DESENHO E IMPLEMENTAÇÃO DO PROTÓTIPO DE UM ELEMENTO DE REDE CAPAZ DE FORNECER *DIFFERENTIATED SERVICES*

Carla Avelans, Gonçalo Quadros, António Alves, Edmundo Monteiro, Fernando Boavida
cavelans@student.dei.uc.pt, quadros@dei.uc.pt, aalves@dei.uc.pt, edmundo@dei.uc.pt, boavida@dei.uc.pt
Departamento de Engenharia Informática, Universidade de Coimbra, Portugal
Telefone: 351-39-790000 Fax: 351-39-701266

Sumário

Este artigo descreve o desenvolvimento do protótipo de um encaminhador, em plataforma *FreeBSD*, capaz de fornecer Qualidade de Serviço (QoS) a tráfego IP. O protótipo baseia-se no modelo *Differentited Services* e é suportado pela métrica de QoS desenvolvida no Laboratório de comunicações Telemáticas (LCT) do DEI[1].

1. Introdução

A qualidade de serviço na Internet e a procura de melhores tecnologias para a garantir tem constituído uma necessidade a satisfazer a curto prazo. Neste sentido tem-se verificado uma intensa actividade de investigação por parte de alguns organismos da comunidade Internet, destacando-se entre eles o IETF [1].

Durante a última década tem-se vindo a presenciar ao crescimento exponencial do número de utilizadores Internet, assim como ao constante emergir de novas aplicações geradoras de tráfego de diferentes naturezas (áudio, vídeo, *real-time* e dados). Muitas destas aplicações são exigentes ao nível do atraso máximo no transporte dos dados, da variação máxima desse atraso (*jitter*), das perdas ou da largura de banda disponível. Assim, para funcionarem correctamente, é necessário que determinados níveis de desempenho sejam garantidos (QoS).

No entanto, a plataforma Internet não está pensada para suportar convenientemente tal heterogeneidade de informação. Como é sabido, o protocolo IP funciona com base no paradigma *best-effort* e portanto, todos os pacotes têm o mesmo tratamento ao longo do seu percurso, não existindo qualquer diferenciação de desempenho. Torna-se clara a necessidade de efectuar alterações específicas ao modelo Internet tradicional, de modo a prover a rede com mecanismos de QoS que lhe permitam suportar de forma interessante grandes quantidades de informação com diferentes requisitos de QoS.

Para tal, diferentes abordagens têm sido propostas pelos diversos grupos de investigação nesta área, destacando-se de entre elas os modelos: *Integrated Services* (IntServ) e *Differentiated Services* (DiffServ). O modelo IntServ [2], proposto pelo grupo INTSERV [3] da IETF, define métodos que se baseiam na especificação de QoS e na reserva de recursos, usados para atribuir parte da capacidade dos sistemas de comunicação a fluxos de dados individuais. Todavia, a utilização deste modelo em algumas redes tem revelado importantes limitações que colocam em causa a sua aplicabilidade em redes alargadas. Perante estas limitações e na tentativa de as superar, o grupo DIFFSERV [4] desenvolveu o modelo DiffServ [5, 6]. Este modelo, mais recente, tem estado sujeito a uma importante actividade de investigação.

O modelo DiffServ baseia-se na classificação do tráfego em classes de serviço (CoS), determinadas por bits específicos nos cabeçalhos dos pacotes IP. Cada classe, de acordo com o nível de QoS que lhe está associada, irá obter dos sistemas de comunicação um tratamento particular, adequado às suas necessidades de desempenho.

Neste artigo é apresentado o desenho e processo de implementação de um protótipo de encaminhador capaz de fornecer QoS a tráfego IP. O protótipo baseia-se no modelo de serviço DiffServ: pretende tornar um elemento de rede (nomeadamente um encaminhador) capaz de fornecer qualidade de serviço diferenciada a classes de tráfego distintas, independentemente do estado de carga, ou congestão, nesse elemento de rede. No sentido de implementar um algoritmo de diferenciação de QoS flexível e dinâmico foi utilizada a métrica de monitorização de QoS [7] desenvolvida no LCT[2].

A secção 2 deste artigo apresenta a arquitectura do protótipo, descrevendo de forma resumida o funcionamento e a implementação dos módulos funcionais que o constituem. A secção 3 descreve os testes realizados ao protótipo, os seus resultados e as primeiras conclusões que é possível deles extrair. Por último são apresentadas na secção 4 as conclusões gerais e as referências a trabalho futuro.

2. Protótipo de um Elemento de Rede Capaz de Fornecer QoS

2.1. Requisitos

As recomendações do grupo DIFFSERV não fazem qualquer referência a algoritmos a adoptar nem tornam obrigatória nenhuma implementação em particular, salientando que as decisões a esse nível devem ser tomadas pelo implementador, tendo em conta os objectivos do serviço a construir. Na especificação e implementação do nosso protótipo houve o cuidado de satisfazer alguns requisitos que a seguir se enumeram:

i) G*eneralidade/Portabilidade*: o protótipo foi desenvolvido para plataformas *FreeBSD*. De forma a melhorar as suas características de portabilidade (*e. g.* tornar possível a sua utilização com tecnologia ATM) evitou-se manipular outro tipo de código que não o que diz respeito à camada IP.

ii) *Transparência*: o código desenvolvido é transparente para o código, ou funcionalidade, já existente. O código original permaneceu inalterado, tendo sido acrescentado o código necessário para implementar as novas funcionalidades do protótipo sem colisão/exclusão das já existentes. Para tal, foi criada uma macro de preprocessamento que inclui, quando activada, o código novo no *kernel* do sistema operativo (SO).

[1]Departamento de Engenharia Informática da Faculdade de Ciências e Tecnologias da Universidade de Coimbra (*http://www.dei.uc.pt*).

[2]Na fase a que este artigo diz respeito apenas foi considerada a característica de QoS atraso de trânsito. O objectivo final é incluir também a característica perdas.

iii) *Flexibilidade*: o código foi implementado de modo a que a configuração dos parâmetros importantes ficasse facilitada. Por exemplo, o número de classes de serviço a suportar pelo DiffServ pode depender do fornecedor de serviços ou do gestor[3], do sistema onde será instalado, ou ainda da disponibilidade de recursos, etc. A manipulação destes parâmetros poderá ser útil para afinar o protótipo de forma a ter um desempenho do sistema onde é utilizado.

iv) *Eficiência/Simplicidade:* o protótipo foi concebido de forma que a sua latência, ou o tempo que ele demora a responder a variações nas condições da rede, seja tão baixa quanto possível. Como tal, e pelo facto de o protótipo estar integrado no *kernel* do SO, evitou-se incluir código em rotinas da pilha protocolar que se consideram críticas no que diz respeito à frequência e prioridade na sua utilização, por forma a minimizar o seu tempo de execução (*e. g.* as macros que manipulam as filas do sistema). Ainda no sentido de manter o nível de desempenho do sistema, houve o cuidado de adoptar algoritmos relativamente simples (não descurando o compromisso entre simplicidade e completude).

2.2. Arquitectura

O modelo DiffServ explicita que os sistemas de comunicação situados no interior de uma rede *Differentiated Services* (DS) apenas necessitam de ter implementados as primitivas de classificação, os mecanismos de gestão e/ou disciplinamento de tráfego, necessários para implementar os PHBs[4]. O protótipo destina-se a um elemento de rede nestas condições e é implementado por três componentes ou módulos principais — Classificador, Disciplinador e Medidor — que, actuando em conjunto, são capazes de fornecer uma QoS diferenciada a distintas classes de serviço. A figura 1 representa a arquitectura do protótipo e a interligação entre os diferentes módulos.

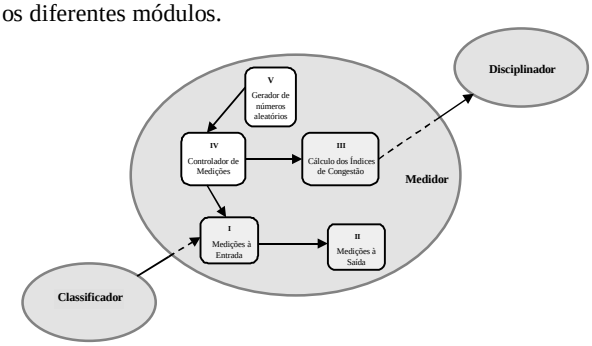


Figura 1 – Arquitectura do Protótipo. Uma seta de um módulo para outro, representa a influência que o primeiro exerce no segundo.

2.2.1. Módulo Classificador

O módulo Classificador implementa a primitiva que é responsável por separar os pacotes IP em classes de serviço e por direcciona-los para as respectivas filas. A classificação dos pacotes IP é feita com base nos indicadores de diferenciação residentes no cabeçalho dos pacotes. Para tal, foram utilizados os 3º e 4º bits do campo TOS[5] do cabeçalho IP dos pacotes, tendo como referência a proposta especificada em [8]. Estes bits indicam o nível de sensibilidade de cada pacote (ou classe) ao atraso, pelo que é possível ter no máximo 4 níveis, ou classes, diferentes. A figura 2 apresenta a semântica adoptada para indicar os níveis de sensibilidade de cada uma das classes ao atraso.

Bit 3	Bit 4	Classe	Semântica
1	0	**1**	**Muito** sensível ao atraso
0	1	**2**	**Sensível** ao atraso
1	1	**3**	**Pouco** sensível ao atraso
0	0	**0**	**Insensível** ao atraso

Figura 2 – Semântica dos bits do campo TOS ao atraso.

O classificador permite, também, identificar os pacotes como pertencentes a uma determinada classe analisando cinco campos do cabeçalho: endereço origem, endereço destino, porto origem, porto destino e protocolo.
A classificação ou diferenciação do tráfego é efectuada à entrada da camada IP, mais especificamente na macro IF_ENQUEUE. Esta macro é executada sempre que a camada lógica pretende colocar um pacote na fila correspondente da camada de rede. Uma vez que o protótipo suporta 4 classes de serviço foi necessário substituir a fila de entrada na camada IP, única, por 4 filas associadas a cada uma das classes. Os pacotes não pertencentes a nenhuma das classes acima definidas são considerados como pertencentes à classe 0, que recebe um tratamento *best-effort*.

2.2.2. Módulo Disciplinador

O disciplinador é um elemento fundamental do protótipo uma vez que é ele o responsável por fornecer o tratamento diferenciado às diferentes classes de serviço. O algoritmo de disciplinamento adoptado é do tipo *weighted-round robin* [9]. O disciplinador visita periodicamente as filas IP, segundo o método *round robin*, retirando de cada uma delas um número de pacotes relacionados com a prioridade (peso) da fila. Neste algoritmo as prioridades das filas são atribuídos de forma dinâmica. Isto é, a métrica de QoS, implementada pelo módulo medidor, permite que o número de pacotes (peso) a processar por fila varie dinamicamente, de acordo com os níveis de serviço (QoS) que as classes de tráfego estão efectivamente a receber e com o estado de congestão verificado no elemento de rede. O pacote seleccionado segue depois para processamento *normal*. A figura 3 representa o esquema do funcionamento dos módulos Classificador e Disciplinador.

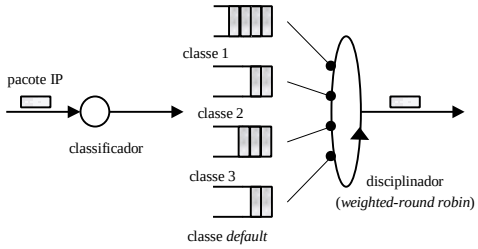


Figura 3 – Funcionamento dos módulos Classificador e Disciplinador.

[3] O modelo DiffServ apenas especifica o número e a estrutura de bits no campo DS do cabeçalho dos pacotes IP que devem ser utilizados para identificar as classes de serviço a que pertencem os fluxos de dados. Cada fornecedor é livre de utilizar o número de classes que considere necessário para disponibilizar os serviços pretendidos.
[4] Per-Hop Behaviors. Cada PHB representa o tratamento específico que cada agregado de tráfego, que identifica determinada classe de serviço, espera receber dos elementos da rede ao longo do seu percurso.
[5] Campo *Type of Service* do cabeçalho IPv4.

2.2.3. Módulo Medidor

O módulo Medidor implementa a métrica de QoS desenvolvida no LCT e tem como principal funcionalidade actuar sobre o módulo Disciplinador de forma a corrigir dinamicamente a QoS associada a cada uma das classes. Segundo esta métrica, os níveis de QoS que cada classe está a receber do sistema, assim como o estado de carga no sistema, podem ser avaliados através do cálculo de índices de congestão. Estes índices de congestão são calculados regularmente para cada uma das classes, através da medição de características de QoS ao longo da camada IP. O medidor apenas mede a característica atraso, uma vez que as classes apenas são definidas de acordo com o nível de sensibilidade ao atraso[6]. Assim sendo, o seu objectivo final consiste em calcular, para cada classe de serviço, o valor dos índices de congestão relativos ao atraso.
O módulo Medidor é constituído por cinco submódulos funcionais que representam as diferentes acções levadas a cabo pelo medidor no desempenho das suas actividades de medição ao longo da camada IP e no controlo dessas actividades.
Os módulos I, II e III estão directamente relacionados com as acções de medição, concretamente a marcação do tempo de chegada dos pacotes ao nível IP, o cálculo do atraso de trânsito quando os pacotes abandonam esse nível e o cálculo dos índices de congestão em função do atraso medido. Os módulos IV e V influenciam e/ou controlam as acções de medição dos módulos I, II, e III, ou seja, determinam os instantes de tempo que espoletam o início de uma actividade de medição. Na figura 1 encontra-se representada a interacção entre estes módulos.

2.2.4. Principais Decisões e Desafios relacionadas com a Implementação do Protótipo

Durante a implementação destes módulos foram tomadas decisões importantes no sentido de manter a simplicidade e escalabilidade do protótipo, entre elas:
- foi utilizada a função de sistema *rdtsc* para devolver o instante exacto de tempo. Esta função acede directamente (através de instruções de *assembler*) ao contador de relógio disponível nos processadores *Pentium* da *Intel*, pelo que é extremamente rápida. Também tem uma granularidade muito fina (ciclos de relógio), importante para o cálculo dos valores de atraso com precisão.
- o instante de chegada de um pacote ao nível IP é armazenado no espaço livre disponível na estrutura de suporte (ou *mbuf*[7]) do pacote. Esta solução optimiza o acesso à informação uma vez que evita o recurso a estruturas específicas para esse efeito e portanto, evita a alocação de memória e a perda tempo na pesquisa da informação residente nessas estruturas.
- parte das actividades relacionadas com a cálculo do atraso são feitas, e têm de ser feitas, em zonas críticas (de elevada prioridade) do código do *kernel*. No entanto, o cálculo dos índices de congestão não foi efectuado nestas zonas. Isto pode resultar em alguma latência na obtenção dos índices, que se considera aceitável, mas minimiza o risco de não serem efectuadas outras tarefas importantes do sistema.
- todas as alterações do código do *kernel* foram efectuadas dentro do nível IP. Deste modo o protótipo pode ser portado, e testado, com diferentes tipos de interfaces.

O leitor interessado em mais pormenores sobre este trabalho deve consultar o documento [10].

3. Testes ao Protótipo

Uma vez implementado o protótipo, este foi submetido a um conjunto de testes diversificados. Estes testes foram realizados no sentido de avaliar a sua robustez, correcção, desempenho e comportamento. Os testes aqui apresentados incidem sobre a avaliação do comportamento do protótipo[8]. Com eles pretende-se aferir a sua capacidade para fornecer uma QoS diferenciada a diferentes classes de serviço. Designadamente, que a classe 1 (a mais sensível) sofre menor degradação de atraso que a classe 2, e essa, menor que a classe 3.

3.1. Ambiente de testes

O ambiente de testes consiste numa rede de local *10Base-T Ethernet*, constituída por duas máquinas Pentium, e está representada na figura 4. Ambas as máquinas correm o sistema operativo *FreeBSD* (v. 2.2.6) e o protótipo encontra-se instalado na máquina *calvin* (encaminhador). A máquina *ideafix* funciona como fonte emissora e receptora do tráfego gerado.

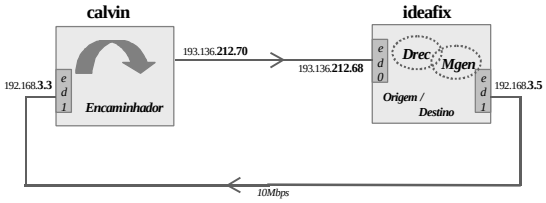


Figura 4 – Ambiente de testes.

Foram executadas duas sequências de testes: na 1ª sequência foi gerado tráfego (fluxos de dados) para duas filas em simultâneo; na 2ª sequência foi gerado tráfego para as quatro filas (classes) em simultâneo. Em ambas as sequências, o encaminhador foi submetido a diferentes níveis de carga e foram medidos os valores de atraso dos pacotes pertencentes a cada classe. Foi utilizada a ferramenta MGEN [11] para gerar fluxos de 64 *bytes*, a diferentes taxas de transmissão, medir os atrasos sofridos pelos pacotes de cada um deles, e efectuar cálculos estatísticos (atraso mínimo, médio, máximo, etc,) por intervalo de medição. O tráfego gerado foi distribuído de forma uniforme por cada fila. Foram simuladas as seguintes situações de carga no encaminhador: 2, 40, 50, 250, 500, 750, 1000, 1250 e 1500 *Kbps*. Cada teste, correspondente a um determinado nível de carga, teve a duração de 3 minutos e foi repetido 3 vezes.

3.2. Análise dos Resultados

Os gráficos que se seguem representam a variação (ou diferença) do atraso médio, verificada entre duas classes para uma mesma situação de carga no encaminhador. Os valores

[6] Como trabalho futuro será incluído o tratamento das perdas.
[7] Os *mbufs* (ou *Memory Buffers*) são estruturas de dados utilizadas pelo *kernel* do SO *FreeBSD* para transportar todo o tipo de informação desde a camada de interface até à camada das aplicações, e vice-versa.

8 A execução de testes de desempenho permitiu concluir que a sobrecarga provocada no *kernel* pelo nosso protótipo é pouco significativa, mesmo em situações de congestão no encaminhador, não tendo reflexos com significado no desempenho do mesmo.

representados em cada gráfico, são obtidos através da diferença entre os valores do atraso médio dos pacotes da classe menos sensível e os mesmos valores da classe mais sensível. Desta forma, um valor positivo no gráfico, indica que a classe mais sensível tem um tratamento melhor do que a classe menos sensível (ou mais desfavorecida), o que corresponde ao comportamento desejado.

1ª Sequência de Testes

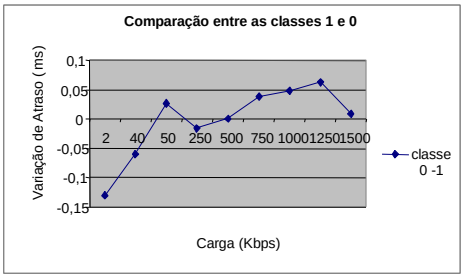
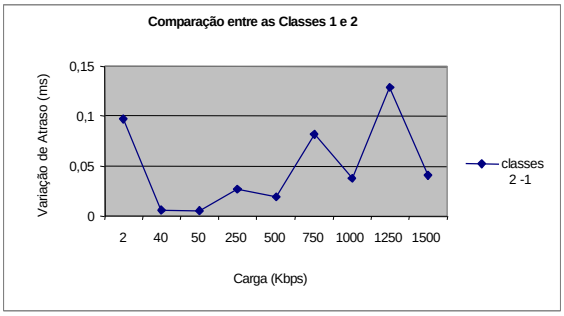
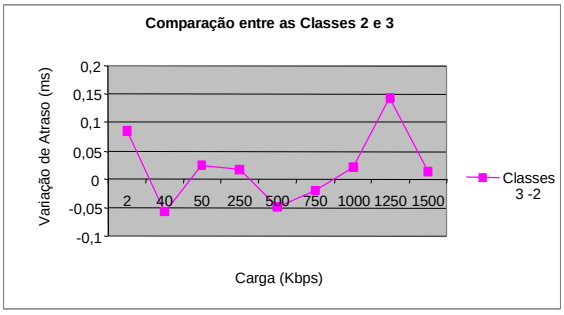


Figura 5 – Comparação entre os atrasos médios sofridos pelas classes 1 e 0 em diferentes situações de carga.

Através do gráfico na figura 5 pode-se verificar que para os maiores valores de carga a classe 1 é sempre melhor tratada do que a classe 0 e que tal tem tendência para se acentuar com o aumento da carga.

2ª Sequência de Testes





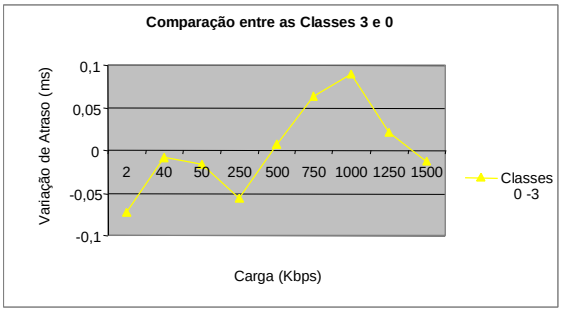


Figura 6 – Comparação entre os atrasos médios sofridos pelas classes 1 e 2; 2 e 3; 3 e 0, em diferentes situações de carga.

Nos testes aqui apresentados, pode verificar-se que:
- em qualquer situação de carga, a classe 1 tem o melhor tratamento do que qualquer uma das restantes classes.
- para situações de maior carga a classe 2 tem um tratamento melhor que a classe 3 e 0.
- a classe 3 verifica, para situações de maior carga, um melhor tratamento que a classe 0, apesar de não ser tão acentuado.

Resumindo, os resultados revelam que, tendencialmente, as classes menos favorecidas são as que apresentam piores valores de atraso. Pode então concluir-se que de uma forma geral, o protótipo está a ter comportamento esperado.

No entanto, apesar destes resultados revelarem tendências interessantes, e consistentes, admitimos que seria desejável que essas tendências se manifestassem de uma forma mais acentuada, com maior proporcionalidade e picos de variação menos intensos.

No nosso trabalho reconhecemos as causas seguintes para essas fraquezas:
- os testes não foram feitos a carga suficientemente elevada (por limitação de *hardware*). Ou seja, a infraestrutra de testes necessita de ser alterada de modo a criar situações de maior congestão no encaminhador.
- o algoritmo de disciplinamento não está a comportar-se conforme o esperado. Vários testes realizados em paralelo no LCT [12], revelaram que mecanismos de disciplinamento do tipo do utilizado neste trabalho apresentam limitações importantes no seu funcionamento. Por forma a contornar este problema considera-se necessário proceder a alguns refinamentos neste algoritmo, ou então, adoptar um algoritmo de disciplinamento diferente.

3.3. Conclusão dos Testes

De um modo geral, os resultados obtidos nestes testes, permitem concluir que o protótipo implementado é capaz de fornecer QoS a diferentes classes de serviço independentemente do estado da carga, ou congestão, a que o elemento de rede está sujeito. Portanto, pode concluir-se que a métrica de monitorização de QoS subjacente ao algoritmo de disciplinamento, concebida para alterar dinamicamente a prioridade das diferentes classes, é interessante para construir sistemas de comunicação capazes de fornecer QoS. No entanto para avaliar com maior precisão o comportamento do protótipo é necessário proceder á execução de outros tipos de testes mais elaborados e num ambiente de teste mais sofisticado tecnicamente. Nesse sentido está previsto o seguinte:
- Proceder á execução dos testes aqui apresentados para situações de cargas mais elevadas. Deverá ser criada uma infraestrutura de testes que possibilite que o encaminhador seja sujeito a uma carga mais elevada e durante mais tempo.
- Efectuar alguns refinamentos aos algoritmos implementados e voltar a executar os testes.

4. Conclusões e Trabalho Futuro

Pretendeu-se com este trabalho desenhar e implementar um protótipo de um elemento de rede capaz de fornecer QoS a tráfego IP. O protótipo baseia-se no modelo *Differentiated Services* e tem como objectivo principal fornecer serviços diferenciados a classes de tráfego distintas, independentemente do estado de carga ou congestão nesse elemento de rede.

A diferenciação de QoS é conseguida através de um mecanismo de disciplinamento extremamente simples, que se baseia num modelo de filas com diferentes níveis de prioridades, que determinam o número de pacotes a serem processados em cada visita do disciplinador. A aplicação da métrica de QoS desenvolvida no DEI permite a alteração dinâmica dos níveis de prioridade atribuídos às filas, de acordo com a qualidade de serviço que as classes de tráfego estão efectivamente a receber e com a carga a que o elemento de rede está sujeito.

Com o objectivo de avaliar o desempenho e comportamento ao protótipo desenvolvido, este foi sujeito a um primeiro conjunto de testes, cujos resultados, e respectiva análise, aqui foram apresentados. Através deles, pode concluir-se que o protótipo atingiu os seus objectivos, e que revelou boas potencialidades no que diz respeito à capacidade de se melhorarem as suas prestações.

Os próximos passos a dar no âmbito deste trabalho consistirão na realização dos testes mencionados no final da secção 3.3, na integração no protótipo do tratamento das perdas e na introdução de melhoramentos no algoritmo de disciplinamento.

5. Referências

[1] Internet Engineering Task Force, http://www.ietf.org/.

[2] R. Braden, D. Clark, S. Shenker, "Integrated Services in the Internet Architecture: An Overview", RFC 1633, June 1994.

[3] Integrated Services Working Group da IETF, http://iwww.ietf.org/html.charters/intserv-charter.html/.

[4] Differentiated Services Working Group, da IETF, http://iwww.ietf.org/html.charters/diffserv-charter.html/.

[5] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, "An Architecture for Differentiated Services", RFC 2475, December 1998.

[6] K. Nichols, S. Blake, F. Baker, D. Black, "Definition of the Differentiated Services Field (DS Filed) in the IPV4 and IPV6 Headers", RFC 2474, December 1998.

[7] G. Quadros, E. Monteiro, F. Boavida, "A QoS Metric for Packet Networks", Technical Report, Dezembro de 1998.

[8] Juha Heinanen, "Use of the Ipv4 TOS octet to Support Differential Services", Internet Draft, November 1997.

[9] Srinivasan Keshav, "On the efficient implementation of fair queueing. Internetworking: Research and Experience", September 1991.

[10] Avelans C., "Desenho e Implementação do Protótipo de um Elemento de Rede Capaz de Fornecer Differentiated Services", Projecto de Licenciatura, Departamento de Engenharia Informática da Universidade de Coimbra, Portugal, Julho 1999.

[11] http://manimac.itd.nrl.navy.mil/MGEN/.

[12] Quadros G., Alves A. ,"Relatório de Testes em Encaminhador com WFQ (Implementação ALTQ – FreeBSD)", Technical Report, Junho 1999.