadas irão possuir menor qualidade de atendimento, mas nunca irão sofrer negação de recursos. [13]

A política de equidade proporcional encontra-se especificada em [14].

2.3 Disciplinas de Escalonamento

Numa plataforma de prestação de serviços onde é necessário providenciar garantias de qualidade de serviço diferenciada por classe de serviço, e com uma partilha de recursos com critérios de equidade, as disciplinas de escalonamento são um componente essencial. Nas secções seguintes são analisadas várias características que permitem distinguir a aplicabilidade das várias políticas de escalonamento.

2.3.1 Disciplina de Escalonamento Ideal

Uma disciplina de escalonamento efectua o serviço das filas de espera segundo uma ordem, que pode ser baseada na prioridade associada a cada classe de serviço, na carga existente nas filas de espera, ou em outros indicadores. Assim, é responsável pela manutenção de um determinado grau de qualidade de serviço.

Para fornecer tais garantias existem características desejáveis, que a disciplina deve possuir. A aplicabilidade de algumas destas características pode comprometer as restantes, pois encontram-se relacionadas, podendo os ideais de uma característica ir contra os princípios de uma outra [15].

As principais características desejadas numa disciplina de escalonamento que permita um serviço justo e eficaz encontram-se descritas nos tópicos seguintes.

- **Simplicidade e Eficiência**

Um algoritmo de escalonamento deve ser simples de implementar, devendo efectuar o

serviço de um modo simples e eficaz.

O processo iterativo de decisão de qual a fila de espera a atender não deve possuir demasiadas instruções ou operações que aumentem a complexidade ao algoritmo. Devem ser evitadas ordenações de elementos em filas de espera, ou métodos de procura sobre os mesmos.

O nível de complexidade induzida no escalonamento é importante na medida que um aumento linear (O(N)) do tempo de processamento com o aumento de ligações activas não é desejável. Assim, a complexidade desejável deve ser inferior à linear, em relação ao número N de ligações activas.

A complexidade ideal será constante, i.e. O(1) [16] [17]. Isto é, o tempo necessário para escolha do cliente que deve ser atendido em cada iteração não tende a crescer à medida que o número de clientes aumenta.

- **Escalabilidade**

O algoritmo deve ser escalável, devendo para isso possuir a menor complexidade possível. O ideal é que a complexidade se mantenha à medida que as filas de espera aumentam, de modo que o escalonamento seja eficiente em baixa e larga escala. Esta preocupação pode ser menos evidente em casos que o número de classes de serviço, e respectivas filas de espera, sejam garantidamente limitados.

- **Diferenciação**

O algoritmo de escalonamento deve efectuar o atendimento diferenciado dos pedidos, fornecendo o comportamento desejado consoante as configurações efectuadas.

O método usual de diferenciação consiste na utilização de filas de espera, com distintas prioridades de atendimento. Esta prioridade deve ser ponderada, para que não seja introduzida demasiada latência no atendimento dos pedidos de menor prioridade, aquando a ocorrência da recepção de rajadas de pedidos.

O atraso que os pedidos existentes em fila de espera sofrem até ao momento do seu atendimento, deve ser previsível. Se o escalonamento for efectuado num determinado período de tempo, e se existir um mecanismo de controlo de ritmo, como o *Leaky Bucket*, então pode ser determinado o tempo de atendimento de cada pedido, segundo a sua posição na fila de espera.

Deste modo, um bom escalonador deve possuir um atraso médio e máximo limitado, para que as filas de espera com menor prioridade não sofram negação de recursos.

- **Equidade**

No contexto deste trabalho é necessário fornecer garantias de equidade no processo de serviço, independentemente do cenário de carga.

As filas de espera representam as fontes de informação donde a informação deve ser escalonada. Para que estas não permaneçam sem atendimento, deve ser garantida uma taxa de atendimento mínima a cada uma, ou seja, deverá ser alocada largura de banda suficiente para garantir que nenhuma fila permanece sem ser atendida indefinidamente (*starvation*).

Como o escalonamento é efectuado iterativamente, sendo atendido um pedido em cada iteração, devem ser concedidas um número de iterações a cada fila, na ordem definida pelo escalonador, tal que a relação dos pedidos atendidos na fila respectiva, com a totalidade de pedidos encaminhados em todas as filas, seja superior a um valor mínimo determinado, de modo que a percentagem de atendimento estabelecida para cada fila seja cumprida.

Para evitar a utilização abusiva de recursos e se preservarem as garantias de atendimento é fundamental garantir que o atendimento de uma determinada classe de serviço não degrada substancialmente o atendimento das restantes. Para este efeito, a disciplina de escalonamento deve proteger as filas de espera bem comportadas, das mal comportadas, devendo ignorar ou penalizar as filas cujo tráfego levaria ao incumprimento das regras de equidade no processo de atendimento.

As garantias dos limites de desempenho devem estar presentes em contratos de serviço, seja por parte o cliente, ou por parte do fornecedor.

- **Adaptação**

Apesar de em alguns cenários ser impossível garantir o serviço de todos os pedidos, este deve ser optimizado de modo a minimizar o número de pedidos não processados. Em cenários de sobrecarga, a degradação da prestação de qualidade e cumprimento dos requisitos deve ser o menos abrupta possível. Deste modo, o algoritmo deve reagir de forma justa face à ocorrência de situações que tendem a afectar a qualidade do serviço.

Em algoritmos adaptativos, o escalonamento efectuado às filas de espera adapta-se à medida as condições se alteram, seja em função do nível de carga das filas, da quantidade de atendimentos de cada fila sofreu ao longo do tempo, ou de outros factores.

Se a condicionante for o nível de carga das filas é permitido que parte da largura de banda reservada a uma fila de espera pode ser consumida por filas de espera com maior carga, sem que a fila respectiva esteja necessariamente desactivada. Se a condicionante for o valor indicador da taxa de utilização que uma fila possui em relação às restantes, esta é atendida de forma a cumprir os níveis de utilização respectivos, não permanecendo demasiadas iterações a ser atendida consecutivamente. Este indicador actua como uma memória, demonstrando quais as filas de espera mais atendidas nas iterações anteriores. Deste modo, Esta medida diminui a rigidez com que a taxa de atendimento deve ser cumprida, mas torna o processo de admissão mais flexível, podendo reduzir o tempo de espera a que os pedidos estão sujeitos.

Seja qual for a condicionante que torna a disciplina adaptativa, a adaptação pode demorar mais ou menos iterações até que as garantias prestadas se verifiquem. É desejável a adaptação no menor número de iterações possíveis para que o serviço seja submetido a pouca degradação.

O peso que estes indicadores devem ter no processo de decisão de qual a fonte de informação a ser atendida em cada iteração, deve ser considerado de modo que as garantias de qualidade fornecidas não sejam degradadas. Tal deve ser tido em consideração, para que as filas de espera com indicadores de peso inferior não sejam excluídas do processo de escalonamento, levando-as à situação de negação de recursos.

- **Qualidade de Serviço**

Garantir níveis de performance, consoante o limite estabelecido na classe desse serviço. A limitação máxima ou média de factores como a de largura de banda, atraso, variação do atraso, perdas, entre outros, permitem garantir desempenho, segundo o perfil de qualidade de serviço desejado para cada serviço. Estes limites podem ser de natureza estática ou dinâmica, sendo em plataformas de prestação de serviços normalmente definidos estaticamente.

2.3.2 Conservação do Trabalho

O comportamento de uma disciplina de escalonamento pode seguir duas abordagens relativas à actividade de atendimento das filas de espera ao longo do tempo. A disciplina pode manter o processo de escalonamento contínuo enquanto existirem pacotes a ser atendidos, designando-se conservativa (*work-conserving*), parando apenas quando não existirem pacotes a transmitir em qualquer uma das filas de espera. Deste modo quando uma fila de espera não contém pacotes, o atendimento desta fila não é considerado, sendo o escalonamento efectuado somente entre as filas de espera activas.

Contudo, uma disciplina de escalonamento pode ser não conservativa (*non-work-conserving*), podendo o escalonamento ficar inactivo no atendimento de determinadas filas de espera, que por decisão do algoritmo de escalonamento possam não ser consideradas no processo de eleição para atendimento. As filas que são atendidas nestas circunstâncias dizem-se elegíveis (*eligible*).

A maioria das disciplinas de escalonamento é conservativa para evitar desperdícios de largura de banda resultantes de um escalonamento não conservativo, melhorando assim os níveis de utilização dos recursos ao longo do tempo. [4]

2.3.3 Medidas de Desempenho

De seguida são descritas e interpretadas algumas expressões cuja validade é demonstrada em [6].

Para melhor descrever o nível de equidade do GPS suponha-se a existência de dois fluxos, i e j, constantemente activos nos períodos considerados, com os custos $\phi(i)$ e $\phi(j)$. Seja S(i, t1, t2) o nível de serviço efectivo de atendimento na fila de espera i, para qualquer que seja o intervalo de tempo $[t_1, t_2]$, temos que:

$$\frac{S(i,t1,t2)}{S(j,t1,t2)} \geq \frac{\phi_i}{\phi_j}$$

Equação 5. Relação entre o serviço de dois fluxos e os seus custos [6]

Isto significa que no intervalo de tempo $[t1, t2]$ a relação entre o nível de serviço dos fluxos *i* e *j* tem de ser igual ao superior à relação dos seus custos, ou seja, a relação de

atendimento dos vários fluxos é permanentemente justa, independentemente do período de tempo considerado, sendo sempre garantida a percentagem de largura de banda mínima a cada fonte de informação.

Deste modo, seja $S(t1,t2)$ a totalidade de serviço fornecida pelo escalonamento GPS, tem-se que:

$$S(i,t1,t2) \geq \frac{\phi_i}{\sum_{j=1}^{N} \phi_j} S(t1,t2)$$

Equação 6. Relação entre o serviço de um fluxo e o ritmo de serviço total [6]

Ou seja, o ritmo de atendimento de um fluxo *i* é sempre igual ou superior ao produto do ritmo total servido pelo escalonador, com a relação entre o custo do fluxo *i* e a soma dos restantes fluxos. Esta condição garante que, independentemente do período de tempo considerado, um fluxo *i* terá cumprida a sua proporção de atendimento, consoante o custo associado aos restantes fluxos activos.

Ora, se o serviço de escalonamento proporcionar uma taxa de atendimento constante *r*, a totalidade do serviço proporcionado pelo escalonador cumpre a condição:

$$S(t1,t2) = r(t2 - t1)$$

Equação 7. Serviço total para um determinado intervalo de tempo

Isto porque a multiplicação do ritmo de atendimento pelo intervalo de tempo considerado reflecte a quantidade de informação atendida nesse período.

Assim, vem que a taxa mínima garantida para o fluxo *i* é de:

$$g_i \geq \frac{\phi_i}{\sum_{j=1}^{N} \phi_j} r$$

Equação 8. Taxa de serviço mínima garantida

Ou seja, é garantida uma taxa de atendimento de modo proporcional entre os vários fluxos, independentemente do período de tempo considerado, visto que o atendimento dos vários fluxos é efectuado em paralelo.

Nenhuma disciplina de escalonamento pode ser tão justa como no modelo GPS. No modelo GPS, o serviço é relativo a fluxos, tendo natureza infinitesimal, cujo atendimento é feito em simultâneo.

Em regime de contenção, como as disciplinas de escalonamento implementáveis têm de atender as várias filas sequencialmente, enquanto uma fila é atendida, outras aguardam estando nesse instante a ser criada injustiça nas restantes sessões.

Assim, como o GPS é o modelo ideal, o grau de justiça de atendimento dos restantes escalonadores pode ser medido relativamente ao GPS. Os vários algoritmos que pretendem atingir um nível de equidade mais próximo possível do nível de equidade perfeito do modelo GPS podem ser avaliados segundo dois indicadores, *o Relative Fairness Bound* (RFB) e o *Absolute Fairness Bound* (AFB) [18].

2.3.3.1. *Relative Fairness Bound* (RFB)

Este índice de desempenho de equidade permite comparar algoritmos de escalonamento com o ideal GPS.

Seja L_i o peso associado ao fluxo i e C a capacidade de linha, então o débito r_i de um fluxo i, é descrito pela equação:

$$r_i = L_i \frac{C}{\sum L_i}$$

Equação 9. Débito de serviço [6]

Para duas sessões com direito aos débitos r_i e r_j, que recebem na prática um serviço de $S(i,t_1,t_2)$ e $S(j,t_1,t_2)$, durante o intervalo de tempo $[t_1,t_2]$, a expressão apresenta o índice *Relative Fairness Bound*.

$$RFB = \left| S(i,t1,t2)/r_i - S(j,t1,t2)/r_j \right| \ , \ \forall i,j,t1,t2$$

Equação 10. *Relative Fairness Bound* entre duas sessões [18]

2.3.3.2. *Absolute Fairness Bound* (AFB)

Para uma sessão com direito aos débitos r_i, mas que recebem na prática um serviço de S(i, t1, t2), e que segundo o algoritmo GPS receberia o serviço G(i, t1, t2), durante o intervalo de tempo $[t_1,t_2]$, temos que:

$$AFB = \left| S(i,t1,t2)/r_i - G(j,t1,t2)/r_j \right| \quad , \forall i,j,t1,t2$$

Equação 11. Absolute Fairness Bound [6]

Este índice demonstra a diferença entre o débito atribuído a uma sessão num algoritmo simulado ao GPS, e o débito dessa mesma sessão pelo algoritmo ideal GPS.

2.3.4 Principais Disciplinas de Escalonamento

Nesta secção são analisados os principais algoritmos de escalonamento, assumindo uma visão crítica sobre os seus níveis de equidade de distribuição de largura de banda. O objectivo destes algoritmos é de atingir a política de escalonamento idealista GPS.

A maioria dos algoritmos apresentados é utilizada em dispositivos de redes orientados a pacotes IP, contudo as conclusões finais do estudo serão aplicadas a nível aplicacional.

As disciplinas estudadas são:

- *First-Come First-Served* (FCFS)
- *Generalized Processor Sharing* (GPS)
- *Round Robin* (RR)
- *Weighted Round Robin* (WRR)
- *Deficit Round Robin* (DRR)
- *Fair Queueing* (FQ)
- *Weighted Fair Queueing* (WFQ)
- *Worst-case Fair Weighted Fair Queueing* (W2FQ)
- *Self-Clock Fair Queueing* (SCFQ)
- *Most Credit First* (MCF)
- *Credit Based Fair Queueing* (CBFQ)

No cenário específico de um mecanismo de admissão aplicacional pretende-se apenas a análise de cada algoritmo de escalonamento como um módulo individual.

2.3.4.1 *First-Come First-Served* (FCFS)

A disciplina de escalonamento mais elementar designa-se FCFS e consiste no atendimento dos pacotes segundo a sua ordem de ingresso na fila de espera. Para isso, é utilizada uma fila de espera FIFO, sendo a inserção de cada pacote é efectuada na última posição da fila, e o atendimento efectuado na primeira posição da fila, de modo que o primeiro pacote a ser inserido é o primeiro pacote a ser removido. Assim, não são efectuadas operações de ordenação ou alteração da posição dos pacotes.

Esta disciplina não suporta diferenciação, ou qualquer distinção de prioridades. Consequentemente não fornece garantias de equidade de largura de banda alocada a cada fluxo.

Um algoritmo de escalonamento deve funcionar em pleno mesmo na presença de fontes mal comportadas, algo que o FCFS não garante, visto que uma fonte mal comportada tende a consumir largura de banda destinada às fontes restantes.

A ordem de chegada dos pacotes IP à fila de espera o único factor que determina a largura de banda, reflectida pelo espaço que cada fonte ocupa na fila. Quando a fila de espera fica lotada são descartados todos os pacotes que ingressarem nestas condições. Com todos os fluxos agregados num só, o atraso de admissão é crescente à medida que o conteúdo da fila de espera aumenta. Em casos onde pacotes de pequena dimensão antecedem de pacotes de dimensão superior, o atraso pode ficar demasiado alto.

Existe a tendência de se tentar combater situações de congestão com o aumento da dimensão das filas de espera, contudo estas devem ser utilizadas apenas como armazenamento a curto prazo, tendo em conta que se forem demasiado grandes proporcionam atrasos elevados [5].

2.3.4.2 *Generalized Processor Sharing* (GPS)

Proposto por *A.Parekh* na sua tese em 1992 [6], o modelo GPS (*Generalized Processor Sharing*) é uma disciplina de serviço idealista que serve os fluxos de informação de natureza infinitesimal segundo o critério *Max-Min*, sendo o escalonamento efectuado de modo contínuo e paralelo, com equidade perfeita.

Neste modelo existe uma fila de espera para cada sessão que efectua a partilha de uma mesma ligação. Nas *N* filas de espera activas, os fluxos são atendidos simultaneamente,

sendo garantida a cada fila uma razão de pelo menos *1/N* da capacidade de largura de banda total.

O GPS trouxe a flexibilidade de diferentes sessões poderem possuir diferentes prioridades, servindo as filas activas proporcionalmente à sua percentagem de largura de banda, mantendo os ideais de equidade mesmo em cenários de congestão.

Como o GPS actua sobre fluxos contínuos de informação, infinitamente divisíveis, é impossível de implementar com a finalidade de escalonamento discreto estatístico. Num escalonamento onde o fluxo de informação não é contínuo, mas sim segmentado, apenas pode ser atendida uma fila de espera. Torna-se assim impossível de cumprir a equidade perfeita ao longo do tempo, pois o seu nível varia, flutuando ao longo das iterações.

Como foi referido na secção 2.3., um algoritmo de escalonamento mantém a equidade ao longo do tempo se a diferença entre a relação do peso da fila com o seu ritmo for nula, distribuindo a taxa de atendimento de uma forma equitativa entre as várias fontes de informação, independentemente do seu comportamento.

Nas disciplinas de escalonamento de natureza discreta, cada fonte dispõe de um período de tempo de atendimento, seja estático ou dinâmico, tendo de possuir algum indicador que traduza a relação entre a quantidade de informação atendida em cada fila de espera e o peso da mesma.

O GPS é utilizado como modelo comparativo utilizado para medir a eficiência de outros algoritmos de escalonamento cujo funcionamento é baseado no num serviço segmentado de informação de perfil discreto. Nesses modelos, a relação de equidade ao longo do tempo nunca é mantida a 100%, podendo apenas aproximar-se do modelo GPS.

2.3.4.3 *Round Robin* (RR)

A disciplina de escalonamento *Round-Robin* (RR) é uma solução que procura tratar N fluxos de forma equitativa, garantindo uma taxa de serviço de 1/N para cada fluxo.

Quando associado a pacotes de dimensão fixa, o método RR é considerado o método mais simples para se atingir equidade no processo de escalonamento [19]. Contudo, apesar de garantir largura de banda a todas as filas de espera activas, esta disciplina não considera a quantidade de informação servida por fila de espera, nem o seu estado [20].

O funcionamento tradicional deste algoritmo consiste no serviço de um pacote por cada fluxo activo. O escalonador atende o fluxo durante um período de tempo. Deste modo

existem garantias de atendimento para cada fluxo activo, sendo impossível a ocorrência de negação de recursos. É também previsível o atraso de atendimento que cada pacote sofre, visto que a fatia temporal de atendimento dos fluxos é fixa.

Se um pacote P de um fluxo *i*, é inserido na fila de espera *i*, no instante *t*, quando existem $P_i(t)$ pacotes na fila, então, assumindo que os pacotes possuem tamanho fixo, o pacote p irá aguardar $P_i(t)$ ciclos, até ser atendido, onde um ciclo pode demorar o tempo de transmissão de 0 a *N* pacotes, dependendo do número de filas de espera activas. Deste modo, se o número de filas de espera *N*, e o número de pacotes existentes em cada fila de espera *i*, $P_i(t)$, forem limitados, limita-se o atraso que um pacote pode sofrer [6].

Em situações que o tamanho do pacote é variável, as filas de espera que possuam uma maior relação de pacotes pequenos do que restantes filas serão penalizadas, sendo o escalonamento efectuado de modo injusto, não equitativo. Uma tentativa para reduzir os efeitos deste problema consiste no atendimento de mais do que um pacote em cada ciclo de atendimento.

Existem soluções semelhantes ao algoritmo RR, como a disciplina *Processor Sharing* (PS), que em vez de efectuarem um atendimento ordenado das filas de espera, efectuam um atendimento aleatório dos fluxos em cada ciclo de atendimento. As principais vantagens desta política de escalonamento consistem no facto de ser simples. Quase todos os algoritmos baseados na política RR possuem baixa complexidade temporal, especialmente vantajosa em cenários onde o poder de computação é limitado.

Contudo, a simplicidade desta política levanta problemas de flexibilidade, pois não pode ser aplicada diferenciação de escalonamento das várias fontes de um modo equitativo. [17]

2.3.4.4 *Weighted Round Robin* (WRR)

O algoritmo *Weighted Round Robin* (WRR) procura aplicar os princípios do escalonamento RR, com o acréscimo da atribuição de pesos às filas de espera, para uma partilha proporcional dos recursos. Pretende aproximar-se ao ideal GPS no que toca à equidade de escalonamento.

Em cenários onde as fontes de informação possuem regimes de tráfego agressivo, ameaçando a atribuição justa de largura de banda entre as fontes de informação

existentes, este algoritmo escalona da informação de um modo substancialmente pior do que WFQ, que será apresentado na secção 2.3.4.7. Também, quando um fluxo é compensado devido a cenários onde o seu atendimento sofreu degradação, os restantes fluxos são prejudicados, podendo não ser atendidos de todo. Esta desvantagem ocorre porque os fluxos prejudicados serão recompensados independentemente do seu ritmo alocado, o que viola os princípios de garantia de equidade.

A essência deste algoritmo encontra-se no cálculo da quantidade mínima de pedidos a atender em cada fila de espera activa de modo que o seu serviço seja homogéneo e proporcional ao peso definido. Com base nesse indicador, as filas de espera activas são servidas ciclicamente.

O WRR simplesmente atribui um peso a cada fonte de informação, de modo que o peso de todas as aplicações seja de uma unidade. O atendimento é cíclico e estático ao longo do tempo, tal como no RR, sendo servida cada fila segundo a proporção do seu peso. O WRR é conservativo, servindo as filas de espera activas e desprezando as inactivas para conservação de um atendimento constante.

Em cenários onde as filas de espera possuem pesos distintos e os pacotes são de tamanho fixo, o escalonamento é efectuado servindo vários pacotes consecutivamente, de modo que os pesos definidos para cada fila de espera sejam cumpridos.

No caso de os pacotes possuírem tamanho variável, o peso de atendimento é normalizado segundo a dimensão do pacote. Torna-se contudo necessário conhecer a dimensão média do pacote para cada sessão, o que é problemático, e no curto prazo pode revelar-se injusto, visto que enquanto a dimensão média do pacote não possuir um valor realista, algumas sessões podem ser mais servidas do que as restantes. Assim, se uma fila de espera de peso inferior possuir pacotes excessivamente grandes em relação às restantes, o seu peso normalizado irá ser injusto em relação às filas restantes.

Contudo, existe compromisso entre a flexibilidade e o atraso que os pacotes sofrem, visto que grandes variações nos pesos das filas provocam o aumento do atraso que os pacotes sofrem nos piores cenários [16].

2.3.4.5 *Deficit Round Robin* (DRR)

A política de escalonamento *Deficit Round Robin* (DRR) [21] é derivada da WRR. Esta disciplina adopta os princípios do WRR, adicionando a capacidade de escalonar pacotes

de tamanho variável sem ter de considerar o seu tamanho médio.

O funcionamento desta disciplina consiste na associação de dois valores a cada fila de espera, designados *quantum* e *deficit counter*. O valor do *quantum* é proporcional ao peso da fila de espera respectiva. O *deficit counter* actua como um acumulador que reflecte a capacidade de largura de banda não utilizada, em relação à reservada para cada fila de espera. Assim, é servido o pacote da fila de espera que possuir um *deficit counter* com valor superior ou igual à dimensão do pacote a servir. O *deficit counter* da fila servida é actualizado subtraindo-lhe o tamanho do pacote. Uma fila transmite até a quantidade de informação servida superar a soma do seu *deficit counter* com o *quantum*.

Sendo este algoritmo baseado no RR, quando uma fila de espera é servida tem de aguardar que as restantes *N*-1 filas transmitam até voltar a ser servida.

Assim, o DRR possui complexidade constante tal como o RR, mas evita a sua falta de equidade através da consideração do tamanho médio do pacote pela estratégia *deficit counter*, que reduz a complexidade do WRR.

Esta disciplina revela-se uma melhor aproximação ao ideal GPS do que as disciplinas RR e WRR.

2.3.4.6 *Fair Queueing* (FQ)

Proposto por *John Nagel* em 1985 [22], o algoritmo *Fair Queueing* (FQ) é a concretização do algoritmo GPS orientado ao escalonamento conservativo de informação não infinitesimal, de um modo segmentado.

Ao contrário da disciplina FIFO, onde uma sessão pode aumentar a quantidade de recursos consumidos através da inserção de uma maior quantidade de pedidos na fila de espera, o objectivo principal da disciplina FQ é a de servir as várias filas de espera de um modo proporcional segundo a política de partilha de recursos, independentemente do nível de carga das mesmas.

Esta política introduziu o conceito da manutenção de uma fila de espera para cada fonte de informação com um atendimento individualizado das mesmas, evitando o problema existente na política *Priority Queue* (PQ), onde uma fila de espera pode monopolizar toda a largura de banda disponível, provocando negação de recursos das filas de espera de prioridade inferior.

A uma taxa de expedição *R*, se existirem *N* filas activas, cada fila é servida ao ritmo de atendimento *R/N*, mas a decisão de qual a fila de espera a atender em cada iteração é feita considerando o tamanho dos pacotes e distinguindo os vários fluxos existentes, de modo a homogeneizar a quantidade de dados emitida por cada fila de espera.

A principal vantagem do FQ consiste na protecção contra fluxos mal comportados, constituídos por altos débitos e rajadas de pedidos, não permitindo que as restantes fontes sejam prejudicadas. A largura de banda residual, que não for utilizada por eventuais filas de espera é partilhada de modo justo pelas filas activas que requisitem mais largura de banda do que a que têm direito. Assim, quando uma fila de espera mal comportada emite demasiados pacotes, esta não irá consumir largura de banda que deve ser utilizada pelas outras fontes, provoca apenas o aumento da fila de espera respectiva. Este algoritmo segue assim os princípios *Max-Min*.

O funcionamento do FQ consiste na computação de um valor de tempo virtual de ingresso e tempo virtual de expedição, para cada pacote, sendo utilizado o tempo real em que o pacote ingressa no serviço. O tempo virtual de expedição consiste na soma do tempo virtual de ingresso com o tempo de transmissão do pacote.

Mais especificamente, seja:

T_i^q - Tempo de transmissão do pacote i na fila de espera q

B_i^q - Tempo virtual em que o pacote i ingressa na fila de espera q

F_i^q - Tempo virtual em que o pacote i é expedido da fila de espera q

τ_i^q - Tempo real de chegada do pacote i à fila de espera q

$R(t)$ - Tempo virtual que representa o número de rondas ocorridas até ao instante t

Então o tempo virtual de expedição é computado segundo as condições:

$$F_i^q = B_i^q + T_i^q$$

$$B_i^q = \max\left[F_{i-1}^q, R\left(\tau_i^q\right)\right]$$

Equação 12. Tempo virtual de expedição Fair Queueing [22]

O algoritmo FQ é assim semelhante ao RR, mas com melhores garantias de equidade. Contudo, a dimensão do pacote não é considerada no processo de escalonamento, o que provoca desigualdade na quantidade de informação transmitida por cada fila de espera, levando também ao aumento do tempo de serviço nas filas de espera que possuem

pacotes com dimensão média superior às restantes. Para além disso, o FQ não atribui pesos distintos pelos vários fluxos, não existindo a diferenciação de atendimento que ocorreria no caso das filas de espera possuíssem pesos distintos.

Este algoritmo possui ainda as desvantagens de um algoritmo que utiliza referências temporais em cada pacote, que reside na computação da função de tempo virtual sempre que um pacote ingressa nas filas de espera.

2.3.4.7 *Weighted Fair Queueing* (WFQ)

Com o objectivo de combater a dependência do recurso ao tamanho médio do pacote, foram apresentados os algoritmos *Packet-by-packet GPS* (PGPS) [6] e *Weighted Fair Queueing* (WFQ) [8]. Ambos os algoritmos representam métodos idênticos, apenas foram apresentados de forma independente.

Este algoritmo foi apresentado como sendo uma aproximação ao GPS, e é uma generalização do FQ, possuindo todas as vantagens deste e adicionando peso a cada fila de espera, o que permite a regulação da quantidade de serviço efectuada por cada fila de espera de um modo conservativo. [6]

A principal ideia do PGPS consiste em gerir o serviço segundo o serviço idealizado pelo modelo GPS, sendo a ordem de serviço determinada com recurso à computação de uma função de tempo virtual, segundo os tempos de serviço previstos no modelo GPS. Assim, o WFQ recorre à computação de uma função de tempo virtual de expedição, no momento de ingresso de cada pacote nas filas de espera, sendo os pacotes servidos segundo a ordem crescente de grandeza dos seus tempos virtuais de expedição. Nessa função é considerada a largura de banda reservada a cada ligação, segundo o peso de cada fila e a sua actividade, assim como o tamanho dos pacotes. Também é utilizado um indicador, designado R(t) que consiste numa função que representa o número de rondas executadas pelo escalonador até ao instante *t*. Quantas mais sessões activas existirem, superior será o tempo de uma ronda, e consequentemente, maior será o atraso de atendimento dos pacotes, sendo a evolução do valor de *R(t)* derivada inversamente proporcional ao número de ligações activas.

Como a taxa de crescimento do valor do número de rondas aumenta segundo o número de ligações activas, o tempo virtual de expedição é calculado independentemente das sessões activas concorrentes, visto que o factor *R* é utilizado nesses cálculos. Uma vez

calculado, o valor do tempo virtual de expedição não é alterado, não considerando as chegadas e partidas futuras de pacotes às filas de espera [6]. É este facto que permite ao WFQ atingir uma boa aproximação ao modelo GPS.

Assim, o processo total que cada pacote sofre antes de ingressar numa fila de espera começa pela leitura do tempo virtual de término do último pacote servido, sendo computado novamente o valor do número de rondas efectuadas, para cálculo do tempo virtual de expedição do pacote. O pacote é inserido na fila de espera respectiva, ordenado segundo o seu tempo virtual, e no caso da fila de espera se encontrar lotada, é descartado o pacote com maior tempo virtual associado. Finalmente, o serviço de escalonamento trata de atender o pacote com menor tempo virtual de expedição.

De um modo informal, seja *R* o número de rondas efectuadas e *f* a função de tempo virtual de serviço, quando um pacote de tamanho *p* ingressa numa fila de espera vazia, o seu tempo de serviço será de *R* + *p*, independentemente do número de sessões existentes. Se contudo, o pacote ingressar numa fila de espera não vazia, o seu tempo de serviço será *f* + *p*, onde *f* representa o valor do tempo virtual de expedição do pacote anterior.

Mais detalhadamente, sendo:

$P(i,k,t)$ a dimensão do pacote k da fila de espera i,

R(t) o tempo de ronda no instante t,

$F(i,k,t)$ o tempo virtual de expedição do pacote k, da fila de espera i

W_i o peso associado à fila de espera i

A expressão que permite a computação do tempo virtual de expedição é:

$$F(i,k,t) = \max\left|F(i,k-1,t), R(t)\right| + P(i,k,t)/\mathrm{w(i)}$$

Equação 13. Tempo virtual de expedição Weighted Fair Queueing [8]

Ou seja, o tempo de término de um pacote deve ser o tempo de ronda ou o tempo do pacote anterior, consoante o que for superior, somado com o tamanho do pacote a atender. O *w(i)* é o factor que permite o atendimento diferenciado a nível de ritmo entre as varias sessões activas existentes.

Como é apresentado em [6], no pior caso, o algoritmo PGPS aproxima-se mais do GPS do que o algoritmo WRR. O tempo de transmissão de um pacote na disciplina WFQ pretende ser equivalente à soma do tempo de transmissão segundo o modelo GPS, nas

mesmas condições, de modo que a diferença nos tempos de serviço nunca supere o tempo de transmissão do pacote de dimensão máxima.

O WFQ tem a desvantagem ter a complexidade computacional alta, derivada da verificação do peso de cada classe de serviço por iteração. Este factor levanta consequentemente problemas de estabilidade, devendo ser utilizadas poucas classes de serviço para limitação da carga computacional.

Duas disciplinas de escalonamento distintas são consideradas semelhantes quando efectuam escalonamento com a mesma velocidade, segundo o mesmo conjunto de sessões e padrões de chegada de tráfego, e se possível a mesma partilha de recursos para cada fila de espera. Contudo, o WFQ não mantém equidade no pior caso, porque um serviço recebido por uma sessão *i* no WFQ conclui serviço mais rápido que o GPS.

2.3.4.8 *Worst-case Fair Weighted Fair Queueing* (W2FQ)

Em [23] encontra-se demonstrado que o algoritmo WFQ pode servir mais do que o GPS, conduzindo a um AFB superior ao pacote de tamanho máximo. No WFQ podemos ter o caso da quantidade de informação servida ser superior ao GPS, no entanto segundo a disciplina *Worst Case Fair Weight Fair Queueing* (WF2Q) é garantida que o nível de serviço nunca ultrapassa o nível de serviço do modelo GPS.

Partilhando os princípios de equidade e garantias de atraso do WFQ, o algoritmo W2FQ foi desenvolvido para colmatar o facto de o WFQ não ser tão próximo do GPS como suposto [23]. Assim, o W2FQ consiste numa melhor aproximação ao modelo GPS, de modo que a diferença nos tempos de serviço nunca supere o tempo de serviço do pacote de dimensão máxima.

As diferenças práticas deste algoritmo em relação ao WFQ são que, segundo o algoritmo W2FQ, a escolha do pacote a ser atendido toma em consideração apenas os pacotes que no serviço GPS já tivessem a sofrer atendimento, escolhendo o pacote que iria terminar primeiro o serviço segundo o modelo GPS. No algoritmo WFQ, a escolha do pacote a ser atendido considera todos os pacotes que se encontram nas filas, escolhendo o pacote que termina o serviço em primeiro lugar segundo o modelo GPS [23].

Este algoritmo é especialmente útil em redes de pacotes, pois optimiza o serviço da informação quando sofre interacção com um algoritmo de controlo de carga, na medida em que a fila de espera não precisa de conhecer o ritmo a que deve enviar a informação.

As desvantagens do algoritmo W2FQ residem no facto de, tal como no algoritmo WFQ, a complexidade temporal ser elevada, devido à computação iterativa de funções de tempo virtual. A função de tempo virtual é utilizada para obtenção dos tempos ideais de término que seriam obtidos segundo o modelo GPS. Assim, para reduzir o índice AFB (relação com o ideal GPS), podem ser escolhidos os pacotes de menor tempo virtual de expedição que tenham começado o serviço no equivalente GPS.

Este algoritmo garante ainda equidade no pior cenário. O pior cenário *Worst Case Fair Index* é uma métrica que permite medir a discrepância entre um escalonamento discreto e o modelo idealista GPS [6]. Esta métrica pode afectar a eficiência de mecanismos de controlo de congestão, que supõem o comportamento de um escalonador tipo GPS.

2.3.4.9 *Self-Clock Fair Queueing* (SCFQ)

Numa tentativa de reduzir a complexidade computacional que caracteriza o algoritmo WFQ, mas mantendo as suas propriedades, foi proposto em [24] o algoritmo SCFQ.

O funcionamento deste algoritmo é semelhante ao algoritmo WFQ, contudo, à medida que os pacotes chegam, é computada a função de tempo de término, considerando a etiqueta temporal do pacote servido na iteração anterior. De seguida o pedido é inserido na fila de espera respectiva e aguarda atendimento. Por sua vez, o escalonador vai transmitindo os pacotes com o menor valor de tempo de término.

Parte da complexidade elevada que o algoritmo WFQ possui resulta da computação da função de tempo virtual, que considera a referência temporal com que cada pacote seria expedido no modelo GPS. O resultado desta função é utilizado para decisão de qual a fila de espera a atender em cada iteração.

Em contraste, o algoritmo SCFQ possui a sua própria referência temporal, computando uma função de tempo virtual que depende apenas do progresso de transmissão ocorrido nas filas de espera. Assim, o tempo virtual de término é indicado em etiquetas temporais já existentes nos pacotes servidos [24].

Deste modo, a função de tempo de término, muito semelhante à do algoritmo WFQ, é expressa na Equação 14.

$$F(i,k,t) = \max\{F(i,k-1,t), F_{serv}\} + \frac{P(i,k,t)}{W(i)}$$

Equação 14. Tempo virtual de expedição Self Clock Fair Queueing [24]

A indicação F_{serv} representa o tempo virtual do pacote em serviço. Assim, é evitada a computação do indicador *R(t)*, utilizado no algoritmo WFQ, que representa o número de rondas executadas pelo escalonador até ao instante *t*. Sempre que o servidor não possuir filas activas para atender, o valor da referência temporal é reiniciado a zero [24]. Estes factos permitem que computação da função de tempo virtual possua a desejável complexidade constante (O(1)).

Para além da computação relaxada do tempo virtual, o SCFQ efectua um escalonamento equitativo, repartindo a largura de banda eficientemente entre as fontes. É mais facilmente implementável que o algoritmo WFQ, e também permite fornecer garantias para limitação do atraso fim a fim, quando limitando o ritmo das filas de espera de cada escalonador através de um regulador.

Contudo, apesar de menos complexo o SCFQ possui desempenho inferior ao WFQ, podendo ser injusto no funcionamento a curto prazo, especialmente quando o número de ligações aumenta, o que demonstra que é menos escalável.

O SCFQ tem o mesmo índice *worst case* que o WFQ, mas no WFQ uma sessão tem de transmitir mais rápido do que o suposto para atingir situação *worst case*, enquanto no SCFQ essa situação pode suceder sem que a transmissão superior ao ritmo esperado ocorra.

Em [15] é apresentado uma implementação eficiente do SCFQ, que recorre à utilização de duas estruturas de filas de espera, uma para armazenamento dos pacotes, e outra gestão dos tempos virtuais de término, permitindo a computação da função de tempo virtual apenas no instante que o pacote se encontra na cabeça da fila de espera. Isto seria impossível segundo o algoritmo WFQ que depende do indicador de número de rondas efectuadas (*R(t)*) no momento de chegada do pacote.

Deste modo aumenta-se a eficiência através da redução da complexidade de ordenação, mantendo-se as garantias de equidade desejadas.

2.3.4.10 *Most Credit First* (MCF)

Nos últimos anos novas espécies de disciplinas de escalonamento foram propostas. Em 2005, foi apresentada em [25] uma disciplina de escalonamento baseada em créditos, denominada *Most Credit First* (MCF).

A missão desta disciplina consiste em minimizar a diferença entre o serviço que um fluxo deve receber, segundo um modelo de equidade ideal, e o serviço que efectivamente é recebido. Para isso, é associado a cada fila das N filas de espera uma quantidade de créditos (c_i), que representam a largura de banda que esta reserva na iteração respectiva. Deste modo, considerando que a capacidade total do canal é de uma unidade, a soma dos créditos de todas as filas activas tem de ser igualmente uma unidade. Assim, o valor do crédito flutua entre 0 e 1, nunca sendo superado. É este factor que permite equidade no serviço prestado, como descreve a condição:

$$\sum_{i=1}^{N} c_i = 1, \quad 0 < c_i \leq 1$$

Equação 15. Condição de créditos alocada às filas de espera [25]

É importante a distinção entre a interpretação de crédito (c_i) e de balanceamento (b_i), pois apesar de o crédito reflectir a largura de banda reservada a cada fila, o termo balanceamento reflecte a largura de banda efectivamente consumida em cada iteração. Assim, é associado a cada fila um terceiro conceito, o crédito acumulado (A_i), que consiste na diferença entre a largura de banda reservada e a efectivamente consumida em cada iteração, como descreve a condição:

$$A_i(t) = \begin{cases} 0, & t = 0 \\ A_i(t-1) + c_i(t-1) - b_i(t-1), & t \geq 1 \end{cases}$$

Equação 16. Condição de crédito acumulado nas filas de espera [25]

Finalmente, o crédito disponível (V_i), consiste na soma dos créditos acumulados com o crédito de largura de banda reservada, como descreve a equação:

$$V_i(t) = A_i(t) + c_i$$

Equação 17. Crédito disponível nas filas de espera

As filas de espera são servidas de modo balanceado segundo este indicador. Para cada iteração, a fila de espera que possuir maior crédito disponível é servida.

Uma vez atendida, o valor de crédito acumulado da fila de espera é reduzido, mantendo o equilíbrio no processo de atendimento, segundo os créditos definidos.

Fluxos que requerem largura de banda e possuem créditos negativos são penalizados, não transmitindo até que os seus créditos aumentem.

O algoritmo MCF proporciona boa equidade graças ao princípio de manter o balanceamento do serviço (b_i) igual ao seus créditos (c_i), que segundo a definição de equidade, permite que cada fluxo consuma a quantidade de largura de banda que merece. É provado em [25] que a disciplina MCF tem melhor desempenho que as disciplinas WFQ e DRR.

Apesar de o MCF estar especificado para pacotes de tamanho fixo, pode ser facilmente adaptado a pacotes de tamanho variável, usando o crédito acumulado como critério de decisão de qual a fila de espera a atender, e considerar o tamanho do pacote nos cálculos do credito e do balanço, como especificado em [25].

2.3.4.11 *Credit-Based Fair Queue* (CBQF)

Uma outra disciplina de escalonamento baseada em créditos é apresentada em [26], denominada *Credit-Based Fair Queueing* (CBFQ).

Semelhante à disciplina MCF, o CBFQ recorre a indicadores de créditos acumulados, que reflectem a largura de banda que deve ser utilizada para cada fila de espera. Contudo, o CBFQ considera aspectos que o tornam adaptativo, como o tamanho dos pacotes e a carga existente nas filas de espera activas.

O princípio de funcionamento do algoritmo de escalonamento CBFQ consiste na utilização de vários indicadores que avaliam a quantidade de créditos acumulados para cada fila de espera, assim como no tamanho das filas de espera activas, sem desprezar a percentagem de atendimento reservada para cada uma. Com base nesses valores o algoritmo decide qual a fila de espera que deve ser servida, baseado na percentagem de transmissão que cada fila deve manter em relação às restantes.

Tal como na disciplina MCF, o CBFQ, associa uma quantidade de créditos (c_i) a cada uma das N filas de espera, que reflecte a largura de banda reserva às mesmas, como indicado na condição:

$$\sum_{i=1}^{n} c_i = 1, \;\; 0 < c_i \leq 1$$

Equação 18. Condição de créditos alocada às filas de espera [26]

Como a política CBFQ é adaptativa, o valor de créditos que reflectem a largura de banda efectivamente utilizada por cada uma das filas (k_i), do conjunto N de filas fila de espera activas, depende do nível de carga (l_i) verificado na fila eleita para transmissão. Apenas à fila eleita para transmissão é atribuído crédito nulo.

Este comportamento é especificado na condição:

$$k_i(t) = \begin{cases} 0 & , i = winner \vee t = 0 \\ k_i(t-1) + \dfrac{l_{winner}(t) - k_{winner}(t-1)}{c_{winner}} c_i & , \forall i \in N \setminus \{winner\}, t \geq 1 \end{cases}$$

Equação 19. Largura de banda associada a uma fila de espera [26]

O indicador de crédito disponível para cada uma das N filas de espera (V_i), é calculado em função dos créditos (c_i), da largura de banda efectivamente utilizada por cada uma das filas (k_i) e do nível de carga (l_i), associados a cada uma das N filas de espera, como é descrito na equação:

$$V_i(t) = \frac{l_i(t) - k_i(t-1)}{c_i} \;, \;\; \forall i \in Q$$

Equação 20. Crédito disponível associado às filas de espera [26]

Esta variável permite a decisão de qual a fila de espera a atender, sendo eleita a fila que possuir menor crédito disponível (V_i) na iteração decorrente, como especificado na condição:

$$V_{winner}(t) \leq V_i(t), \;\; \forall i \in N \setminus \{winner\}$$

Equação 21. Condição de crédito disponível na fila de espera eleita [26]

O serviço é assim balanceado para que a equidade seja mantida entre as filas ao longo do tempo. Deste modo, a disciplina CBFQ atinja níveis de equidade e garantias de atraso análogo às obtidas com as disciplinas que recorrem à computação de funções de tempo virtual, evitando as suas desvantagens, e proporcionando um método mais simples de implementação.

2.4 Conclusões

Este capítulo apresentou um estudo relativo a algumas metodologias subjacentes a um mecanismo de controlo de admissão. Foram abordadas soluções de controlo de ritmo, de distribuição de largura de banda, e algumas das mais importantes disciplinas de escalonamento.

As metodologias analisadas são, em muitos casos, orientadas a redes IP, sendo a sua actividade exercida sobre pacotes IP. Contudo, no contexto deste trabalho, o controlo efectuado pelas metodologias adoptadas é exercido sobre referências de pedidos de requisição de serviço, sendo o funcionamento destes mecanismos adaptado para uma actividade em tudo semelhante ao que é proporcionado nas redes IP. O conceito de dimensão, associado aos pacotes IP, não é considerado no âmbito do pedido, visto que os pedidos são simples invocações de serviços.

Assim, como resultado deste estudo, são adoptados as metodologias que se revelam mais convenientes para implementação numa solução de controlo de admissão destinada ao módulo UIF, tomando em consideração os objectivos apresentados na secção 1.2 desta dissertação.

A apresentação e especificação da solução são efectuadas no capítulo 3.

Capítulo 3

Solução Proposta

Neste capítulo é apresentada uma solução para efectuar o controlo de admissão diferenciado e adaptativo no módulo que implementa a interface UIF (*Unified Interface*).

A solução, destinada a ser integrada numa interface de acesso a uma plataforma de prestação de serviços, visa efectuar o controlo do ritmo com que os pedidos de invocação dos serviços são admitidos na plataforma, modulando a sua admissão de um modo diferenciado de acordo com a classe de serviço que é invocada em cada pedido.

O UIF recebe pedidos de requisição de serviço, emitidos por entidades externas à plataforma NGIN, descodificando-os e invocando os serviços respectivos. Cada pedido consiste numa invocação para a prestação de um serviço, sendo os serviços executados após o pedido ter sido admitido na plataforma.

Independentemente da distribuição com que os pedidos são emitidos na interface de acesso à plataforma, deve ser garantido um nível mínimo de taxa de admissão e do atraso sofrido pelos pedidos, em função das classes de serviço. Assim, no processo de controlo de admissão, a ocorrer ao nível da interface de acesso, devem ser fornecidas tais garantias de qualidade de serviço através da aplicação de políticas de escalonamento, com recurso a mecanismos de filas de espera, de modo que o escalonador defina a ordem com que os pedidos são admitidos.

A maioria dos métodos analisados no estudo do capítulo 2 são focados no controlo de admissão de pacotes em redes IP, contudo a solução de controlo de admissão presente não necessita de considerar a grandeza dos pedidos que admite, pois a carga encontra-se na execução do pedido uma vez admitido, e não no processo de admissão. Este facto simplifica consideravelmente a prestação de garantias de equidade entre as várias classes de serviço, sendo o escalonamento efectuado como se fosse focado a pacotes de tamanho fixo.

Nas seguintes secções são inicialmente apresentados os requisitos que a solução deve obedecer assim como o ambiente onde se deverá integrar. Seguidamente é apresentado o procedimento que os pedidos de requisição de serviço percorrem no seu processo de admissão, assim como a arquitectura que o sustenta. Finalmente, é apresentado o funcionamento global da solução após a sua integração no UIF.

3.1 Requisitos Funcionais

A solução deve considerar os seguintes requisitos:

- limitação da taxa de admissão com que os pedidos são admitidos para a plataforma NGIN;
- armazenamento temporário dos pedidos rejeitados em cenários onde o ritmo limite é superado, impedindo que estes sejam rejeitados directamente;
- diferenciação justa, garantindo equidade de serviço em função da prioridade definida para cada classe de serviço;
- alternativa de serviço adaptativo, segundo a carga associada a cada classe de serviço, proporcionando a atribuição flexível de largura de banda ao longo do tempo;
- minimização do tempo de admissão dos pedidos em cenários onde o ritmo é superado;
- capacidade de configurar a granularidade com que a limitação do ritmo é efectuada, assim como os perfis que caracterizam as classes de serviço. As alterações devem ser aplicadas sem que a interface precise de ser reiniciada.

3.2 Ambiente de Integração

Como foi introduzido no início deste capítulo, o mecanismo de admissão integra-se no UIF, que consiste numa interface de acesso à plataforma NGIN. Esta interface permite a satisfação de pedidos de requisição de serviço *HTTP URL Encoded*, efectuados por entidades externas à plataforma.

O UIF encontra-se desenvolvido num módulo, em *Java2EE*, sendo compilado recorrendo ao *Ant*, e executado em plataformas que suportem os servidores aplicacionais *WebLogic* e *JBoss* [28].

A interface de acesso encontra-se dividida em dois níveis operacionais: o nível das *Interfaces*, que estabelecem a ligação das entidades externas com o segundo nível, o *Engine*, que possui toda a lógica de funcionamento e regras de acesso aos serviços.

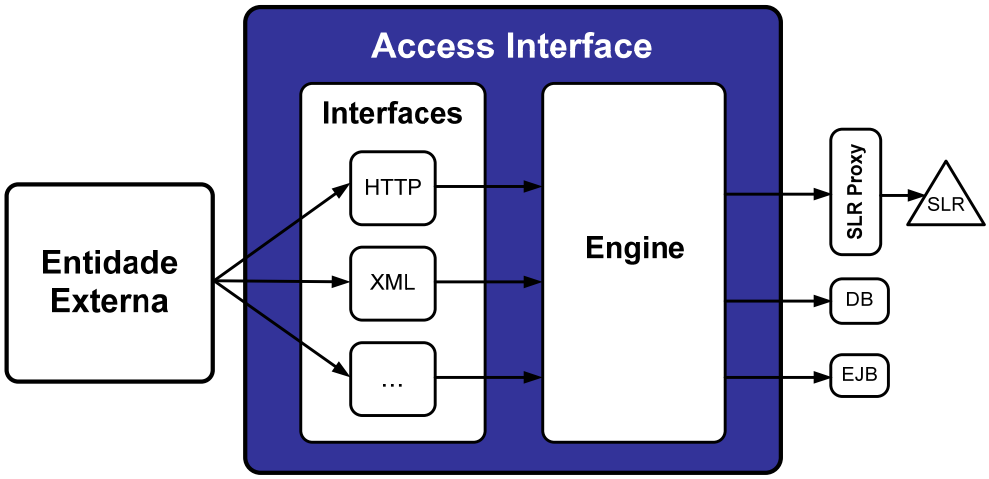

Figura 5. Arquitectura simplificada da Interface de Acesso

Os pedidos efectuados pelas entidades externas às *Interfaces* sofrem um processo de interpretação e verificação de admissão antes de serem executados pelo *Engine*.

Inicialmente, os pedidos são encaminhados para a Interface correspondente, sendo-lhes atribuída uma identificação. De seguida, o *Engine* verifica se a entidade externa tem permissão para aceder ao serviço invocado. Também é verificado se a invocação do pedido é efectuada no período horário permitido, assim como se o número máximo de ligações permitidas à entidade externa associada não é excedida.

Se for atribuída permissão ao pedido, é então executado o serviço nele invocado, podendo este consistir na invocação de primitivas, procedimentos ou funções de base de

dados, ou na invocação de métodos implementados em *EJBs*. A resposta é finalmente entregue pelas *Interfaces* à entidade externa, no formato XML.

As entidades externas emitem os pedidos a um ritmo indeterminado. Assim, na transição do pedido das *Interfaces* para o *Engine*, antes da execução de cada pedido, estes são submetidos a um processo de controlo de admissão que efectua a autenticação e validação. Se for concedida permissão, o serviço invocado em cada pedido é executado, caso contrário, este não pode prosseguir e é devolvida à entidade externa uma reposta com uma mensagem esclarecedora do motivo.

No UIF, o processo de admissão dos pedidos apenas considera a validade dos serviços neste invocados. A Interface carece de um processo de admissão que limite a taxa de aceitação dos pedidos, para que em situações em que são recebidos pedidos em excesso, o acesso destes à plataforma seja regulado de modo a limitar indirectamente a carga a que a plataforma é sujeita.

Outra carência da Interface consiste na incapacidade de efectuar diferenciação na admissão dos pedidos. Como existe interesse em que determinados serviços possuam precedência em relação a outros, e que sejam fornecidas garantias de qualidade de serviço, revela-se útil a agregação de determinados conjuntos de pedidos em classes de serviço.

3.3 Processo de Admissão

Em situações onde a taxa de ingresso dos pedidos na interface de acesso é superior ao ritmo limite, é necessária a intervenção de um mecanismo de controlo de admissão que, com uma determinada margem, permita a satisfação dos pedidos que superam o ritmo máximo sem que estes sejam imediatamente rejeitados.

Assim, em cenários em que o ritmo limite é superado, os pedidos de requisição de serviço devem ser inseridos numa fila de espera onde aguardam até serem atendidos por um escalonador, ou rejeitados por excesso de tempo de espera pelo seu serviço.

Neste âmbito e com o objectivo de colmatar as carências de controlo de admissão manifestadas na Interface de acesso, o processo de admissão foi implementado de modo a satisfazer os requisitos apresentados na secção 3.1.

O processo de admissão aplicado a cada pedido que ingressa na interface é ilustrado na Figura 6.

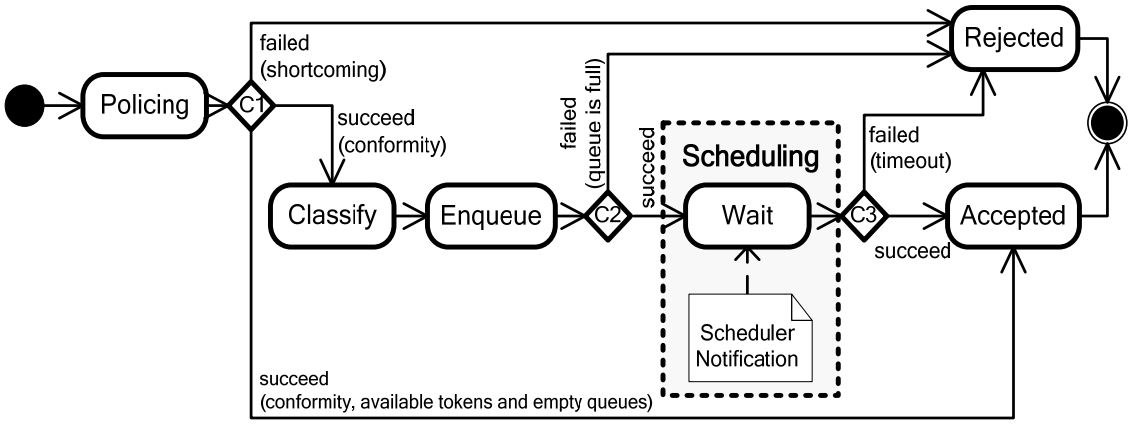

Figura 6. Algoritmo de controlo de admissão

Na solução apresentada, a admissão de um pedido é constituída por três fases principais: policiamento, classificação e escalonamento. Estas fases assentam sobre um método de controlo da taxa de admissão. Cada uma das fases é condicionada, podendo o processo de admissão terminar sem que todas as fases se processem.

3.3.1 Policiamento

O processo de policiamento é a primeira fase do processo de admissão. Este processo já se encontra implementado na interface de acesso para o qual esta proposta é destinada.

O policiamento consiste na verificação de conformidade dos pedidos com o contrato especificado. Quando um pedido ingressa na interface respectiva é verificado se este possui permissão para aceder ao serviço que invoca e se este pode ser invocado no período actual, de acordo com o contrato especificado.

Contudo, uma vez verificadas as permissões para execução do pedido, inicia-se todo um processo de admissão que a Interface de acesso não possui. Começa por ser verificado se a eventual admissão do pedido provoca a infracção do ritmo de admissão máximo (condição C1 da Figura 6). Se tal não se verificar, o pedido é admitido e o seu processo de admissão termina. Porém, se a admissão do pedido não for possível pelo facto da taxa permitida de admissão ter sido superada, o pedido terá de aguardar um determinado período de tempo até que a admissão lhe seja concedida, se possível. Neste último caso o pedido procede à fase de Classificação.

O processo de admissão é assim simples no caso da taxa média de ingresso dos pedidos ser inferior ao limite, prosseguindo o processo de admissão apenas em situações que a taxa limite é superada.

3.3.2 Classificação

Num mecanismo de diferenciação, a informação pode ser diferenciada por fluxo, individualmente, ou diferenciada segundo conjuntos agregados de fluxos, através de classes de serviço.

A agregação de vários serviços numa classe de serviço trás claros benefícios de escala. A nível de eficiência, a principal vantagem resulta da redução do número de filas de espera que seriam utilizadas numa diferenciação por serviço, onde seria utilizada uma fila de espera por serviço.

O processo de Classificação consiste na identificação de qual a classe de serviço associada ao serviço invocado pelo pedido, sendo criada uma referência ao mesmo, que é rotulada com a classe de serviço respectiva. Todos os serviços registados na base de dados passam a ser mapeados com a classe de serviço respectiva. Deste modo, são estaticamente determinados quais os serviços que devem pertencer a determinada classe de serviço, agregando-os segundo o seu perfil.

A informação que permite a Classificação deve ser carregada da base de dados para memória local para que o processo de classificação de cada pedido seja efectuado o mais rapidamente possível, sem prejudicar o processo de admissão. Contudo, a utilização de memória local pode revelar-se indesejável na presença de um grande número de serviços mapeados. Tal opção deverá ser ponderada no processo de configuração da solução.

Assim, a agregação deve tomar em consideração todos os serviços que requerem as mesmas prioridades, e que possuam perfis semelhantes segundo os critérios da sua utilização.

Como existe uma fila de espera para cada classe de serviço, a Classificação é efectuada antes de uma referência do pedido ser armazenada na fila de espera, para que seja determinada qual a fila de espera a considerar.

Se a fila de espera, onde a referência de um pedido deve ser inserida, se encontrar lotada no momento de inserção (condição C2), o pedido não pode ser aceite, sendo a sua admissão imediatamente rejeitada. Caso contrário, se a referência do pedido é inserida

com sucesso na fila de espera, esta é automaticamente submetida ao processo de escalonamento.

3.3.3 Escalonamento

O escalonamento das filas de espera é efectuado de forma iterativa, permitindo o controlo da ordem pela qual os pedidos das diferentes classes de serviço são admitidos na plataforma.

Idealmente uma disciplina de escalonamento deve proporcionar da melhor forma as características apresentadas na secção 2.3.1. Estas são as características ideais que uma política de escalonamento deve possuir, sendo algumas algo contraditórias entre si. O principal ponto que se pretende satisfazer é o de proporcionar uma diferenciação equitativa no processo de admissão dos pedidos.

Uma segunda vertente consiste na possibilidade de se considerar a carga existente para cada classe de serviço, para adaptação das condições de equidade de uma forma dinâmica ao longo do processo de escalonamento.

No processo de escalonamento, as referências dos pedidos relativas a uma mesma classe, e consequentemente inseridas na mesma fila de espera, são servidas segundo a ordem de chegada, pelo método FIFO. Não existe necessidade de utilizar outra ordem de atendimento, visto que, como todos os pedidos existentes numa fila de espera pertencem à mesma classe de serviço, é implícito que possuem a mesma prioridade, sendo desnecessária a atribuição de preferências no atendimento de uns pedidos em relação aos restantes.

Nenhuma fila deverá permanecer indefinidamente sem ser atendida, de modo a maximizar o seu serviço, sem sacrificar a justiça e equidade.

Uma vez atendida uma referência a um pedido, o pedido que se encontrava em suspenso à espera de uma notificação por parte do processo de Escalonamento é reactivado. A notificação indica se o pedido foi ou não admitido (condição C3). A notificação de que o pedido é rejeitado ocorre no caso em que o tempo limite de espera pelo serviço é atingido.

A restrição de um intervalo de tempo permitido para que a referência do pedido seja atendida pelo processo de escalonamento advém do facto de que os pedidos devem ser atendidos em tempo real, necessitando de resposta em determinado tempo útil. Assim, a

necessidade de execução do pedido deixa de ter significado decorrido o período de tempo especificado na restrição. Nesse caso, o processo de escalonamento do pedido é prematuramente terminado, sendo este removido da fila de espera.

3.3.4 Controlo da Taxa de Admissão

No capítulo 2, na secção 2.1.3, foi apresentado o método *Leaky Bucket* para fins de regulação efectiva da taxa de admissão. A aplicação deste método no presente contexto iria provocar que todos os pedidos fossem encaminhados através de uma fila de espera, mesmo em situações em que a taxa máxima de admissão permitida não era excedida, recorrendo à fila de espera sem necessidade.

Preferencialmente, o algoritmo a adoptar deverá ser flexível, possuindo uma complexidade imperceptível enquanto a taxa de admissão dos pedidos não atingisse o limite definido, de modo a que os pedidos fossem encaminhados como se não existisse qualquer mecanismo de limitação do ritmo, sem grande complexidade adicional ao processo de admissão dos pedidos.

Adicionalmente, o algoritmo a implementar deverá permitir a admissão de rajadas de pedidos. Tal possibilidade pode ser vantajosa, aumentando a flexibilidade do processo de regulação de ritmo e diminuindo a necessidade de inserção de pedidos na fila de espera.

Assim, no caso de ser desejável, a admissão directa de pequenas rajadas de pedidos, com limitação do número de pedidos por rajada (conceito *Token Bucket*) pode ser aplicado a um algoritmo que permita efectuar uma limitação da taxa de admissão mais flexível, mas que ao mesmo tempo permita um atendimento directo de um número limitado de pedidos (pequenas rajadas de pedidos).

Tendo em consideração estas ideias, o processo de admissão apresentado utiliza o conceito *Token Bucket* para limitação da taxa de admissão ao longo do tempo. Deste modo, a admissão de pedidos através das filas de espera apenas é realizada no caso de não existirem *tokens* no *bucket* e das filas de espera se encontrarem vazias.

Deste modo o processo de admissão de pedidos é mais imediato, sendo apenas adicionada a capacidade da fila de espera no caso do ritmo de recepção dos pedidos exceder o ritmo permitido.

No caso do processamento de pequenas rajadas de pedidos não ser desejado basta limitar o contador a uma unidade, que apesar de efectuar o mesmo propósito do que a

solução *Leaky Bucket*, não possui a complexidade de inserção e remoção dos pedidos numa fila de espera sem necessidade.

Deste modo, o mecanismo de modelação de tráfego não se confina à limitação do ritmo, fornecendo também a flexibilidade desejada no atendimento dos pedidos que são recebidos em situações em que o ritmo limite foi superado.

3.4 Arquitectura da Solução

A arquitectura proposta para implementar o processo de admissão apresentado na secção 3.3. encontra-se ilustrada na Figura 7.

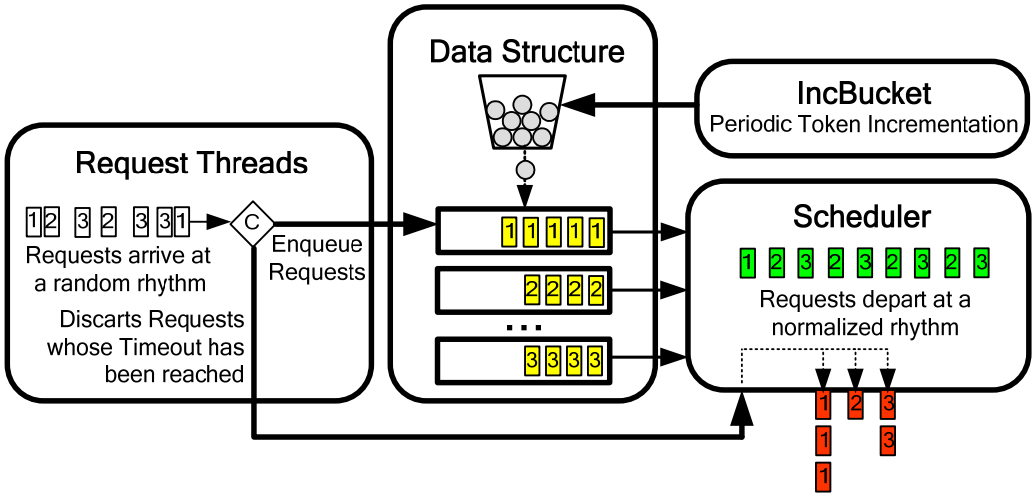

Figura 7. Arquitectura do mecanismo

Esta arquitectura possui vários componentes, coordenados entre si, de modo que o processo de escalonamento dite as permissões de admissão que os pedidos deverão receber para prosseguirem a sua execução. Adicionalmente, é possibilitado o ajuste da rigidez com que o controlo da taxa de admissão é efectuado, permitindo a configuração da sua granularidade.

Os pedidos ingressam na plataforma a ritmos imprevisíveis, sendo o mecanismo responsável pela sua admissão segundo uma taxa normalizada.

O processo de admissão é suportado por quatro módulos: as *Threads* de Pedido, Estrutura de dados, Incrementador e o Escalonador. Estes módulos são apresentados nas secções seguintes.

3.4.1 *Threads* de Execução dos Pedidos

Segundo o funcionamento da Interface de acesso, o processo de execução de cada pedido é efectuado através de uma *thread*. Assim, existe uma *thread* associada a cada pedido que é efectuado.

A funcionalidade de controlo de admissão apresentada é invocada em cada *thread*, antes da execução do pedido. Os processos de Policiamento, Classificação e inserção dos pedidos em fila de espera, indicados no processo de admissão especificado na secção 3.3., são executados nestas *thread*s.

De acordo com o processo de admissão, quando o ritmo não é superado, quando o *bucket* possui *token*s disponíveis, o pedido é admitido imediatamente, evitando o recurso às filas de espera e respectivos escalonadores. Contudo, basta existir um elemento em qualquer uma das filas de espera para que a admissão directa dos pedidos não seja permitida, sendo atribuída prioridade ao atendimento das filas de espera. Em cenários em que o ritmo máximo permitido é superado e as referências aos pedidos são inseridas nas filas de espera, a *thread* aguarda a notificação do escalonador com a respectiva resposta de admissão. Se a notificação não for recebida durante um período predefinido de tempo, ocorre a negação de admissão do pedido a sua referência é removida da fila de espera.

3.4.2 Estrutura de Dados

As estruturas de dados que suportam o controlo de admissão consistem num contador, designado *bucket*, e em várias filas de espera, uma para cada classe de serviço. Segundo a terminologia *Token Bucket*, o *bucket* define a granularidade e elasticidade com que os pedidos em fila de espera são servidos. Para isso, são configurados os parâmetros de limite das rajadas de pedidos (pedidos admitidos consecutivamente) e o número de unidades a incrementar periodicamente. Quando o *bucket* fica vazio é sinal que o ritmo de admissão permitido foi atingido, o que força a que o escalonador aguarde pela incrementação de mais unidades de *tokens* no *bucket*.

As filas de espera permitem o armazenamento dos identificadores dos pedidos. Deve existir uma fila de espera para cada classe de serviço, visto que as filas de espera serão atendidas de modo a atingirem diferentes níveis de performance, consoante a prioridade de atendimento pretendida para determinada classe de serviço. Como o nível de

prioridade dos pedidos pertencentes à mesma classe de serviço não varia, não é necessária a ordenação das filas de espera. Assim, cada fila é servida segundo a disciplina *first-in first-out*.

Não é suportado nenhum mecanismo de gestão activa sobre as filas de espera, pois o descarte de pedidos não é desejado. Uma vez introduzido na fila de espera o pedido deverá ser atendido. Se contudo o período de tempo a que o pedido está disposto a esperar for excedido, este deve ser removido da fila. É preferível que o serviço seja negado à partida, em situações de excesso de carga, do que ser aceite nessas circunstâncias.

Os conceitos apresentados são largamente utilizados para fornecer qualidade de serviço em redes de comunicações, onde o controlo é orientado ao pacote, sendo comum a execução do escalonamento tendo em consideração o tamanho dos mesmos. Esta consideração permite o equilíbrio da relação de carga atendida em cada fila de espera. Contudo, o modelo que se apresenta é orientado ao pedido e, portanto, não considera a dimensão do mesmo.

3.4.3 Incrementador

O Incrementador é responsável pela adição periódica de unidades no *bucket*.

A adição de unidades ao contador é limitada consoante o número máximo de pedidos que podem ser encaminhados consecutivamente (rajadas de pedidos), e é definida em função da taxa de admissão máxima de pedidos permitida por segundo.

A taxa de incremento pode ser alterada variando o número de *token*s adicionados em cada iteração. Quantos mais *token*s forem incrementados por iteração, maior é o período de tempo entre cada operação de incrementação, ocorrendo menos iterações por segundo. Contudo a granularidade do ritmo de aceitação é sacrificada, passando os pedidos a ser admitidos em rajadas mais longas. A decisão da quantidade de *token*s a adicionar ao *bucket* em cada iteração deve ser cuidadosamente ponderada devido à sua influência na granularidade de admissão. Assim, apenas devem ser adicionados vários *tokens* por operação quando o ritmo de admissão é muito elevado, para evitar que a operação de incrementação seja efectuada com um ritmo elevado.

O período de tempo decorrido entre cada operação de incrementação (*sleepTime*) é definido segundo a Equação 22.

$$sleepTime(ms) = \frac{nTokens * 1000}{admissionRate(requests/\sec)}$$

Equação 22. Período de incrementação do *bucket*

Assim, o *sleepTime* resulta da relação entre a quantidade de *tokens* que deve ser adicionada ao *bucket* em cada iteração e a taxa máxima de admissão permitida. Esta relação é multiplicada por 1000 para conversão de segundos a milissegundos. O Incrementador revela-se assim o elemento da solução responsável pelo modo como a modulação das taxas de admissão é controlada.

3.4.4 Escalonador

O Escalonador é responsável pelo serviço equitativo das filas de espera. As referências dos pedidos são servidas iterativamente, sendo decidida em cada iteração qual a fila de espera que deve ser servida.

Um escalonamento eficiente deve ser caracterizado pelas características apresentadas na secção 2.3.1. Das disciplinas de escalonamento analisadas na secção 2.3.4, as disciplinas MCF e CBFQ foram consideradas as mais adequadas a aplicar na solução. Todas as restantes disciplinas são consideradas inadequadas, seja por recorrerem à computação iterativa de funções de tempo virtual, ou por não proporcionarem níveis de equidade satisfatórios comparativamente com as disciplinas de escalamento baseadas em créditos.

Assim, o escalonamento pode ser efectuado segundo as duas disciplinas de escalonamento, MCF e CBFQ, conforme desejado e configurado.

A disciplina MCF [25] deve ser utilizada em cenários onde se pretende a atribuição justa de largura de banda com um comportamento linear, devendo ser o CBFQ [26] utilizado para um escalonamento adaptativo, consoante as condições de carga das filas.

Com a finalidade de cumprir o ritmo definido, o escalonador somente serve as filas de espera no caso de existirem *token*s a consumir no *bucket*. Para cada referência de pedido que é atendida da fila de espera, o *bucket* é decrementado um *token*.

3.4.5 Coordenação entre Módulos

As operações efectuadas pelos vários módulos são executadas de modo paralelo, mas dependem umas das outras. Este facto levanta a necessidade de uma coordenação cuidada entre as operações. Existem três níveis de coordenação:

(i) no caso das filas de espera se encontrarem vazias o escalonador pára, até que um novo pedido chegue, sendo emitida uma notificação da *thread* do pedido para o escalonador arrancar de novo o seu atendimento;

(ii) no caso de não existirem unidades disponíveis no *bucket* o escalonador pára, até que sejam incrementadas unidades no *bucket*, sendo emitida uma notificação do Incrementador para o escalonador arrancar de novo o seu atendimento;

(iii) sempre que o escalonador serve a referência de um pedido, envia uma notificação para a *thread* do pedido respectivo, que se encontra à espera de uma notificação que traduza a resposta de admissão do pedido;

(iv) sempre que o *bucket* está cheio e não chegam novos pedidos, o Incrementador suspende, até que uma notificação seja recebida por parte do escalonador, indicando a necessidade de activar o Incrementador de novo.

3.5 Integração da Solução Proposta

A solução apresentada foi integrada numa interface de acesso a uma plataforma de prestação de serviços.

A solução foi desenvolvida a nível aplicacional na linguagem Java EE, sendo compilada recorrendo ao *Ant*, e executada em plataformas que suportem os servidores aplicacionais *WebLogic* e *JBoss*.

Nas seguintes secções é apresentada a arquitectura da interface UIF com a solução de controlo de admissão integrada. São apresentados ainda alguns detalhes de concepção e implementação que tiveram lugar no processo de integração.

3.5.1 Arquitectura Resultante do Módulo UIF

A arquitectura resultante do módulo UIF, após a integração da solução de controlo de admissão, é ilustrada na Figura 8.

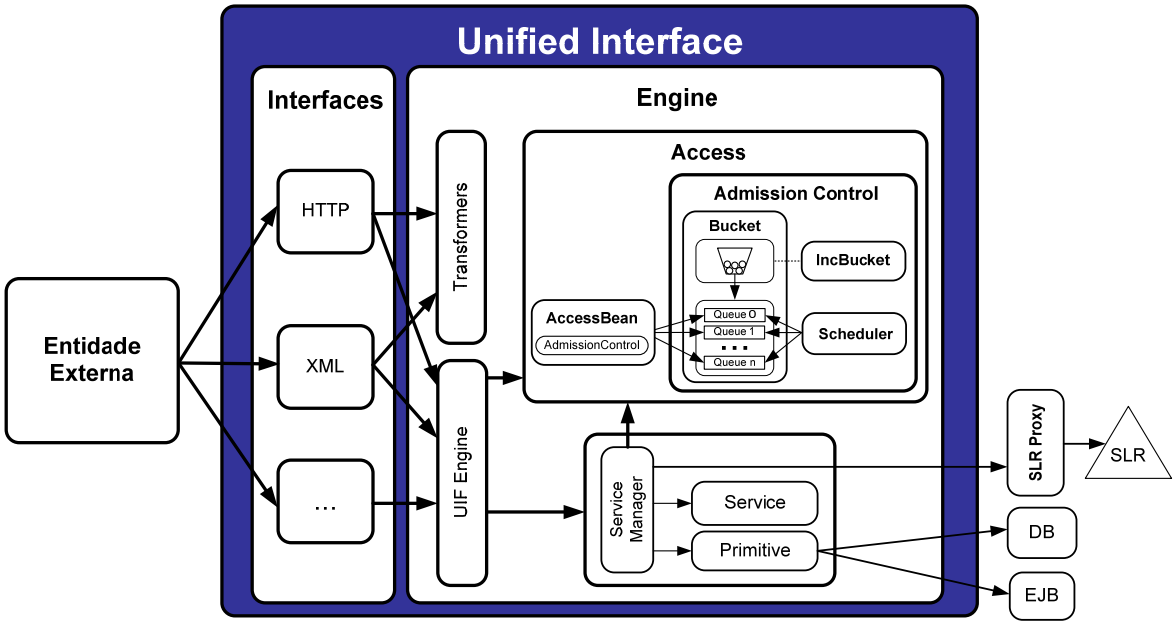

Figura 8. Arquitectura resultante do UIF

O núcleo do processo de controlo de admissão encontra-se implementado no *package AdmissonControl*, que é constituído fundamentalmente pela classe *Bucket*, pelo módulo incrementador designado *IncBucket*, e pelo escalonador designado *Scheduler*.

Estes componentes asseguram o comportamento da arquitectura da solução conforme o apresentado na secção 3.4. A classe *Bucket* instância a estrutura de dados necessária ao controlo de admissão, a *thread IncBucket* é responsável por adicionar periodicamente unidades de crédito ao contador do *Bucket*, e a *thread Scheduler* efectua o escalonamento das referências aos pedidos existentes nas várias filas de espera, segundo o algoritmo configurado.

A interacção deste *package* com a plataforma UIF é realizada ao nível do módulo Access, antes da autenticação de cada pedido, no método *login*. Este método encontra-se implementado no EJB *AccessBean.java,* e é executado para cada pedido.

3.5.1.1 Processo Global de Admissão

Os pedidos de requisição de serviço são emitidos pelas entidades externas à interface de acesso a um ritmo indeterminado. Os pedidos são descodificados a nível das *Interfaces*, sendo encaminhados para o UIF *Engine*, responsável pela execução do pedido. A primeira tarefa executada no UIF *Engine* é a execução do método *login* implementado no *AccessBean*. Neste método é verificada a validade de cada pedido.

Anteriormente à implementação da solução de controlo de admissão proposta, já eram efectuados alguns métodos de Policiamento apresentados na secção 3.3, nomeadamente para validação dos pedidos. Assim, com a integração da solução proposta, antes dos pedidos serem executados pelo UIF *Engine* é invocado dentro do método *login* o método *admissionControl*, responsável pelo processo de admissão proposto.

O retorno deste método vai definir o resultado de admissão. O pedido pode ser admitido, sendo concedida permissão para este prosseguir para o UIF *Engine*, para execução do serviço neste invocado. Contudo, o pedido também pode não ser admitido, sendo proibida a sua execução ao nível do UIF *Engine*. Nestes casos, é devolvida à entidade externa uma resposta com uma mensagem esclarecedora do motivo.

A execução do mecanismo de controlo de admissão adiciona a possibilidade de restringir a taxa de pedidos admitidos por segundo. Esta capacidade permite o controlo indirecto da carga de processamento que é induzida na plataforma. Tal carga resulta da posterior execução de procedimentos, métodos e funções necessários para satisfação do serviço invocado em cada pedido.

Uma vez executado o serviço é devolvida uma mensagem de resposta à entidade externa que invocou o pedido.

3.5.1.2 *AccessBean*

É no módulo *AccessBean* que é invocado o método *admissionControl*. Neste método começa por ser verificado se existem condições para admissão directa do pedido. Tal admissão só pode ocorrer quando cumpridas duas condições: a taxa de admissão máxima definida para o escalonamento dos pedidos não foi superada; as filas de espera não podem conter referências de pedidos a aguardar por autorização de admissão. Nesse caso

o pedido pode ser servido directamente, sem recorrer às filas de espera, evitando atrasos desnecessários.

A taxa de admissão máxima é superada sempre que o *bucket* se encontrar vazio, sendo necessário esperar o tempo predefinido para que o *bucket* seja incrementado e um novo pedido possa ser encaminhado.

A condição de garantir que todas as filas de espera se encontrem vazias, resulta da necessidade de que estas sejam processadas com prioridade, bastando a existência de uma referência de um pedido em qualquer uma das filas de espera para que o processo de admissão directa não seja possível. Ou seja, um pedido recém-chegado nunca é admitido directamente se existirem elementos em fila de espera à aguardar atendimento.

Quando um pedido não é admitido directamente, é inserido numa fila de espera para posterior admissão. Face a essa operação o método *admissionControl* suspende durante um tempo determinado, aguardando pela recepção de uma notificação proveniente do *Scheduler* que indica se o pedido respectivo pode prosseguir a sua execução.

Se o tempo de espera definido para aguardar pela notificação for excedido, o objecto referência do pedido será removido da fila de espera em que se encontra, sendo devolvido um código de erro que identifica que o pedido não pode ser atendido por excesso de carga submetido à plataforma UIF (*timeout*). O código é devolvido ao *MBean UIFEngine*, que trata de o mapear na mensagem que deve ser apresentada à entidade externa.

3.5.1.3 *Scheduler*

A *thread Scheduler* é responsável pelo escalonamento das filas de espera, assim como pela notificação das threads *AccessBean* associadas às referências dos pedidos que são servidos. Como foi referido na secção 3.3.4, foram implementadas as disciplinas de escalonamento MCF e CBFQ.

A disciplina MCF, apresentada na secção 2.3.4.10 e especificada em [25], foi implementada na solução segundo o algoritmo representado na Figura 9.

A disciplina CBFQ, apresentada na secção 2.3.4.11 e especificada em [26], foi implementada na solução segundo o algoritmo representado na Figura 10.

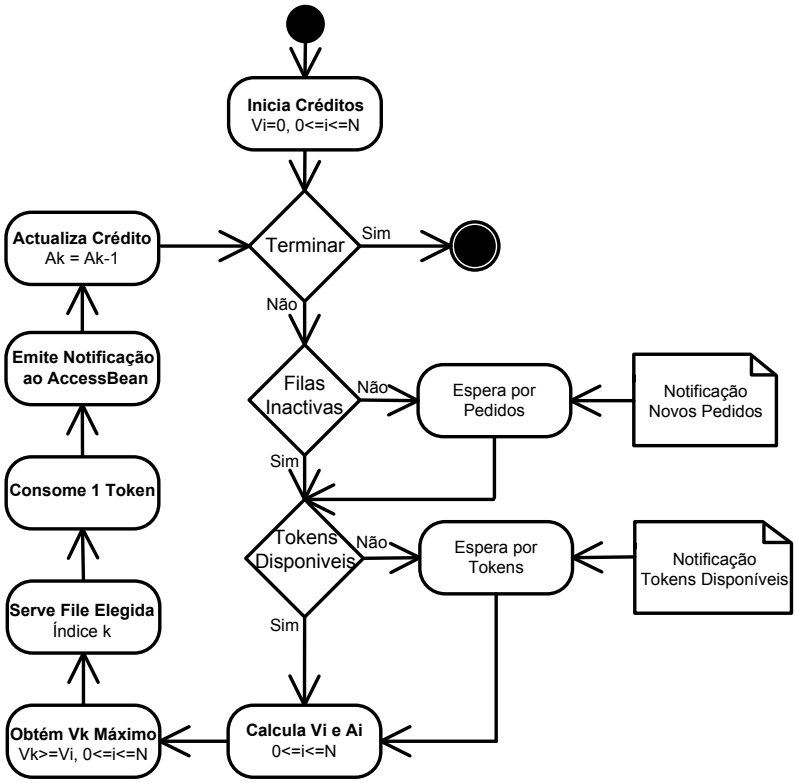

Figura 9. Algoritmo de escalonamento MCF

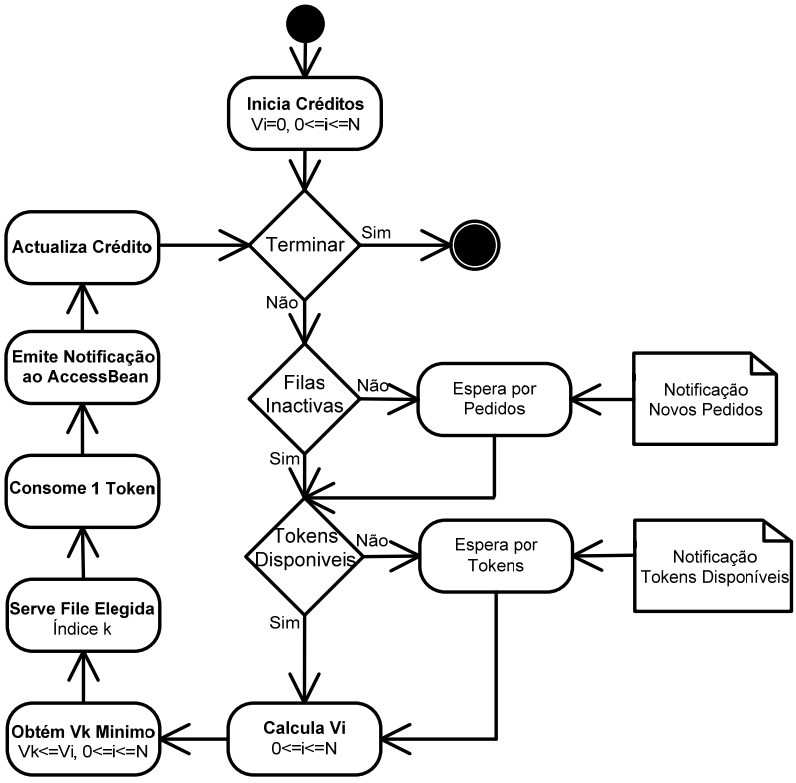

Figura 10. Algoritmo de escalonamento CBFQ

Ambos os algoritmos apresentados nas figuras 9 e 10 atuam de acordo com o funcionamento do escalonador descrito na secção 3.4.4.

Assim, o *Scheduler* interage com os restantes módulos, de modo que a taxa de admissão seja coordenada pelo *IncBucket* e a inatividade das filas de espera permita a suspensão do processo de escalonamento e que seja efectuada a notificação ao *AccessBean* associado a cada pedido.

3.5.2 Detalhes de Implementação

A implementação da solução proposta foi realizada em duas fases. Numa primeira fase a solução foi implementada de forma modular, para simplificar a sua integração no UIF. Nesta fase a solução foi desenvolvida em *Java SE*, sendo executada localmente. Foi simulada a interacção do módulo desenvolvido com o módulo UIF para validação do funcionamento antes da integração.

Numa segunda fase, o módulo da solução foi integrado no UIF, através da invocação de um método que executa o controlo de admissão de cada pedido.

Nesta fase, a solução foi enriquecida com funcionalidades do *Java EE*, tirando proveito das vantagens que o UIF possui ao ser executado por um servidor aplicacional. Foram adicionadas mensagens de *logs*, criados e interpretados novos códigos de erro, a base de dados foi estendida e o acesso à mesma recorreu a *EntityBeans* e à capacidade de cache do servidor aplicacional.

São brevemente descritas de seguida algumas características da solução desenvolvida.

3.5.2.1 Sintaxe dos Pedidos

O mecanismo de admissão irá actuar sobre os pedidos de requisição de serviço efectuados às *Interfaces* do UIF. Estes pedidos possuem um perfil *online*, devendo ser atendidos o mais brevemente possível.

O formato dos pedidos suportados pelo UIF pode ser HTTP URL *Encoded* ou XML. No caso de ser URL *Encoded*, o pedido consiste num endereço URL que contém os campos necessários à interpretação do mesmo, sendo esta efectuada pela interface URL *Encoded*.

A sintaxe de um pedido http URL *Encoded* é a seguinte:

```
http://<host>/<url_base>/<serviço>?InSid=<sid>&inParam1=x&...&inParamN=n
```

Tabela 1. Sintaxe de um pedido URL Encoded do UIF

host – endereço do servidor;

url_base – contexto da aplicação;

serviço – serviço que se pretende executar;

sid – parâmetro que identifica o sistema externo;

inParamX – parâmetros do serviço, dependendo da configuração do serviço.

A interface http XML é semelhante à http URL Encoded, requerendo os mesmos campos mas com uma sintaxe distinta, em XML.

A sintaxe de um pedido XML é a seguinte:

```
<requestservice="<serviço>" sid="<sid>" timestamp="2009-06-07T13:47:00">
    <record>
        <fieldname="<inParam1>" value="String"/>
        <fieldname="< inParam2>" value="32"/>
        …
        <fieldname="< inParamN>" value="4.3"/>
    </record>
</request>
```

Tabela 2. Sintaxe de um pedido XML do UIF

Seja qual for a interface, o formato das respostas devolvidas aos pedidos suportados pelo UIF é sempre em XML.

Apresenta-se de seguida um exemplo de uma resposta do UIF:

```
<?xmlversion="1.0" encoding="ISO-8859-1"?>
<response success=”yes”>
      <body>
            <record>
                  <fieldname="<outParam1>"  value=”exemplo” />
                  <fieldname="<outParam2>"  value=”QWE” />
                    …
                   <fieldname="<outParamN>"  value=”X” />
            </record>
      </body>
</response>
```

Tabela 3. Sintaxe de uma resposta do UIF

A ordem devolvida dos parâmetros de saída é a definida nos parâmetros da primitiva de invocação do serviço.

3.5.2.2 *Logs*

As operações que decorrem ao longo do processo de admissão são registadas em *logs*, recorrendo à *Framework commons-logging*. Podem deste modo ser utilizados uma variedade de *loggers*. A quantidade de *logs* gerados depende do modo como o serviço de *logs* se encontra configurado, segundo os níveis hierárquicos: *debug*, *info*, *warn*, *error*, e *fatal*.

Os *logs* do tipo *info* são gerados apenas para registar o resultado do controlo de admissão de cada pedido. Os *logs* do tipo *debug* encontram-se distribuídos por todo o código como notificação de operação. Este nível só deve assim ser definido em casos onde é necessária a verificação de funcionamento. Deste modo evita-se que os ficheiros de *logs* adquiram tamanhos insustentáveis.

Os *logs* do tipo *error* são gerados em situações de erro não fatais para a continuação de funcionamento do módulo. No caso de um erro comprometer o funcionamento do processo de admissão é gerado um *log* do tipo *fatal*.

3.5.2.3 Códigos e Mensagens

Em situações em que o processo de admissão de um pedido falha, é devolvida uma resposta à entidade externa que invocou o pedido.

A reposta devolvida é gerada pela interface que recebeu o pedido, contudo o *Engine* é apenas responsável pela devolução de um código de erro à Interface respectiva. Assim, as *Interfaces* efectuam o mapeamento de eventuais códigos de erro com a mensagem que deve ser devolvida à entidade externa que invocou o pedido. Deste modo, em cada interface existente no modulo UIF existe um ficheiro responsável por mapear o código de erro com a resposta que deve ser devolvida pela interface.

No âmbito do controlo de admissão implementado são definidos dois códigos de erro: o código devolvido no caso de um pedido exceder o tempo de espera definido para uma resposta de admissão, e o código devolvido no caso de as filas de espera se encontrarem cheias, não sendo possível a execução do pedido em ambos os casos.

Os códigos são definidos estaticamente de forma a serem utilizados de forma legível, como se apresentado na Tabela 4.

```
public static final int AC_SCHEDULING_TIMEOUT=140013;
public static final int AC_FULL_QUEUES=140014;
```

Tabela 4. Definição estática dos códigos de erro.

No ficheiro existente para cada interface encontra-se o mapeamento entre os códigos utilizados e a mensagem devolvida como é apresentado na Tabela 5.

```
140013=ACESSO NAO AUTORIZADO: ADMISSION CONTROL TIMEOUT
140014=ACESSO NÃO AUTORIZADO: ADMISSION CONTROL FULL QUEUES
```

Tabela 5. Mensagens mapeadas com os códigos de erro

3.5.2.4 Base de Dados

O módulo UIF possui uma base de dados para efectuar a parametrização dos seus serviços e interfaces, assim como definir as permissões que cada entidade externa possui.

No funcionamento do módulo UIF, quaisquer parâmetros de configuração sujeitos a alterações não devem ser aplicados no código mas sim armazenados na base de dados. Assim, a base de dados do UIF foi estendida com novas tabelas contendo os parâmetros de configuração necessários à solução de controlo de admissão.

As tabelas criadas relativas ao mecanismo de controlo de admissão encontram-se representadas no diagrama de relacionamento ilustrado na Figura 11:

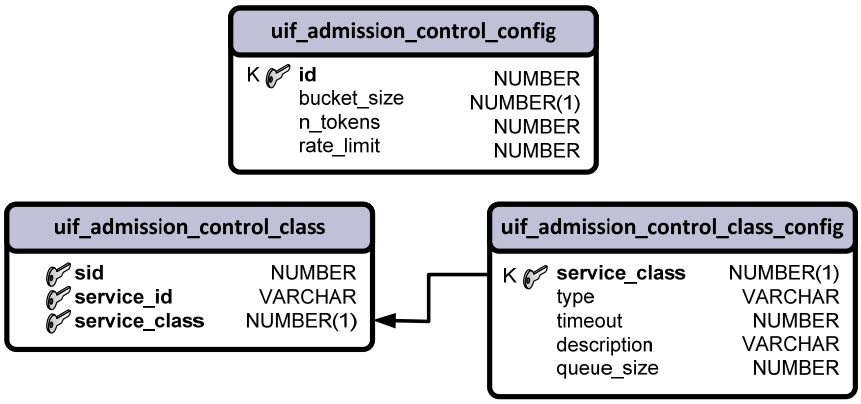

Figura 11. Tabelas de configuração do controlo de admissão

Numa perspectiva geral, os campos contidos nas tabelas permitem a afinação do controlo de ritmo e da diferenciação, consoante o comportamento pretendido. É apresentado de seguida o propósito e o conteúdo de cada uma das tabelas da Figura 11.

- **uif_admission_control_class**

Esta tabela indica qual a classe de serviço associada a cada tuplo (identificação de serviço, identificação da sessão). É este o método de classificação dos pedidos de requisição de serviço que são submetidos ao mecanismo de controlo de admissão, sendo feito o mapeamento de cada serviço com a classe de serviço respectiva.

O campo *sid* representa a identificação da sessão e o campo *service_id* representa a identificação do serviço. O tuplo constituído por estes dois campos constitui a chave primária da tabela, visto que existe interesse em distinguir as classes de serviço que os vários serviços possuem em relação à sessão segundo a qual estes são invocados.

O campo *service_class* é o identificador da classe de serviço, cujas configurações se encontram na tabela *uif_admission_control_class_config*, razão pela qual este campo se encontra como chave estrangeira.

- **uif_admission_control_class_config**

Esta tabela contém as configurações associadas a cada classe de serviço.

O campo *service_class* representa o identificador da classe de serviço, sendo a chave primária desta tabela. O campo *type* representa a designação da classe de serviço respectiva, visto que o campo *service_class* apenas identifica numericamente a classe. O campo *timeout* indica qual o tempo de espera que cada pedido da classe de serviço respectiva está disposto a aguardar para que o processo de controlo de admissão seja concluído. O campo *description* pode apresentar uma descrição do perfil da classe de serviço. O campo *queue_size* representa o tamanho da fila de espera associada à classe de serviço respectiva.

- **uif_admission_control_config**

Esta tabela contém as configurações que permitem afinar o comportamento do mecanismo de controlo de admissão. Um conjunto de parâmetros de configuração é armazenado numa só linha da tabela, podendo existir, no entanto, várias configurações. Deste modo o campo *id* representa a identificação do conjunto de configurações. O campo *bucket_size* representa o tamanho do Bucket utilizado no controlo de ritmo, definindo o tamanho das rajadas de pedidos que podem ser admitidos. Este campo possui uma restrição que garante que o seu valor é obrigatoriamente inferior ao campo *n_tokens*, pois

seria incongruente acrescentar ao *bucket* uma quantidade de *tokens* que este não suporte. O campo *n_tokens* representa o número de *tokens* que são adicionados ao *bucket* em cada operação de incrementação. Este limite é definido no campo *rhythm_limit*.

Com a adição das tabelas apresentadas a base de dados do UIF fica pronta a fornecer todos os dados de configuração que o mecanismo de controlo necessita, de um modo independente do resto do mecanismo de admissão.

3.5.2.5 Acesso à Base de Dados

No acesso à informação de configuração armazenada na base de dados, é evitada a dependência de APIs de acesso a base de dados, assim como de *drivers* dependentes da tecnologia de acesso utilizada. Tais APIs não actuam de um modo uniforme no acesso a sistemas díspares, provocando ainda a perda de portabilidade [27].

Como o módulo UIF é executado por um servidor aplicacional *JBoss*, a informação contida em base de dados é carregada através dos critérios especificados em *Entity Beans*. Cada *Entity Bean* encontra-se relacionado a uma tabela da base de dados. Assim, foram criados três *Entity Beans* que possuem métodos para obter e armazenar provisoriamente os dados de configuração da solução de controlo de admissão existentes nas tabelas da base de dados.

A informação obtida através dos *Entity Beans* é armazenada na cache do servidor aplicacional por um determinado período de tempo. Este facto permite que a classificação seja efectuada mais rapidamente, sem acesso constante à base de dados. Qualquer alteração ao nível da base de dados, pode ser aplicada à interface sem que esta necessite de ser reiniciada, bastando para isso limpar os dados armazenados na *cache* do servidor aplicacional.

Uma vez obtidos os dados de configuração, estes são encapsulados em objectos segundo os princípios de *Data Access Object* (DAO) [27]. Um objecto DAO actua como intermediário entre a base de dados e os componentes que necessitam de aceder a esses dados. Deste modo, os componentes que recorrem ao DAO não necessitam de considerar os métodos de acesso à base de dados existentes nos *Entity Beans* implementados. Bastando a instanciação de um objecto DAO para efectuar a invocação desses métodos.

A interface que o DAO fornece não é alterada na ocorrência de uma alteração no método de acesso à BD. Deste modo é alcançada uma abstracção e encapsulamento de todo o acesso à base de dados, facilitando uma eventual migração de tecnologia de base de dados.

3.5.2.6 Modularidade

A solução implementada possui uma arquitectura modular que pode ser aplicada não só em plataformas de prestação de serviços, como em outros tipos de ambientes desenvolvidos em *Java* que necessitem de controlo de admissão.

No UIF, a interacção com a funcionalidade de controlo de ritmo ocorre ao nível do *AccessBean* que é instanciado para cada pedido. Em outros ambientes esta invocação será efectuada de um modo diferente, sendo necessária a modificação dos parâmetros de entrada que permitiram identificar a classe de serviço do pedido.

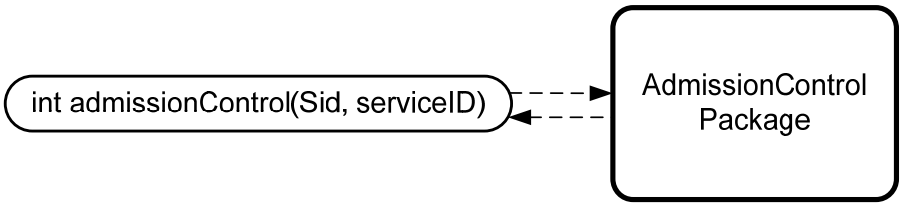

Figura 12. Invocação de controlo de admissão

Do ponto de vista do *AccessBean*, o controlo de admissão é efectuado pela simples invocação do método *admissionControl(Sid, service id)*, sendo devolvido um inteiro cujo valor pode assumir os seguintes significados:

- 2: pedido admitido com recurso a fila de espera.
- 1: pedido admitido directamente.
- 0: pedido rejeitado por timeout.
- -1: pedido rejeitado por fila lotada.
- -2: pedido rejeitado por classe de serviço desconhecida.
- -3: pedido rejeitado por erro desconhecido.

A necessidade de invocar o método com os parâmetros de entrada *Sid* e *service id* resulta do facto que a diferenciação dos perfis das classes de serviço é efectuada em função deste tuplo.

Não obstante, os campos de informação obtidos através da consulta à base de dados, relativos às classes de serviço, também têm de ser modificados para o novo cenário, sendo necessária a edição dos *Entity Beans* que obtêm essa informação, assim como a alteração dos DAOs que a armazenam.

Como a referência de cada objecto sujeito a controlo de admissão é constituída pela informação contida no DAO, uma vez obtida essa informação pode ser efectuado controlo de admissão independentemente dos elementos que se pretende controlar.

Deste modo, o mecanismo pode ser facilmente integrado em outros módulos *Java*, sendo invocado no ponto em que os objectos a coordenar e controlar são recebidos no módulo ou plataforma.

3.5.2.7 Instalação do UIF

Para se efectuar uma instalação do UIF têm de ser realizadas as seguintes operações:

- compilação do módulo e implantação dos arquivos resultantes num servidor aplicacional;
- criação da base de dados do UIF, com as tabelas dedicadas ao processo de controlo de admissão;
- parametrização dos serviços, respectivas classes de serviço e configurações do módulo;
- configuração dos *Data Sources* do servidor aplicacional.

O módulo UIF é compilado através do *Ant*, sendo adicionalmente geradas as classes *Local* e *LocalHome* relativas a cada *Entity Bean*. Como resultado da compilação do UIF, o *Ant* gera ficheiros de extensão EAR. Estes ficheiros são depositados no servidor aplicacional, sendo este responsável pela sua execução.

Durante o processo de compilação e execução do UIF, é necessária a conectividade à base de dados do mesmo, tendo de existir um *Data Source* devidamente configurado para acesso ao *schema* onde a base de dados se encontra instalada.

Na base de dados deverão ser parametrizados os serviços desejados, assim como as classes de serviço respectivas. Podem ser utilizados os *scripts* de teste desenvolvidos no âmbito do controlo de admissão do UIF, para de um modo fácil preparar a base de dados para um cenário de testes completo.

Quando é emitido o primeiro pedido à primeira instanciação do UIF, a estrutura de dados necessária ao controlo de admissão também é instanciada e são arrancados o *IncBucket* e o *Scheduler*, passando o sistema a estar totalmente activo.

Os módulos *IncBucket*, *Scheduler* e a estrutura de dados devem ser instanciados uma só vez. Assim, estes módulos são implementados pelo padrão *singleton*, para que exista apenas um objecto associado a cada módulo respectivamente. Desta forma é mantido um único objecto referente a cada módulo, sendo este devolvido sempre que for efectuada uma tentativa de instanciação dos módulos.

3.6 Conclusões

Este capítulo apresentou uma solução proposta para colmatar a carência de controlo de admissão no módulo UIF, segundo os objectivos apresentados na secção 1.2.

Foi especificado o processo de admissão sofrido por cada pedido, assim como a arquitectura que sustenta esse processo.

Adicionalmente, foram apresentadas algumas características especificas, relativas à implementação da solução.

A solução implementada foi submetida a testes que permitem demonstrar o seu correcto funcionamento. A especificação e análise dos resultados desses testes são efectuadas no capítulo 4.

Capítulo 4

Cenários de Utilização e Testes

Neste capítulo são apresentados os cenários e procedimentos utilizados nos testes efectuados sobre a nova versão do módulo UIF, suportando este a solução de controlo de admissão apresentada no capítulo 3.

São apresentados os resultados dos testes efectuados, sendo estes analisados e comparados em função das disciplinas de escalonamento e configurações realizadas.

A realização de testes é importante para a verificação de funcionamento dos vários aspectos que a solução de controlo de admissão sustenta, como o controlo da taxa de admissão, diferenciação do escalonamento efectuado entre as filas de espera e o nível de equidade prestado entre as classes de serviço.

Foram desenvolvidos dois tipos de testes sobre o mecanismo de controlo de admissão: testes unitários e testes de carga.

Os testes unitários procuram garantir o correcto funcionamento de cada método implementado. Estes testes verificam se o resultado retornado por cada método é o esperado, sob diversas situações. Deste modo, se a implementação de um método for alterada, e o seu propósito se mantiver, são fornecidas garantias que o método mantém o seu funcionamento como suposto.

Os testes de carga procuram submeter o módulo UIF a situações de elevada carga, verificando o comportamento assumido pelo mesmo. Estes testes exploram as várias vertentes do mecanismo de admissão, como a verificação do controlo da taxa de admissão, da largura de banda atribuída a cada classe de serviço ao longo do tempo, o comportamento face à variabilidade das condições, e a percentagem de pedidos rejeitados segundo as políticas de escalonamento MCF e CBFQ.

4.1 Testes Unitários

Os testes unitários foram efectuados sobre um projecto *Maven* [29], recorrendo aos métodos da biblioteca *JUnit* [30], antes da integração da solução no módulo UIF, mantendo o seu desenvolvimento independente do código da solução desenvolvida.

Os testes foram direccionados a todas as classes definidas na solução desenvolvida. Para cada classe existe uma classe de testes associada.

Uma classe de testes contém testes unitários cujo propósito é garantir a validade lógica dos resultados retornados para cada método, comparando o resultado esperado com o realmente recebido, garantindo assim a preservação da sua lógica em função dos vários parâmetros de entrada recebidos.

No caso do código do método sofrer optimizações, se os testes unitários forem executados com sucesso, então a validade do método mantém-se, caso contrário irá ser facilmente detectado o erro em tempo de compilação.

Assim, a robustez de cada método implementado é garantida através da invocação parâmetros de entrada válidos e inválidos, e verificação dos valores de retorno esperados para os diversos cenários testados.

Os testes são executados em cada compilação do projecto *Maven*. Deste modo, esta metodologia conduz a que os métodos a testar sejam aprimorados ao longo do seu processo de desenvolvimento.

4.2 Testes de Controlo da Taxa de Admissão

Os testes efectuados nesta secção pretendem demonstrar o comportamento da solução de controlo de admissão, mais especificamente do mecanismo de controlo da taxa de admissão ao longo do tempo. Neste âmbito, os testes foram efectuados segundo várias configurações do mecanismo de controlo da taxa de admissão.

4.2.1 Objectivos do Teste

Mais especificamente, os testes efectuados nesta secção pretendem avaliar o mecanismo de controlo da taxa de admissão com a perspectiva de:

- verificar que a taxa máxima de admissão nunca é superada;
- avaliar o comportamento da solução face à emissão de diferentes taxas de emissão de pedidos, e segundo configurações distintas;
- garantir que todos os pedidos gerados são satisfeitos.

4.2.1 Cenários de Teste

Os testes apresentados no âmbito da verificação do controlo da taxa de admissão foram realizados num projecto *Maven* desenvolvido antes da solução ter sido integrada no UIF.

Deste modo foi possível gerar dados estatísticos mais detalhados do que os obtidos nos testes realizados ao módulo UIF. É possível distinguir o número de pedidos admitidos directamente ou através da fila de espera, assim como o número de pedidos rejeitados por *timeout*. Estes dados são analisados e reiniciados todos em cada segundo.

Nestes testes não é examinada a capacidade de diferenciação, sendo o processo de escalonamento efectuado sobre uma só fila de espera.

Em detalhe, as configurações efectuadas neste teste resumem-se a:

- instanciação de apenas uma fila de espera;
- limitação da taxa máxima de admissão a 50 pedidos por segundo;
- emissão de 500 pedidos em testes com várias taxas de emissão constantes e uma taxa de emissão variável;
- limitação do tempo de espera, por uma notificação de admissão, de 4 segundos;
- limitação do tamanho máximo das filas de modo que os pedidos emitidos nunca provoquem a sua lotação, para que este factor não influencie os resultados.

A taxa de emissão dos pedidos no módulo foi efectuada segundo os vários perfis:

- taxa de emissão de 100 pedidos por segundo, para simulação da chegada de pedidos a um ritmo bastante superior à taxa máxima de admissão;
- taxa de emissão de 55 pedidos por segundo, para simulação da chegada de pedidos a um ritmo um pouco superior à taxa máxima de admissão;
- taxa de emissão de 50 pedidos por segundo, para simulação da chegada de pedidos ao mesmo ritmo que a taxa máxima de admissão;
- taxa de emissão variável de chegada de pedidos por segundo, obtida aleatoriamente, ilustrada na Figura 13.

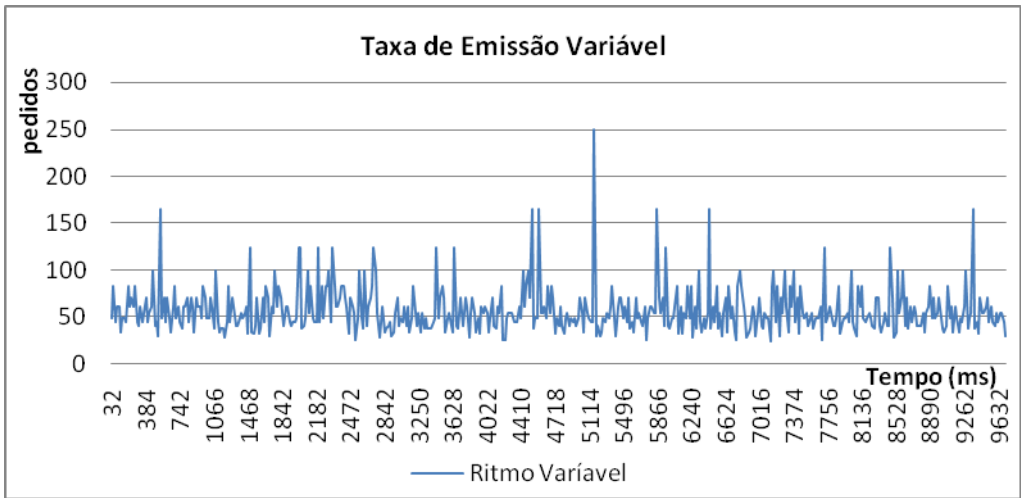

Figura 13. Taxa de emissão variável

Na taxa de emissão ilustrada na Figura 13, o intervalo médio de tempo decorrido entre a emissão de cada pedido situa-se nos 20 milissegundos, o que equivale a uma taxa de emissão média de 50 pedidos por segundo.

Para cada uma das taxas de emissão, os testes são realizados várias vezes, segundo configurações distintas do mecanismo de controlo da taxa de admissão:

- sem possibilidade de rajadas de pedidos, incrementando o contador com uma unidade entre períodos de 20 milissegundos, legendado nos gráficos como “B1T1” (referência de *bucket* = 1 e *token* = 1);

- com permissão de rajadas de 10 pedidos, incrementando o contador com 1 unidade entre períodos de 20 milissegundos, legendado nos gráficos como "B10T1" (referência de *bucket* = 10 e *token* = 1);
- Com permissão de rajadas de 10 pedidos, incrementando o contador com 5 unidades entre períodos de 100 milissegundos, legendado nos gráficos como "B10T5" (referência de *bucket* = 10 e *token* = 5);
- Com permissão de rajadas de 20 pedidos, incrementando o contador com 1 unidade entre períodos de 20 milissegundos, legendado nos gráficos como "B20T1" (referência de *bucket* = 20 e *token* = 1);
- Com permissão de rajadas de 20 pedidos, incrementando o contador com 5 unidades entre períodos de 100 milissegundos, legendado nos gráficos como "B20T5" (referência de *bucket* = 20 e *token* = 1).

Durante a execução de cada um dos testes, apresentados nesta secção, são recolhidos periodicamente dados estatísticos que permitem contabilizar a quantidade de pedidos admitidos directamente, através de uma fila de espera, ou rejeitados no caso de excesso de tempo de espera por admissão. A periodicidade definida para este processo de monitoria é de segundo a segundo.

4.2.2 Resultados

São apresentados nesta secção os resultados dos testes efectuados de acordo com os perfis de configuração anteriormente referidos.

4.2.2.1 Taxa de Emissão de 100 Pedidos por Segundo

Os resultados obtidos para uma taxa de emissão de 100 pedidos por segundo apresentam-se nos gráficos da Figura 14.

Como a taxa de emissão de 100 pedidos por segundo é bastante superior à taxa máxima de admissão permitida, de 50 pedidos por segundo, os pedidos são encaminhados quase exclusivamente através da fila de espera. Alguns pedidos chegam a perder a validade, visto

que a taxa de serviço da fila de espera é metade da taxa de emissão dos pedidos, conduzindo ao aumento do atraso sofrido pelos pedidos à medida que estes se acumulam na fila.

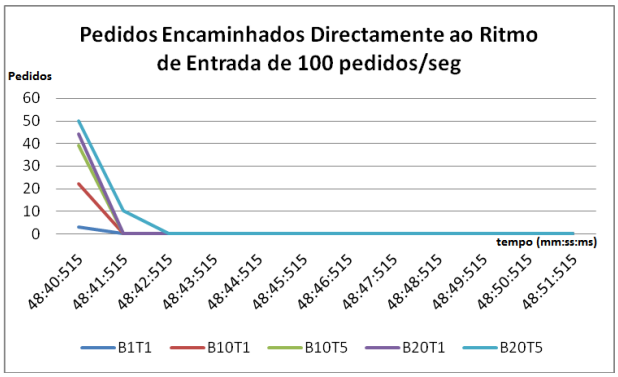

Figura 14. Admissão directa para taxa de emissão de 100 pedidos por segundo

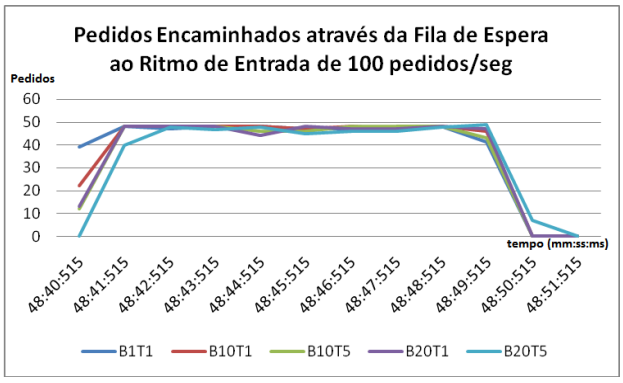

Figura 15. Admissão por fila de espera para taxa de emissão de 100 pedidos por segundo

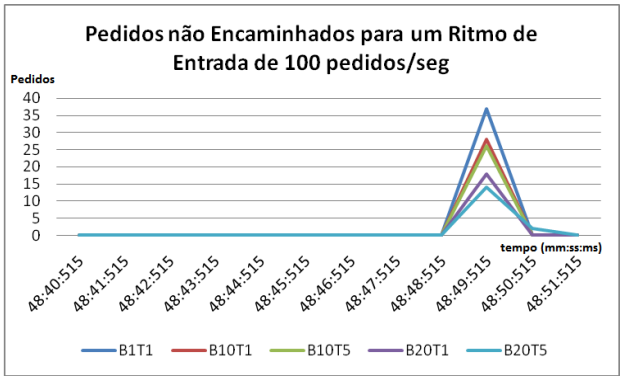

Figura 16. Rejeição para taxa de emissão de 100 pedidos por segundo

Verifica-se que quanto maior o *bucket*, menor quantidade de pedidos são rejeitados. Ou seja, a quantidade de pedidos rejeitados é superior nas configurações em que não são permitidas rajadas de pedidos. O aumento de rejeições é também influenciado pelo número de pedidos servidos directamente nos instantes iniciais, pois quanto mais pedidos

forem servidos sem recurso à fila de espera, menos pedidos são rejeitados quando ocorre *timeout*.

4.2.2.2 Taxa de Emissão de 55 Pedidos por Segundo

Os resultados obtidos para uma taxa de emissão de 55 pedidos por segundo apresentam-se nos gráficos da Figura 15.

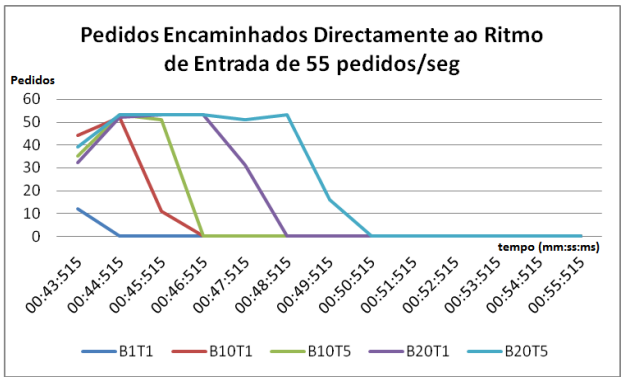

Figura 17. Admissão directa para taxa de emissão de 55 pedidos por segundo

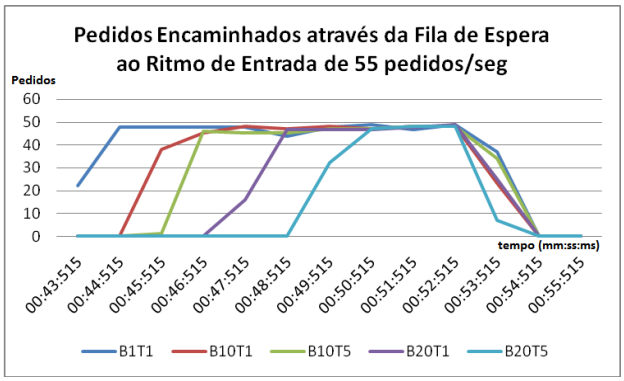

Figura 18. Admissão por fila de espera para taxa de emissão de 55 pedidos por segundo

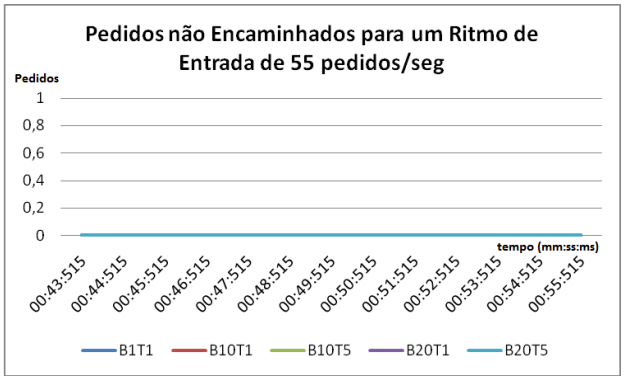

Figura 19. Rejeição para taxa de emissão de 55 pedidos por segundo

Neste cenário a taxa de emissão de 55 pedidos por segundo encontra-se bastante próxima da taxa máxima de admissão permitida, de 50 pedidos por segundo.

Nas configurações que permitem o serviço directo de rajadas de pedidos verifica-se um aumento substancial do número de pedidos admitidos directamente, sendo este comportamento mais evidente quanto maior for o tamanho do *bucket*. Contudo, na configuração em que o *bucket* é limitado a uma unidade, a admissão de pedidos é efectuada quase exclusivamente através da fila de espera, como esperado. Como o ritmo é constante e superior ao limite, uma vez que um pedido entre na fila de espera, mais nenhum pedido é encaminhado directamente até ao fim do teste.

Observa-se que quanto mais acentuada for a diferença entre a taxa de emissão dos pedidos e a taxa de admissão máxima, mais rapidamente os *token*s são consumidos, esgotando a capacidade de admitir pedidos directamente.

Confirma-se assim que o aumento do limite do *bucket* conduz a um esgotamento de *token*s mais tardio, aumentando a capacidade de encaminhar pedidos directamente.

4.2.2.3 Taxa de Emissão de 50 Pedidos por Segundo

Os resultados obtidos para uma taxa de emissão de 50 pedidos por segundo apresentam-se nos gráficos da Figura 16.

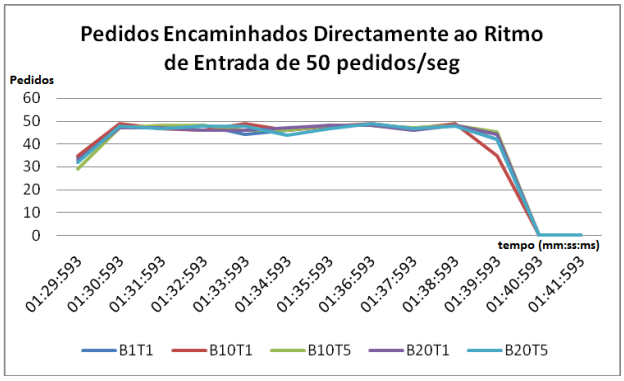

Figura 20. Admissão directa para taxa de emissão de 50 pedidos por segundo

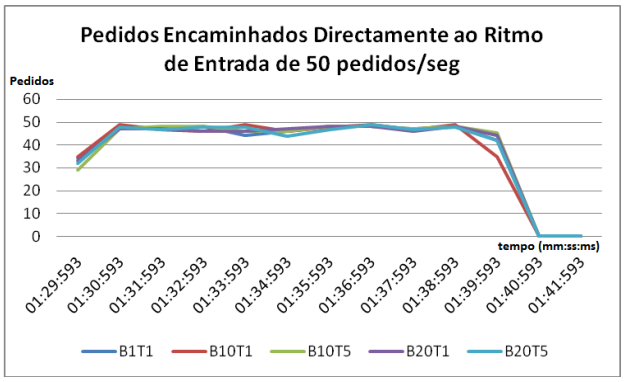

Figura 21. Admissão por fila de espera para taxa de emissão de 50 pedidos por segundo

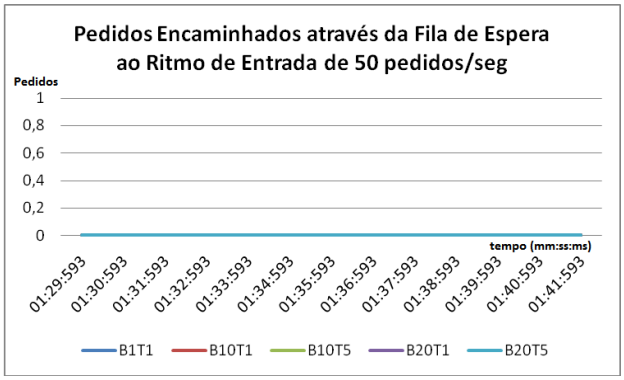

Figura 22. Rejeição para taxa de emissão de 50 pedidos por segundo

Como a taxa de emissão de 50 pedidos por segundo é igual à taxa máxima de admissão permitida, os pedidos são encaminhados directamente, na sua totalidade.

Tal sucede pelo facto da taxa de emissão não provocar um consumo de *token*s que ultrapasse o tamanho do *bucket*, i.e., o número de unidades do *bucket* é sempre suficiente para garantir a admissão directa dos pedidos.

Nestas circunstâncias, não é efectuada admissão com recurso às filas de espera, visto que a taxa máxima de admissão nunca é superada.

Como a uma taxa de 50 pedidos por segundo nenhum pedido é encaminhado através das filas de espera, o mesmo comportamento verifica-se necessariamente para taxas de emissão inferiores.

4.2.2.4 Taxa Variável de Emissão

Os resultados obtidos para uma taxa de emissão variável apresentam-se nos gráficos da Figura 17. Neste cenário, a taxa média de emissão de pedidos é de 50 pedidos por segundo.

Como a taxa de emissão é variável, a taxa máxima de admissão permitida é frequentemente superada. Este acontecimento devesse ao facto da taxa variável de emissão conter ritmos bastante elevados em relação à taxa máxima de admissão.

Na configuração onde a limitação da dimensão do *bucket* é de um *token* os pedidos são encaminhados quase exclusivamente através da fila de espera, nunca chegando esta a ficar vazia para dar lugar ao processo de admissão directa. Se tal ocorresse, a admissão directa seria efectuada até que o *bucket* ficasse novamente vazio.

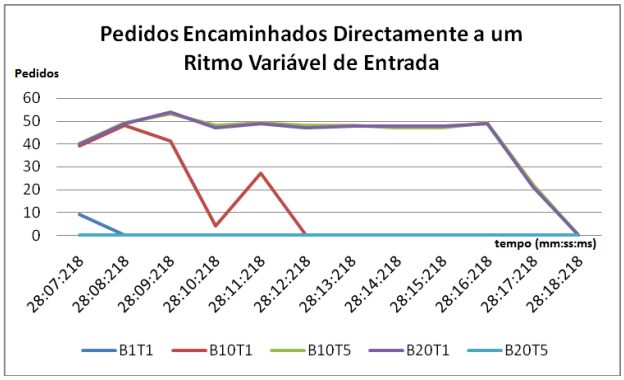

Figura 23. Admissão directa para taxa de emissão variável

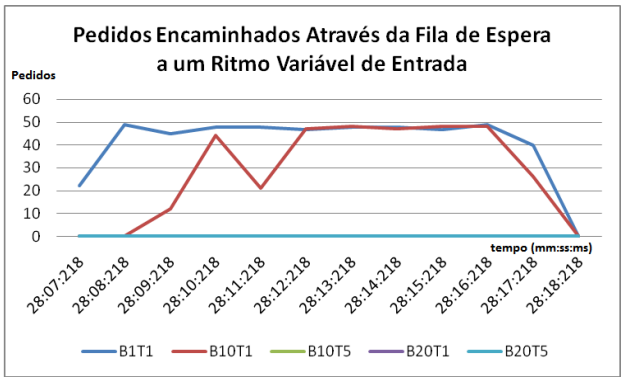

Figura 24. Admissão por fila de espera para taxa de emissão variável

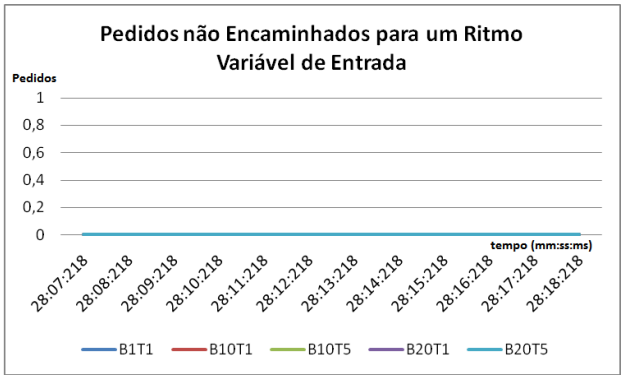

Figura 25. Rejeição para taxa de emissão variável

Nas configurações onde a limitação da dimensão do *bucket* é superior a 10 *tokens*, a admissão dos pedidos deixa de recorrer à fila de espera, contrariamente ao verificado nas configurações anteriores.

Contudo, apesar das configurações que permitem a admissão de rajadas de pedidos induzirem menores níveis de complexidade que as configurações que não as permitem, estas também provocam a perda de rigidez e granularidade no controlo e modulação da taxa de admissão.

4.2.2.5 Atrasos de Admissão

Apresenta-se de seguida o conjunto de tempos de atraso de admissão sofrido pelos pedidos submetidos aos testes cujos resultados foram apresentados nas subsecções anteriores.

Os tempos de atraso de admissão são obtidos pelo tempo decorrido desde o instante em que cada pedido é recebido até ao instante em que o mesmo é admitido ou rejeitado. Assim, para cada taxa de emissão de pedidos foram obtidos os seguintes atrasos:

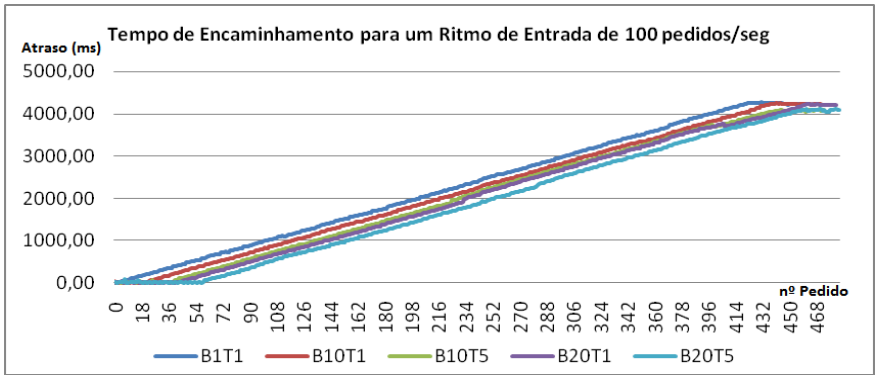

Figura 26. Atraso de admissão para taxa de 100 pedidos por segundo

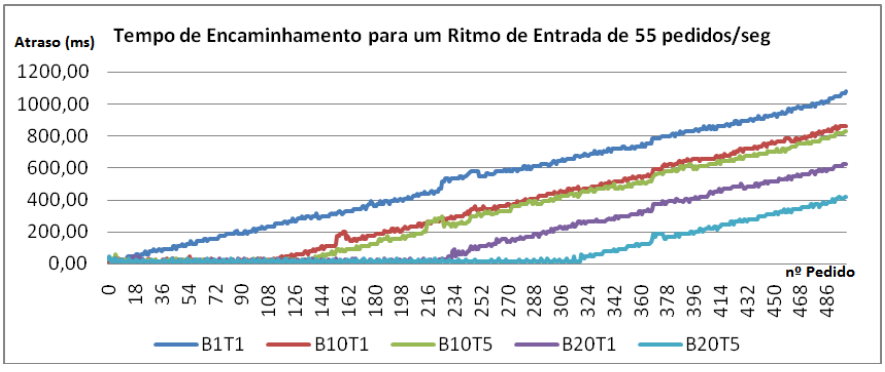

Figura 27. Atraso de admissão para taxa de 55 pedidos por segundo

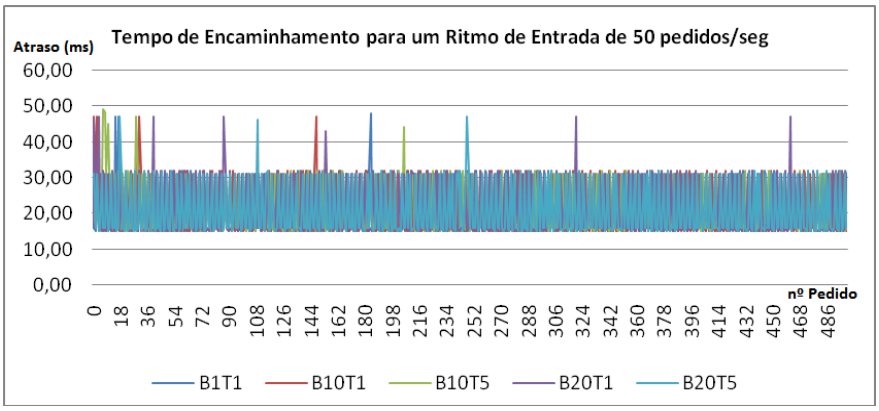

Figura 28. Atraso de admissão para taxa de 50 pedidos por segundo

Observa-se que nos casos em que a taxa máxima de admissão não é superada, a duração do processo de admissão dos pedidos é predominantemente constante, como esperado, pois face à taxa constante de emissão dos pedidos ser inferior à taxa máxima de admissão, os pedidos nunca chegam a ser encaminhados através das filas de espera.

Para taxas de emissão constante superiores à taxa máxima de admissão, verificam-se tempos de atraso constantes enquanto os pedidos são admitidos directamente, e uma tendência crescente linear a partir do momento que os pedidos passam a ser admitidos através das filas de espera.

Podemos observar que quanto maior for o *bucket*, i.e., quanto maior o número de pedidos permitidos para admissão directa de uma rajada, e quanto maior o número de *token*s incrementados periodicamente no contador, maior será a flexibilidade da admissão. Não obstante, a diminuição da complexidade de execução, esse cenário reflecte-se na perda de rigidez e granularidade do controlo da taxa de admissão.

Como as taxas de emissão são constantes, à medida que o ritmo aumenta o número de elementos admitidos através da fila de espera também aumenta, o que conduz à diminuição da componente constante e aumento da componente linear dos tempos de processamento. O comportamento de atraso crescente estabiliza quando o tempo de processamento do pedido atinge o limite tolerado por uma resposta de admissão, sendo assim removido da fila de espera.

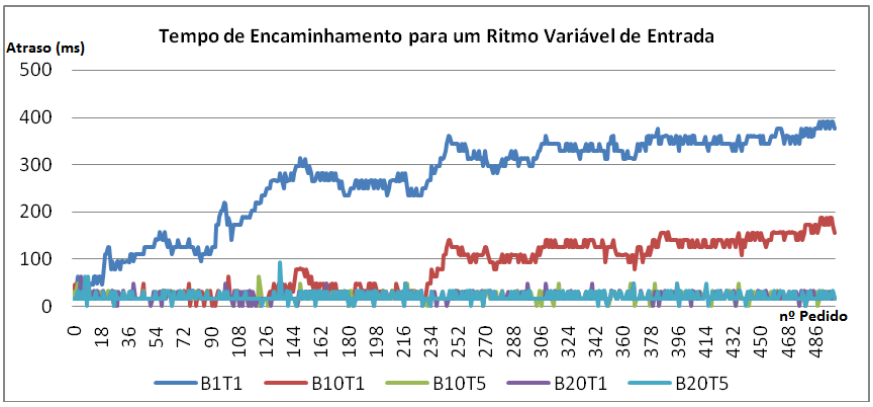

Figura 29. Resultados para taxa variável de emissão de pedidos

Para uma taxa variável de emissão, o tempo de serviço dos pedidos é constante para as configurações que proporcionam maior flexibilidade na limitação da taxa de admissão, pois nesses casos os pedidos são admitidos directamente. Em situações em que é necessária a admissão dos pedidos recorrendo à utilização de filas de espera verifica-se um aumento do atraso, que varia em função do nível de carga da fila de espera ao longo do tempo.

4.2.3 Conclusões

O âmbito destes testes consistiu na análise do comportamento que a funcionalidade de controlo da taxa de admissão, integrada no mecanismo de controlo de admissão, assume face às várias configurações efectuadas.

Dos resultados apresentados pode-se concluir que quanto maior for a dimensão do *bucket* maior é a tolerância de admissão directa dos pedidos, aumentando a margem de aceitação a rajadas de pedidos com a dimensão especificada no *bucket*.

A configuração da funcionalidade de controlo da taxa de admissão deve basear-se em manter o *bucket* com *tokens* suficientes para admitir uma determinada quantidade de pedidos directamente, de modo que a quantidade média de pedidos recebidos não tenda a impossibilitar a capacidade de admissão directa.

O valor do parâmetro que define o número de *tokens* a adicionar ao *bucket* em cada iteração também define a flexibilidade da solução. Um maior valor do *bucket* permite aumentar o número de pedidos admitidos directamente, diminuindo consequentemente o

número de pedidos encaminhados através da fila de espera, que seriam sujeitos a atrasos resultantes do processo de complexidade linear de inserção e remoção da fila de espera.

Não existe uma configuração ideal para a solução que abranja todos os casos pois a distribuição do ritmo de chegada dos pedidos é, muitas vezes, indeterminada. Contudo, caso fosse conhecida a distribuição dos ritmos de chegada típicos na interface do UIF, os parâmetros da solução podiam ser configurados com valores que optimizassem o processo de admissão em função dessa distribuição. Estudos de caracterização de tráfego são, neste contexto, de grande relevância.

4.3 Testes de Diferenciação

Os testes efectuados nesta secção pretendem demonstrar o comportamento da solução de controlo de admissão relativamente ao modo como o escalonamento dos pedidos é efectuado, sendo avaliado o nível de equidade das disciplinas de escalonamento implementadas.

De modo a demonstrar o funcionamento da solução são efectuados testes de carga através da submissão concorrente de um número configurável de pedidos fictícios ao UIF.

4.3.1 Objectivos dos Testes

Os testes efectuados nesta secção pretendem demonstrar o comportamento da solução de controlo de admissão nos seguintes aspectos:

- verificar que a largura de banda atribuída a cada classe de serviço é efectuada como suposto, segundo os princípios de alocação de largura de banda dos algoritmos MCF e CFBQ;
- verificar que o taxa de atendimento de todas as classes de serviços não excede o limite configurado;
- verificar o comportamento face à emissão de conjuntos de pedidos concorrentes entre si num curto período de tempo, de modo a submeter a solução a condições de carga elevada.

4.3.2 Cenário de Teste

Para atingir os objectivos destes testes, foi criada uma aplicação *Java* que inicia conexões HTTP que permitem a invocação de endereços URL a um serviço *web*, neste caso fornecido pelo servidor aplicacional *JBoss*. Foi utilizado o servidor aplicacional *JBoss* versão 4.0.5.GA. e o ambiente de execução *Java* versão 1.5.0.

A aplicação instancia três *pools* de *thread*s, uma para cada classe de serviço, para que os pedidos sejam efectuados em situações de concorrência. As respostas aos pedidos URL efectuados são visualizadas no *System.out*.

As configurações efectuadas para este teste encontram-se descritas nos tópicos apresentados de seguida.

1. Limitação da dimensão do *bucket* para suportar apenas um *token*.

O *bucket* foi limitado de modo que a admissão de rajadas de pedidos não seja permitida, com a finalidade de testar mais eficazmente a equidade que cada disciplina de escalonamento fornece. Assim, a solução é configurada para um controlo rígido da taxa de admissão.

2. Definição de três classes de serviço segundo dois perfis de prioridades: com prioridades idênticas e com prioridades de 50%, 30% e 20%.

As classes de serviço foram definidas para estabelecer dois cenários distintos. Um cenário consiste na definição de iguais prioridades por todas as filas de espera. Outro cenário consiste na definição de prioridades distintas. Neste cenário a classe 1 possui o nível de prioridade superior, de 50%, a classe 2 possui um nível de prioridade intermédio, de 30%, e a classe 3 possui um nível de prioridade inferior, de 20%.

3. Parametrização de três serviços fictícios em base de dados.

Foram parametrizados três serviços, exclusivamente para testes, denominados service_dummy_1, service_dummy_2 e service_dummy_3. Cada um dos serviços encontra-se associado a cada uma das classes de serviço definidas, classe 1, 2 e 3 respectivamente. Em cenários onde as classes possuam prioridades distintas, o serviço service_dummy_1 é

configurado para ter maior prioridade que o service_dummy_2, e este por sua vez para ter maior prioridade que o service_dummy_3.

A invocação dos serviços pode ser feita através dos URLs:

```
http://localhost:8080/sacs/service_dummy_1?InSid=sid1
http://localhost:8080/sacs/service_dummy_2?InSid=sid1
http://localhost:8080/sacs/service_dummy_3?InSid=sid1
```

Tabela 6. URLs de invocação dos serviços de teste

4. Invocação de um procedimento de teste.

A execução dos serviços fictícios criados recorre à invocação de um procedimento PL/SQL responsável por registar a classe de serviço e o instante em que cada pedido é executado. Este procedimento foi implementado num *package* pertencente à base de dados do UIF.

```
PROCEDURE service_dummy(p_class IN INTEGER) IS
  BEGIN
      INSERT INTO UIF_TEST_ADMISSION_CONTROL(v_class, v_timestamp)
             VALUES (p_class, systimestamp);
      COMMIT;
  EXCEPTION
      WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR (-20001,
                    p_class || ':$:' ||
                    SQLERRM, TRUE);
END service_dummy;
```

Tabela 7. Procedimento PL/SQL invocado nos serviços de teste

O procedimento apresentado efectua a inserção de uma entrada na tabela *uif_test_admission_control*.

5. Tratamento de dados contidos na tabela *uif_test_admission_control*.

Os resultados dos testes pretendem demonstrar a taxa de pedidos servida por unidade de tempo (segundos) para cada classe de serviço.

Os resultados de cada teste executado são armazenados na tabela *uif_test_admission_control*. Como cada entrada da tabela *uif_test_admission_control* possui a classe de serviço e o instante em que cada pedido é executado, foi criado o script SQL apresentado na Tabela 8. Este *script* permite gerar um histograma que apresenta o número de pedidos servidos por segundo, para cada classe de serviço respectivamente.

```
select  to_char(v_timestamp, 'dd-mm-yyyy hh24:mi:ss'),
        SUM(CASE WHEN v_class = 0 THEN 1 ELSE 0 END) AS "class0",
        SUM(CASE WHEN v_class = 1 THEN 1 ELSE 0 END) AS "class1",
        SUM(CASE WHEN v_class = 2 THEN 1 ELSE 0 END) AS "class2"
from uif_test_rate_control
group by to_char(v_timestamp, 'dd-mm-yyyy hh24:mi:ss')
order by to_char(v_timestamp, 'dd-mm-yyyy hh24:mi:ss');

select count(*) from uif_test_rate_control;

truncate table uif_test_rate_control;
```

Tabela 8. Script SQL gerador de histogramas

Com base nos dados do histograma são gerados os gráficos apresentados em cada teste efectuado.

6. Emissão concorrente de 1200 pedidos de cada classe.

O procedimento que cada teste executa consiste na emissão de 1200 pedidos por classe de serviço, com intervalos de 4 segundos por emissão de pedidos.

O teste realizado possui uma distribuição da emissão dos pedidos por cada classe ao longo do tempo, como ilustrado na Figura 24.

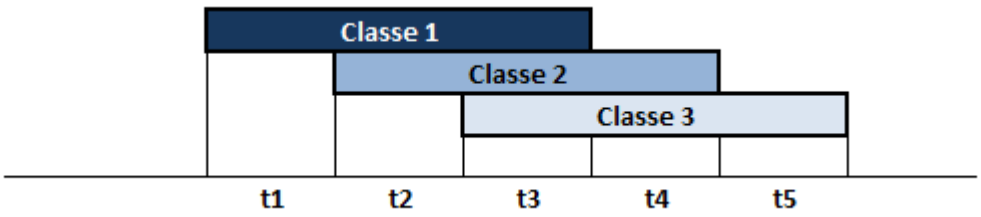

Figura 30. Esquema de emissão de pedidos em função do tempo

Deste modo:

- no intervalo t1 podemos testar o comportamento para apenas uma fila activa;
- no intervalo t2 testa-se a distribuição de largura de banda entre duas classes, concorrendo entre si;
- no intervalo t3 testa-se a distribuição de largura de banda entre as três classes de serviço;
- no intervalo t4 testa-se o comportamento do escalonador quando os pedidos da primeira classe de serviço deixam de ser emitidos;

- no intervalo t5 testa-se o comportamento do escalonador quando os pedidos da segunda classe de serviço deixam de ser emitidos.

O ritmo de admissão dos pedidos limitado a 100 pedidos por segundo. Isto leva a que, quando pedidos de classe 2 e 3 são submetidos, ainda existam pedidos de classe 1 a aguardar atendimento. Deste modo pode ser verificada a equidade da partilha de largura de banda ao longo do tempo. O teste é efectuado usando os algoritmos de escalonamento MCF e CBFQ separadamente.

4.3.3 Resultados

Para cada teste realizado os gráficos representam as taxas de pedidos admitidos por segundo, o rácio de admissão verificado ao longo do tempo, e o atraso sofrido no processo de admissão dos pedidos.

Os gráficos são obtidos a partir dos histogramas associados à tabela *uif_test_admission_control*.

4.3.3.1 Resultados MCF sem Diferenciação de Prioridades

A Figura 23 apresenta os resultados dos testes para o cenário onde é utilizado o algoritmo MCF sem a aplicação de diferenciação de prioridades entre as filas de espera.

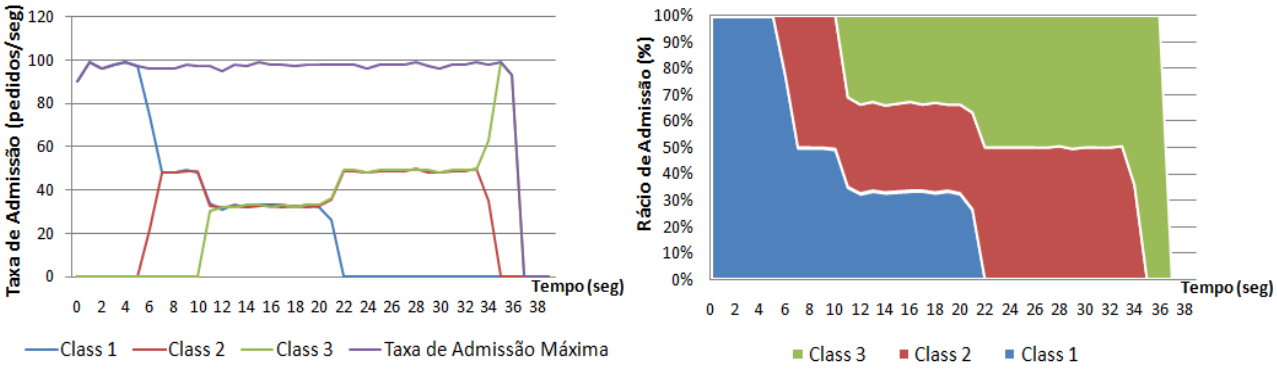

Figura 31. Resultados MCF sem diferenciação de prioridades

Verifica-se que nos instantes iniciais apenas são emitidos pedidos de classe1, de modo que esta classe consome toda a capacidade de admissão. Contudo quando os pedidos de classe 2 começam a ser emitidos, a capacidade de admissão é dividida pelas duas classes equitativamente, sendo alocada 50% da capacidade total para cada uma.

De modo análogo, quando os pedidos da classe 3 começam a ser emitidos, a capacidade de admissão é atribuída de modo equitativo entre as três classes de serviço activas, sendo alocada aproximadamente 33% da capacidade total para cada uma das classes.

Quando os pedidos de classe1 são totalmente satisfeitos, a taxa de admissão reservada a esta classe é distribuída pelas classes 2 e 3, com alocação de 50% da capacidade total para cada uma das classes.

Finalmente quando a totalidade dos pedidos de classe 2 são satisfeitos, a classe3 passa a deter um rácio de admissão equivalente à taxa máxima de admissão permitida, não existindo contenção com as restantes classes.

Este teste demonstra o perfil de equidade prestado pela disciplina de escalonamento MCF.

O gráfico apresentado na Figura 24 representa os atrasos sofridos na admissão dos pedidos neste teste, para a disciplina de escalonamento MCF.

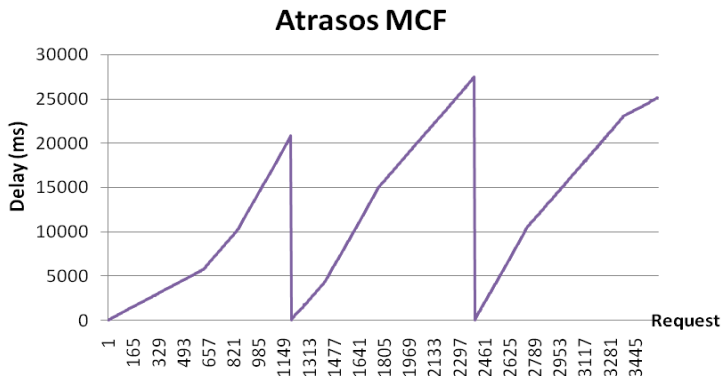

Figura 32. Atrasos de admissão MCF sem diferenciação de prioridades

Atrasos Médios MCF

Classe 1	7515ms
Classe 2	14317ms
Classe 3	14779ms

Tabela 9. Atrasos médios para disciplina MCF sem diferenciação de prioridades

	Classe 1	Classe 2	Classe 3
Atraso Máximo	20841ms	27480ms	25108ms
Atraso Mínimo	16ms	16ms	36ms

Tabela 10. Atrasos extremos para disciplina MCF sem diferenciação de prioridades

Destacam-se três variações relativas às classes 1, 2 e 3 respectivamente.

Apesar de todas as filas de espera possuírem a mesma prioridade, o atraso médio das filas é distinto entre as mesmas. Este comportamento provém do facto que a contenção entre as filas de espera alterna ao longo do período do teste, devido à variação da sua actividade. Existem períodos de tempo onde apenas uma classe de serviço está activa, e outros onde as três classes de serviço competem pelo serviço do escalonador. Por esse facto, os pedidos da classe 1 revelam menor atraso médio e máximo do que os pedidos das restantes classes. Do mesmo modo, os pedidos da classe 2 revelam um atraso médio e máximo superior, visto que foram submetidos a um maior período de contenção.

4.3.3.2 Resultados MCF com Diferenciação de Prioridades

A figura 25 apresenta os resultados dos testes para o cenário onde é utilizado o algoritmo MCF sendo efectuada diferenciação entre as filas de espera.

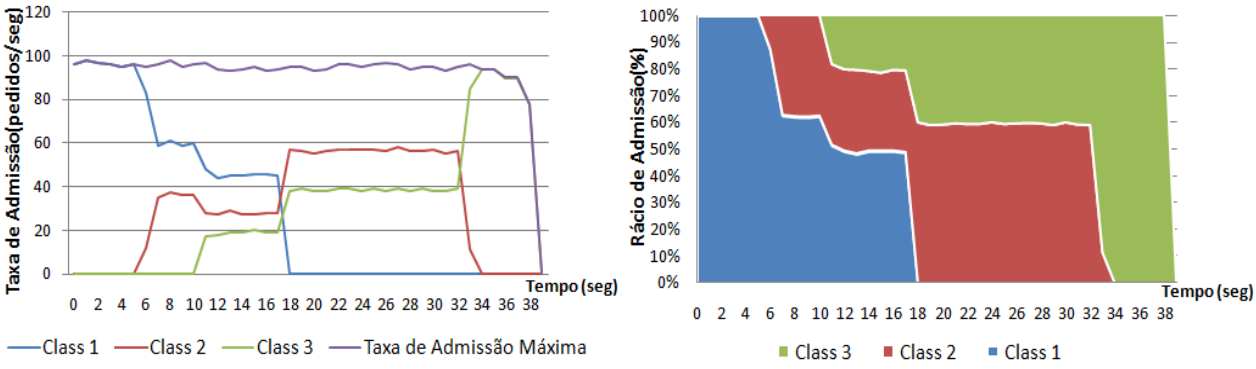

Figura 33. Resultados MCF com diferenciação de prioridades

Analogamente ao caso em que não é efectuada diferenciação das prioridades das filas de espera, verifica-se que nos instantes iniciais, apenas são emitidos pedidos de classe1, consumindo esta classe toda a capacidade de admissão disponível.

Contudo, no instante em que os pedidos de classe 2 começam a ser emitidos, o escalonador passa a distribuir capacidade de admissão pelas classes 1 e 2 de modo

proporcional à sua prioridade, de aproximadamente 60% e 40% respectivamente. Analogamente, no instante em que os pedidos da classe 3 são emitidos, capacidade de admissão é distribuída proporcionalmente às prioridades definidas para as três classes, de 50%, 30% e 20% respectivamente. Quando os pedidos de classe1 são totalmente satisfeitos, a taxa de admissão reservada a esta classe é distribuída pelas classes 2 e 3, com alocação de aproximadamente 60% e 40% respectivamente. Finalmente quando a totalidade dos pedidos de classe 2 são satisfeitos, a classe3 passa a deter o total da capacidade de admissão, não existindo contenção com as restantes classes.

Analogamente ao teste em que não é efectuada diferenciação das prioridades, este teste demonstra que o escalonamento é efectuado com equidade. Não obstante, este teste demonstrou ainda que a distribuição da capacidade de admissão é efectuada de modo proporcional às prioridades definidas para cada fila de espera, ao longo da sua actividade.

O gráfico apresentado na Figura 26 representa os atrasos sofridos neste teste, para a disciplina de escalonamento MCF.

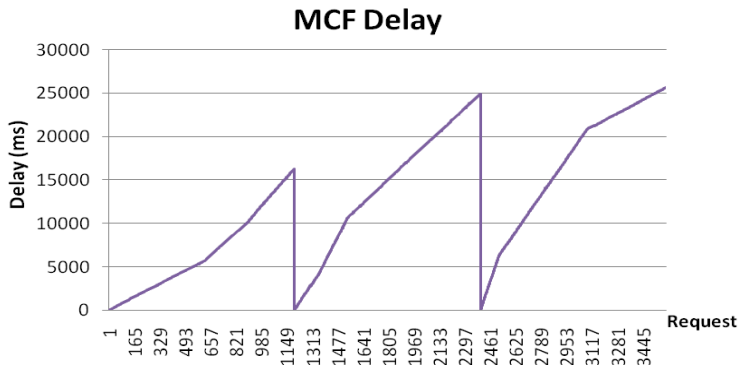

Figura 34. Atrasos de admissão MCF com diferenciação de prioridades

Atrasos Médios MCF	
Classe 1	6697ms
Classe 2	14213ms
Classe 3	16710ms

Tabela 11. Atrasos médios para disciplina MCF com diferenciação de prioridades

	Classe 1	**Classe 2**	**Classe 3**
Atraso Máximo	16346ms	25015ms	25732ms
Atraso Mínimo	17ms	31ms	68ms

Tabela 12. Atrasos extremos da disciplina MCF com diferenciação de prioridades

Destacam-se três variações relativas às classes 1, 2 e 3 respectivamente.

Numa análise global, tendo em consideração o tempo médio de atraso de admissão sofrido por cada classe de serviço, verifica-se que os pedidos das classes de serviço de maior prioridade atingem em média atrasos inferiores.

Contudo, o motivo pelo qual os pedidos da classe 2 possuem um atraso médio pouco inferior aos pedidos da classe3, de menor prioridade, reside no facto que durante o período em que a classe 2 se encontra a ser servida, esta está constantemente sujeita a contenção com as restantes classes de serviço. Os pedidos de classe 3 por sua vez são servidos sem contenção no período final, utilizando a totalidade da capacidade de admissão.

O atraso médio de admissão dos pedidos de classe 1 destaca-se por ser bastante inferior ao atraso das restantes classes de serviço. Este comportamento resulta do facto desta classe de serviço ter uma prioridade superior face às restantes, assim como do facto desta classe utilizar a totalidade da capacidade de admissão durante os instantes iniciais.

Verifica-se ainda que o atraso mínimo verificado na classe 3 é superior aos restantes pois quando os pedidos desta classe são emitidos existe a contenção entre as três classes de serviço.

4.3.3.3 Resultados CBFQ sem Diferenciação de Prioridades

A Figura 27 apresenta os resultados dos testes para o cenário onde é utilizado o algoritmo CBFQ, não sendo efectuada diferenciação de prioridades entre as filas de espera.

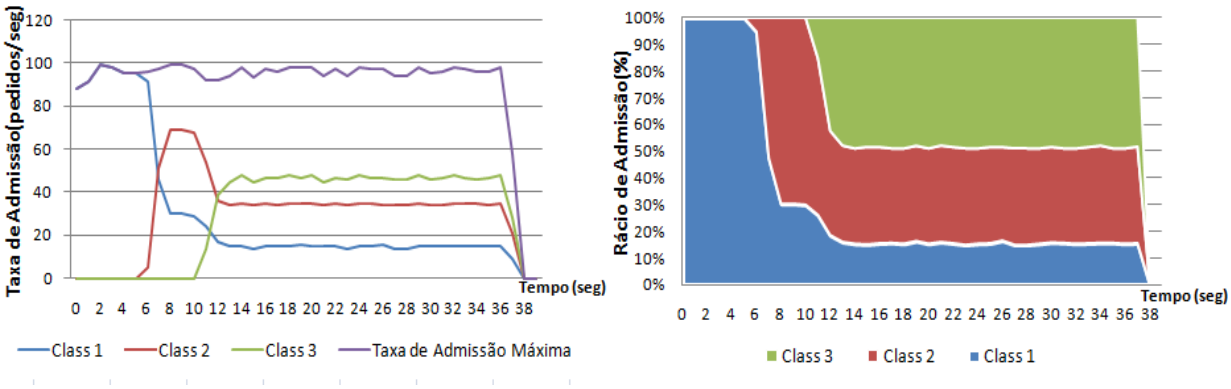

Figura 35. Resultados CBFQ sem diferenciação de prioridades

Verifica-se que nos instantes iniciais não existe contenção. Apenas são emitidos pedidos de classe1, consumindo esta classe a totalidade da capacidade de admissão.

Quando os pedidos de classe 2 são emitidos, é tida em conta não só a divisão da do rácio de admissão de modo proporcional ao custo das filas de espera activas da classe 1 e 2, como também o nível de carga existente em cada fila de espera. Contudo, como as prioridades das filas são iguais e não existe variação na relação entre os níveis de carga das filas de espera activas, o serviço das classes é mantido com o mesmo nível ao longo do tempo, até que não existam mais pedidos a admitir nas filas de espera. Assim, quando os pedidos de classe 3 começam a ser emitidos não se verificam variações de serviço sendo o atendimento efectuado de um modo homogéneo até que não existam mais pedidos em fila de espera.

Neste caso, o comportamento adaptativo desta disciplina não se destaca devido ao facto do peso das filas ser igual, contudo num cenário com diferenciação dos pesos da fila de espera o aumento do nível de serviço das filas de maior prioridade deverá provocar variação na relação do nível de carga de cada fila em relação às restantes, tornando o serviço adaptativo mais evidente.

O gráfico apresentado na Figura 28 representa os atrasos sofridos neste teste pela disciplina CBFQ.

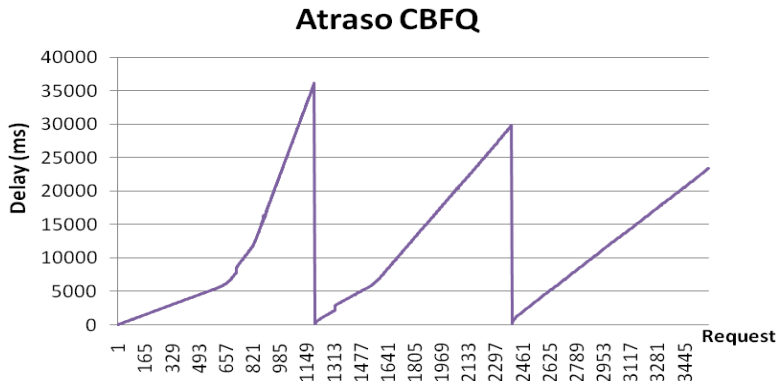

Figura 36. Atrasos de admissão CBFQ sem diferenciação de prioridades

Atrasos Médios MCF	
Classe 1	10353ms
Classe 2	13533ms
Classe 3	12051ms

Tabela 13. Atrasos médios para disciplina CBFQ sem diferenciação de prioridades

	Classe 1	Classe 2	Classe 3
Atraso Máximo	36188ms	29849ms	23405ms
Atraso Mínimo	14ms	134ms	214ms

Tabela 14. Atrasos extremos para disciplina CBFQ sem diferenciação de prioridades

Destacam-se três variações relativas às classes 1, 2 e 3 respectivamente.

Verifica-se que as classes de serviço de maior prioridade possuem menor atraso médio, contudo a distribuição do atraso não é linear, contrariamente à disciplina MCF. Deste modo os valores máximos do atraso sofrido para cada classe de serviço atingem maior amplitude nas classes de prioridade superior. Este comportamento é justificado pelo perfil adaptativo da disciplina CBFQ, que embora pondere a prioridade das classes de serviço, também considera a carga associada a cada fila de espera como contributo para a decisão de escalonamento.

4.3.3.4 Resultados CBFQ com Diferenciação de Prioridades

Para destacar o comportamento adaptativo da disciplina CBFQ apresenta-se na Figura 29 os resultados dos testes para o cenário onde é efectuada diferenciação entre as filas de espera.

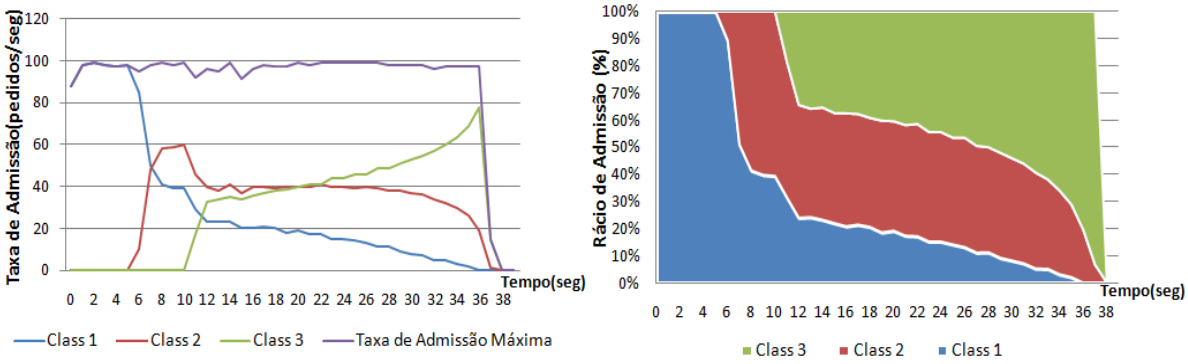

Figura 37. Resultados CBFQ com diferenciação de prioridades

Também neste caso se verifica que nos instantes iniciais não existe contenção, sendo emitidos apenas os pedidos de classe1, consumindo esta classe a totalidade da capacidade de admissão.

Face à diferenciação de prioridades, no instante em que os pedidos de classe 2 passam a concorrer pelo serviço com os pedidos de classe 1, verifica-se que a capacidade de

admissão é distribuída tendo em consideração não só a prioridade associada a cada classe de serviço, como também o nível de carga que cada fila de espera possui ao longo do tempo. No instante em que os pedidos de classe 3 começam a ser emitidos, verifica-se que o atendimento destes pedidos é efectuado segundo uma taxa crescente de modo exponencial.

Esta tendência crescente provém do facto de, apesar de a classe 3 ter prioridade inferior às restantes, o seu nível de carga ser superior em relação às restantes filas, sendo esta diferença cada vez mais evidente à medida que o tempo passa. Deste modo, para balancear o nível de carga das várias filas de espera, a fila da classe3 é servida com maior prioridade em relação às restantes.

Este facto torna o CBFQ apropriado para lidar com cenários de congestão onde as classes de serviço possuem uma afluência desproporcional que pode conduzir à negação de serviço das classes de menor prioridade.

O gráfico apresentado na Figura 30 representa os atrasos sofridos neste teste para a disciplina CBFQ.

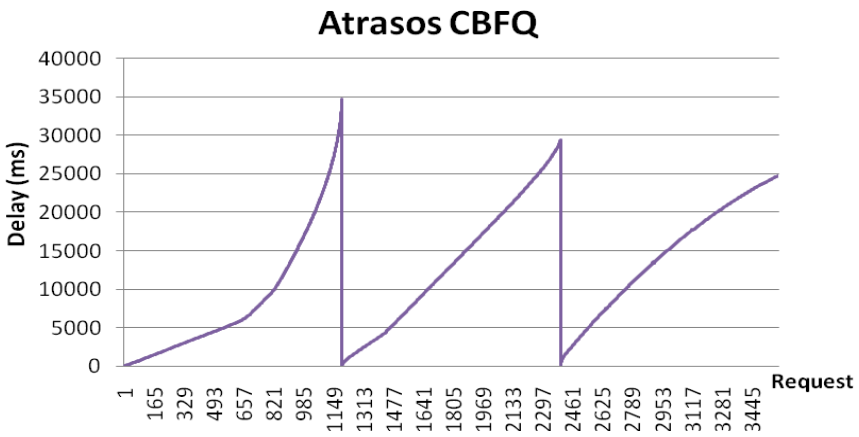

Figura 38. Atrasos de admissão CBFQ com diferenciação de prioridades

Atrasos Médios CBFQ	
Classe 1	8660ms
Classe 2	13412ms
Classe 3	14514ms

Tabela 15. Atrasos médios para disciplina CBFQ com diferenciação de prioridades

	Classe 1	**Classe 2**	**Classe 3**
Atraso Máximo	34770ms	29412ms	24810ms
Atraso Mínimo	8ms	189ms	366ms

Tabela 16. Atrasos extremos para disciplina CBFQ com diferenciação de prioridades

Destacam-se três variações relativas às classes 1, 2 e 3 respectivamente.

Analogamente aos atrasos apresentados no teste da disciplina CBFQ, onde não foi aplicada diferenciação no serviço das filas de espera, verifica-se que os pedidos de classe3, de prioridade inferior, possuem um atraso máximo de admissão inferior às restantes classes. Verifica-se ainda que o atraso médio de admissão dos pedidos de classe 1 é inferior em relação às restantes classes, sendo os pedidos de classe 3 detentores do atraso médio superior.

Mais uma vez se conclui que, para além do nível de prioridade associado a cada classe de serviço ser um contributo para a decisão de escalonamento, demonstrado pelo facto dos tempos de atraso de admissão médio serem inferiores nas filas de prioridade superior, que a consideração da carga associada a cada fila de espera provoca o aumento do tempo de atraso máximo nas filas de menor carga, visto que estas cedem parte da sua largura de banda reservada a filas de espera com um nível de carga superior.

A diferença entre os atrasos máximos de admissão resulta portanto, do balanceamento de carga que é efectuado, de modo que os últimos pedidos emitidos segundo as várias classes podem sofrer atrasos provenientes da cedência de prioridade às filas que possuam um nível de carga superior.

Podemos concluir com este teste que a disciplina CBFQ induz atraso na classe de maior prioridade de forma a poder reduzir a carga das classes de menor prioridade, nivelando a nível de carga geral nas filas de espera activas.

4.4 Conclusões

Neste capítulo foram apresentados os testes mais relevantes efectuados ao módulo UIF, uma vez detentor da solução de controlo de admissão implementada. Foram apresentadas as análises dos testes respectivos, que permitiram evidenciar o efeito obtido pela aplicação de configurações distintas no controlo da taxa de admissão, assim como a distinção entre o comportamento equitativo da disciplina MCF, e o comportamento adaptativo da disciplina CBFQ.

Os resultados obtidos revelaram-se esclarecedores e demonstraram que a solução actua como previsto nos vários cenários em que foi testada.

No capítulo 5 serão apresentadas as conclusões e as principais contribuições das várias fases desta dissertação.

Capítulo 5

Conclusões

Neste capítulo são apresentadas as principais conclusões alcançadas ao longo da dissertação, sendo efectuada uma breve síntese das contribuições proporcionadas.

São ainda referidos vários aspectos que podem ser abordados para trabalho futuro.

O principal objectivo deste trabalho consistiu na concepção de uma solução de controlo de admissão que actua de modo diferenciado e adaptativo, fornecendo garantias de qualidade de serviço à interface UIF.

5.1 Principais Conclusões e Contribuições

Nesta secção são apresentadas as conclusões relativas às três fases que esta dissertação abordou: o estudo de metodologias de controlo de admissão apresentado no capítulo 2; a proposta e implementação de uma solução de controlo de admissão diferenciado e adaptativo, apresentada no capítulo 3; e os cenários de utilização e teste a que a solução foi submetida, apresentados no capítulo 4.

No âmbito das conclusões são simultaneamente apresentadas as principais contribuições de cada fase.

5.1.1 Metodologias de Controlo de Admissão

Esta dissertação apresentou uma solução que colmata a carência de controlo de admissão da interface UIF. A concepção da solução nasceu de um estudo referente às principais abordagens que um mecanismo de controlo de admissão pode adoptar.

O estudo focou-se na análise de métodos de controlo de ritmo, gestão de largura de banda e disciplinas de escalonamento. Foi prestado especial destaque ao modo de funcionamento e nível de eficácia das principais disciplinas de escalonamento.

Como resultado deste estudo foram seleccionados os métodos considerados mais adequados para a aplicação de uma solução de controlo de admissão que colmatasse a carência de controlo da taxa de admissão, assim como a necessidade de diferenciação da precedência na prestação de serviços.

Foram seleccionadas disciplinas de escalonamento com preocupações de equidade, que utilizam algoritmos de escalonamento inovadores e eficazes, tendo-se revelada essencial a utilização de distintas filas de espera para a prestação de diferenciação no processo de controlo de admissão.

Foi concluído neste estudo que a preferência pela adopção de disciplinas de escalonamento baseadas em créditos é benéfica, na medida em que estas não assumem a complexidade das disciplinas de escalonamento baseadas na computação de funções de tempo virtual, mantendo contudo níveis semelhantes de equidade e eficácia.

Este estudo contribui assim como uma síntese que visa facilitar a compreensão dos métodos de alocação de largura de banda e das principais disciplinas de escalonamento, tendo como preocupação o âmbito aplicacional da sua utilização.

5.1.2 Solução Proposta

A proposta de uma solução para colmatar a carência do controlo de admissão proporcionado pelo módulo UIF, de um modo diferenciado e adaptativo, teve em conta os aspectos referenciados no estudo efectuado, tendo recorrido às metodologias estudadas que melhor se adequam aos requisitos definidos.

A implementação da solução proposta no módulo UIF, permitiu não só regular a taxa de admissão dos pedidos de requisição de serviço com uma granularidade configurável, como também a optimização do processo de admissão face a situações de congestão, em que o número de pedidos de requisição de serviço efectuados em função do tempo ultrapassa os limites estipulados.

A solução proporciona a atribuição flexível de largura de banda ao longo do tempo, tendo sido fornecida a capacidade de diferenciação adaptativa em função da carga associada a cada classe de serviço.

Através da introdução do conceito de classe de serviço, a interface UIF ficou dotada com a potencialidade de diferenciação do nível de precedência dos pedidos que invocam os serviços, segundo os perfis definidos para as classes de serviço respectivas.

A nível da concepção da arquitectura assinala-se o perfil modular da solução, podendo esta ser facilmente adaptada para integração em outros ambientes.

Contribuiu-se assim para a prestação de garantias de qualidade de serviço ao nível do controlo de admissão efectuado no módulo UIF, assim como para a capacidade de diferenciar a prestação de alguns serviços segundo a óptica de negócio desejada.

5.1.3 Cenários de Utilização e Testes

A solução foi sujeita a testes de carga, cujos resultados revelam a eficácia e robustez da mesma, cumprindo os requisitos definidos e proporcionando qualidade de serviço essencial para enriquecer a fiabilidade do processo de admissão.

Os resultados obtidos nos testes efectuados permitiram confirmar que o comportamento da disciplina de escalonamento MCF é baseado na prioridade associada a cada fila de espera, sem considerar o nível de carga das mesmas. Verificou-se que o rácio de admissão é atribuído de forma adaptativa pelas várias classes de serviço, tendo em consideração o nível de carga existente nas filas de espera, sendo efectuado o balanceamento de carga entre as mesmas. Confirmou-se que, quanto maior for a carga de uma fila de espera em relação às restantes filas activas, maior é a sua prioridade de serviço.

Face à submissão da solução a condições de carga elevada, concluiu-se que a limitação e controlo da taxa de admissão e a largura de banda atribuída a cada classe de serviço são efectuadas como suposto.

Assim, podemos concluir que a solução de controlo de admissão implementada no módulo UIF cumpre todos os objectivos propostos.

5.2 Trabalho Futuro

De modo a melhorar a solução desenvolvida, consideram-se como desenvolvimentos futuros os seguintes aspectos:

- a solução poderá ser alvo de optimizações a nível da sua implementação, de modo a aumentar a sua eficiência;

- poderão ser adicionados novos algoritmos de escalonamento que se revelem mais vantajosos face à visão de negócio aplicada na prestação de serviços. Os algoritmos implementados têm o propósito de satisfazer os requisitos impostos para a funcionalidade de controlo de admissão que o módulo UIF implementa. Contudo, para outras plataformas que possuam requisitos distintos devem ser utilizados algoritmos de escalonamento com diferentes finalidades;

- poderão ser implementados cenários de teste adicionais que comparem a eficácia das disciplinas de escalonamento implementadas com outras não implementadas;

- a análise do comportamento das disciplinas de escalonamento utilizadas poderá também ser enriquecida face a cenários de teste distintos.

5.3 Observações Finais

Como observação final, julgo poder considerar que os objectivos desta dissertação foram efectivamente cumpridos.

O desenvolvimento da dissertação no contexto de uma empresa como a PT INOVAÇÃO permitiu-me desenvolver aptidões em matérias que desconhecia, e interagir com tecnologias presentemente utilizadas no panorama empresarial.

Todo o estudo de investigação realizado permitiu-me uma melhor compreensão do funcionamento das principais metodologias de controlo de admissão existentes até à actualidade.

Evidencio ainda a participação na *9ª Conferência sobre Redes de Computadores*, em que apresentei o artigo "*Adaptative Admission Control in Real Time Systems*" desenvolvido no âmbito desta dissertação.

Está ainda prevista a participação na *5ª Conferência de Engenharia 2009 – Inovação e Desenvolvimento*, com o artigo "*Controlo de Admissão Diferenciado e Adaptativo em Sistemas de Tempo Real*", também desenvolvido no âmbito desta dissertação, sendo abordada a solução proposta com mais detalhe.

Assim, considero que este projecto de dissertação me proporcionou benefícios pessoais com a aquisição de competências e experiência que considero úteis para o meu futuro.

Referências Bibliográficas

1. Brandauer, C., Iannaccone, G., Diot, C. and Fdida, S. "*Comparison of Tail Drop and Active Queue Management Performance for Bulk-Data and Web-Like Internet Traffic",* Computers and Communications, IEEE Symposium on, 2001. volume 0: p. 0122.

2. Jacobson, V. and Floyd, S. "*Random Early Detection Gateways for Congestion Avoidance",* IEEE/ACM Transactions on Networking, 1993, volume 1: p. 397-413.

3. Keshav, S. "*An Engineering Approach to Computer Networking",* Addison-Wesley Professional Computing Series, 1997.

4. Sousa, P., Carvalho, P. and Freitas, V. "*A Multi-Constrained QoS Aware Scheduler for Class-Based IP Networks",* 4th Symposium on Communication Systems, 2004.

5. Astuti, D., "*Packet Handling",* Seminar on Transport of Multimedia Streams in Wireless Internet.

6. Parekh, A. and Gallager, R. "*A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single Node Case",* IEEE/ACM Transactions on Networking, IEEE Computer Society Press, 1993, volume 1: p. 344-357.

7. Goyal, P., Vin, H. and Cheng, H. "*Start-Time Fair Queueing: A Scheduling Algorithm for Integrated Services Packet Switching Networks",* IEEE/ACM Transactions on Networking, IEEE Computer Society Press, 1997, volume 5: p. 690-704.

8. Demers, A., Keshav, S. and Shenker, S. "*Analysis and Simulation of a Fair Queueing Algorithm",* SIGCOMM: ACM Special Interest Group on Data Communication, 1989, volume 19: p. 1-12.

9. Boudec, J., "*Rate adaptation, Congestion Control and Fairness: A Tutorial",* 2000.

10. Eryilmaz, A. and Srikant, R. "*Fair Resource Allocation in Wireless Networks Using Queue-Length-Based Scheduling and Congestion Control",* IEEE/ACM Transactions on Networking, 2007, volume 15: p. 1333-1344.

11. Tassiulas, L. and Ephremides, A. "*Stability Proprieties of Constrained Queueing Systems and Scheduling Policies for Maximum Throughput in Multihop Radio Networks",* IEEE Transactions on Automatic Control, 1992, volume 37: p. 1936-1948.

12. Jalali, A., Padovani, R. and Pankaj, R. "*Data Throughput of CDMA-HDR a High Efficiency-High Data Rate Personal Communication Wireless System",* Vehicular Technology Conference Proceedings, 2000, volume 3: p. 1854--1858.

13. Yang, J., Yifan, Z., Ying, W. and Ping, Z., "*Average Rate Updating Mechanism in Proportional Fair Scheduler for HDR",* IEEE Global Telecommunications Conference, 2004, volume 6: p. 3464-3466.

14. Whiting, H. and Whiting, P. "*Convergence of Proportional-Fair Sharing Algorithms under General Conditions",* IEEE Transactions on Wireless Communications, 2003, volume 3: p. 1250--1259.

15. Rexford, J., Greenberg, A. and Bonomi, F. "*Hardware-Efficient Fair Queueing Architectures for High-Speed Networks"*, In Proceedings of INFOCOM, 1996, volume 2: p. 638-646.

16. Wong, W., Tang, H. and Leung, V. "*Token Bank Fair Queuing: A New Scheduling Algorithm for Wireless Multimedia Services"* Research Articles, International Journal of Communication Systems, 2004, volume 17: p. 591-614.

17. Xie, D. and J, Y. "*Probability Based Weighted Fair Queueing Algorithm with Adaptive Buffer Management for High-Speed Network",* Springer Berlin / Heidelberg, 2006, volume 4222/2006: p. 428-437.

18. Sethu, H., Zhou, Y. "*On the Relationship Between Absolute and Relative Fairness Bounds",* IEEE Comm. Letters, 2002, volume 6: p. 37--39.

19. Hahne, E., "*Round-Robin Scheduling for Max-Min Fairness in Data Networks",* IEEE Journal on Selected Areas in Communications, 1991, volume 9: p. 1024--1039.

20. Alborz, N., Chen, B. and Trajkovic, L. "*Modeling Packet Scheduling Algorithms in IP Routers"*, School of Engineering Science, Simon Fraser University, 2001.

21. Shreedhar, M. and G. Varghese, "*Efficient Fair Queueing Using Deficit Round Robin"*, in Proceedings of the Conference on Applications, ACM, 1995, volume 4: p. 231--242.

22. Nagle, J., "*On Packet Switches With Infinite Storage"*, in Innovations in Internetworking, Artech House, 1988, p. 136-139.

23. Bennett, J. and Zhang, H. "*Wf2q : Worst-case Fair Weighted Fair Queueing",* 1996.

24. Golestani, S. "*A Self-Clocked Fair Queueing Scheme for Broadband Applications",* INFOCOM '94, 13th Proceedings IEEE, 1994, volume 2: p. 636--646.

25. Pan D. and Yang, Y. "*Credit Based Fair Scheduling for Packet Switched Networks",* INFOCOM 2005, Proceedings IEEE, 2005, volume 2: p. 843--854.

26. Bensaou, B., Tsang, D. and Chan, K. "*Credit-Based Fair Queueing (CBFQ): A Simple Service-Scheduling Algorithm for Packet-Switched Networks",* IEEE/ACM Transactions on Networking, 2001, volume 9: p. 591-604.

27. *Core J2EE Patterns - Data Access Object*. Available from: http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html.

28. JBoss Homepage: http://www.jboss.org/

29. Maven Homepage: http://www.maven.apache.org/

30. JUnit Homepage: http://www.junit.org/