UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

CENTRO TECNOLÓGICO

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

SALIM SUHET MUSSI

DIFERENCIAÇÃO DE FLUXOS SEM MANUTENÇÃO DE ESTADOS EM ROTEADORES

VITÓRIA

2011

SALIM SUHET MUSSI

DIFERENCIAÇÃO DE FLUXOS SEM MANUTENÇÃO DE ESTADOS EM ROTEADORES

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Engenharia Elétrica, na área de concentração Telecomunicações.
Orientador: Prof. Dr. Moisés Renato Nunes Ribeiro

VITÓRIA

2011

Dados Internacionais de Catalogação-na-publicação (CIP)
(Biblioteca Central da Universidade Federal do Espírito Santo, ES, Brasil)

Mussi, Salim Suhet, 1985-
M989d Diferenciação de fluxos sem manutenção de estados em roteadores / Salim Suhet Mussi. – 2011.
102 f. : il.

Orientador: Moisés Renato Nunes Ribeiro.
Dissertação (Mestrado em Engenharia Elétrica) – Universidade Federal do Espírito Santo, Centro Tecnológico.

1. Roteadores (Rede de computador). 2. Interconexão em rede (Telecomunicações). 3. Telecomunicações - Equipamento e acessórios. I. Ribeiro, Moisés Renato Nunes. II. Universidade Federal do Espírito Santo. Centro Tecnológico. III. Título.

CDU: 621.3

SALIM SUHET MUSSI

DIFERENCIAÇÃO DE FLUXOS SEM MANUTENÇÃO DE ESTADOS EM ROTEADORES

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Engenharia Elétrica, na área de concentração Telecomunicações.

Aprovada em 15 de Dezembro de 2011.

COMISSÃO EXAMINADORA

Prof. Dr. Moisés Renato Nunes Ribeiro
Universidade Federal do Espírito Santo
Orientador

Prof. Dr. Magnos Martinello
Universidade Federal do Espírito Santo

Dr. Marcos Rogério Salvador
Fundação CPqD

À família e aos amigos.

Agradecimentos

Gostaria de agradecer, em primeiro lugar, a minha família que sempre me amou, acolheu, confortou, incentivou e educou. Todos vocês tem fundamental importância em minha formação moral e intelectual.

Aos amigos, a família que escolhemos para nós, pelos momentos fundamentais de lazer, conversas, conselhos e incentivo. Em todo esse tempo dividimos alegrias que recordo com felicidade e angústias que nos fizeram crescer.

Não poderia deixar um agradecimento especial a todos os amigos que fiz no PoP-ES/RNP, com quem tive o prazer de trabalhar, aprender e talvez ensinar. Lá desenvolvi grande parte deste trabalho, utilizando a estrutura física e as máquinas cedidas. Considerável porção da motivação e incentivo veio do espírito de desenvolvimento e progressão que criamos.

Ao professor Moisés pela atenção, paciência e todo o tempo dedicado ao desenvolvimento deste trabalho.

Cultivamos a Ciência e queremos merecer a Poesia.

A Ciência é para os que estão aprendendo e a Poesia, para os que já sabem.

(Irvênia Prada e Hermínio C. Miranda)

Resumo

O tráfego da Internet é dominado por transações de curta duração. Todavia, apesar da grande quantidade, os fluxos curtos são responsáveis por uma pequena porção da carga total dos enlaces e ainda disputam, injustamente, recursos com conexões que transportam grandes volumes de dados. O desempenho de sessões TCP (*Transmission Control Protocol*) operando em fase de *slow-start* ou em regime de pequenas janelas sofre de forma significativa ao compartilhar *buffers* e capacidade dos enlaces com grandes rajadas oriundas de sessões na fase de controle de congestionamento. Uma forma de amenizar essa desigualdade é tratar diferenciadamente fluxos curtos e longos. Neste trabalho estudamos, desenvolvemos e implementamos técnicas sem manutenção de estados (*stateless*) de forma a atingirmos um bom compromisso entre eficiência e complexidade na diferenciação de serviço entre fluxos curtos e longos. Do ponto de vista metodológico, optamos pela implementação experimental sobre roteadores físicos e utilizando tráfego real. Esta escolha trouxe maior confiabilidade aos resultados, uma vez que eles não ficaram atrelados à qualidade dos modelos de simuladores, os quais são frequentemente simplórios demais para corresponder ao real comportamento de uma rede. Para a implementação de roteadores que viabilizassem a alteração dos esquemas de tratamento de pacotes utilizou-se a plataforma Click. Em relação à metodologia de testes, propomos um ambiente controlado que possibilita comparações entre diferentes técnicas sob tráfego real, oriundo de um backup de um *hard-disk* via FTP (*File Transfer Protocol*). Dentre as técnicas de diferenciação de fluxos, apresentamos, no conhecimento dos autores, a primeira implementação física de um roteador com o mecanismo RuN2C (*Running Number 2 Class*). Trazemos ainda a contribuição da investigação de seu desempenho quando diferentes técnicas de escalonamento são aplicadas. Propomos ainda um novo método de diferenciação de fluxos, denominado RAFLE (*Random Assorter of Flow LEngths*), que não exige nenhuma alteração de protocolos hoje existentes, o que facilitaria sua implantação em ambiente em operação. A classificação de pacotes pertencentes a fluxos longos e curtos é inferida a partir de uma pequena tabela com as informações de identificação dos últimos pacotes encaminhados, não sendo necessário manter estados dos fluxos ativos. Como resultados relevantes podemos destacar que o desempenho do RAFLE supera o RUN2C e aproxima-se bastante do desempenho da diferenciação com conhecimento completo dos fluxos (*full-state*) em diferentes cenários de tráfego.

Abstract

Internet traffic is dominated by short data transfers. However, short flows account for a small portion of the total link capacity. In addition, short flows unfairly compete for resources with connections that carry large volumes of data. The performance of TCP (Transmission Control Protocol) operating at slow-start phase (or under the small transmission windows) are impaired when sharing buffers and with long bursts coming from sessions at the stage of congestion control. One way to mitigate this inequality is to treat short and long flows differently. In this work we investigate stateless techniques in order to achieve a good compromise between efficiency and complexity of service differentiation between short and long flows. From the methodological point of view, we adopt an experimental approach using PC-based routers under real traffic. The goal is to achieve trustworthy to the results, since they are not tied to particular features of simulation models, which are often overlook the actual behavior of a network. We use the Click platform for routers' implementation. The evaluation methodology for the different mechanisms is based on a controlled environment considering a backup of a hard-disk via FTP (File Transfer Protocol). We present what is, in the authors' best knowledge, the first physical implementation of a router with the RUN2C mechanism (Running Class Number 2). In addition, we investigate its performance for different scheduling techniques. Our main contribution is a new method for flow differentiation called RAFLE (Random Lengths Assorter of Flow). In contrast with RuN2C, RAFLE requires no change of current network protocols. The classification of packets belonging to long and short flows is inferred from a small table with identification information of the last forwarded packets, but with no need for keeping the full flow states. Results show that RAFLE's performance exceeds RUN2C's. Moreover, RAFLE is able to virtually reach the performance of systems with full knowledge flows (full-state) in different traffic scenarios.

Lista de Figuras

Lista de Tabelas

Lista de Siglas

AF	Assured Forwarding
BRR	Bit-by-bit Round Robin
CDF	Cumulative Distribution Function
DiffServ	Differenciated Services
DT	Drop Tail
DRR	Deficit Round Robin
DRR-SFF	DRR Short Flow First
FQ	Fair Queueing
FTP	File Transfer Protocol
FIFO	First-In, First-Out
FPGA	Field-Programmable Gate Array
GPS	Generalized Processor Sharing
IntServ	Integrated Services
IP	Internet Protocol
NS2	Network Simulator 2
PQ	Priority Queue
PrioSched	Priority Scheduling
PS	Processor Sharing
QoS	Quality of Service
RAFLE	Random Assorter of Flow Lenghts
RED	Random Early Detection
RCP	Rate Control Protocol
RIO	RED with In and Out
RR	Round Robin
RuN2C	Running Number 2 Class
SLA	Service Level Agreement
TCP	Transmission Control Protocol
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuits
WFQ	Weight FQ
WRED	Weight Random Early Detection
WDRR	Weight DRR

Sumário

1 O Trafego na Internet e a Diferenciação de Fluxos

1.1 Introdução

A interconexão das diversas redes de dados existentes no mundo acelerou o desenvolvimento científico global. Inicialmente, as redes eram destinadas a atividades acadêmicas e militares para a troca de informações entre diferentes instituições. Entretanto, a revolução dos meios de comunicação tornou acessível a conexão do usuário doméstico a este meio de criação e difusão de conteúdo, a Internet, inegavelmente, uma das coisas mais importantes da sociedade moderna.

A popularização da Internet se deu a partir da diminuição dos custos para transmissão de dados. Isto provém, dentre outros motivos, do melhor aproveitamento dos recursos investidos nas redes de comunicação pela diminuição da ociosidade dos enlaces. O compartilhamento do meio de transmissão deixa de orientar a comunicação à alocação de circuitos dedicados, comutação de circuitos, passando a transmissão de datagramas, surge a era da comutação de pacotes.

Um datagrama é uma entidade de dados completa e independente que contém informações suficientes para ser roteada da origem ao destino sem precisar confiar em permutas anteriores entre essa fonte, a máquina de destino e a rede de transporte (MARINE; REYNOLDS; MALKIN, 1994). Dentro da família de protocolos TCP/IP, é o protocolo IP o responsável por padronizar os datagramas, cuidando inclusive da fragmentação de grandes datagramas, para que estes possam ser transportados sob qualquer rede que limite o comprimento máximo. O controle da transmissão, de forma a garantir a integridade da informação entregue deve ser realizado nas camadas superiores.

Assim, para que as inúmeras aplicações atuais pudessem se comunicar, técnicas específicas para a troca de informações foram desenvolvidas. Pesquisas tendo o objetivo de interconectar cada vez mais nós às diferentes redes, aproveitar melhor os recursos dos enlaces existentes e

prover tratamento diferenciado, qualidade de serviço (QoS), a aplicações que compartilham uma mesma rede física mas que possuem requisitos mínimos distintos.

Essa crescente exigência por qualidade de serviço adiciona complexidade ao processamento dos pacotes, uma vez que perdas e atrasos dos datagramas devem ser controlados. Portanto, políticas mais eficientes são necessárias para adequar estas diferentes taxas de crescimento de tráfego e da capacidade de processamento. Neste sentido, esforços estão sendo direcionados para o desenvolvimento de técnicas que possibilitem, a partir de um baixo processamento eletrônico e baixa exigência de memória, atender a esta demanda. O uso mais equânime dos recursos disponíveis é também uma crescente preocupação presente. Uma destas técnicas é a priorização de fluxos curtos sem manutenção de estados em roteadores para garantir a sua escalabilidade (MUSSI; RIBEIRO, 2009) (AVRACHENKOV; BROWN; NYBERG, 2004).

Neste capítulo abordaremos, primeiramente, a caracterização do tráfego e sua importância na Seção 1.2. Logo após, na Seção 1.3 será apresentado os desafios da diferenciação de tráfego e os meios de realizá-la. Em seguida, na Seção 1.4, faremos um levantamento dos principais trabalhos relacionados ao trabalho da Dissertação e na Seção 1.5 apresentaremos uma proposta para atender o requisito de alocação equânime dos recursos disponíveis em um roteador IP.

1.2 Caracterização de tráfego

A diferenciação e a priorização de tráfego podem ser aplicadas em diferentes cenários. Sua eficiência será decorrente da qualidade do método empregado e da adequação dos parâmetros às características do tráfego a ser beneficiado. Dois cenários possuem especial destaque no aprovisionamento de serviços e consequentemente na necessidade de melhor aproveitamento dos recursos de rede disponíveis: a Internet e os *data centers*.

O cenário de tráfego da Internet é difícil de ser bem caracterizado analiticamente. A alta variabilidade do tamanho das transferências de arquivos, o uso de um protocolo extremamente elástico (i.e., *Transmission Control Protocol*, TCP) que se adapta às diversas condições da rede e o surgimento constante de novos serviços são apenas alguns dos fatores complicadores na tarefa de modelagem desta rede. No entanto, um fato evidente é que o tráfego da Internet é dominado por fluxos TCP de curta duração (LABOVITZ et al., 2009). Fluxos inferiores a dez pacotes representam 95% do tráfego do cliente pro servidor TCP e 70% do tráfego em sentido oposto (CIULLO; MELLIA; MEO, 2009). Em parte, isto se dá pelo predomino da interação dos navegadores com *Web* pages. Seguindo a tendência do aumento das conexões curtas, os aplicativos *torrent* subdividem grandes transferências num conjunto de pequenos arquivos os

quais se dispersam na rede, em virtude da diversidade de origens. Todavia, apesar da grande quantidade, os fluxos curtos ainda são responsáveis por uma pequena porção da carga total dos enlaces e ainda disputam, injustamente, recursos com conexões que transportam grandes volumes de dados. O desempenho de sessões TCP operando em fase de *slow-start* ou em regime de pequenas janelas sofre de forma significativa ao compartilhar *buffers* e capacidade dos enlaces com grandes rajadas oriundas de sessões na fase de controle de congestionamento (ANELLI; LOCHIN; DIANA, 2011).

Grande parte do conteúdo acessado pelos usuários da Internet fica, naturalmente, hospedado em *data centers*. Assim, numa análise fim-a-fim, a topologia de interconexão dos servidores entre si e com a Internet pode influir na percepção do usuário final. Quanto mais usuários um serviço possui, mais servidores serão alocados para este fim.

A demanda por largura de banda entre servidores dentro de um *data center* apresenta maior crescimento que a de *hosts* externos. A rede é um gargalo para a computação. Frequentemente vê-se *switches* de topo de *rack* cujos uplinks estão acima de 80% de utilização. No entanto, assim como na Internet, o tráfego no interior de um *data center* é composto em sua maioria por pequenos fluxos, que transportam poucos *KBytes* (BENSON; AKELLA; MALTZ, 2010). Em sua grande parte, estes fluxos são compostos de *hellos* e requisições de meta-dados em sistemas de arquivos distribuídos (GREENBERG et al., 2009).

Para melhor quantizar a caracterização de fluxos em um data center é apresentada na Figura 1.1 uma distribuição dos tamanhos dos fluxos levantada a partir de um *cluster* de 1500 servidores conectados através de 75 *switches* de topo de *rack*. Pode-se notar através da C.D.F. (*Cumulative Distribution Function*) que 80% dos fluxos transportam menos de 10 *KBytes*, e que mais de 95% do volume de *bytes* transportados estão entre os fluxos de 10MB e 1G.

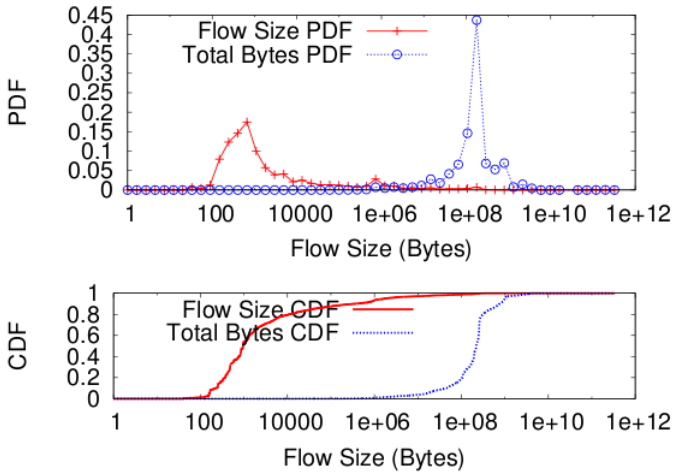


Figura 1.1: Caracterização do tráfego de um *data center* (GREENBERG et al., 2009)

Nos dois cenários vê-se que a maior parte dos fluxos são pequenos, mesmo que nos data centers a distribuição deles seja mais simples e uniforme (GREENBERG et al., 2009). No entanto, apesar da grande quantidade, os fluxos curtos ainda são responsáveis por uma pequena porção da carga total dos enlaces e ainda disputam, injustamente, recursos com conexões de altíssima duração resultantes da transferência de grandes arquivos de uma única origem.

Assim, esforços também estão sendo direcionados para que se desenvolvam técnicas de baixa complexidade que tentem diminuir o tempo de transferência de pequenos arquivos e que promovam, portanto, uma alocação mais justa dos recursos disponíveis sem sobrecarregar o processador e a memória do roteador. Entende-se como uma alocação justa de recursos quando os atrasos experimentados pelas sessões são apenas proporcionais ao volume de dados a serem transferidos. Em outras palavras, que as sessões com baixo volume de tráfego, que eventualmente não passam da fase de *slow start* do TCP, não sejam influenciadas pelas grandes rajadas características dos fluxos de longa duração já operando na fase de controle de congestionamento.

1.3 Disciplinas de Diferenciação de Tráfego

Na Internet, diferentes aplicações compartilham os mesmos enlaces. Todavia, os requisitos para transmissão de dados de cada uma destas podem ser distintos, trazendo a necessidade de tratamento diferenciado. Aplicações de tempo real necessitam de baixo atraso médio na entrega dos dados. Por outro lado, aplicações de transferência de grandes volumes de dados, normalmente necessitam de disponibilidade de banda nestes enlaces, não tendo tanta importância o atraso na entrega de seus pacotes, desde que a taxa de entrega deles seja alta. Aplicações críticas possuem a exigência básica que os dados sejam transmitidos integramente. Outro requisito em aplicações em tempo real é a variância do atraso (*jitter*).

Estes requisitos de transmissão devem ser atendidos desde a origem até o destino. Porém, quando se trata de tráfego na Internet isto é muito difícil de ser garantido, já que os dados trafegam por diferentes redes sob diferentes administrações. Entretanto, garantir ganho no requisito necessário salto-a-salto em redes conhecidas pode ser uma forma de mitigar o problema.

A composição do atraso na transmissão de um pacote em um nó (a_{no}) é formada pela soma dos atrasos de processamento destes pacotes (a_{proc}), atraso médio no sistema de enfileiramento (a_w) e do atraso médio de propagação (a_{prop}), como formalizado na equação 1.1.

$$a_{no} = a_{proc} + a_w + a_{prop} \tag{1.1}$$

$$a_w = E(T) = E(n)/\lambda \tag{1.2}$$

$$a_{prop} = D/s \tag{1.3}$$

O atraso de processamento (a_{proc}) é o tempo gasto para processar o pacote desde a identificação dos cabeçalhos até a entrega no *buffer* da interface de saída do nó. A diminuição deste tempo é possível pela alteração de *hardware* ou otimização dos algoritmos do sistema operacional do nó.

Já o tempo médio no sistema de enfileiramento em regime permanente (a_w) é relacionado pela Lei de Little, mostrada na Equação 1.2. O tempo médio de resposta, $E(T)$, é dado pela divisão entre o número de pacotes no sistema, $E(n)$, e a taxa de chegada de pacotes, λ. A redução deste atraso pode se dar através de algoritmos de escalonamento que priorizem a transmissão dos pacotes de aplicações sensíveis a atraso. É justamente nesta componente que nesta componente que o presente trabalho visa atuar.

Os demais atrasos são alterados por questões de *hardware* e questões físicas do meio de propagação. O atraso de transmissão é contabilizado pelo tempo levado para transformar todos os bits do pacote em sinais compatíveis com o meio de propagação. E o atraso de propagação (a_{prop}) é dado pela relação entre a distância do enlace (D) e a velocidade de propagação do meio (s) evidenciada na Equação 1.3.

Para tentar atender a estes requisitos, primeiramente é necessário distinguir a transferência de dados de cada aplicação, identificando cada datagrama, nó-a-nó da rede, para, em seguida, atribuir prioridades diferentes no encaminhamento destes até o próximo salto. Isto pode ser feito de duas formas, identificando os datagramas prioritários através do valor de um dos campos no cabeçalho da camada de rede, ou identificando, de forma mais específica, os pacotes pertencentes às sessões estabelecidas pelas aplicações através de um padrão de fluxo de dados. Ao conjunto de pacotes identificados por este padrão denominaremos a entidade “fluxo”.

Todavia, cada um dos métodos tem diferentes implicações nos requisitos de processamento e de memória nos roteadores. Eles podem ter que manter ou não informações sobre os dados trafegados ou sessões estabelecidas pela rede caracterizando a manutenção ou a não manutenção de estados.

1.3.1 Diferenciação de Pacotes

A marcação das classes prioritárias na diferenciação de pacotes pode ser realizada através de métodos dinâmicos ou através de atribuições realizadas pelo administrador da rede. Para

conseguir qualidade de serviço fim-a-fim em grandes redes é necessário que haja negociação entre os provedores de acesso envolvidos. Os acordos gerados por essas negociações, definindo quais serão as classes de tráfego e quais serão os requisitos mínimos a cada uma, são chamados de acordos de nível de serviços (*Service Level Agreements, SLAs*).

Um dos métodos clássicos propostos, neste contexto, é o DiffServ, *Differentiated Services* (BLACK et al., 1998). Neste método, as prioridades dos pacotes são definidas pelo emissor conforme os acordos de nível de serviço. A marcação dos pacotes é realizada atribuindo diferentes valores ao campo ToS do cabeçalho IP. Desta forma, os datagramas podem ser classificados com bastante simplicidade, e em seguida aplicam-se as políticas de escalonamento de prioridades acordadas, não exigindo muito processamento, nem muita memória dos roteadores envolvidos.

Todavia, há várias desvantagens no uso do DiffServ. A simplicidade da marcação dos pacotes com o campo ToS gera a necessidade de determinar que valores corresponderão a que classes de prioridades. No entanto, essa escolha é um tanto quanto arbitrária, e isto é mais um assunto para os acordos de SLA. Quando o tráfego passa por mais de uma rede, a negociação destes valores torna-se mais complexa, uma vez que os valores escolhidos para as classes devem se correspondentes em todas as redes envolvidas. Outra questão a ser analisada é a facilidade que um usuário mal intencionado tem de interferir na solução. Atribuindo o valor correto do campo ToS a todos os seus pacotes, este usuário possui prioridade sobre os demais e, dependo do volume de dados originado ou recebido por ele, pode comprometer o desempenho dos serviços que realmente precisariam de tratamento diferenciado.

Assim, para que a diferenciação de pacotes seja menos susceptível a este tipo de ataque, é necessário que se desenvolva métodos de classificação mais automáticos, através de outras informações providas pelo cabeçalho do datagrama. Estas soluções mais elaboradas podem ser conseguidas dispondo de informações oferecidas por cabeçalhos de diferentes camadas.

1.3.2 Diferenciação de Fluxos

Há diversas formas de se definir um fluxo na literatura. Em (TSAI; CHUNG; TSAI, 2011), defini-se um fluxos como uma unidade lógica que representa uma sequência de pacotes. Associam-se, ainda, os fluxos a conexões de dados entre diferentes nós de uma rede. Cada pacote que entra em um roteador estaria associado a um fluxo e, na prática, pacotes pertencentes ao mesmo fluxo, frequentemente compartilhariam dos mesmos requisitos de qualidade de serviço.

Assim como esta definição, muitos estudos anteriores se concentraram em definições de fluxos em termos de manutenção de estados, derivadas da observação de abertura e fechamento explícitos de conexões TCP. Porém, em (CLAFFY; BRAUN; POLYZOS, 1995), apresenta-se uma metodologia para determinar os perfis de fluxos inspirada no modelo de trem de pacotes (JAIN; ROUTHIER, 1986). Um trem de pacotes é definido como uma rajada de pacotes que chegam ao mesmo destino originados de uma mesma fonte. Se o tempo entre a chegada de dois pacotes excede um tempo determinado, *timeout*, diz-se que eles pertencem a diferentes trens de pacotes, e logo a fluxos diferentes.

Mesmo que esta definição de fluxos não esteja associada, efetivamente, às conexões de dados (i.e. mecanismo SYN/FIN), mas a condições temporais e de localização espacial observadas dentro de cada nó da rede, ela, assim como a primeira definição exposta, necessita de uma forma de caracterizar os pacotes dentro de uma especificação de fluxo. A definição das estruturas de especificação de um fluxo pode ser agrupada sob quatro aspectos (CLAFFY; BRAUN; POLYZOS, 1995):

1. **Direcionalidade**: pode-se definir um fluxo como unidirecional ou bidirecional. O Tráfego orientado a conexão TCP geralmente exibe bidirecionalidade. Cada fluxo TCP de A para B gera um fluxo reverso de B para A para confirmação de entrega dos pacotes, e que normalmente transporta pequena quantidade de dados. Todavia, este tipo de tráfego poderia ser analisado separadamente, como dois fluxos unidirecionais distintos.

2. **Pontos de agregação de tráfego**: Os fluxos podem ser agregados na fonte ou no destino do tráfego, ou em ambos simultaneamente. Um exemplo é a diferenciação entre todo o tráfego destinado a uma rede A ou todo tráfego vindo da rede B e destinado a rede A. Também poderia-se associar este aspecto às portas de origem e destino dos protocolos de transporte, TCP/UDP.

3. **Granularidade do destino**: os fluxos podem ser agregados quanto à granularidade do destino. Granularidades potenciais incluem aspectos de tráfego como aplicação, usuário final, *hosts*, número IP da rede, Domínio Administrativo, provedor de serviços de rede. O roteamento IP utiliza de granularidades relacionadas aos números IPs, como por exemplo, as máscaras CIDR.

4. **Camadas de protocolos**: este aspecto especifica os fluxos pelas camadas funcionais da rede. Podem-se definir os fluxos pela camada de transporte via pacotes SYN-FIN, ou sob grupos de campos dos protocolos denominados tuplas. Um exemplo seria definir

um fluxo como os pacotes que possuem em seu cabeçalho os mesmos (IP de Destino, Identificação do Protocolo de Transporte, Porta de Destino).

Um dos métodos utilizados para diferenciação e priorização de fluxos é o Intserv (*Integrated Services*) (BRADEN; CLARK; SHENKER, 1994). Ele caracteriza o fluxo de forma unidirecional com base na tripla citada anteriormente (Endereço IP de Destino, Identificação do Protocolo de Transporte, Porta de destino) e atribui prioridade ao encaminhamento destes fluxos ao longo de toda a rede Intserv através de um mecanismo dinâmico de reserva de recursos de rede, o RSVP (*Resource Reservation Protocol*) (BRADEN et al., 1997). O método não se tornou popular quando foi publicado, pois um de seus requisitos é o armazenamento dos múltiplos estados relativos às reservas de recursos em todos os roteadores da rede ocupando muita memória, recurso custoso na época. Também, torna-se difícil gerir todas as reservas à medida que as redes se tornam maiores, tornando o Intserv inadequado para a Internet.

Há, portanto, diversas formas de se segregar o tráfego de uma rede em classes distintas, através de informações extraídas dos cabeçalhos dos pacotes. A priorização das classes de tráfego com maiores requisitos de SLA, de forma a melhorar a qualidade de serviço no atendimento destas, só é eficaz se os métodos de classificação automática e políticas de serviço, ou disciplinas de escalonamento, do tráfego segregado em classes distintas forem eficientes.

1.4 Trabalhos Relacionados

Em uma rede de pacotes podem ocorrer congestionamentos temporários ocasionados quando a taxa de pacotes encaminhados a uma interface é maior que a capacidade do enlace que a interconecta a outro nó da rede, ou quando a taxa de chegada de pacotes é superior à capacidade de processamento do nó em questão. Para controlar a situação, em cada interface de rede há uma memória temporária para armazenamento dos pacotes que chegam ou que devem ser transmitidos, os *buffers*. Uma disciplina de escalonamento define a forma como o nó, por exemplo, um roteador IP, irá tratar os pacotes armazenados nos *buffers* de entrada e saída. O armazenamento dos pacotes pode se dar em fila única ou em múltiplas filas. Quando é realizado em mais de uma fila, efetua-se uma classificação dos datagramas que serão direcionados a cada uma delas, como mostrado nas subseções anteriores.

Tratando-se de fila única, uma das formas mais simples de realizar este escalonamento é a disciplica FIFO (*First-In, First-Out*). Consiste em tratar todos os pacotes de forma igualitária, não utilizando nenhum conceito de priorização de tratamento. Os datagramas são enfileirados e tratados na mesma ordem que chegaram a interface de rede. Porém, o tamanho dos *buffers*

das interfaces é limitado e consequentemente limita o número de pacotes possíveis em uma fila. Quando o número de pacotes que chegam a uma interface excede a capacidade de armazenamento desta, nesta disciplina, datagramas são descartados até que ocorra a liberação de memória no *buffer*. Denominamos a ação de descarte dos últimos pacotes a chegar a uma fila pela expressão *Drop Tail*. O principal problema neste tipo de escalonamento é a possibilidade de fontes de tráfego intensas consumirem toda a largura de banda disponível, ou de tráfego em rajadas de longa duração causarem atrasos inaceitáveis a certos tipos de aplicações. Neste trabalho utilizaremos a expressão *Drop Tail* para indicar a política de descarte de uma fila FIFO, já que este é o termo amplamente usado e designa mais claramente a ação sobre os pacotes enfileirados.

Ainda, há políticas aplicadas a filas únicas que atuam no controle de congestionamento dos *buffers*. Uma delas é a Random Early Detection (RED). Esta disciplina monitora a média de ocupação de uma fila e descarta pacotes baseando-se na probabilidade estatística do esgotamento da fila. Quando uma fila cresce, a probabilidade de descarte de um pacote antes de ser enfileirado também aumenta, chegando a 100% quando a fila está cheia. Porém, há dois *bugs* no algoritmo RED clássico, descobertos e provados por Kathy Nichols. Suas melhorias, apesar desenvolvidas, nunca foram publicadas ocasionando a sua não disseminação e consequente não implementação de forma comercial (GETTYS, 2010).

Além disto, o algoritmo RED clássico não possuía a possibilidade de diferenciação para qualidade de serviço. Novos algoritmos derivados deste foram desenvolvidos de forma a prover esta capacidade, utilizando mais de uma fila de prioridades, tais como, o *Weighted RED* (WRED) e o *RED with In and Out* (RIO) (CLARK; WROCLAWSKI, 1997). Porém a utilização eficiente destes depende do ajuste de vários parâmetros que caracterizam o comportamento do tráfego na rede, o que, atualmente, com o surgimento de cada vez mais aplicações distintas, se torna demasiadamente complexo.

Quando se trata de múltiplas filas, após a classificação em diferentes classes e armazenamento dos pacotes em filas distintas, os pacotes devem ser encaminhados de acordo com a prioridade desejada para cada classe. Para isto, há vários algoritmos de escalonamento com diferentes graus de complexidade. Um dos mais simples é o algoritmo *Round Robin*, que consiste em alocar uma janela de tempo igualitária para serviço sequencial de cada fila classificada. Esta janela pode durar tempo suficiente para encaminhar apenas um pacote, menor unidade de dados em uma rede TCP/IP. Um melhoramento deste algoritmo denominado *Deficit Round Robin*. Nele a janela de tempo de serviço é destinada apenas a filas ocupadas (SHREEDHAR; VARGHESE, 1995). E quando se trata de escalonamento de prioridades, há a variação *Weight*

Round Robin (WRR), na qual um peso é atribuído a cada classe de tráfego. A determinação do percentual da banda que será alocada para cada classe de tráfego se dá através da expressão $P_i = (W_i/S) * B$. Onde P_i representa a taxa em bps alocada para o tráfego da classe i; W_i, o peso associado à classe de tráfego i; S, o somatório de todos os pesos atribuídos as classes de tráfego; e B, a banda total do enlace. (CISCO..., 2002)

Há também, o algoritmo de Prioridade Estrita, também chamada de *Priority Queue* (PQ) ou *Priority Scheduling* (PrioSched). Consiste em encaminha os pacotes de uma fila com maior prioridade e apenas quando esta se esvazia destina o tempo de serviço a próxima classe prioritária. Neste algoritmo deve-se ter uma preocupação com as classes de tráfego de maior prioridade, pois um volume grande de tráfego prioritário pode reter fluxos de menor prioridade por um tempo inaceitávelmente elevado. Este fenômeno é conhecido como *starvation*. Uma forma de contornar este problema é utilizando ferramentas de formatação de tráfego, de modo a restringir a taxa de chegada de tráfego de alta prioridade. (CISCO..., 2001)

No sentido de equidade de serviço aos fluxos passantes por um nó é importante descrever o algoritmo *Bit-by-bit Round Robin* (BRR), também chamado de *Generalized Processor Sharing* (GPS). Nesta disciplina cada fluxo identificado é mantido numa fila exclusiva de saída e um *bit* de cada fluxo é enviado pelo enlace de saída a cada ciclo de serviço. Desta forma, o BRR dividiria de forma exata a banda disponível entre os fluxos ativos. Isto faz com que o BRR seja um algoritmo ótimo do ponto de vista de equidade, mas de implementação inviável, já que um canal de controle que compartilhe a informação de quais fluxos estão ativos e a qual fluxo pertence cada bit transmitido teria que ser mantido, gerando demasiado *overhead*. Contudo, é nesta disciplina que a política de escalonamento *Fair Queueing* (FQ) é inspirada (MATA, 2002). No FQ existe uma única fila de saída. A prioridade de serviço de um novo pacote é dada de acordo com um cálculo do instante de tempo (t_p) em que o último bit deste pacote seria transmitido através do algoritmo BRR. Os pacotes com menor t_p são posicionados na fila de forma a ter maior prioridade sobre os que possuem maiores t_p, e assim são transmitidos de forma integral, e não mais bit-a-bit. Nota-se que no FQ ainda há a necessidade de determinar e manter as informações sobre quais fluxos estão ativos a cada instante, ou de forma mais genérica identificar padrões de fluxos, classes, que serão tradados de forma igualitária.

No sentido de prover QoS a aplicações, o FQ pode não atender os requisitos de necessários. Nem sempre equidade de tratamento é suficiente, às vezes, é necessário prioridade de tratamento. Para isto uma variação do algoritmo foi criada de forma a se poder atribuir pesos as classes identificadas, o *Weight Fair Queueing* (WFQ)(CISCO..., 2001). Esta disciplina atua sobre as classes atribuindo a cada pacote um peso no cálculo do parâmetro t_p, mesma expressão

utilizada disciplina FQ. Isto implica na transmissão de mais pacotes de uma classe com maior prioridade.

Nota-se a existência de inúmeras possibilidades para as disciplinas de escalonamento de tráfego. No presente trabalho, adotaremos como referência a utilização de uma única fila única, FIFO. Os resultados obtidos serão comparados com o escalonamento de tráfego classificado em duas filas através das disciplinas PrioSched e RR. A escolha está relacionada à complexidade dos algoritmos. Como se pretende obter métodos de classificação de fluxos de forma automática e sem a necessidade de interação com o usuário ou aplicações executadas sobre a rede, experimentando os rotedores sob tráfego real, diminuir a complexidade de processamento das disciplinas de escalonamento das classes de prioridades se faz necessário.

Há uma infinidade de métodos de diferenciação e priorização de fluxos na literatura. Em (NOUREDDINE; TOBAGI, 2002) são discutidas duas possíveis soluções. A primeira é baseada na diferenciação de serviços (DiffServ), utilizando *Assured Forwarding* (AF) e *RED with In and Out* (RIO). Esta solução requer a escolha não trivial de inúmeros parâmetros para os mecanismos AF e RIO. Outra solução apontada é um mecanismo com manutenção de estados dos fluxos TCP, que utiliza do tamanho das janelas de cada sessão e conta com a observância dos *hosts* de destino. Desta vez, a solução requer o *tuning* de parâmetros para o *Weighted Round Robin* (WRR) que novamente requer uma escolhas não evidentes e consequentemente se trata de outra solução não robusta.

Em (SUN et al., 2007) é proposto um método para realizar a diferenciação de fluxos através da contabilização dos *bytes* transmitidos em cada sessão denominado *Deficit Round Robin with Short Flow First* (DRR-SFF). Para isto, é necessária a manutenção da quíntupla de caracterização de cada fluxo (IP e porta de origem e destino, além do protocolo) em uma tabela, além do total de *bytes* transmitidos. Os pacotes de cada fluxo são divididos através de um limiar em duas filas distintas escalonadas via *Weighed Deficit Round Robin* (DRR). Mostram-se resultados positivos, todavia, somente através de simulações numéricas, na redução do tempo médio de transmissão dos fluxos em relação ao SSF, que utiliza prioridade estrita para a fila de fluxos curtos. Porém, tal mecanismo demanda muita memória para armazenamento dos estados e elevado processamento diferenciação dos fluxos sendo passível de implementação apenas em roteadores de borda.

Em (DUKKIPATI, 2007) é proposto um protocolo de transporte com controle de congestionamento em roteadores, alternativo ao TCP, denominado *Rate Control Protocol* (RCP). Ele visa proporcionar o menor tempo possível de finalização para os fluxos ativos. O RCP é uma emulação da divisão ideal de Processor Sharing (PS), onde o roteador tenta dividir igualmente os

recursos entre os fluxos ativos propiciando ao usuário de aplicações mais interativas com trocas de dados de pequenos volumes seja melhor atendido num ambiente com competição com fluxos que intercabiam grandes arquivos. Esta política possui a vantagem de a duração dos fluxos ser invariante à distribuição granulométrica dos fluxos e de ser intrinsecamente justa. Contudo, a sua incompatibilidade com o TCP torna custosa a sua implementação em larga escala.

Em (AVRACHENKOV; BROWN; NYBERG, 2004) o mecanismo sem manutenção de estados *Running Number 2 Class* (RuN2C) é proposto. Tal mecanismo consiste na divisão entre fluxos curtos e longos em filas distintas utilizando a informação de números de *bytes* transferidos dos números de sequência dos pacotes TCP para comparação com um limiar de decisão. Uma política de prioridade estrita é adotada para o escalonamento destas filas. Nesta política a fila de fluxos longos não é servida até que a de fluxos curtos esteja vazia. Este método pode ser implementado em diferentes roteadores, tanto em redes de acesso como em redes de *backbone*, por não ser necessária a manutenção de estados dos fluxos (i.e., *stateless*). Além disto, a implantação pode se dar de maneira gradual não necessitando de prévio acordo entre as operadoras. Entretanto, somente resultados de simulação (em NS2) e com tráfego sintético são apresentados para demonstrar a validade da técnica.

Recentemente, tornou-se popular uma tecnologia de identificação e tratamento de fluxos denominada OpenFlow (MCKEOWN et al., 2008). Baseia-se em um *switch* Ethernet, com uma interface padrão para adicionar e retirar regras de encaminhamento em uma tabela interna de fluxos. Estas regras são definidas por um controlador exterior aos elementos de encaminhamento de pacotes da rede. Desta forma, um controlador pode ser responsável por definir regras para inúmeros *switches* em uma rede, simplificando a gerência de rede e tráfego tanto em ambientes empresariais quanto em *data centers*, provendo visibilidade global dos fluxos.

Todo pacote que chega ao *switch* é processado de forma a extrair todas as informações possíveis para identificação do fluxo a que pertence. Essas informações podem incluir campos desde a camada de enlace até a de protocolos de transporte. Em seguida a tabela de encaminhamento de fluxos é consultada para verificação de alguma regra compatível com a identificação do pacote em questão. Caso esta seja encontrada, o pacote é enviado para a interface de saída correta. Caso não seja encontrada nenhuma regra, no modelo mais comum de funcionamento, este pacote é encapsulado e encaminhado via o canal seguro de comunicação para o controlador, que enviará como resposta uma entrada para tabela de fluxos de todos os *switches* sob seu controle que estão sobre o caminho definido para este fluxo. Assim, agiliza-se o processo nos demais nós. Trabalhando com o modelo regras pré-instaladas, o pacote que não encontra regra compatível é descartado.

O OpenFlow como apresentado hoje, não é escalável. O controlador centraliza a gerência de encaminhamento de toda rede e pode se tornar um ponto grave de falha. E mesmo em soluções onde o controlador da rede opera de forma distribuída, os próprios *switches* OpenFlow podem ser um grande gargalo para a escalabilidade da solução. A melhor implementação conhecida pode iniciar apenas poucas centenas de fluxos por segundo. Uma estimativa deste valor no equipamento desenvolvido pela fabricante HP, o ProCurve 5406z (HP. . . , 2011), este número é de 275 fluxos por segundo (CURTIS et al., 2011). Ainda, para que o controlador tenha completa visibilidade da rede é necessário que os *switches* se comuniquem com ele a cada início de um novo fluxo. Isto adiciona muita carga ao plano de controle e muito atraso durante o início de um fluxo em uma nuvem OpenFlow (CURTIS et al., 2011). Esta introdução de latência, mesmo quando da ordem de 1ms, é intolerável para aplicações que possuem fortes requisitos (ALIZADEH et al., 2010).

Por mais que a tecnologia OpenFlow não esteja ainda suficiente madura para implementar efetivamente o núcleo da Internet, soluções como o DevoFlow (CURTIS et al., 2011) propõem algumas alterações não tão radicais ao projeto desta tecnologia. As principais alterações estão no sentido de diminuir as interações *switch*-controlador introduzindo mecanismos que permitam aos *switches* tomarem decisões de roteamento locais para encaminhamento de fluxos que não exijam habilitação do controlador. Propõe-se ainda, que não seja necessário possuir visibilidade total das interações na rede. Além disto, demonstra-se que as decisões sejam tomadas a partir de estatísticas relativas aos enlaces associada a alguma visibilidade dos fluxos ativos na rede aumentando assim o número de iniciação de fluxos por segundo e a vazão nos enlaces. Porém, ainda não há implementação física desta proposta. Os resultados foram obtidos através de simulações.

Ainda no escopo da tecnologia OpenFlow, acreditando na possibilidade de seu desenvolvimento de modo que permita a escalabilidade, é apresentado o RouteFlow como uma forma de prover um novo serviço de rede que viabilize flexibilidade na interconexão de diferentes pontos através de uma operadora: o fornecimento de rede lógica composta por roteadores virtuais (NASCIMENTO et al., 2011). Ele é composto por um controlador OpenFlow, um servidor RouteFlow independente que gerencia um ambiente de rede virtual para interligar roteadores IP virtualizados, como por exemplo o Quagga (QUAGGA. . . , 2006), melhor definido na Capítulo 2.

Dado o custo de capacidade dos enlaces oferecido pelas operadoras, é sensato pensar na limitação finita deste recurso. E, que tanto para um serviço como RouteFlow, quanto para enlaces físicos contratados, há a necessidade de melhor uso da banda disponível de forma a

atender várias classes de aplicações com diferentes requisitos, melhorando, consequentemente, a impressão que os usuários têm da rede.

1.5 Proposta

O presente trabalho tem como objetivo principal a alocação justa dos recursos de rede compartilhados entre diversos perfis de usuário. Entende-se, aqui, como justa a alocação de recursos, quando os atrasos experimentados pelas sessões sejam proporcionais ao volume de tráfego transmitido. Utilizam-se, então, técnicas de diferenciação entre fluxos curtos e longos, estudando diferentes métodos de escalonamento de tráfego para estas duas classes, atuando na componente de atraso de enfileiramento, conforme descrito na seção 1.3, entendendo que a medida que a capacidade disponível aumenta, o peso desta componente em relação ao atraso total decresce proporcionalmente.

O crescimento da capacidade de transmissão pelos enlaces ópticos é superior à capacidade de processamento eletrônico das informações de roteamento/encaminhamento. Uma demanda crescente é pela alocação justa de recursos quanto ao tempo de serviço das sessões que devem ser proporcionais ao volume de tráfego transmitido. Assim, optou-se neste trabalho pelo estudo e desenvolvimento de técnicas sem manutenção de estados (*stateless*) para atingirmos um bom compromisso entre eficiência e complexidade na diferenciação de serviço entre fluxos curtos e longos.

Do ponto de vista metodológico, acreditamos que experimentos em roteadores físicos sujeitos a tráfego real trazem maior confiabilidade para os resultados. Simuladores ficam sujeitos aos erros introduzidos pelos modelos utilizados, muitas vezes demasiadamente simplórios para corresponder ao real comportamento de uma rede. Nesta linha optamos em utilizar a plataforma Click para a implementação dos roteadores que viabilizem a alteração dos esquemas padrões de tratamento dos pacotes. Para os experimentos utilizaremos tráfego TCP, protocolo difundidamente utilizado nas redes de computadores, através da iniciação em diferentes regimes de sessões FTP para transferência de arquivos.

A implementação dos roteadores deve ser realizada em plataformas abertas que viabilizem a alteração do esquema de enfileiramento/encaminhamento de pacotes, não provido por roteadores comerciais. Assim, no Capítulo 2 será discutida as tecnologias de desenvolvimento dos roteadores, tanto na visão de implementação via *hardware*, como na visão de utilização de *softwares* executados em computadores.

Na Figura 1.2 é destacado o caminho das tecnologias adotadas no desenvolvimento deste

trabalho. As escolhas aqui apontadas servem para evidenciar as diretrizes básicas e foram tomadas segundo alguns argumentos já apresentados neste capítulo e detalhamentos que serão apresentados em seções posteriores.

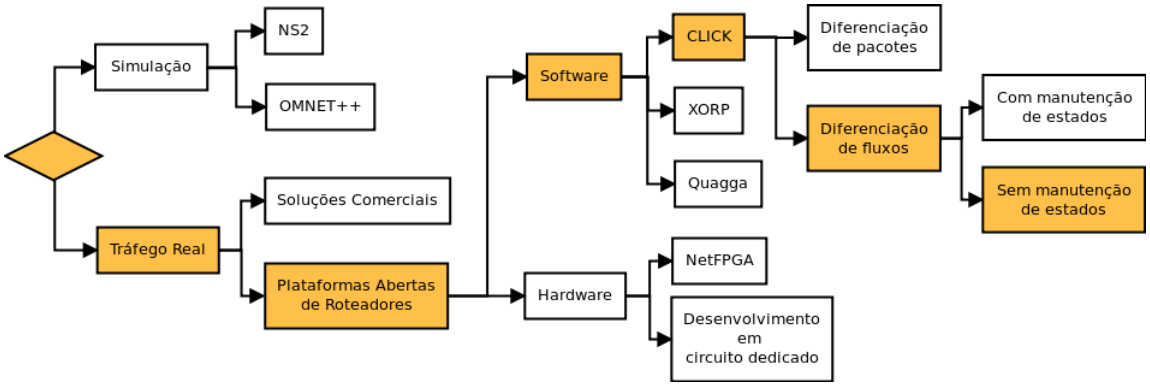


Figura 1.2: Tecnologias adotadas no desenvolvimento do trabalho

Assim, a nossa proposta para atingir maior equidade entre os fluxos ativos é atuar através de técnicas de escalonamento de tráfego na componente de atraso de enfileiramento em sessões TCP, utilizando por simplicidade duas classes: fluxos longos e fluxos curtos. Fluxos curtos normalmente são finalizados após a transmissão de poucos pacotes e assim, em geral deixam a fase de *slow-start*. Desta forma, durante toda a sua transmissão, competem com fluxos longos, já em fase de controle de congestionamento, alcançado taxas de transmissão muito abaixo da taxa justa, se a banda fosse compartilhada igualmente entre os fluxos ativos (DUKKIPATI; MCKEOWN, 2006). Diminuir o atraso dos pacotes pertencentes a estes fluxos nas filas, através da priorização deste tráfego é uma das formas de alcançar equidade no compartilhamento de recursos.

De modo análogo, para alcançar uma divisão mais equânime de recursos, pode-se limitar as taxas dos fluxos longos aumentando-se o atraso de seus pacotes. Isto acarreta consequente aumento do RTT (*Round Trip Time*) nas sessões. O tempo de resposta deste tipo de fluxo depende, principalmente, da fase de controle congestionamento, onde são transmitidos a maior parte dos dados. Enquanto a taxa de perda de pacotes, p, não for tão alta e a janela de recepção não for limitante, a taxa de transferência, T, obtida por um fluxo em regime permanente é dada pela relação aproximada (FREDJ et al., 2001):

$$T(p) \approx \frac{K}{RTT \cdot \sqrt{p}} \tag{1.4}$$

onde K é uma constante que depende de estatísticas de perda de pacotes de segunda ordem ($K = \sqrt{3/2}$ para perdas pediódicas (ALTMAN; AVRACHENKOV; BARAKAT, 2005)).

O mecanismo de controle de congestionamento do TCP proporciona que as conexões se

adaptem às condições de banda limitada de um enlace que atua como gargalo. Em regime, quando muitas conexões passam por um mesmo enlace o efeito do controle de congestionamento é o de compartilhar a banda disponível entre elas. No entanto, assumindo que todos os fluxos experimentem a mesma probabilidade de perda de pacotes, p, a equação 1.4 sugere que a banda é inversamente proporcional ao RTT. Em (FREDJ et al., 2001), este resultado é demonstrado, através de simulação, conforme apresentamos na Tabela 1.1. Na simulação utilizou-se 20 conexões TCP compartilhando um enlace com capacidade de 10Mbps; dentre elas 10 com $RTT = 50ms$ e 10 com $RTT = 100ms$, juntamente com um fluxo de 1Mbps de tráfego UDP *on-off* (incluído apenas para atenuar efeitos indesejáveis de sincronização devido a homogeneidade de conexões simuladas). Nota-se na Tabela 1.1 que as conexões TCP de mesmo RTT alcançam taxas muito semelhantes e que, como esperado, a banda atribuída é inversamente proporcional ao RTT; dos 9Mbps destinados às conexões TCP, 6Mbps são compartilhados pelas conexões de $RTT = 50ms$ e 3Mbps pelas de $RTT = 100ms$.

Tabela 1.1: Taxas de transferências em um enlace de 10Mbps compartilhado por conexões persistentes UDP e TCP (FREDJ et al., 2001)

RTT (*ms*)	Taxa de transferência dos fluxos TCP (Kbps)	Total (Mbps)
50	646 570 578 605 577 629 642 592 535 667	6,04
100	273 277 376 352 248 320 311 306 252 288	3,00

Desta forma, para atingirmos o objetivo de equidade entre os fluxos, utilizaremos mecanismos de diferenciação e priorização de fluxos curtos em detrimento dos longos, atuando no atraso de enfileiramento. Emprega-se o mecanismo RuN2C, inicialmente proposto em (AVRACHENKOV; BROWN; NYBERG, 2004), e o RAFLE, mecanismo desenvolvido pelos autores.

Do ponto de vista de metodologia de teste, a contribuição está nos experimentos em ambiente controlado, para possibilitar comparações entre as técnicas de diferenciação de fluxos, com tráfego TCP real. O TCP é um protocolo extremamente elástico e se adapta às condições da rede. Assim, para estudar a real influência das técnicas de diferenciação de fluxos utilizamos um cenário composto por um cliente e um servidor FTP interconectados através de um único roteador, de forma a podermos isolar os efeitos destas técnicas. Em cada tipo de experimento, as sessões FTP são iniciadas da mesma forma para cumprir as tarefas propostas, ficando sujeitas às reações do TCP sobre as condições da rede propiciadas pela técnica de diferenciação de fluxos utilizada.

No presente trabalho é apresentado, no conhecimento dos autores, a primeira implementação física de um roteador com o mecanismo RuN2C. Trazemos ainda a contribuição da investigação do seu desempenho sob diferentes técnicas de escalonamento. Ao contrário das conclu-

sões do artigo original (AVRACHENKOV; BROWN; NYBERG, 2004), onde o escalonamento com prioridade estrita é apontado como o de melhor desempenho, os nossos resultados experimentais apontam o *Round Robin* como a melhor estratégia de equidade no escalonamento. Assim como no RCP, nossa implementação experimental do RuN2C apresentou resultados bastante próximos ao ideal PrioSched, porém sem exigir alterações significativas do TCP requeridas pelo RCP. Os resultados destes experimentos já foram publicados na comunidade acadêmica em (MUSSI; RIBEIRO, 2009).

Propomos, também, um método de diferenciação de fluxos denominado RAFLE (*Random Assorter of Flow LEngths*). Ele é inspirado nos princípios da técnica CHOKe (WANG; TANG; LOW, 2003) desenvolvido para proteger fluxos TCP de inundações UDP. Estes princípios, aplicados ao contexto deste projeto, consistem em inferir a que classe de fluxo (longos ou curtos) pertence um pacote através de uma pequena memória recente, com a identificação de fluxo dos últimos pacotes encaminhados.

O RAFLE utiliza da característica de transmissão de pacotes em rajadas em redes TCP/IP, e que estas rajadas alcançaram maior número de pacotes (i.e. maiores janelas de transmissão) na fase de controle de congestionamento, onde normalmente estão os fluxos longos. Assim, quanto mais pacotes de uma rajada de um fluxo longo forem encaminhados, maior a probabilidade da identificação deste fluxo se encontrar na memória. Então, a inferência de que um pacote pertence a um fluxo longo é realizada através da constatação de igualdade entre a identificação de fluxo de um novo pacote que chega ao roteador com uma identificação sorteada de forma aleatória na memória recente. Dado a igualdade, o pacote é encaminhado a classe de fluxos longos, caso contrário é classificado como curto.

Uma vantagem do RAFLE é que não há necessidade de alterações nos protocolos já existentes. Além disto, os benefícios desta técnica são obtidos salto a salto, possibilitando a implementação gradual em ambientes em produção. Também, os mecanismos de classificação não utilizam manutenção de estados, possuindo baixos requisitos de memória e de processamento, recursos custosos que devem ser bem aproveitados de modo a possibilitar maior tratamento de tráfego.

2 *Fundamentos*

Para a realização da diferenciação dos fluxos em uma rede de pacotes é fundamental o conhecimento dos protocolos que serão utilizados. Através das informações de seus cabeçalhos em cada pacote, os fluxos poderão ser classificados. As características dos protocolos utilizados neste trabalho serão abordadas na Seção 2.1.

Para realização de pesquisas no âmbito de tratamento diferenciado do tráfego nos roteadores, sem o uso de simuladores, é necessário alterar o modo de funcionamento de um roteador existente ou que se desenvolva um roteador de fato. Roteadores comerciais têm o desenvolvimento de seu *hardware* e seus sistemas operacionais fechado à comunidade acadêmica, o que inviabiliza a alteração destes para este propósito. Na Seção 2.2 serão abordadas as principais plataformas abertas para o desenvolvimento de roteadores de forma a justificar a escolha de uma para realização deste trabalho.

2.1 Protocolos

De forma a diminuir a complexidade do desenvolvimento das redes de computadores facilitando a integração dos diversos padrões desenvolvidos, elas foram organizadas em camadas empilhadas. Cada uma destas camadas é responsável por abstrair parte do tratamento da informação que será transmitida. Neste modelo, uma camada inferior provê determinadas funcionalidades à camada superior. Os meios de transmissão, sejam linhas de transmissão metálicas, rádio frequência, fibra ótica, ou outro qualquer outro, estão localizados na camada mais básica, a camada física. Acima dela, localizam-se as camadas responsáveis pelo controle dos enlaces e acesso ao meio, pelas interconexões de diferentes redes, pelo controle do transporte das informações e pelas aplicações.

Há vários modelos de referência na literatura dividindo essas camadas, tais como o modelo TCP/IP (BRADEL, 1989) e o modelo OSI. Há ainda, um modelo híbrido definido em (TANENBAUM, 2003). A comparação destes modelos é apresentada na Figura 2.1.

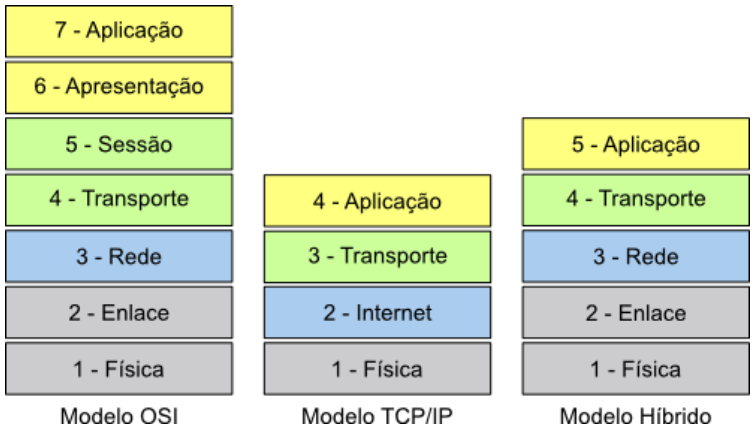


Figura 2.1: Modelos de referência de divisão em camadas (TANENBAUM, 2003)

Uma crítica realizada ao modelo TCP/IP é a não há diferenciação entre camada física e de enlace, que tratam de questões completamente diferentes. A primeira se preocupa com a característica dos meios de transmissão, já a segunda, com delimitar o início e o final dos quadros e enviá-los de um lado a outro da rede com o grau de confiabilidade desejado. A divisão do modelo OSI foi realizada durante a criação do modelo de forma mais política que técnica (TANENBAUM, 2003). Desta forma, o modelo híbrido, onde ocorre a divisão da camada física e de enlace, e a união das camadas foi o adotado no desenvolvimento da Internet.

Para realizar a diferenciação dos datagramas pertencente às diversas conexões estabelecidas em uma rede IP, é necessário acesso às informações dos cabeçalhos de protocolos em diferentes camadas. Assim, nesta seção, descreveu-se os protocolos das camadas de rede, transporte e aplicação, base dos experimentos com tráfego real realizados nesta dissertação, os protocolos IP, TCP e FTP, respectivamente.

2.1.1 IP

Para realizar a interconexão das diferentes redes locais mundiais sem a necessidade de unificar o sitema de MAC e de endereçamento das redes locais, é necessário que exista um esquema de endereçamento unificado projetado para sistemas de comunicação orientados à comutação de pacotes.

O *Internet Protocol* (IP) tem a função transmitir blocos de dados, denominados datagramas, da origem ao destino, identificando estes pontos por endereços de tamanho fixo. O protocolo prevê um mecanismo de fragmentação e agregação de grandes datagramas, se necessário, para a transmissão destes através de redes com tamanho máximo de pacotes limitado. Porém, não há agregado às funcionalidades deste protocolo, funcionalidades capazes de prover o aumento

da confiabilidade de entrega de dados fim-a-fim no sistema, o controle de fluxo e a entrega sequencial dos pacotes transmitidos. Estas funcionalidades, quando necessárias, ficam a cargo dos protocolos de camadas superiores (IP..., 1981).

Assim, a primeira exigência para interconexões de redes através do IP é a atribuição de endereços unívocos que identifiquem os participantes das redes interligadas, lembrando que um nó pode ser identificado por mais de um endereço. Quando se pensa em um modelo de interligação global, surge a necessidade de organização de regulação para a alocação destes endereços. A entidade responsável por essa função é a IANA (*Internet Assing Numbers Authority*). A ela estão subordinadas outras organizações de atuação continental, como é o caso do LACNIC (*Latin American and Caribbean Internet Addresses Registry*), responsável pela alocação e administração dos endereços IP na América Latina e Caribe.

Os endereços IPv4 (versão 4 do protocolo) são compostos de 4 octetos binários, totalizando 32 *bits*. E apesar da distribuição do último bloco de endereços IPv4, realizada pela IANA em abril 2011, restando agora apenas as reservas regionais, o esquema de endereçamento utilizado hoje na Internet ainda é baseado nesta versão do protocolo. A expansão dos endereços é realizada na versão 6 do protocolo, definindo seu tamanho em 128 bits. Por ora, o IPv6 não se encontra operacional em toda a rede mundial. Ainda há receio de vulnerabilidades e, além disto, é necessário tempo para a adequação das redes, serviços e conteúdo disponibilizados. É importante ressaltar, que não será realizada transição brusca entre as duas versões do protocolo. Serão utilizadas técnicas de coexistência das duas versões, como pilha dupla de protocolos, tunelamento, e tradução de endereços, não dispensando, assim, a atenção de pesquisas sobre a versão 4 do protocolo.

Para possibilitar o roteamento dos pacotes através destes endereços, eles são distribuídos aos Sistemas Autônomos (AS) em forma de blocos. Um bloco de endereços é designado pelo endereço nome da rede e uma máscara que indica a parte fixa dos endereços alocados. Ela é composta de 1's nos bits fixos, e de 0's nos bits variáveis onde cada combinação designa uma máquina da rede. Alguns blocos de endereços IPv4 são reservados para comunicações entre redes privadas, e estes não podem ser roteados globalmente.

Porém, o protocolo IP não é composto apenas do endereço de origem e destino. Outras funcionalidades foram acopladas, tornando-o mais flexível e confiável, através da definição de outros campos em seu cabeçalho. O cabeçalho IPv4 é apresentado na Figura 2.2. Nele destacamos o campo Protocolo que, como o próprio nome sugere, identifica o protocolo que será utilizado na camada superior, o campo Tempo de Vida (TTL) que atribui um número de saltos máximo possível que um datagrama pode efetuar, evitando-se, assim, que um erro de rotea-

mento encaminhe-o pela rede eternamente ocupando os recursos dos enlaces. É importante salientar que há ainda o campo de Checksum que se trata um código de verificação de integridade dos dados do cabeçalho.

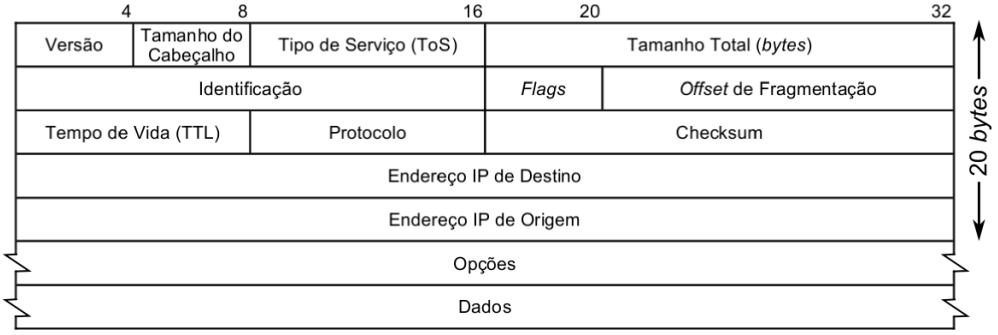


Figura 2.2: Cabeçalho IPv4 (IP..., 1981)

O mecanismo de fragmentação de datagramas é controlado pelos campos Identificação, *Flag* e *Offset de Fragmentação*. O primeiro indica o "número de série" do datagrama. No caso de fragmentação a qual datagrama pertence aquele fragmento, já que inúmeros pacotes podem estar sendo fragmentados na comunicação de dois *hosts*. As Flags podem ser utilizadas de forma a marcar um pacote IP não permitindo sua fragmentação. Existem alguns *hosts* que não aceitam datagramas fragmentados. Caso o datagrama não possa ser entregue sem fragmentação, ele é descartado. Há uma *flag* utilizada para indicar se o pacote atual é parte de um pacote fragmentado, ou se é o último fragmento deste. A ordem dos fragmentos é indicada no campo de *Offset* de Fragmentação.

Um usuário ou aplicação pode informar através o campo Tipo de Serviço (ToS) uma indicação de prioridade do pacote. Evidentemente, para que esta indicação seja acatada, todos os roteadores no caminho devem possuir funcionalidades de classificação e priorização como foi explanado na Seção 1.3.

Cabe ainda a observação de que o tamanho do cabeçalho IP não é fixo, seu tamanho mínimo é de 20 *Bytes* sendo extensível a outros fins, por meio do campo de bits Tamanho do Cabeçalho. As extensões do cabeçalho devem ser incluídas no campo Opções. O campo Tamanho Total inclui os campos de cabeçalho e o de Dados. É composto de 16 bits, e pode variar de 20 *Bytes* (mínimo tamanho do cabeçalho) a 64 K*Bytes*.

2.1.2 TCP (*Transmission Control Protocol*)

O IP não provê mecanismos de confiabilidade de entrega de pacotes, uma vez que estes podem ser descartados devido ao esgotamento dos *buffers* de roteadores e comutadores de pacotes.

Assim como não garante a entrega em ordem sequencial de envio, necessária para recomposição de arquivos transmitidos em mais de um datagrama. Além disto, o IP só traz a garantia de integridade de seu cabeçalho, mas não dos dados transmitidos. Cabe à camada de transporte definir, quando necessário, quais mecanismos de proteção serão implementados para sanar essas questões. Assim, uma das principais funções da camada de transporte é proteger as camadas superiores das imperfeições da rede.

O *Transmission Control Protocol* (TCP) foi projetado com a finalidade de obter alta confiabilidade em comunicações ponto-a-ponto sobre uma rede de computadores baseada em comutação de pacotes (TCP..., 1981). A abordagem efetuada nesta seção será apoiada nas RFCs 793, 1323, 2018, 2581, que padronizam o TCP, com foco na implementação Reno, utilizada nos sistemas Linux, que são de grande interesse o presente trabalho.

Uma das principais características do TCP é a de ser orientado à conexão. Desta forma, antes que a transmissão dos dados propriamente ditos seja iniciada, é necessário o estabelecimento de uma sessão entre as máquinas envolvidas. A abertura de conexões TCP se dá em três tempos, através de um mecanismo denominado de *three-way handshake*. Após o estabelecimento de uma sessão, as máquinas envolvidas podem trocar dados de forma bidirecional, caracterizando-a como uma conexão *full-duplex*.

A identificação de origem e destino dos datagramas, bem como a localização do caminho a ser percorrido para a entre destes, é realizada pelas funcionalidades do IP. Todavia, em uma máquina conectada a uma rede IP são executadas inúmeras aplicações simultaneamente que devem poder utilizar a rede de forma independente. A distinção dos datagramas gerados por cada uma destas aplicações é realizada na camada de transporte através dos campos de porta origem e destino presentes no início do cabeçalho deste protocolo. Na Figura 2.3 é apresentado o cabeçalho do TCP. Observa-se que os campos de porta de origem e destino têm comprimento de 16 *bits*, possibilitando o estabelecimento de 65535 conexões entre dois nós identificados com apenas um endereço IP cada.

A fim de proporcionar integridade na entrega dos dados, o TCP possui em seu cabeçalho um número de sequência (SN - Sequence Number) que identifica cada pacote de uma dada sessão. É através deste número que o protocolo é capaz de ordenar a entrega dos dados às aplicações e detectar a perda de um pacote dentro da rede. No estabelecimento da conexão TCP é gerado um número de sequência inicial (conhecido como ISN - *Inicial Sequence Number*) aleatoriamente para cada um dos hosts. Este número não inicia em zero por questões de segurança. Se os ISNs fossem sempre iguais, introduziria-se a possibilidade de segmentos de diferentes conexões fossem confundidos possibilitando, por exemplo, o sequestro de conexões via IP spoofing

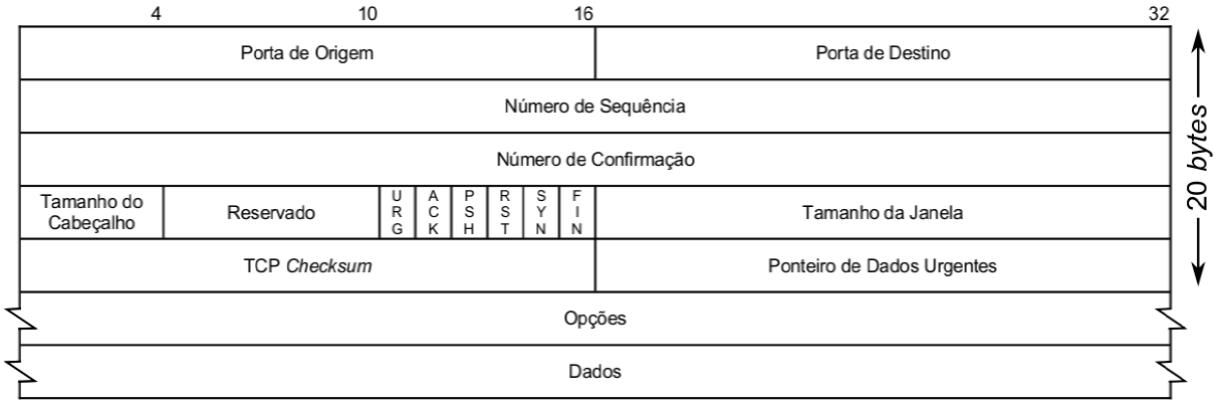


Figura 2.3: Cabeçalho TCP (TCP..., 1981)

(KOZIEROK, 2005).

No TCP, a cada segmento enviado o número de sequência é incrementado do número de bytes transmitido, diferentemente de outros protocolos da classe ARQ (*Automatic Repeat Request*) onde o número de sequência é incrementado de acordo com o número de segmentos transmitidos. Os protocolos desta classe são capazes de retransmitir os pacotes perdidos automaticamente. Estes pacotes são identificados, pois no cabeçalho TCP há um campo chamado Número de Confirmação, válido apenas quando a *flag* ACK é marcada com um *bit* 1, que indica o próximo número de sequência esperado.

O processo de confirmação de recebimento de pacotes no TCP é chamado de confirmação cumulativa. O número de confirmação indica que todos os *bytes* anteriores foram recebidos. Isto possibilita o aumento de desempenho do protocolo por um processo de retransmissão rápido de pacotes perdidos. Na Figura 2.4 é ilustrado o funcionamento deste processo. Imagine que a máquina A transmitirá oito pacotes sequencialmente de 1 *byte* cada à máquina B e que o numero de sequência inicial é 1 (SN=1). Ao receber o primeiro pacote B envia um pacote de confirmação, com a *flag* ACK marcada, com o número de confirmação igual ao próximo SN esperado (ACK=2). Imagine também, que o pacote com SN=2 é perdido na rede por qualquer motivo. Ao receber o pacote com SN=3, B envia outro pacote ACK=2, já que não recebeu o pacote SN=2. O mesmo ocorre para os pacotes SN=4, 5 e 6. Porém, ao receber o terceiro pacote com ACK=2, a máquina A infere que o pacote SN=2 foi perdido, e o retransmite. B ao receber este pacote, também já recebeu os pacote com número de sequência até SN=6. Por a confirmação ser feita de forma cumulativa, ao enviar um pacote com número de confirmação ACK=7, B confirma o recebimento de todos os pacotes de SN inferiores a 7. O processo de confirmação continua. e após a confirmação do pacote com SN=8, B espera receber um pacote com SN=9.

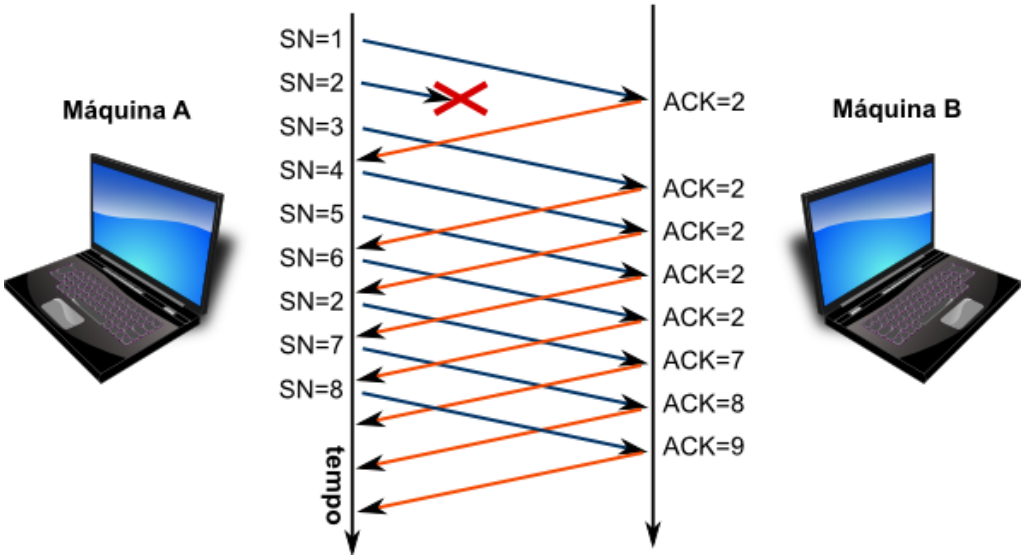


Figura 2.4: Processo de retransmissão rápida do TCP

Outro mecanismo importante provido pelo TCP é o de controle de fluxo. Não adiantaria o emissor da informação enviar tantos dados quanto possa, se não há recursos para tratamento destes no receptor. O resultado disto seria a ocupação inútil de recursos dos enlaces disponíveis. Para este controle, o receptor pode indicar quantos *bytes* ele está disposto a receber. Este valor é indicado no campo do cabeçalho Tamanho da Janela. Ao receber um pacote do receptor com o anúncio do tamanho da janela, o emissor que só poderá ter em trânsito aquela quantidade de informação até esperar pela confirmação (ACK) de um dos pacotes, que por sua vez trará uma atualização da janela disponível.

Porém, o que acontece se o gargalo da rede não for o receptor? Quantos pacotes deveriam ser enviados em uma rede com enlaces congestionados? A resposta sensata seria que só devem ser enviados os números de pacotes que não serão descartados. Mas como prever este número? A solução encontrada no TCP foi a utilização de um mecanismo de controle de congestionamento inferido através da informação de perda de pacotes. A taxa de transmissão é aumentada, incrementando o tamanho da janela de transmissão, até o limite informado pelo receptor, ou até que o emissor detecte a perda de um pacote, por *timeout* ou pelo recebimento de 3 ACKs com números de confirmação iguais e inferiores ao último número de sequência utilizado. O tamanho da janela de transmissão até que detectada a perda é denominado janela de congestionamento. Quando detectado a fase de congestionamento o crescimento da janela de transmissão se dá de forma aditiva. Incrementa 1 pacote com o tamanho máximo permitido pela rede (MSS - Maximum Segment Size) por RTT (*Round Trip Time*), tempo gasto para que um pacote seja enviado, recebido e confirmado. Mas, para que o congestionamento seja rapidamente controlado, quando é detectada a perda de um pacote, o decrescimento da janela é dado de forma

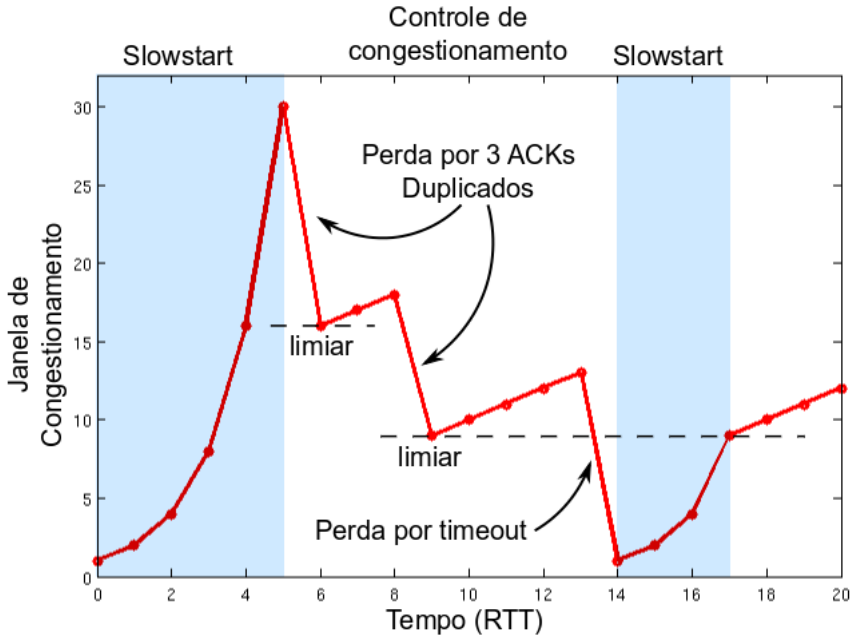


Figura 2.5: Mecanismos TCP de controle de congestionamento e *slow start*

multiplicativa, o tamanho da janela cai pela metade.

Entretanto, não há como saber o estado de congestionamento da rede quando uma conexão TCP é iniciada. Desta forma, para que conexões novas não inundem uma rede já sem recursos disponíveis foi criado um mecanismo de inicio lento, ou *slow start*. No *slow start* a janela de congestionamento inicia em 1 MSS. A cada ACK recebido, a janela de congestionamento cresce em 1 MSS. Assim, o processo se inicia com o transmissor enviando 1 MSS e esperando o pacote de confirmação, o ACK. Quando ele é recebido, o transmissor envia 2 MSS, e a cada ACK recebido aumenta a janela de transmissão em mais 1 MSS e a janela de congestionamento cresce para 4 MSS. O mecanismo provê um crescimento exponencialmente (2^n, $n \in \mathbb{N}$), entretanto, o crescimento não é exatamente exponencial, pois o receptor pode atrasar seus ACKs, tipicamente enviando um ACK a cada dois segmentos recebidos (STEVENS, 1997). Este crescimento acontece até que seja detectada a primeira perda de pacote. Nesta etapa o último valor válido da janela de congestionamento é memorizado como um limiar. Se esta for detectada via *timeout*, o processo de *slow start* é novamente indicado com a janela de congestionamento em 1 MSS, mas dura até o limiar memorizado, a partir daí entra em fase de controle de congestionamento. Todavia, se a primeira perda for detectada via 3 ACKs duplicados, a conexão entra em fase de congestionamento diretamente, como evidenciado na Figura 2.5. Na fase de controle de congestionamento, como observado anteriormente, o crescimento é de 1 MSS por RTT, e quando é detectada perda por 3 ACKs duplicados, o tamanho da janela cai pela metade e este passa a ser o novo limiar memorizado.

É importante salientar, novamente, que as considerações realizadas neste capítulo são referentes à implementação do TCP Reno, utilizado na implementação de sistemas Linux. No TCP Tahoe (FALL; FLOYD, 1996), por exemplo, a decisão tomada quando a perda de pacotes é detectada por 3 ACKs duplicados é a mesma que a tomada na perda por timeout, reiniciando a janela de congestionamento para 1 MSS e entrando em fase de *slow start*.

2.1.3 FTP (*File Transfer Protocol*)

O FTP é um protocolo da camada de aplicação baseado na arquitetura cliente-servidor. Provê o serviço de transferência ou compartilhamento de arquivos entre computadores. Para isto, o protocolo prevê dois tipos de conexões paralelas entre cliente e servidor: a de controle e a de transferência de dados. O primeiro tipo é utilizado para o envio de comandos para autenticação, listagem os arquivos e diretórios disponíveis ou início da transferência de arquivos. Porém, nenhum arquivo é transmitido nesta sessão. Quando, através desta conexão, solicita-se a transferência de um arquivo, uma nova conexão, agora do segundo tipo, é aberta para que isto ocorra. Ao termino do envio do arquivo em questão esta é finalizada. O início da conexão de controle é realizado pelo cliente que direciona o pacote de solicitação para a porta TCP que o servidor estiver escutando. Usualmente, a porta utilizada é a 21 (POSTEL; REYNOLDS, 1985).

Há dois modos de funcionamento de um servidor FTP: ativo ou passivo. A diferença está em quem origina a conexão de transferência de dados, pois a conexão de controle existe em ambos os casos. No modo ativo, é o servidor quem origina a conexão de dados de sua porta TCP número 20 (porta padrão) para a porta informada pelo cliente pela conexão de controle. Já no modo passivo, as conexões são originadas pelo cliente de sua porta TCP disponível para a porta qualquer informada pelo servidor.

O protocolo FTP prevê quatro tipos de transferência de dados. ASCII, EBCDIC, Binário e Local. Cada tipo é utilizado para atender as necessidades de padrões distintos de dados para implementar as funcionalidades da camada de apresentação:

- **ASCII**: utilizado para transferência de texto. É o tipo padrão adotado pelas aplicações, mas é inapropriado para transferências de outro tipo de dados.
- **EBCDIC**: semelhante ao ASCII, mas é mais eficiente para transferência de texto entre dois *hosts*, pois utiliza o conjunto de caracteres EBCIDIC (*Extended Binary Coded Decimal Interchange Code*).

- **Binário**: destinado à transferência de dados binários sem nenhuma conversão. É recomendado que todas as aplicações possuam a implementação deste tipo.

- **Local**: destinado para transferência de dados em modo proprietário. Ambos os computadores envolvidos devem possuir o módulo proprietário implementado.

Além dos tipos de transferência, ainda há três modos em que os dados podem ser transmitidos: fluxo, blocos e compressão. O Primeiro modo, a transferência é realizada apenas com o controle do TCP. Nenhum tipo de processamento é realizado pelo FTP. No modo de blocos, é o mais convencional. Neste modo, os dados são divididos em blocos e encapsulado em blocos individuais FTP, ou registros. Cada registro tem um cabeçalho de três *bytes* que indica o seu comprimento e contém informações sobre os blocos de dados a ser enviado. Um algoritmo especial é usado para manter o controle dos dados transmitidos e para detectar e reiniciar uma transferência interrompida. No último modo é realizada uma compressão dos arquivos antes que estes sejam enviados ao destino, onde serão descomprimidos de forma transparente ao usuário.

O *software* servidor adotado foi o Proftpd (PROFTPD, 2011) sobre a plataforma Linux. Ele suporta o tipo de transferência binário e ASCII. O tipo utilizado neste trabalho foi o ASCII no modo fluxo, pois os arquivos a serem transmitidos foram gerados por um *script* através da inclusão de caracteres aleatórios em arquivos até que os seus tamanhos chegassem ao desejado.

2.2 Plataformas Abertas para Roteadores

Há duas formas de se desenvolver um roteador: em *hardware* ou em *software*. A primeira, normalmente, é a mais custosa, já que exige o desenvolvimento de circuitos especializados para cada função. Isto vem se tornando mais acessível à produção em pequena escala com o uso de kits de desenvolvimento com FPGAs (*Field-Programmable Gate Array*). As FPGAs possibilitam a criação de *hardware* dedicado através de programação da interconexão de matriz de circuitos lógicos de que é composta. Para isso, utilizam-se linguagens específicas como o Verilog ou o VHDL. O desenvolvimento de códigos utilizando estas linguagens é custoso, já que se trata de linguagens de baixo nível de abstração. Assim, a alternativa mais facilmente implementada é o desenvolvimento de *softwares* para computadores pessoais, devido ao habitual acesso a estes equipamentos. Outra vantagem deste meio é que o requisito de processamento destes programas é baixo, podendo gerar o aproveitamento de máquinas com *hardware* de tecnologia mais antiga.

Uma opção atrativa para o desenvolvimento em *hardware* de roteadores é a utilização da

plataforma NetFPGA (LOCKWOOD et al., 2007). Esta plataforma permite a estudantes e pesquisadores construir sistemas de rede de alto desempenho. Seu desenvolvimento é aberto para a comunidade acadêmica e a documentação, bibliotecas, e códigos são disponíveis no sítio *web* do projeto (NETFPGA, 2011).

Há apenas algumas opções de roteadores baseados em PC, com código aberto, derivados do esquema de encaminhamento por *kernel* do *Berkeley Software Distribution* (BSD). Os mais completos no quesito de protocolos de roteamento são o Quagga (QUAGGA..., 2006) e o XORP (HANDLEY; HODSON; KOHLER, 2003). Todavia, esta característica torna os códigos destes *softwares* mais complexos. Isto pode ser interessante quando se desejam alterar ou criar novos protocolos de roteamento. Porém, quando o interesse é pela forma de encaminhamento de datagramas e a diferenciação do tratamento destes, plataformas mais simples, mesmo que utilizem apenas de rotas estáticas, tornam-se vantajosas.

Para atender ao escopo deste trabalho, vamos optar pela tecnologia CLICK (KOHLE, 2000), hoje incorporado à plataforma XORP, que se baseia em uma arquitetura flexível e modular de *software* para criar e encaminhar pacotes, facilitando a implementação de novas técnicas de tratamento destes.

2.2.1 NetFPGA

O objetivo da plataforma NetFPGA é possibilitar a rápida prototipagem de *hardware* de alto desempenho para rede de computadores, com foco em ensino e pesquisa (NAOUS; GIBB; BOLOUKI, 2008). Uma NetFPGA é uma placa se conecta a um computador pessoal. Existem dois modelos de NetFPGA. Um deles, mostrado na Figura 2.6, possui barramento PCI e 4 portas *Gigabit Ethernet*. O outro modelo utiliza um barramento PCI-*Express* e possui quatro portas SFPs (*Small Form-factor Pluggable transceiver*) com capacidade de 10Gbps. Os dois tipos de barramentos PCI podem ser utilizados para programação da FPGA da placa e como interface de comunicação entre a FPGA e o processador do computador. A placa também possui a facilidade de acesso direto à memória, DMA, tornando esta comunicação menos custosa do ponto de vista de processamento.

Há três projetos básicos desenvolvidos e disponíveis para NetFPGA: um interface de rede com 4 portas *Ethernet*, um *switch Ethernet* e um roteador IPv4. A maior parte dos projetos a serem desenvolvidos com a plataforma serão extensões destes três disponíveis. Isto torna a plataforma mais atrativa. Nesta seção nos ateremos apenas ao roteador.

O roteador IPv4 básico pode tratar o tráfego das 4 interfaces disponíveis em *full-duplex*

e inclui o encaminhamento de pacotes em *hardware*, dois programas que permitem construir rotas e tabelas de roteamento, e uma linha de comandos de terminal e interface gráfica para gerência do roteador. O software permite construir as tabelas de rotas através de protocolos de roteamento, como o Pee-Wee OSPF (PEE-WEE..., 2011), executando em um espaço de usuário completamente diferente do sistema operacional Linux, ou utilizando a própria tabela de rotas.(NAOUS; GIBB; BOLOUKI, 2008)

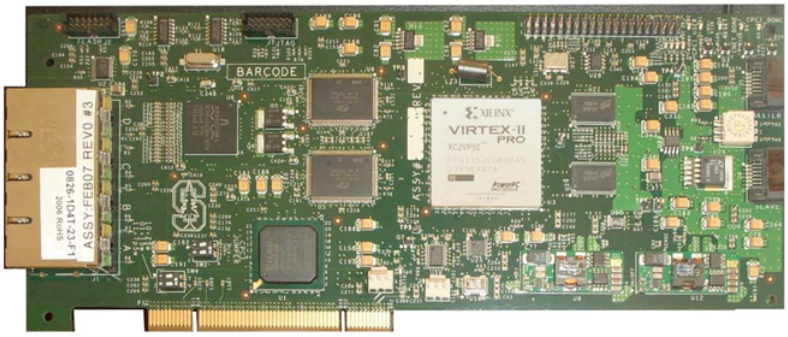


Figura 2.6: Foto de uma NetFPGA v2.1 - 4 x 1GE (LOCKWOOD et al., 2007)

Apesar de todos os benefícios de desempenho do desenvolvimento de *hardware*, a utilização desta NetFPGA requer um aprendizado profundo de sua linguagem de programação, Verilog ou VHDL, cuja curva de aprendizagem pode ser demorada, por ser uma linguagem não procedural de baixo nível de abstração. Além disto, é necessário que haja um computador para que a placa possa ser instalada, tornando o custo maior que o de roteadores implementados diretamente sobre estes mesmos PCs.

2.2.2 QUAGGA

Quagga é um *software* roteador, distribuido sob a licença GNU GPL - *General Public Lincense*, que provê serviços de roteamento baseados na pilha TCP/IP com suporte a protocolos de roteamento como RIPv1, RIPv2, RIPng, OSPFv2, OSPFv3, BGP-4, and BGP-4+. Além dos protocolos tradicionais de roteamento IPv4, o Quagga já suporta protocolos de roteamento IPv6 e possui as facilidades de gerenciamento via SNMP, provendo MIBs específicas dos protocolos de roteamento.

Um sistema com o Quagga instalado atua como um roteador dedicado. Com o Quagga, uma máquina troca informações de roteamento com outros roteadores podendo utilizar diversos protocolos de roteamento padronizados por RFCs. Através destas informações, ele atualiza a tabela de rotas do *kernel* e então encaminha dos datagramas para o destino correto. Há suporte para alteração da configuração dinamicamente. Além disto, as informações da tabela de roteamento

podem ser facilmente visualizadas através do terminal Quagga de cada funcionalidade.

A principal diferença entre um *software* de roteamento tradicional e o Quagga é na concepção de sua arquitetura. A arquitetura tradicional é composta de um único processo que provê todas as funcionalidades dos protocolos de roteamento (QUAGGA. . . , 2006). A abordagem da arquitetura Quagga é composta de uma coleção de diversos processos que juntos constroem a tabela de rotas. Pode haver vários processos de protocolos de roteamento específicos que se comunicam com um processo de gerência, denominado Zebra, que atualiza a tabela de rotas do sistema.

2.2.3 XORP

Assim como o Quagga, o XORP (eXtensible Open Router Platform) é um *software* para implementação de um roteador em computadores padrão. O objetivo do XORP é ser uma ferramenta de pesquisa e uma plataforma de desenvolvimento estável, possibilitando a transição de novas ideias de tratamento do encaminhamento de dados em redes de computadores para o mundo real.

Apesar de outros programas, como o Quagga, possibilitarem o desenvolvimento de *daemons* para roteamento sobre sistemas operacionais Linux ou FreeBSD, o principal problema é em questão a desempenho e robustez. O encaminhamento de pacotes nestes sistemas não são otimizados para serem submetidos à carga pesadas de tráfego, os daemons para roteamento não são robustos o suficiente e normalemente não são desenvolvidos de forma a prover facilidade de extensibilidade e por fim, não possuem uma interface integrada para o usuário (HANDLEY; HODSON; KOHLER, 2003).

O XORP é dividido em dois subsistemas, o *higher-level* (também chamado de *"user-level"*) e o *lower-level*. O primeiro subsistema consiste dos protocolos de roteamento propriamente ditos, juntamente com as bases de informações de roteamento e processos de suporte. Já o segundo, executa dentro do *kernel* do sistema operacional, gerenciando o encaminhamento de cada pacote e provendo uma API (*Aplication Programming Interface*) de acesso para o *higher-level*.

A arquitetura do *user-level* é composta de múltiplos processos. Assim como o Quagga, há um processo para cada protocolo de roteamento e processos extras para gerenciamento, configuração e coordenação. Já o *lower-level* utiliza o CLICK *modular router* (KOHLE, 2000), que será discutido posteriormente. Dentro do XORP o CLICK atua integrado ao kernel do sistema operacional o tornando mais robusto e eficiente para o encaminhamento de pacotes.

Para o objetivo de nosso trabalho, que é estudar formas de tratamento diferenciado para categorias de fluxos, identificando-os e priorizando seu encaminhamento quando necessário a fim de manter equidade na utilização dos recursos de rede disponíveis sem a utilização de manutenção de estados, não necessitaremos de mecanismos avançados de roteamento. Isto torna a utilização apenas da plataforma CLICK mais interessante, já que assim eliminaremos a ocupação de recursos computacionais, como processamento e memória, do *higher-level* do XORP.

2.2.4 CLICK Router

A arquitetura CLICK é flexível, modular e de alto desempenho para encaminhamento e análise de pacotes de rede. Diversos aspectos desta arquitetura foram inspirados diretamente por propriedades dos roteadores (KOHLE, 2000). Um exemplo disto é que a passagem de pacotes ao longo de uma conexão pode ser iniciada tanto pela extremidade da fonte (pacote é 'empurrado') quanto pela extremidade de destino (pacote é 'puxado'), modelando a maior parte dos padrões de fluxo de pacotes em roteadores.

Há duas formas de se utilizar o *software* CLICK: como aplicação a nível de usuário (*driver user-level*), ou como módulo do *kernel* Linux (*driver linux-module*). As duas possibilidades se destinam a funções diferentes. O *driver user-level* é útil para depuração e execução de testes repetitivos; ele pode receber pacotes da rede usando mecanismos do sistema operacional através de bibliotecas originalmente programadas para *sniffers*. Entretanto, o *user-level* não se restringe da pilha de rede do sistema operacional para encaminhar um pacote. Já o driver em nível de *kernel*, o *linux-module*, substitui completamente a pilha de rede do SO, transformando um computador pessoal convencional num roteador, alcançando alto desempenho (KOHLE, 2000). Desta forma só se utilizará o último *driver* neste trabalho.

Arquitetura CLICK

A elaboração de roteadores na arquitetura CLICK se dá através de módulos processadores de pacotes denominados elementos. Para a construção de um roteador é necessária apenas uma coletânea destes elementos interconectados, possibilitando a adição de novas funcionalidades ao roteador pela modificação de elementos existentes ou criação de novos elementos.

Um elemento CLICK representa uma unidade de processamento do dispositivo de rede. Representa, também, uma computação simples, tal como, decrementar o campo TTL do pacote a ser roteado a um novo nó da rede. Toda ação executada pelo software CLICK é encapsulada

em um elemento.

Para a construção de um roteador utilizando a tecnologia CLICK é necessária uma coletânea de elementos interconectados através de uma linguagem visual própria de configuração. Assim, se torna possível a expansão da arquitetura a fim de oferecer suporte a novas funções de roteamento, tais como novos protocolos, modificando elementos já existentes ou criando novos elementos.

Dentre as propriedades dos elementos se destacam a classe do elemento, que indica o código a ser executado no processamento de um pacote, assim como a função de iniciação do elemento e seu formato de dados; a *string* de configuração, que pode ser utilizada para realizar um ajuste fino no comportamento de um elemento; e os *handlers*, que tornam possível o acesso a parâmetros intrínsecos ao roteador, tais como tamanhos das filas, contadores, e tabelas de roteamento. Todas estas propriedades podem ser melhor identificadas no padrão visual mostrado na Figura 2.7 que descreve as propriedades do elemento Tee, que tem a função de copiar um pacote de sua entrada para o número de saídas discriminado na *string* de configuração. A verificação destes parâmetros se dá através do terminal da máquina roteador com a execução do comando cat direcionado a um dos arquivos ASCII criados dinamicamente para cada elemento instanciado na configuração do roteador instalada.

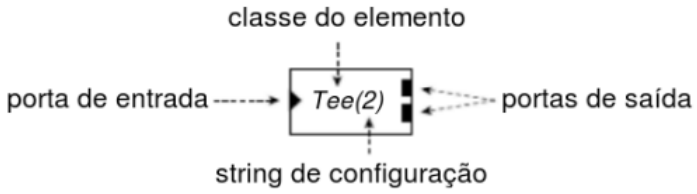


Figura 2.7: Descrição do elemento CLICK *Tee* (KOHLE, 2000)

Roteador Simples

Este roteador foi configurado para tarefas de roteamento IP que envolvam apenas informações locais. O roteador conta com duas interfaces de rede e as rotas são adicionadas de forma estática. Para facilitar o entendimento de interconexões de elementos CLICK este roteador é mostrado na Figura 2.8. Um pacote chega através de uma interface física do computador ao *software* pelo elemento *FromDevice* e é diretamente classificado, em pacotes *ARP Queriers*, *ARP Responses*, e *pacotes IP*. Cada tipo de pacote segue rumo próprio dentro do roteador. Os pacotes *ARP Queries* são encaminhados ao *ARPResponder*, elemento que se encarrega de responder as requisições ARP pela interface que o pacote foi recebido. Os pacotes de respostas ARP são encaminhados ao Linux da máquina roteador, e aos elementos *ARPQueriers* do roteador. Este

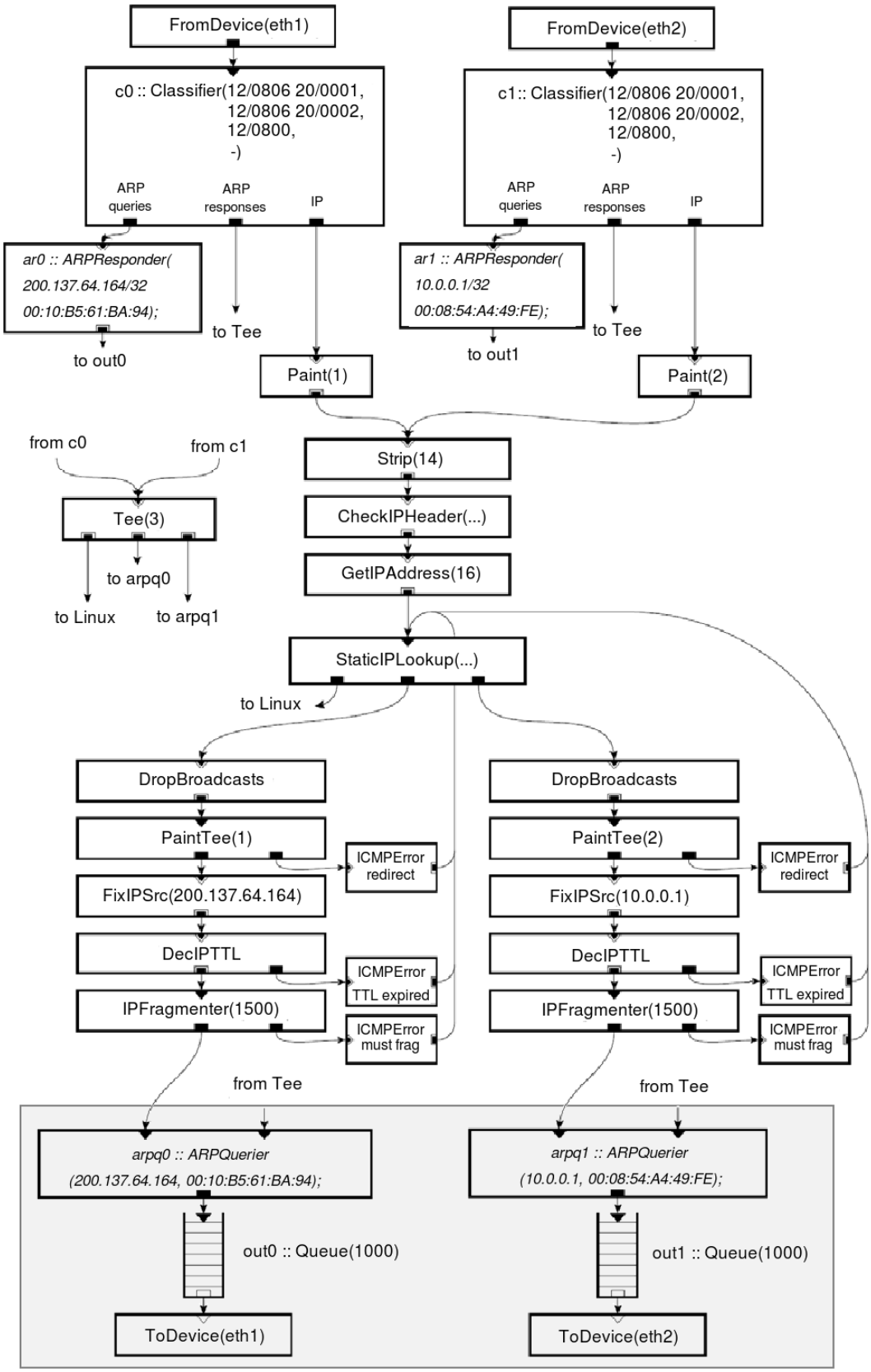


Figura 2.8: Roteador Simples implementado no CLICK (KOHLE, 2000)

elemento salva estas informações em uma tabela ARP, caso a requisição tenha sido realizado por ele. Por fim, os pacotes IPs são marcados a partir de anotações inseridas no cabeçalho e encaminhados para as funções de roteamento propriamente ditas.

A própria estrutura de pacotes do Linux, *sk_buff*, fornece 48 *Bytes* de espaço para anotações. Estas são utilizadas no roteador para informar se um pacote está saindo pela mesma interface que pela qual chegou, através da anotação *Paint*, para descartar pacotes de *broadcast* não úteis, anotação *Link-level broadcast flag*, e para encaminhar pacotes de erro ICMP com o endereço IP fonte da interface em que este é emitido, anotação *Fix IP Source flag*.

Depois de totalmente roteado, o pacote é enviado ao elemento *ARPQuerier*, que realiza quase todas as funcionalidades do protocolo ARP. Este elemento é quem faz a requisição dos endereços físicos, mantém uma tabela com essas informações e se encarrega das tarefas de inclusão destes endereços fixos nos pacotes a serem encaminhados. Por fim, os pacotes devem ser enviados a uma fila que realiza o armazenamento destes até que possam ser transmitidos pela interface de rede através do elemento *ToDevice*. No roteador básico mostrado nesta seção, esta fila é única e do tipo FIFO (*First In, First Out*). Porém, é justamente nesta porção que aplicaremos mecanismos capazes de identificar os datagramas pertencentes a diferentes fluxos e de atribuí-los prioridades distintas, acarretando a capacidade de percepção diferenciada da utilização dos mesmos recursos disponíveis na rede. Na Figura 2.8 é apresentado o diagrama de blocos de interligação dos elementos CLICK, com suas respectivas *strings* de configuração. Em destaque, evidencia-se onde ocorrerão as alterações no roteador simples para que nele sejam incorporadas as funcionalidades de diferenciação de tratamento de tráfego.

2.3 Conclusão

Para a experimentação em ambientes físicos de rede, evitando que os mecanismos fruto de uma pesquisa tenham suas repostas limitadas à premissa da qualidade e simplicidade dos modelos utilizados em simuladores, é necessário que se desenvolva roteadores experimentais, uma vez que os roteadores comerciais não têm o desenvolvimento de seus sistemas operacionais abertos à comunidade acadêmica. A construção destes roteadores pode ser realizada em *hardware* e em *software*. A primeira forma é mais custosa tanto do ponto de vista financeiro, como do gasto de recurso humanos, uma vez que mesmo com a utilização de FPGAs, o desenvovimento dos códigos necessários para programação do *hardware* não é trivial. Já a segunda, é mais atrativa, visto que os requisitos de memória e processamento dos programas utilizados para implementar os mecanismos de roteamento são baixos, podendo gerar inclusive o aprovei-

tamento de máquinas de *hardwares* obsoletos para os sistemas operacionais e programas atuais destinados a usuários domésticos.

Neste capítulo, além da caracterização do funcionamento dos protocolos envolvidos neste trabalho, abordamos as principais plataformas abertas para desenvolvimento de roteadores em computadores pessoais. Entre e elas destacamos o QUAGGA e o XORP por implementarem a maior parte dos protocolos de roteamento padronizados. Porém, o QUAGGA utiliza o mesmo sistema de encaminhamento de pacotes que o sistema operacional, que não é otimizado para o encaminhamento de pacotes tornando-o menos eficiente quando submetemos o roteador a alta carga. Já o XORP utiliza a plataforma CLICK em sua base utilizando outros processos em nível de usuário para implementação dos protocolos de rede. Desta forma, optamos por utilizar apenas a plataforma CLICK neste trabalho, uma vez que nosso objetivo desenvolver mecanismos capazes de tratar de forma diferenciada o encaminhamento de pacotes e não de desenvolver ou alterar protocolos de roteamento, eliminando assim, parte da complexidade de desenvolvimento oferecida por plataformas mais complexas como o XORP.

3 *Proposta de Classificadores e de Metodologia de Testes*

Este capítulo trata da descrição do funcionamento dos diferenciadores de tráfego propostos, além disto, trata do desenvolvimento de ambiente de testes controlado para a utilização de tráfego real, das métricas utilizadas para avaliação dos diferentes métodos de classificação e escalonamento de tráfego de modo a proporcionar maior equidade no compartilhamento dos recursos de rede disponíveis.

Primeiramente, na Seção 3.1, aborda-se o objeto de estudo, os escalonadores de tráfego. Nesta seção são abordadas as características dos algoritmos de classificação RuN2C, RAFLE e do classificador Ideal, além de como foi realizada as suas implementações na arquitetura CLICK. Na Seção 3.2, apresenta-se as escolhas realizadas para realização dos experimentos. Em seguida, na Seção 3.2.1, discutem-se o cenário de tráfego utilizado e quais adequações nas aplicações e no sistema operacional foram realizadas para que ele correspondesse aos objetivos do trabalho. Logo após, na Seção 3.2.2, tratam-se dos meios utilizados para geração de tráfego em ambiente controlado. Segue-se, na Seção 3.3, com o padrão de equidade adotado. Finalmente, na Seção 3.4 apresentamos uma discussão sobre os experimentos realizados e sobre as métricas utilizadas para cada caso.

3.1 Objeto de Estudo: Diferenciadores de Fluxo

Uma rede de comutação de pacotes proporciona maior utilização dos recursos dos enlaces em relação às redes baseadas em comutação de circuitos. Porém, as redes baseadas em comutação de pacotes podem levar a problemas cruciais quando pacotes associados a diferentes classes de qualidade de serviço compartilham os mesmos enlaces que possuem capacidades finitas. Quando, em um enlace, a carga de tráfego é alta, o encaminhamento dos pacotes de acordo a ordem de chegada, disciplina FIFO, pode não ser uma maneira eficiente de satisfazer aos requisitos de QoS dos usuários (TSAI; CHUNG; TSAI, 2011).

O processo de atribuição intencional de prioridade aos pacotes de diferentes usuários, ou aplicações, para proporcionar um certo nível de justiça no compartilhamento de recursos e de garantia de desempenho é realizado pelos escalonadores de tráfego. De forma mais específica, os escalonadores de tráfego visam alcançar as seguintes propriedades:

- **Eficiência**: a função básica dos algoritmos de escalonamento de pacotes é definir a ordem de transmissão dos pacotes enfileirados no sistema. Um algoritmo é dito mais eficiente que outro quando tem maior capacidade de atender as exigências de QoS, mesmo quando há conexões que produzam alta carga de tráfego ou quando muitos usuários utilizam simultaneamente os recursos da rede.

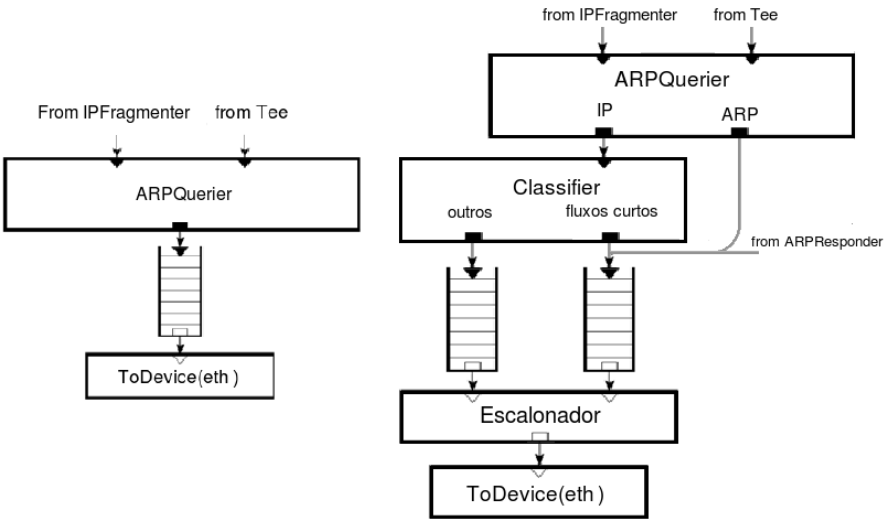
- **Proteção**: além da garantia de QoS, outra propriedade desejada em um algoritmo de escalonamento de tráfego é possibilidade de tratar os fluxos de forma individual, diminuindo os efeitos do proporcionamento de QoS à outros fluxos tanto quanto possível. Aqui, definimos fluxos como uma simples conexão de dados de um usuário.

- **Flexibilidade**: outra propriedade importante é a capacidade de atender a diferentes classes de QoS. Hoje em dia, cada vez mais surgem aplicações com diferentes requisitos para funcionamento adequado.

- **Baixa complexidade**: Um algoritmo de escalonamento de tráfego deve possuir uma complexidade computacional razoável, de forma a possibilitar sua implementação. Devido ao rápido crescimento da largura de banda e das taxas de transmissão nos sistemas de comunicação de dados atuais, a velocidade de processamento dos pacotes torna-se cada vez mais crítica.

Assim, um bom escalonador de tráfego é aquele que consegue atingir os objetivos de QoS definidos atingindo os melhores níveis de eficiência, proteção, flexibilidade e baixa complexidade.

O modelo proposto para diferenciação de fluxos é apresentado na Figura 3.1. Na Figura 3.1(a) é apresentado o sistema de referência composto por apenas uma fila e sem nenhum método de classificação de tráfego. Já na Figura 3.1(b) é mostrada a estrutura dos diferenciadores de fluxos propostos, compostos por uma política de fila que classifica o tráfego e armazena-o para futura transmissão. As classes distintas têm suas respectivas filas e um escalonador de tráfego encarrega-se de decidir a ordem de serviço para a transmissão dos pacotes de cada fila. Aqui já são indicados os nomes de alguns elementos da arquitetura CLICK utilizados, mas a implementação será melhor desenvolvida na Seção 3.1.2.



(a) Sistema de referência (b) Estrutura proposta

Figura 3.1: Diferenciador de fluxos CLICK

3.1.1 Políticas de Fila

As políticas de fila podem ser compostas por uma única fila única, tratando a ordem de transmissão pacote-a-pacote e de descarte de pacotes enfileirados quando a capacidade de armazenamento esgota, ou uma política de múltiplas filas, separando as classes de QoS em filas com tratamento distintos. O tratamento destas filas é definido por uma disciplina de escalonamento de tráfego. As duas disciplinas de serviço adotadas neste trabalho, de modo a viabilizar a baixa complexidade dos escalonadores de tráfego propostos foram a *Round Robin* (RR) e a *Priority Scheduling* (PrioSched).

Referência: Drop Tail Fila Única

A disciplina de enfileiramento mais simples é a FIFO. Assim como a disciplina de descartes de menor complexidade é aquela na qual os pacotes que na sua chegada encontram o *buffer* de enfileiramento cheio são descartados. Como referência, utiliza-se a composição destas disciplinas em uma única política de fila, a qual denominaremos apenas por *Drop Tail*. Utilizaremos esta referência para o ganho alcançado pelas propostas estudadas neste trabalho, já que este é o tratamento mais básico que se pode ter em um sistema de comutação de pacotes amplamente utilizado na prática.

Porém, para que seja possível comparar esta política de escalonamento de tráfego com as demais políticas propostas, compostas por duas filas, sem prejuízos causados por descartes originados pela desigualdade no tamanho dos *buffers* empregados, determinou-se o tamanho dos *buffers* experimentalmente de forma a sobredimensionar a sua capacidade para que descartes

não ocorressem.

Premissas para a Divisão dos fluxos

As aplicações TCP reagem ao congestionamento e às perdas em uma rede de três formas distintas: através da alteração do tamanho de suas janelas (congestionamento e de recepção), de processos rápidos de retransmissão de pacotes ou da retransmissão destes após o *timeout* (PAXSON; ALLMAN, 2000). Para conexões curtas, o que implica em pequenas janelas de congestionamento, a perda é normalmente detectada após o *timeout* e, provavelmente, após a finalização da transmissão de todos os dados. Como resultado disto, temos que a retransmissão após o *timeout*, para conexões curtas, não é muito eficaz na redução do tráfego global de uma rede e em sua estabilização.

Quando é utilizada uma fila única, há desigualdade na disputa entre fluxos longos e curtos na ocupação desta. Os fluxos longos, em fase de controle de congestionamento, alcançam maiores janelas de transmissão implicando, assim, na transmissão de dados em longas rajadas de pacotes e consequente ocupação de maiores porções do *buffer*. Dado que este recurso é limitado, o esgotamento do *buffer* acarreta em descartes que oneram em maior proporção o desempenho de fluxos curtos, que alcançam pequenas janelas de transmissão.

A separação de fluxos longos e curtos em duas filas distintas, além de aumentar a equidade na disputa na ocupação do buffer entre estas classes, pode proporcionar economia deste recurso. Resultados em (PSOUNIS et al., 2005) indicam que a somatória do tamanhos dos *buffers* necessários para a classe de fluxos longos e para a classe de fluxos curtos é menor que o tamanho necessário quando se utiliza um único *buffer* compartilhado.

***Running Number 2 Class* (RuN2C)**

O mecanismo de diferenciação RuN2C é um escalonador de tráfego TCP/IP que processa duas classes de prioridades. Nele, os pacotes são classificados como baixa ou alta prioridade de acordo com o número de *bytes* transmitidos numa sessão TCP. Os pacotes pertencentes a cada uma dessas classes são lotados em um fila FIFO. Após isso, utiliza-se uma disciplina de escalonamento para transmissão destes pacotes. Na proposta original do RuN2C (AVRACHENKOV; BROWN; NYBERG, 2004), a disciplina de escalonamento utilizada é a PrioSched. No entanto, também se experimenta neste trabalho a disciplina RR.

A diferenciação dos pacotes pelo RuN2C é realizada sem manutenção de estados das conexões TCP ativas. Quando uma conexão é estabelecida, um número de sequência inicial, ISN

(*Initial Sequence Number*), de 32 bits é gerado de maneira aleatória. A cada pacote transmitido é somado a este ISN o número de *Bytes* transferidos. Isto possibilita a diferenciação dos fluxos ativos no roteador, através deste campo do cabeçalho TCP, apenas com o controle da geração destes números iniciais nas máquinas fontes de tráfego e da comparação deles com um limiar nos roteadores.

Após a transferência do primeiro pacote um número de sequência inicial é incrementado de um número máximo de *bytes* igual ao MSS (*Maximum Segment Size*) da rede. Para um MSS igual a 1460 *bytes*, temos que, aproximadamente os 10 bits menos significativos, $\log_2(MSS = 1460) \approx 10$, do número de sequência seriam utilizados logo no primeiro pacote. Assim, estes bits poderiam ser utilizados, sem prejuízos ao limiar, para geração de um ISN. No entanto, 10 bits gerariam apenas 1024 possíveis ISNs, não provendo aleatoriedade suficiente a estes números, possibilitando problemas de segurança, tais como IP *spoofing* para sequestro de conexões e sessão *hijacking* (SUN et al., 2007).

Desta forma, a divisão do número de sequência é feita de acordo com a Figura 3.2. Utilizam-se além dos 10 bits menos significativos, os 10 bits mais significativos, campo R, para a geração dos ISN, totalizando em $2^{20} \approx 1$ milhão de possíveis números iniciais.

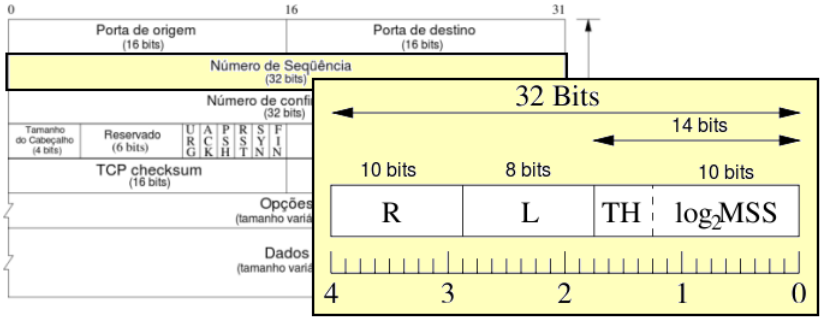


Figura 3.2: Padrão do número de sequência utilizado pelo RuN2C

A geração dos ISNs se dá através de uma função específica no kernel dos sistemas operacionais. Desta forma, para que o Run2C funcione adequadamente é necessário que uma adaptação na geração destes números de sequência iniciais seja realizada. Esta adaptação deve-se dar pelo menos na máquina que envia os arquivos. Para os experimentos, optamos pela alteração nas duas máquinas. Como mostrado, requisito para que o mecanismo Run2C funcione adequadamente é a geração dos 32 bits dos números de sequência inicial na forma: 20 bits, os 10 bits mais e os 10 bits menos significativos são gerados de maneira aleatória possibilitando um total de 1 milhão de ISNs e os 12 bits restantes devem ser iniciados com zero quando uma conexão TCP é estabelecida. Para isto, alteraram-se as funções do *kernel* Linux responsável pela geração destes ISNs aplicando-se ao final da função de geração existente uma máscara sobre o número

gerado pelo algoritmo. Ela mantém os 20 bits necessários aleatórios e zera os demais. O arquivo alterado foi o Random.c, localizado dentro do *kernel* em linux<versão>/drivers/char/.

O parâmetro de ajuste do RuN2C é o limiar de separação, TH. O mecanismo segrega as conexões que transmitirem um número de *bytes* superior a este limiar a fila de fluxos longos. Assim, o limiar deve ser escolhido de forma que os fluxos curtos se beneficiem do mecanismo de diferenciação, todavia, mantendo-se baixa a carga de alta prioridade para que fluxos longos não sejam prejudicados demasiadamente. A determinação desse limiar será discutida no Capítulo 4, com base em uma análise experimental.

Random Assorter of Flow LEngths (RAFLE)

A dependência de alterações na implementação dos protocolos básicos da Internet (TCP/IP) para que uma política de diferenciação de fluxos funcione adequadamente é um ponto complexo na implantação desta em ambientes em operação. Assim, mecanismos de diferenciação de fluxos que consigam realizar a isolação de fluxos através de informações do tráfego passante em um roteador, sem que o grau de complexidade dos algoritmos cresça pela necessidade de manter os estados de todas as sessões ativas, tornam-se necessários.

Para inferir que pacotes pertencem a fluxos de longa duração que ocupem uma grande porção dos recursos disponíveis nos enlaces, o escalonador de tráfego RAFLE, proposto neste trabalho, utiliza da característica de transmissão em rajadas dos fluxos na Internet.

De acordo com o princípio de funcionamento do TCP, fluxos que transferem grandes volumes de dados alcançam altos valores de janela de transmissão, o que implica em longas rajadas de pacotes. Mantendo-se uma pequena memória da identificação de fluxo, que denominaremos de *flowID*, dos últimos pacotes processados pelo roteador é possível inferir que um pacote pertence a uma rajada de longa duração. Porém, não se deve demandar muito tempo varrendo toda a memória, o que acarretaria em aumento do atraso na componente de processamento.

A cada pacote pertencente a uma rajada de um fluxo que é processado pelo roteador, maior a probabilidade do *flowID* de um novo pacote que ingressa neste estar armazenado na memória. Desta forma, a estratégia utilizada pelo RAFLE é sortear uma posição da memória (P_a), segundo uma distribuição uniforme, e então efetuar a comparação do *flowID* existente nesta posição da memória ($FlowID\_memoria(P_a)$) com o do novo pacote ($FlowID\_atual$) que chega ao roteador. Caso os *FlowIDs* sejam idênticos, o novo pacote é encaminhado para a fila de fluxos longos. Caso contrário ele ingressa na fila de fluxos curtos.

Durante iniciação do mecanismo, quando a memória ainda não se encontra completa, caso

não haja *FlowID* na posição da memória sorteada, o pacote é enviado para a fila de fluxos curtos e seu *FlowID* é inserido na tabela normalmente. A memória do classificador RAFLE é utilizada de forma rotativa, para que não seja necessário deslocar todos os *FlowIDs* nela inseridos para as posições posteriores a cada novo pacote. A última posição onde foi inserido um FlowID é marcada através de ponteiro incremental (P_u).

Assim, o processo de decisão do algoritmo RAFLE é apresentado na Figura 3.3. O desempenho do classificador depende do ajuste do parâmetro N, que indica o tamanho da memória, em número de linhas, que o classificador irá utilizar.

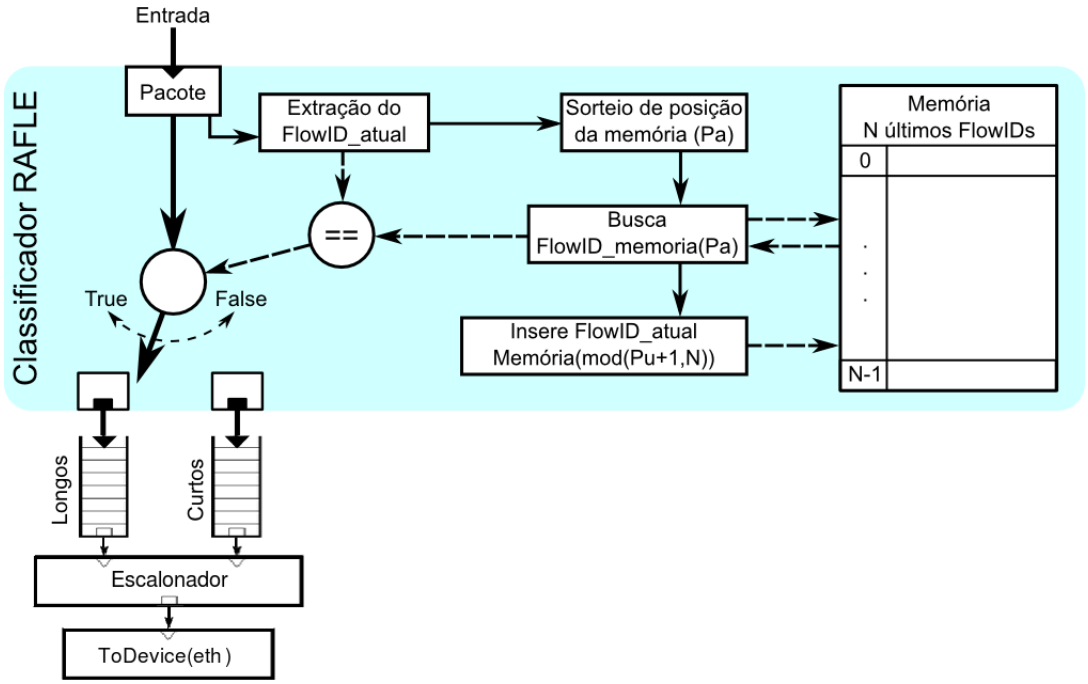


Figura 3.3: Diagrama de fluxo do algoritmo RAFLE

Uma das vantagens do RAFLE é a possibilidade de uso do algoritmo para diferentes definições de fluxo. Neste trabalho, a identificação dos fluxos TCP, padrão do FlowID, utilizada é composta da tupla (IP de Origem, Porta de Origem, IP de Destino, Porta de Destino) totalizando o armazenamento de apenas 12 *bytes* por posição da memória.

3.1.2 Módulos CLICK Desenvolvidos

Para implementação de um roteador que priorize as conexões curtas no CLICK, foi adicionada uma segunda porta de saída ao elemento CLICK *ARPQuerier*, conforme mostrado na Figura 3.1. Quando esta saída é adicionada, os pacotes IP são encaminhados à primeira saída, e pacotes do protocolo ARP à outra, possibilitando inclusão dos pacotes do último tipo na mesma fila com prioridades dos pacotes TCP de fluxos curtos. Isto proporciona ganhos, uma vez que

enquanto não são obtidas as respostas das requisições ARP os pacotes não são encaminhados.

Os dois métodos de diferenciação de fluxos propostos se baseiam na mesma estrutura de elementos CLICK. A diferença se encontra no classificador de pacotes.

No RuN2C a classificação dos pacotes é feita através da análise do campo número de sequência do cabeçalho TCP. Desta forma, incluiu-se o elemento *IPClassifier*, já disponível na biblioteca de elementos CLICK, que pode ser configurado para separar pacotes do protocolo TCP de acordo com o número de sequência. Este elemento utiliza o mesmo padrão de configuração do *software* Linux *TCPDump*. Nesta configuração, ao número de sequência, situado nos *bytes* de 5 a 8 do cabeçalho TCP, aplica-se uma máscara ocultando os 10 bits do campo R, mostrado anteriormente na Figura 3.2. Em seguida compara-se o resultado com o limiar adotado. Os pacotes que após esta operação possuam números de sequência inferiores ao limiar são destinados a uma interface e os demais à outra.

As filas são finalmente ligadas a um escalonador de prioridades que é encarregado de encaminhar os pacotes à interface. Também são utilizados elementos da biblioteca disponível que implementam as disciplinas de serviço adotadas neste trabalho. São eles: o elemento *PrioSched* para a disciplina *Priority Scheduling* e o *RoundRobinSched* para a RR. O *PrioSched* funciona como o esquema de prioridades proposto para o RuN2C, a fila de fluxos longos não é servida até que a de fluxos curtos esteja vazia. Já o *RoundRobinSched* faz o escalonamento do tráfego transmitindo seqüencialmente um pacote de cada entrada. Se uma entrada acabou de ser servida, o escalonador busca o próximo pacote na fila da entrada seguinte. *RoundRobinSched* utiliza-se de um mecanismo de notificação para não servir filas vazias.

Para o desenvolvimento do RAFLE utilizou-se a mesma estrutura do RuN2C evidenciada na Figura 3.1. Porém, foi necessário o desenvolvimento de um elemento classificador específico, de acordo com as especificações dadas na Seção 3.1.1, utilizando a linguagem C++. Este elemento foi incorporado à biblioteca do CLICK utilizada.

O classificador ideal, assim como o RuN2C, foi implementado com base no elemento *IPClassifier* disponível. Este classificador deixa a cargo das fontes de tráfego declararem que tipo pertencem cada fluxo. A declaração é feita através do IP de destino utilizado. O servidor espera conexões curtas através de um IP A e longas através de um IP B. Assim, comparando este campo, os pacotes são encaminhados no roteador para as filas correspondentes com 100% de acerto.

Definida as propostas de diferenciadores de fluxos e do sistema de referência, passa-se agora a metodologia para descrição do ambiente de testes, padrões de comparação e, por fim,

dos experimentos realizados e das métricas de avaliação utilizadas.

3.2 Metodologia e Ambiente de Testes

Na literatura é corriqueiro o uso de simuladores para a comparação de eficiência de diferentes métodos de tratamento de fluxos. Todavia, os processos de geração de tráfego, assim como o processo de tratamento ao longo da rede, se baseiam em modelos muitas vezes demasiadamente simplificados. Portanto, os resultados de simulação frequentemente não representam de maneira fidedigna o comportamento dos sistemas físicos. Desta forma, foi objetivo neste trabalho a submissão dos roteadores implementados a um cenário de tráfego real, mas no qual pudéssemos ter algum grau de controle nos testes de forma a permitir comparações de desempenho. Para isto, optou-se pela transferência simultânea, gerando concorrência entre os fluxos, de uma base real de arquivos via FTP. O objetivo principal é avaliar o efeito dos esquemas propostos sobre sessões TCP, daí para podermos isolar o efeito da política de diferenciação de fluxos devemos trabalhar, e isso justifica também a topologia de rede, com conexões com apenas um salto.

Na implementação dos roteadores em PC utiliza-se o *driver linux-module* da arquitetura CLICK, que atua a nível de *kernel* em sistemas Linux. A escolha foi realizada no capítulo 2, com base no compromisso entre custo e benefícios com relação à flexibilidade, à facilidade de implementação e ao desempenho da solução.

Para a análise dos classificadores implementados, RuN2C e RAFLE, que inferem se os pacotes pertencem a fluxos curtos ou longos, direcionando-os a filas distintas que serão servidas de acordo com as disciplinas *Round Robin* (RR) e *Priority Schedulling* (PrioSched) será adotado o seguinte método. Primeiramente, é realizada a comparação a um padrão de equidade estipulado através de uma modelagem analítica da política *Processor Sharing*, onde os recursos da rede são divididos igualmente entre todos os fluxos ativos. Este padrão é proposto na seção 3.3. Em seguida, computam-se os ganhos em relação aos resultados obtidos quando nenhuma estratégica de priorização é utilizada. O roteador implementado, neste caso, possui uma fila única com a disciplina FIFO em cada interface de saída. Denominaremos esta política de tratamento de tráfego de *Drop Tail* fazendo menção aos sistemas reais, onde a decisão de descartar os pacotes que chegam a filas cheias é normalmente tomada.

Para estabeleceremos os limites máximos de ganho para cada disciplina de escalonamento das filas classificadas, RR ou PrioSched, implementou-se um classificador ideal. Ele é capaz distinguir sem erros os pacotes pertencentes a cada classe, pois neste teste os pacotes são classi-

ficados pela fonte de tráfego de acordo com o IP de destino, ficando a cargo do roteador apenas classificar os pacotes dos fluxos segundo esta marcação.

3.2.1 Ambiente de Testes

A opção de utilizar tráfego real, mas utilizando um ambiente de testes controlado onde pudéssemos avaliar o desempenho dos métodos de classificação e escalonamento de tráfego através de diversas disciplinas foi tomada com base nas discussões dos capítulos anteriores.

O tráfego utilizado será unicamente TCP, por se tratar de um dos protocolos de transporte mais utilizados na Internet. Esta é, também, uma das razões da utilização de tráfego real nos experimentos. Por ser um protocolo que ajusta as fontes de tráfego dinamicamente de acordo com as condições impostas pela rede, torna-se difícil e custoso simular o comportamento de diversas conexões simultâneas que compartilham o mesmo enlace. As sessões TCP serão iniciadas por clientes FTP que se conectam a um servidor remoto através de um roteador dotado das diferentes políticas de tratamento de tráfego.

No entanto, ao se transmitir múltiplos arquivos simultaneamente a um servidor, é possível que este não consiga tratar todo o tráfego recebido, levando ao enfileiramento destes datagramas recebidos em um *buffer* por cada sessão no servidor. Quando isto ocorre, o servidor diminui o tamanho da janela de recepção anunciada, resultando na redução do tráfego que o emissor irá injetar na conexão. Se este *buffer* se esgota, o tamanho da janela é anunciado como zero, voltando a subir apenas quando o servidor tratar os pacotes recebidos (STEVENS, 1997).

Para avaliarmos o desempenho de cada política de escalonamento de tráfego temos que garantir, então, que o gargalo seja o roteador e não o servidor FTP. Assim, adicionou-se após cada elemento de entrada *FromDevice* do roteador um elemento limitador de banda chamado *BandwidthShaper*, configurado para taxa de 1Mbps. Este elemento limitador de banda deve ser utilizado associado a uma fila que o antecede, pois o elemento em si não realiza descartes. Desta forma, só haverá descartes quando as filas associadas às interfaces de entrada do roteador esgotem sua capacidade. O espaço de armazenamento padrão do CLICK é de 1000 pacotes e este foi o valor utilizado. Além disto, para compararmos os mecanismos dotados de duas filas a outro com uma única fila evitaremos possíveis descartes de pacotes nas interfaces de saída dos roteadores, tornando estas filas grandes o suficiente. O tamanho adotado de 10000 pacotes, obtido experimentalmente.

Para avaliar o efeito da política de diferenciação de fluxos sobre as sessões TCP, precisamos limitar os experimentos a conexões de apenas um salto. Testes realizados em redes com

topologias diferentes desta, incluindo inserção e retirada de tráfego em diferentes nós, acarretariam inúmeras combinações. A análise do impacto causado pela aplicação de uma política de tratamento de tráfego estaria sujeita a incontáveis correlações, não deixando claros os ganhos alcançados.

Para todos os experimentos foram utilizadas três máquinas conectadas diretamente umas as outras com cabos cruzados, como mostrado no quadro interno da Figura 3.4, evitando-se assim, a interferência nas medidas por outros equipamentos de rede. As máquinas têm a seguinte configuração de hardware: processador Intel Pentium *III*, 933MHz, NIC PCI 10/100 com clock 33MHz, e 512M*Bytes* de memória RAM 133MHz. Apenas na máquina roteadora utilizou-se memória de 768M*Bytes*.

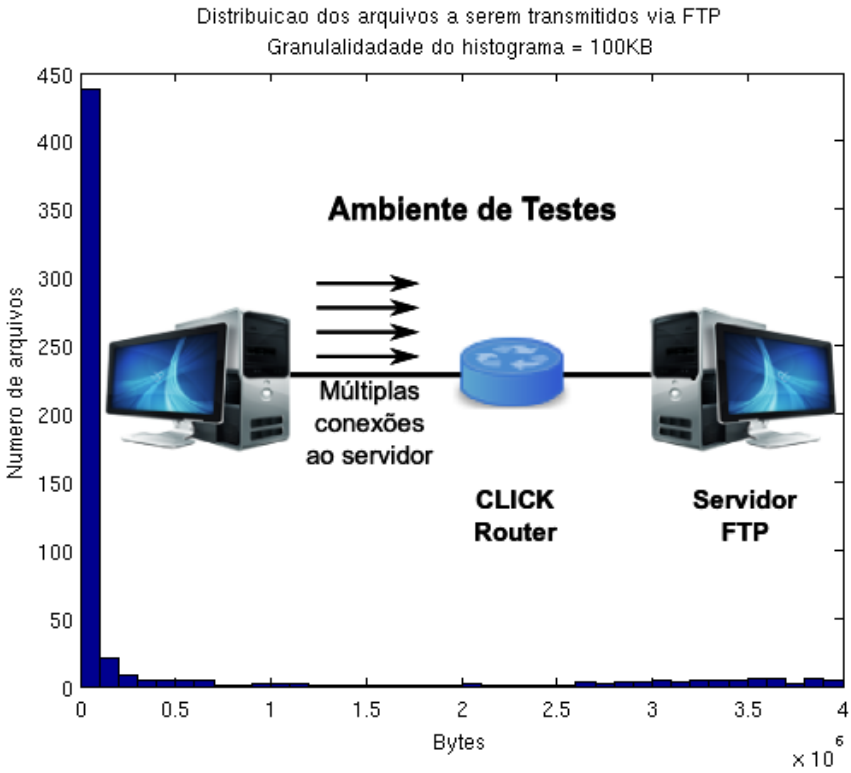


Figura 3.4: Ambiente de testes e base de arquivos transmitidos ao servidor FTP

A distribuição Linux utilizada neste trabalho para implementação dos roteadores é a Debian Etch. Já, para a máquina que será o servidor FTP e para a máquina que iniciará as conexões utilizou-se a distribuição Ubuntu 10.04. Todo sistema operacional é dotado de um escalonador de processos para compartilhamento da CPU entre os processos. Porém, os sistemas Linux não possuem um limite de *threads* simultâneas por processos, há somente um limite de número total de processos executando no sistema (*threads* são em essência processos com um espaço de endereço compartilhado). O número de máximo suportado por um sistema é definido por uma variável no *kernel* acessível no diretório /proc/sys/kernel/threads-max. Por padrão o valor

é igual a um quarto do número de páginas de memória. É possível aumentar esta capacidade alterando diretamente o valor da variável, ou editando o arquivo /etc/security/limits.conf. Evidentemente, o valor efetivamente alcançado dependerá dos recursos de *hardware* disponíveis.

3.2.2 Geração de Tráfego em Ambiente Controlado

Para o estabelecimento de sessões TCP simultâneas que disputassem os recursos da rede, desenvolveu-se um *script* em *python* que inicia transferências de arquivos em um diretório via FTP. Há a opção de transmissão de todos os arquivos simultaneamente, ou de se manter um número constante de *threads* de transferência ativas durante um tempo determinado. Neste caso, ao término de cada *thread* uma nova é criada, transmitindo um arquivo da base escolhido aleatoriamente. É importante observar que o adequado funcionamento da segunda opção advém da diminuição do tempo de estabelecimento da conexão FTP de controle, já que muitas vezes ele pode ser superior a duração da conexão para transferência de dados.

O servidor FTP utilizado foi o Proftpd. Por padrão este servidor realiza uma consulta reversa de DNS para o IP da máquina que tenta se conectar a ele. Quando não há servidor DNS configurado, é necessário esperar o *timout* da consulta para que a conexão FTP seja estabelecida, e este tempo gira em torno de dezenas de segundos. Para desabilitar esta opção utilizamos a diretiva *UseReverseDNS off* no arquivo de configuração do serviço. De forma análoga utilizamos a diretiva *IdentLookups off* para desabilitar o outro método de identificação da máquina remota, definido na RFC1413 (JOHNS, 1993).

Base de Arquivos Transmitida

Visando obter uma base consistente de arquivos com a qual se pudesse estudar um procedimento de *backup* de disco, realizou-se uma amostragem de 564 arquivos na faixa de 1KB até 4MB oriundos da listagem de todos os arquivos de um computador pessoal, resultando na distribuição mostrada na Figura 3.4. A granularidade adotada no histograma é de 100K*Bytes* e esta será utilizada em todo o trabalho.

O tamanho da base é limitado ao número máximo de *threads* concomitantes de transferência FTP, conseguido empiricamente através de um *script* em *python* para dar inicio às transferências. A limitação em 4MB no tamanho dos arquivos é dada por uma particularidade da técnica de classificação do RuN2C. Esta técnica propõe um grau de controle na geração dos números de sequência iniciais das sessões TCP, abordado na Seção 3.1.1.

Os arquivos da base foram gerados por um *script* que insere caracteres aleatoriamente den-

tro de arquivos obedecendo a distribuição do histograma. Todavia, todos os arquivos de uma faixa têm o mesmo tamanho, definido pelo valor médio dos arquivos de cada faixa da base originalmente amostrada. Os arquivos da primeira faixa, que definirão as tarefas caracterizadas como fluxos curtos, tem o tamanho de 10KB.

3.2.3 Testes de Vazão do Roteador CLICK

Para aferir o desempenho de um roteador implementado com a tecnologia CLICK, apontado na literatura como possuidor de uma capacidade de encaminhamento de pacotes superior a dos roteadores comerciais Cisco da série 7200 (KOHLE, 2000), utilizou-se de dois métodos de medidas em testes que executamos sobre a plataforma. Um destes métodos, interno ao roteador CLICK, utiliza das informações de *bytes* transmitidos providas pelos *handlers* dos elementos para gerar um *log* em tempo de execução. A partir deste *log* pode-se calcular a taxa média de transferência. O outro método foi a utilização do Iperf, renomado *software* das distribuições Linux para medição da banda disponível em enlaces, que também foi utilizado como fonte de tráfego. Os resultados destes testes com duas placas de rede 100Mbps mostram a coerência das medidas pelos dois métodos, alcançando taxas máximas de 72.8 Mbps, o que representa mais que 70% da capacidade bruta da camada física utilizada. Também é importante ressaltar o funcionamento adequado em nossos testes utilizando as duas modalidades de medidas do elemento *BandWidthShaper* limitando a banda em 1Mbps como desejado.

3.3 Padrão de Equidade

Propomos como padrão de equidade uma modelagem analítica da política *Processor Sharing* que distribuirá os recursos igualmente entre todos os fluxos ativos, NF_i (MUSSI; RIBEIRO, 2009). Na medida em que os fluxos são finalizados, a banda, B, disponível no roteador é redistribuida entre os fluxos que permanecem. Considerando A um vetor de n posições contendo em cada uma o número de arquivos de uma faixa de um histograma de granularidade G, o número de fluxos ativos será o indicado na Equação 3.1.

$$NF_i = \sum_{j=i}^{n} A(j) \tag{3.1}$$

No *Processor Sharing* a banda é igualmente distribuída disponível entre os fluxos ativos. Desta forma, a banda disponível para cada arquivo, B_E, é contabilizada pela Equação 3.2. No primeiro momento, quando todos os fluxos estão ativos, o tempo médio de transmissão completa

dos arquivos da primeira faixa é apresentado na Equação 3.3. O tempo médio de transmissão dos arquivos da i-ésima faixa do histograma, T_i, é dado pela Equação 3.4. Ele é calculado com base na média dos tamanhos dos arquivos desta faixa. Portanto, o desvio padrão segundo a equidade gerada pelo *Processor Sharing* deve ser igual a zero uma vez que todos os fluxos de uma mesma faixa terminariam exatamente no mesmo instante quando compartilhando banda sob tal regime.

$$B_E(i) = \frac{B}{NF_i} \tag{3.2}$$

$$T_1 = \frac{\frac{G}{2} \cdot 1024 \cdot 8}{B_E(1)} \tag{3.3}$$

$$T_i = T_{i-1} + \frac{G \cdot 1024 \cdot 8}{B_E(i)} \tag{3.4}$$

3.4 Dinâmica de Tráfego e Métricas Adotadas

Os roteadores serão avaliados sob duas circunstâncias distintas. Primeiramente, serão geradas tarefas concorrentes com início simultâneo, visando avaliar o comportamento das sessões no pior caso de concorrência de recursos. Porém, em regime normal de funcionamento novas sessões são iniciadas a todo tempo. Para avaliar este regime tráfego, submeteu-se os roteadores à tarefas concorrentes com início não simultâneo de forma a manter a carga, número de fluxos ativos, constante nos roteadores.

Em nossos experimentos utilizaremos como tarefa padrão será a transmissão de um arquivo dentro de uma base definida na Seção 3.2.1. Para isto, no FTP, é necessária a abertura de uma sessão TCP de controle e uma de dados. Há apenas poucos *bytes* transferidos na sessão TCP de controle. Sua influência é desprezível quando as tarefas têm inicio simultâneo, porém essa transação de controle necessária no FTP e em outros protocolos que necessitam de autenticação influi no tempo total de realização da tarefa encontrando uma rede com um dado regime de carga. Tal fato nos leva a propor dois cenários de teste correspondentes aos regimes anteriormente citados. Os tempos computados para a realização de cada tarefa a ser realizada (considera um fluxo na análise dos resultados) englobam desde o estabelecimento da sessão de controle até a sua finalização.

Dada a opção de realizar os experimentos com tráfego real, cada realização destes passa por situações específicas, não sendo possível garantir que a mesma conexão alcançará um valor de janela de congestionamento em determinado momento, ou que os mesmos pacotes dos fluxos

serão descartados no limitador de banda. Isto emula bem condições reais evitando a sincronização das janelas, mas gera um certo grau de ruído nos resultados. Assim, para equalizar os resultados dos experimentos realizados de forma a filtrar efeitos aleatórios de cada realização, todos os resultados apresentado será fruto do valor médio obtido entre três execuções do mesmo experimento.

3.4.1 Tarefas concorrentes com início simultâneo

Neste tipo de experimento, inicia-se a transmissão de toda a base de arquivos simultaneamente. Assim, avalia-se o comportamento das sessões no pior caso de concorrência de recursos. Esta avaliação acontece, inicialmente, através da análise temporal do cumprimento das tarefas impostas, apresentada por uma curva de número de fluxos ativos em relação ao tempo, que evidenciará a velocidade na finalização das sessões iniciadas em relação ao esperado pelo padrão de equidade proposto, onde os recursos são compartilhados igualmente entre os fluxos ativos. Quanto mais justa for a alocação de recursos pela política de tratamento de tráfego aplicada, mais próxima da curva computada pelo padrão de equidade sua estará a curva obtida experimentalmente.

Outra métrica adotada será a tempo médio de finalização dos fluxos iniciados para transmissão dos arquivos da base, definida na Seção 3.2.2. A média e desvio padrão serão computados sobre grupos de arquivos separados de acordo com o tamanho dos mesmos. O critério de separação dos grupos é relacionado à granularidade do histograma da base transmitida, apresentado na Figura 3.4. Arquivos pertencentes a uma mesma faixa de tamanho do histograma pertencem ao mesmo grupo.

Este experimento visa reproduzir situações reais como a execução de *backups*, de buscas distribuídas em *datacenters*, distribuição de tarefas em *clusters* de servidores, entre outros.

3.4.2 Tarefas concorrentes com início não simultâneo

Para uma análise em regime permanente das políticas de tratamento de tráfego desenvolvidas, este experimento visa simular um ambiente de tráfego real, onde usuários disputam a utilização de um servidor o submetendo ao limite capacidade de atendimento de requisições simultâneas. O tráfego das sessões estabelecidas entre os clientes e o servidor é tratado pelos roteadores implementados os submetendo a uma carga praticamente constante, do ponto de vista de número de sessões ativas. A cada requisição atendida, uma nova é iniciada, entrando na disputa pelos recursos de rede em desigualdade com sessões de longa duração, que alcançaram

grandes janelas de transmissão.

Cada requisição desta será a transmissão de um arquivo aleatoriamente escolhido, segundo uma distribuição de probabilidade uniforme, dentro da mesma base de arquivos utilizada no experimento anterior. Neste caso, o tempo médio de finalização das sessões não é uma boa métrica, já que elas não são submetidas sempre às mesmas condições de ocupação dos enlaces. Analisaremos, então, o número de requisições atendidas pelo servidor ao longo do tempo, pois se as conexões de curta duração, como a transmissão de um arquivo pequeno, forem priorizadas, menor será o tempo de duração desta, abrindo a possibilidade de que uma nova requisição seja atendida.

Porém, para que a análise seja justa, devemos constatar que o regime de requisições e serviço está em estado estacionário. Segundo a teoria de processos estocásticos, um processo aleatório é estacionário quando a natureza da aleatoriedade do mesmo não muda com o tempo. Em outras palavras, uma observação realizada num intervalo de tempo (t_0, t_1) exibe o mesmo padrão de aleatoriedade de uma observação realizada sobre outro intervalo $(t_{0+\tau}, t_{1+\tau})$ (LEON-GARCIA, 1994). Tomando, então, como variável aleatória o número de sessões servidas para transmissão de arquivos da base escolhidos de forma aleatório, segundo uma distribuição uniforme de probabilidades, devemos observar que a distribuição do número de arquivos com relação ao tamanho destes em processo de transmissão ou com a transmissão concluída, em um intervalo (t_i, t_{i+1}), deve ser igual a do instante $(t_{i+\tau}, t_{i+1+\tau})$.

4 Resultados

Serão expostos inicialmente neste capítulo os resultados obtidos para os experimentos de tarefas concorrentes com início simultâneo. Realizaremos, primeiramente, na Seção 4.1, um resgate das considerações realizadas nos capítulos anteriores. Na Seção 4.2 são apresentados os resultados para os padrões de comparação definidos, sendo seguido dos resultados obtidos para os classificadores de tráfego RuN2C na Seção 4.3 e para o RAFLE na Seção 4.4. Discutiremos ainda, nas Seções 4.3.2 e 4.4.2, a definição dos limiares e parâmetros utilizados nos mecanismos de classificação de tráfego implementados, RuN2C e RAFLE, respectivamente. Por fim, na Seção 4.5 será realizada uma comparação de todos os experimentos apresentados neste capítulo, adicionando neste ponto, também os resultados obtidos nos experimentos com tarefas concorrentes com início não simultâneo.

4.1 Introdução

Neste capítulo apresentaremos os resultados dos experimentos, sempre a média de três realizações de modo a filtrar ruídos comportamentais, de tarefas concorrentes com início simultâneo, através das métricas de número de fluxos ativos em relação ao tempo, tempo médio de finalização destes fluxos e seu respectivo desvio padrão, para o roteador limitado à taxa de 1Mbps, bem abaixo da capacidade de encaminhamento de 73 Mbps, prevenindo assim que limitações do servidor FTP afetem nossos resultados, e aumentando os *buffers* dos roteadores suficientemente para que se evite descartes nas filas de saída destes, possibilitando a comparação entre políticas de classificação dotadas de um número diferente de filas.

Iniciaremos a exposição pelos resultados dos padrões de comparação que serão utilizados para mensurar os ganhos obtidos pelos classificadores de tráfego. São eles: padrão de equidade analítico, roteador com disciplina *Drop Tail* e o roteador com classificador ideal. Logo após, determinaremos o limiar a ser utilizado no RuN2C e seguiremos com a apresentação dos resultados deste método. Prossegue-se com a analise para decisão do parâmetro de memória do RAFLE, sucedendo-se com os resultados por ele alcançados. Por fim, realiza-se uma compara-

ção de todos os resultados obtidos.

4.2 Padrões de Comparação

Normalmente, um sistema de comunicação baseado em comutação de pacotes que não implementa nenhum mecanismo de priorização de tráfego com a finalidade de prover QoS aos usuários utiliza a política de de uma fila FIFO com disciplina de serviço *Drop Tail*. Assim, esta política provê o tratamento mais simples possível aos pacotes em uma interface de saída de um roteador. Desta forma, este se torna uma base para o estabelecimento dos ganhos obtidos pelos classificadores de tráfego, em relação ao aumento da complexidade computacional. Desempenho inferior ao método *Drop Tail* implica na inviabilidade do classificador proposto.

O padrão de comparação analítico nos traz a ideia de equidade da solução implementada. Quanto mais às métricas adotadas se aproximarem do padrão mais os recursos estão sendo compartilhados de forma justa. No entanto, nem sempre a equidade na distribuição dos recursos disponíveis pode ser a melhor solução para aumentar o desempenho da rede. Priorizar a classe de fluxos que possuam curta duração pode liberar recursos para fluxos que demandam maior capacidade de banda. Fluxos curtos serão definidos como os que transferem um volume de dados maior do que 16KB. Os motivos desta definição serão detalhados em seções posteriores.

O último padrão adotado é um classificador que, assim como o RuN2C e o RAFLE, possui duas classes de tráfego. Porém, este classificador é ideal do ponto de vista de segregação dos fluxos iniciados no ambiente de testes. Com ele será possível delimitar os máximos valores que serão atingidos pelas técnicas RuN2C e RAFLE propostas neste trabalho.

4.2.1 Tarefas Concorrentes com Início Simultâneo

Neste experimento todos os arquivos da base serão enviados de forma simultânea, acarretando em concorrência dos recursos disponíveis. Implica na análise de pior caso, evidenciando o regime transitório do sistema. Prossegue-se com a análise das métricas adotadas.

Análise Temporal de Número de Fluxos Ativos

Como a transmissão dos arquivos é dada de forma simultânea, através de um *script* que inicia conexões FTP em diferentes *threads*, analisaremos a finalização destas conexões ao longo do tempo. É esperado que, quando se utiliza priorização estrita dos fluxos curtos pela disciplina PrioSched, que ocorra um decaimento mais veloz no número de fluxos ativos, já que estes são

muito maiores em número transferindo pequenas quantidades de *bytes*. No entanto, quando as classes de tráfego, curtas e longas, são servidas com a disciplina RR, espera-se que pouca diferença nesta métrica seja notada em comparação ao *Drop Tail*, já que o tempo serviço será igualmente distribuído para ambas as classes.

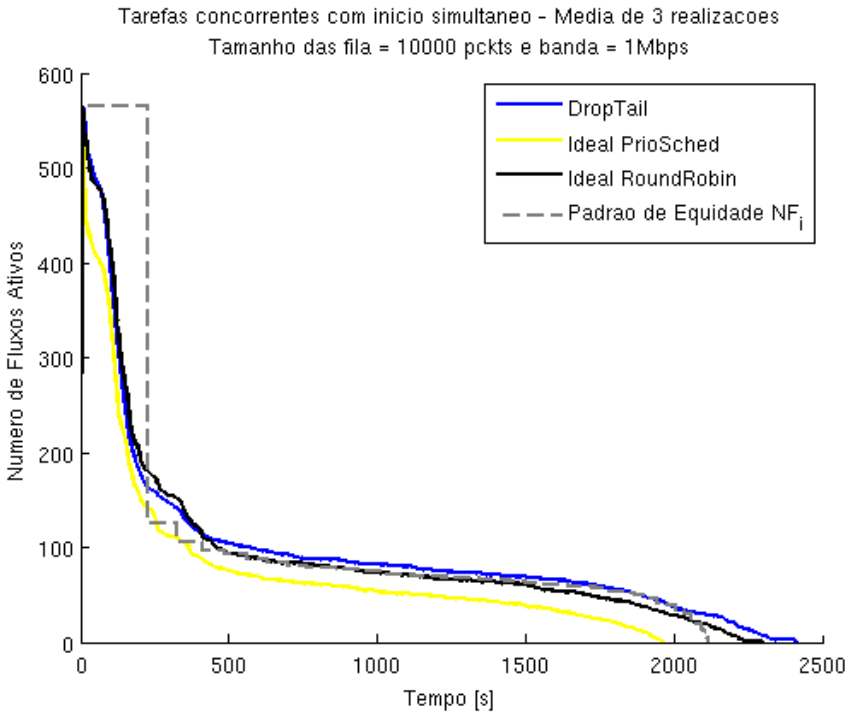


Figura 4.1: Diagrama comparativo de fluxos ativos por tempo para o classificador Ideal

Na Figura 4.1 é apresentada as curvas de números de fluxos ativos ao longo do tempo para as políticas de fila única e sem nenhuma diferenciação entre fluxos (denominada aqui como *DropTail*), RuN2C com escalonamento de prioridade estrita à fila de fluxos curtos (PrioSched) e RuN2C com escalonamento *Round Robin*.

Conforme mencionado na Sessão 3.4 um fluxo ativo refere-se ao cumprimento de uma tarefa, transmissão de um arquivo da base que engloba a conexão de controle e de dados do protocolo FTP. Portanto, o número de sessões TCP simultâneas é o dobro do número de fluxos ativos. Apresentamos ainda o padrão de referência analítico, $NF_i(t)$, para o caso ideal de divisão equânime de banda entre os fluxos. Nota-se que em $t = 0\ s$ são iniciadas simultaneamente as sessões FTP. Em $t = 225\ s$, quando se dá o término da transmissão dos arquivos localizados na primeira faixa do histograma da base de arquivos pelo método analítico, todas as políticas de tratamento de tráfego empregadas possibilitam a finalização de algumas sessões em tempo menor que o calculado por este padrão de equidade, indicando que, como esperado, algum privilégio foi dado a tais sessões em termos de divisão de banda. Nota-se que a priorização estrita para fluxos curtos possibilita maior finalização em número de fluxos. Entre este ponto e

$t = 410\ s$ a curva Ideal PrioSched é a que mais se aproxima do padrão de equidade.

Em um segundo momento, a partir de $t = 488\ s$ há a aproximação da curva Ideal RR a do padrão, quase que se sobrepondo a ela. Também, nota-se que a maior finalização inicial dos fluxos, quando se utiliza a disciplina PrioSched, libera recursos mais rapidamente, possibilitando o término dos demais fluxos de forma mais veloz até que o padrão de equidade. Percebe-se ainda, a proximidade do *Drop Tail* e do Ideal RR durante todo o tempo de experimento, sendo que ambos os métodos finalizam a transmissão dos arquivos em tempo superior ao padrão de equidade, já que a finalização inicial dos fluxos para o Ideal RR se dá de forma mais lenta que no PrioSched, atrasando a liberação dos recursos. Todavia, mesmo mais tarde, ela ocorre, beneficiando o Ideal RR em relação ao *Drop Tail* a partir do cruzamento das curvas em $t = 410\ s$. No *Drop Tail*, as grandes rajadas de fluxos longos podem atrasar a finalização de fluxos menores e, assim, não há como saber a que classe pertence os fluxos que estão sendo finalizados. E apesar dos resultados da Figura 4.1 já indicarem alguns benefícios dos mecanismos de priorização de fluxos curtos em relação ao *Drop Tail*, uma análise mais detalhada é necessária em termos do tratamento dado às sessões em função do tamanho dos arquivos por elas transferidos. Prossegue-se então, com a análise das demais métricas adotadas.

Tempo Médio de Finalização dos Fluxos

O tempo de transmissão de um arquivo é contado desde o início da conexão com o servidor FTP, incluindo a autenticação, até o fim da transmissão do mesmo, incluindo, assim, os ganhos proporcionados também pela priorização da conexão de controle de ambas as classes. Espera-se que os tempos médios de transmissão dos fluxos nos roteadores com mecanismo de diferenciação e priorização de fluxos curtos sejam menores que no roteador sem este mecanismo.

Na Figura 4.2 apresentamos os tempos médios para finalização dos fluxos contidos em cada faixa do histograma da base de arquivos para o roteador sem nenhum esquema de priorização de tráfego, o *Drop Tail*. Cada ponto deste é marcado um intervalo que indica o desvio padrão da medida. O padrão de equidade é apresentado por uma curva tracejada que liga os pontos das médias obtidos analiticamente, uma vez que o desvio padrão é sempre zero, facilitando a visualização da tendência dos valores experimentais.

Para a política de tratamento de tráfego *Drop Tail*, percebe-se que há uma alta variabilidade dos tempos médios entre faixas vizinhas. Esta variabilidade acontece, também, para o desvio padrão destas médias. O mesmo ocorre com valores obtidos para o classificador de tráfego Ideal RR, apresentados na figura 4.3. Nota-se que nos dois casos as médias obtidas, mesmo entre faixas vizinhas, tanto alcançam valores superiores, quanto inferiores aos definidos pelo

padrão de equidade, indicando aleatoriedade no beneficio de algumas conexões em detrimento de outras. Neste ponto, é importante observar que ocorrem ao centro dos gráficos pontos com desvio padrão igual a zero. Isto se deve a limitações no número de arquivos da base transmitida, onde pode haver um único arquivo situado em determinada faixa do histograma.

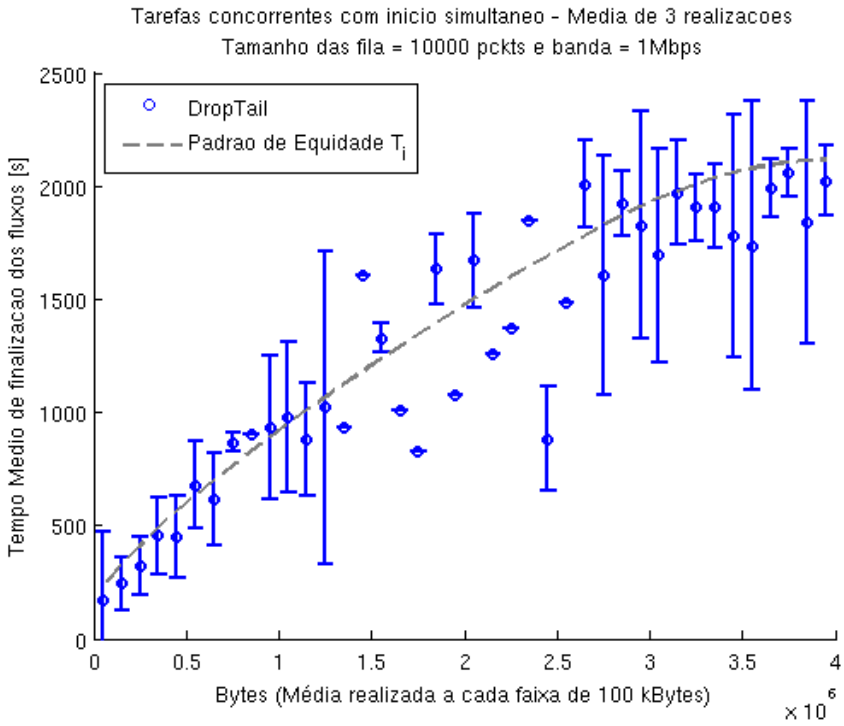


Figura 4.2: Tempos médios de finalização de fluxos e desvio padrão para o roteador sem priorização de tráfego, *Drop Tail*

Já os resultados obtidos para o classificador Ideal PrioSched, onde os fluxos curtos são estritamente priorizados, são apresentados na Figura 4.4. Para este caso, é importante salientar que a grande maioria dos tempos médios de finalização dos conjuntos de fluxos se encontra abaixo da curva do padrão de equidade. É possível visualizar que os tempos médios para a finalização das primeiras faixas de arquivos são menores que para as outras políticas experimentadas.

Ainda, nota-se que a finalização de fluxos das primeiras faixas diminui o número de conexões disputando recursos da rede, o que possibilita o término antecipado dos demais fluxos ainda ativos, reforçando os resultados obtidos na análise temporal do número de fluxos ativos.

Para evidenciar os ganhos alcançados para a faixa de interesse, fluxos curtos, é explicitado na Tabela 4.1 os valores de média e desvio padrão das cinco primeiras faixas de arquivos transmitidos das políticas que servirão como padrão de comparação aos demais classificadores. A partir desta tabela, é indicado que a priorização dos fluxos curtos surte efeito principalmente na faixa que contém os arquivos menores (i.e., primeira faixa do histograma). Neste caso, devido ao regime de privilégio dos fluxos curtos, os roteadores com o classificador ideal apresentam

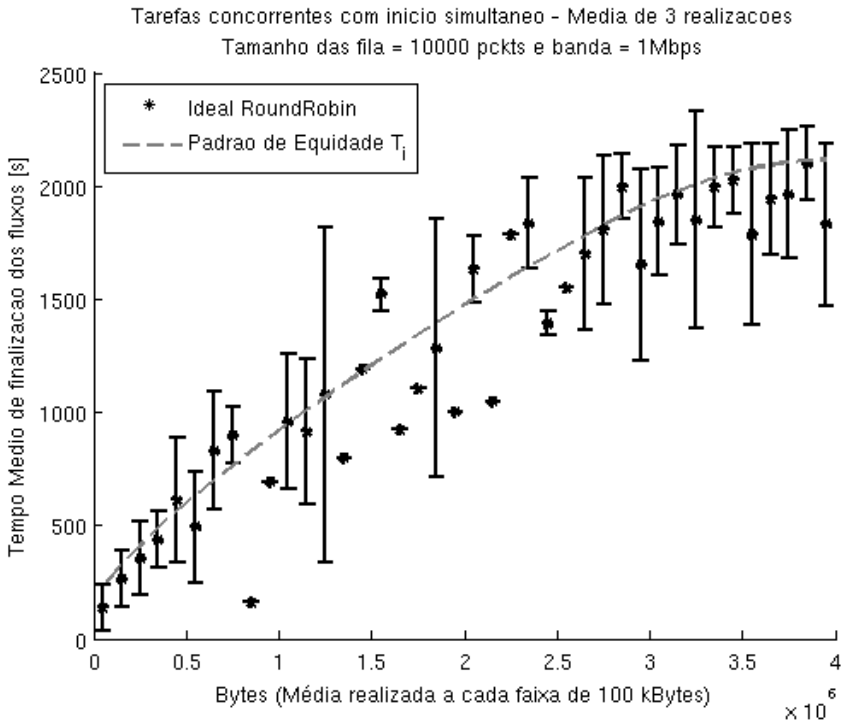


Figura 4.3: Tempos médios de finalização de fluxos e desvio padrão para o classificador ideal com escalonador *round robin*

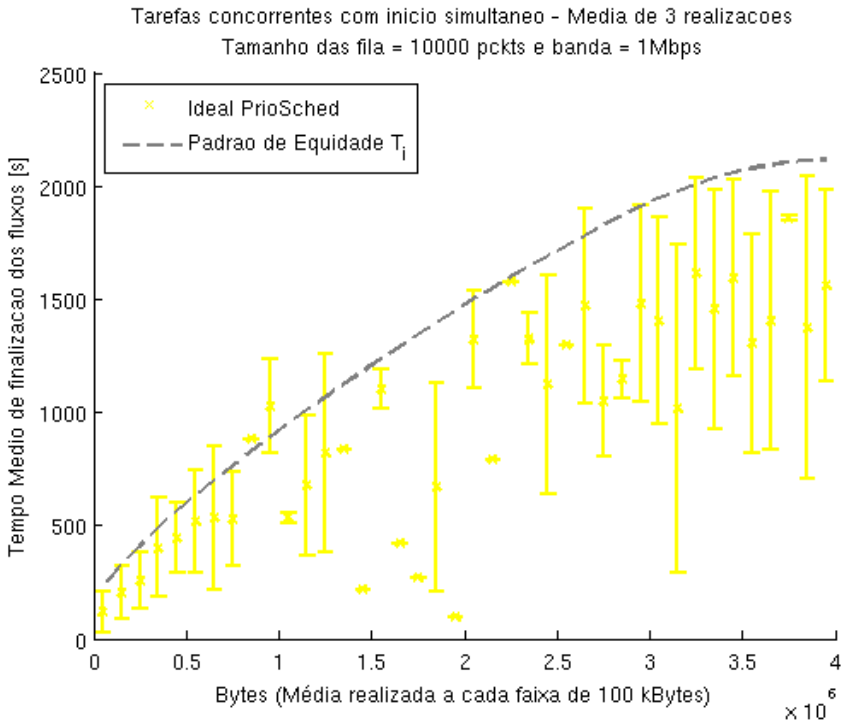


Figura 4.4: Tempos médios de finalização de fluxos e desvio padrão para o classificador ideal com prioridade estrita

ganhos em relação ao *Drop Tail*, destacando-se a disciplina *Priority Scheduling*. Os ganhos relativos ao tempo médio de transferência de arquivos pequenos são de $21,87\%$ e $44,69\%$ quando se emprega o classificador Ideal com disciplina de serviço RR e PrioSched, respectivamente. Já em relação ao desvio padrão os ganhos são de $198,44\%$ e $233,27\%$ apresentados na mesma ordem.

Tabela 4.1: Tempos médios de finalização e desvio padrão para o classificador ideal

Faixa	Padrão (T_i)		*Drop Tail*		Ideal PrioSched		Ideal RR	
[KB]	$\bar{t}$ [s]	σ [s]	$\bar{t}$ [s]	σ [s]	$\bar{t}$ [s]	σ [s]	$\bar{t}$[s]	σ [s]
0 à 100	225,600	0	168,267	304,951	116,297	91,475	138,074	102,183
100 à 200	326,400	0	242,231	117,352	205,309	116,348	263,371	124,156
200 à 300	410,400	0	320,642	126,085	257,593	126,818	352,689	161,392
300 à 400	488,000	0	453,563	169,347	402,133	218,720	436,262	126,313
400 à 500	562,400	0	450,703	182,594	444,927	153,426	612,494	274,780

Os ganhos obtidos pelo classificador ideal representam que se conseguiu um tratamento mais igualitário para os fluxos dentro da faixa de sessões de curta duração com ambas as disciplinas de serviço, RR e PrioSched. Estes resultados demarcam os limites que poderão ser alcançados pelos métodos RuN2C e RAFLE, já que estas propostas possuem classificadores que segregam fluxos curtos e longos de maneira estimada pois não mantêm o estado das sessões TCP.

Prosseguiremos a apresentação dos resultados obtidos para os classificadores propostos começando pelo RuN2C e passando em seguida para o RAFLE. Em cada seção específica será discutida a determinação dos parâmetros utilizados por estes classificadores de tráfego.

4.3 RuN2C

O *Running Number 2 Class* propõe diferenciar o tratamento dado aos pacotes pertencentes a fluxos curtos e longos (AVRACHENKOV; BROWN; NYBERG, 2004). Para este classificador de tráfego, a diferenciação destas duas categorias se dá através do número de *bytes* transferidos na sessão desde o momento de sua iniciação. Para que não seja necessário manter o estado de todos os fluxos ativos nos roteadores, o mecanismo de classificação utiliza a informação transportada pelo número de sequência do protocolo TCP.

Esta é, no conhecimento dos autores, a primeira implementação do classificador de tráfego RuN2C em roteadores físicos, nos trabalhos publicados são utilizados o simulador NS2 para realização dos experimentos. Dividiremos essa seção em três partes. Inicialmente realizaremos uma breve introdução ao mecanismo RuN2C, resgatando alguns conceitos. Em seguida, na

seção 4.3.2, é abordada a determinação do limiar de decisão para a classificação entre fluxos curtos e longos. E após isso, serão apresentados os resultados obtidos para este classificador de tráfego na Seção 4.3.

4.3.1 Introdução

No estabelecimento de uma sessão TCP um número de sequência inicial (ISN) é gerado aleatoriamente. A cada segmento enviado, este número é incrementado do número de *bytes* transportado. Desta forma, só o número de sequência dos pacotes não trariam informação suficiente para segregar fluxos longos dos curtos. Seria necessário armazenar os ISNs de cada fluxo, utilizando algoritmos complexos para controle de quais fluxos ainda estão ativos, ou quais foram interrompidos, reduzindo o custo com a memória total utilizada neste fim.

Para contornar esta situação, não sendo necessário manter estados das sessões nos roteadores é sugerida uma adequação simples na geração dos ISNs. Após a geração aleatória do número de sequência que possue 32 *bits*, aplica-se uma máscara que zere os 12 *bits* centrais, restando ainda 20 *bits* para garantir a aleatoriedade deste número inicial, o que equivale a aproximadamente um milhão de possíveis números de sequência iniciais. Com este controle é possível identificar o volume de *bytes* transferidos através de um limiar demarcado entre os 12 *bits* centrais até o máximo de 4MB. Este limite é imposto já que para comparação seria necessário ocultar os 10 *bits* mais significativos. Os 10 *bits* menos significativos provocariam um erro não relevante de no máximo 1KB, uma vez que este volume de dados pode ser transportado em apenas um pacote TCP.

Fluxos que transportam volume de dados superior à 4MB retornariam à categoria de fluxos curtos após o estouro dos primeiros 22 *bits*. Se o limiar for escolhido adequadamente, apenas alguns pacotes da classe de fluxos longos serão beneficiados, já que as janelas de transmissão desta classe frequentemente alcançam altos valores. A discussão para determinação deste limiar é apresentada a seguir.

4.3.2 Metodologia

Para uma solução de determinação de um limiar que satisfaça o compromisso de favorecer os fluxos curtos, sem que sejam notados grandes prejuízos aos longos, é importante ressaltar que os fluxos TCP curtos são altamente propensos a *timeouts* após a perda de um pacote. O impacto disto é extremamente relevante para o tempo de resposta deste tipo de fluxos, uma vez que o tempo mínimo para *timeout* é de 1 segundo (PAXSON; ALLMAN, 2000). Para evitar este

impacto, deve-se dar prioridade a estes fluxos antes que a janela de congestionamento do TCP atinja 3 ou 4 pacotes, o que ocorre, por exemplo, se são transmitidos 8 pacotes. Isto totaliza um limiar de aproximadamente 12KBytes, para um MSS igual a 1460 KBytes.

Outro ponto é que se os fluxos TCP na Internet são descritos por uma distribuição cauda-pesada (THOMPSOM; MILLER; WILDER, 2004)(JR.; RIBEIRO; WALDMAN, 2005), fluxos abaixo de 8 pacotes representam um significante número do total dos fluxos TCP, mas uma pequena porção da carga total. Assim, ao se dar prioridade a estes fluxos, atendemos ao requisito de não haver prejuízo digno de nota aos fluxos longos.

A Figura 3.2, apresentada anteriormente, define o padrão para o número de sequência utilizado pelo RuN2C. A decisão do limiar, TH, ideal é amplamente discutida em (AVRACHENKOV; BROWN; NYBERG, 2004). Neste artigo de referência o limiar determinado em 12 KB sendo necessários os $TH = log_2(12 \times 1024) \approx 14$ bits menos significativos do campo do número de sequência. Ressalta-se ainda que a escolha deste limiar é feita em número de bits, assim ele é determinado seguindo potências de dois (2^x).

Tendo como referência teórica o limiar de 14 *bits*, 16KB, iniciou-se então a etapa experimental para determinação do limiar ideal. Executando o experimento de tarefas concorrentes com início simultâneo, implementamos um roteador em que as filas do classificador RuN2C são cascateadas com um classificador ideal. Assim, contabilizou-se os pacotes classificados como pertencentes a fluxos curtos pelo RuN2C nas categorias de acerto e erros de classificação. O mesmo foi realizado para os pacotes classificados como fluxos longos. Em seguida calculou-se a probabilidade de um pacote ser classificado corretamente pelo RuN2C (Probabilidade de acertos nos fluxos curtos (PA_c)) e a dessa cassificação ser realizada de forma errônea (probabilidade de erros nos fluxos curtos (PE_c)). Seguindo este padrão calculou-se, também, a probabilidade de acertos de erros de classificação para fluxos longos, PA_l e PE_l, respectivamente.

Definiu-se, então, um índice que aponta a qualidade da classificação do RuN2C, tendo como base as probabilidades de acertos de classificação, como o produto $PA_c \cdot PA_l$. Na tabela 4.2 são apresentadas as probabilidades calculadas para cada limiar testado, seguido do respectivo índice de acerto. Limiares acima de 32KB (i.e. 64KB ou 128KB) não são evidenciados nesta tabela, já que um fluxo de 64KB transmitiria cerca 44 pacotes, o que ultrapassaria em mais que o dobro do número médio de pacotes trocados por fluxos curtos na Internet, cerca de 20 datagramas (THOMPSON; MILLER; WILDER, 1997). E se também considerarmos o tratamento dos fluxos em *data center* as sessões de curta duração alcançam no máximo 10KB de transferência (GREENBERG et al., 2009), ficando muito distante destes limiares.

Analisando-se os índices na Tabela 4.2 vemos que apesar de os pacotes de fluxos curtos

Tabela 4.2: Definição experimental do limiar para o RuN2C

Limiar	Fluxos Curtos		Fluxos Longos		Índice
TH [KB]	PA_c	PE_c	PA_l	PE_l	$PA_c \cdot PA_l$
8	0,885	0,115	0,982	0,018	0,870
16	1	0	0,979	0,021	0,979
32	1	0	0,969	0,031	0,969

$PA_c \rightarrow$ Probabilidade de acerto para pacotes de fluxos curtos
$PE_c \rightarrow$ Probabilidade de erro para pacotes de fluxos curtos
$PA_l \rightarrow$ Probabilidade de acerto para pacotes de fluxos longos
$PE_l \rightarrow$ Probabilidade de erro para pacotes de fluxos longos

serem classificados corretamente tanto no limiar de 16KB, quanto no de 32KB, devido a limitação da base de arquivos transmitida em 4MB, a diferença no índice de acerto de pacotes se encontra nos fluxos classificados como longos pelo RuN2C, favorecendo a escolha do limiar de 16KB. Somando a este resultado o fato de que na Internet ou nos data centers as transferências não são limitadas a este volume, quanto maior for o limiar, mais pacotes serão classificados como pertencentes a fluxos curtos incorretamente, quando a transferência de dados ultrapassar de 4MB. Portanto, considera-se o limiar de 16KB mais adequado, assim como o utilizado em (AVRACHENKOV; BROWN; NYBERG, 2004).

4.3.3 Resultados: Tarefas Concorrentes com Início Simultâneo

A partir da determinação do limiar de classificação iniciou-se os experimentos de transmissão simultânea dos arquivos da base, via FTP, para os escalonadores de tráfego combinados com o RuN2C. Aqui, os resultados serão inicialmente comparados ao padrão de equidade e aos números obtidos nos experimentos para os classificadores ideais, que indicam os limites máximos de ganho. Em seguida evidenciaremos também os ganhos alcançados em relação ao roteador *Drop Tail*.

Análise Temporal de Número de Fluxos Ativos

Na Figura 4.5 temos o número de fluxos ativos ao longo do tempo para o RuN2C com escalonamento de prioridades estrita à fila de fluxos curtos (RuN2C 16k PrioSched) e para o RuN2C com escalonamento *Round Robin*. Juntamente a estas curvas são apresentadas as curvas para os classificadores de tráfego com classificador ideal e o padrão de referência analítico, para o caso ideal de divisão equânime de banda entre os fluxos.

Nota-se que desde o início da transferência simultânea dos arquivos, em $t = 0\ s$, até o instante $t = 100\ s$ o RuN2C possibilita a finalização de algumas sessões em tempo menor que o indicado pelo padrão de equidade. Deste fato, conclui-se que algumas sessões foram privilegiadas em termos de divisão de banda, como era esperado. Todavia, contrariando a intuição, de que o escalonamento com prioridade estrita seria a melhor forma de oferecer este privilégio, a política *Round Robin* apresenta maior número de fluxos finalizados em menor tempo, enquanto com a utilização da disciplina PrioSched, há um atraso no término de fluxos.

O RuN2C RR finaliza mais fluxos inclusive o classificador ideal utilizando a disciplina RR, dentro desta faixa de tempo de transferência. Este ganho é decorrente do processo de classificação inicial do RuN2C, que classifica todos os pacotes que ainda não transferiram o volume de dados indicado pelo limiar adotado como fluxos curtos, possibilitando o crescimento inicial da janela de transmissão sem tratamento diferencial. Este efeito é mais significativo no desenvolvimento das sessões das faixas iniciais do histograma da base de arquivos transmitidas, já que estas faixas transmitirão um número menor de pacotes, permanecendo por um tempo pequeno na fila de fluxos longos.

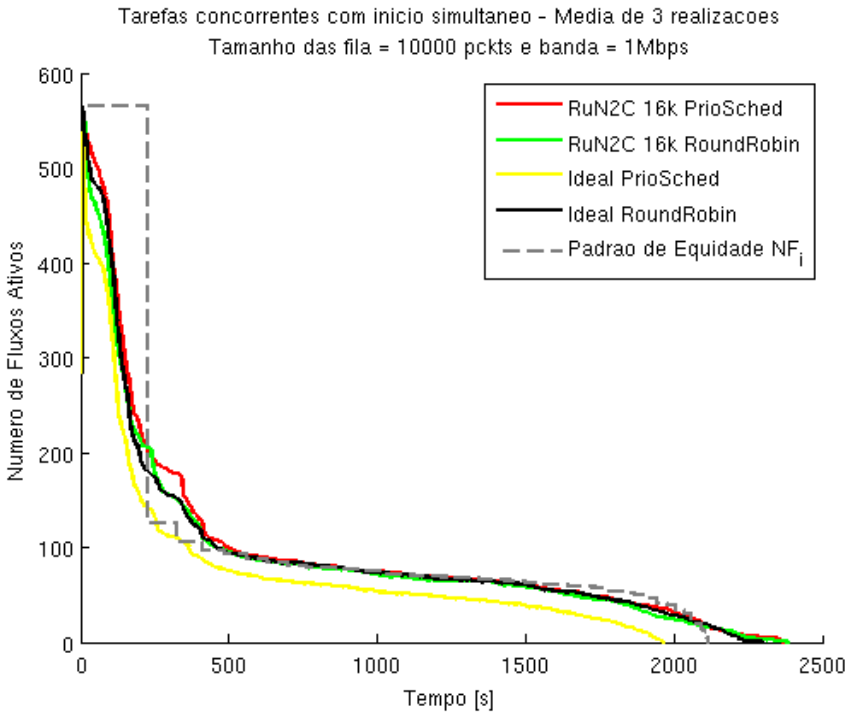


Figura 4.5: Diagrama comparativo de fluxos ativos por tempo para o classificador RuN2C

Nos próximos 100 segundos, até $t = 200\ s$ a finalização de fluxos dos classificadores RuN2C RR e PrioSched, ficam praticamente justapostos com a curva do classificador ideal RR, porém uma pequena diferença é percebida quando se utiliza a disciplina PrioSched no classificador RuN2C, que se deve principalmente ao atraso inicial ocorrido para esta disciplina. Deste ponto

até $t = 500\ s$ há um distanciamento seguido de nova aproximação da resposta do RuN2C PrioSched em relação as demais curvas. A partir daí, as três curvas se sobrepõem ao padrão de equidade até o instante $t = 1500\ s$. Neste momento, o padrão de equidade apresenta um decaimento mais brusco no número de fluxos ativos até o instante analítico de término da transferência em $t = 2150\ s$. O término da transferência nos classificadores que utilizaram priorização de tráfego durante o início do experimento decai de maneira mais suave ao final deste, de forma a compensar o tempo dedicado à priorização. O experimento para os classificadores RuN2C termina por volta de $t = 2400\ s$.

Tempo Médio de Finalização dos Fluxos

São apresentados nas Figuras 4.6 e 4.7 os tempos médios de finalização dos fluxos por faixa do histograma da base de arquivos transmitida de forma simultânea. Nestas figuras, também é traçado do padrão de equidade analítico. Quanto mais próximo desta curva, mais justa é alocação dos recursos para os fluxos ativos na rede.

Percebe-se que o comportamento global dos tempos médios de ambos classificadores RuN2C se assemelha ao do classificador Ideal RR, apresentado anteriormente na Figura 4.3, assim como notado na análise temporal de número de fluxos ativos. Todavia, é possível notar que no RuN2C os tempos médios imediatamente subsequentes à primeira faixa, que representa o tempo médio de transferência dos fluxos curtos, se aproximam fortemente do padrão de equidade, evidenciando que para estes fluxos, não houverão prejuízos decorridos da priorização efetuada. Isto ocorre até nas faixas de arquivos de 1KB até 1MB, e de 3MB à 4MB. No intervalo intermediário os tempos médios se tornam mais aleatórios, se afastando do padrão de equidade, tanto para tempos maiores, quanto para menores que os demarcados por este. Isto pode ser fruto de uma limitação da base de arquivos transmitida, já que ela contém muito poucos arquivos por faixa nesta porção do histograma e, às vezes, apresenta um único arquivo, o que implica no desvio padrão ser igual a zero.

A exemplo da análise anteriormente realizada quantificam-se, agora, os ganhos obtidos no desempenho dos fluxos curtos através dos dados exibidos na Tabela 4.3, já que pôde-se notar que não houve prejuízos dignos de nota ás conexões de longa duração. Assim, contabilizam-se ganhos de $7,13\%$ e $14,98\%$ para os tempos médios de transmissão de fluxos curtos quando se utiliza os classificadores de tráfego RuN2C PrioSched e RR, respectivamente. Estes resultados, mostram que ao contrário do que se se acredita no artigo de proposição do RuN2C (AVRACHENKOV; BROWN; NYBERG, 2004), uma vez que apenas o escalonador PrioSched é utilizado, o RR se apresenta mais eficiente.

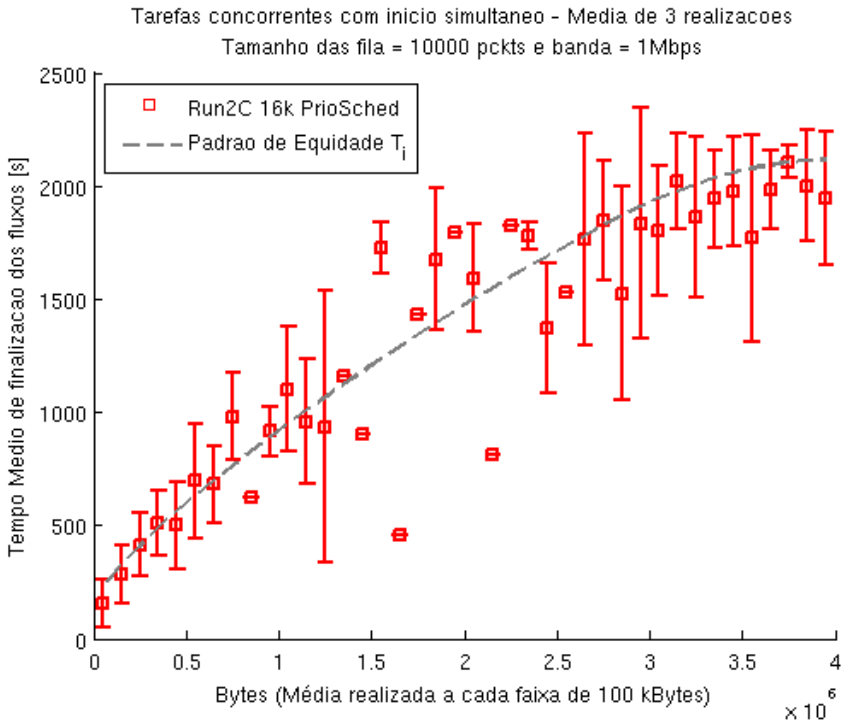


Figura 4.6: Tempos médios de finalização de fluxos e desvio padrão para o classificador RuN2C com escalonador de prioridade estrita

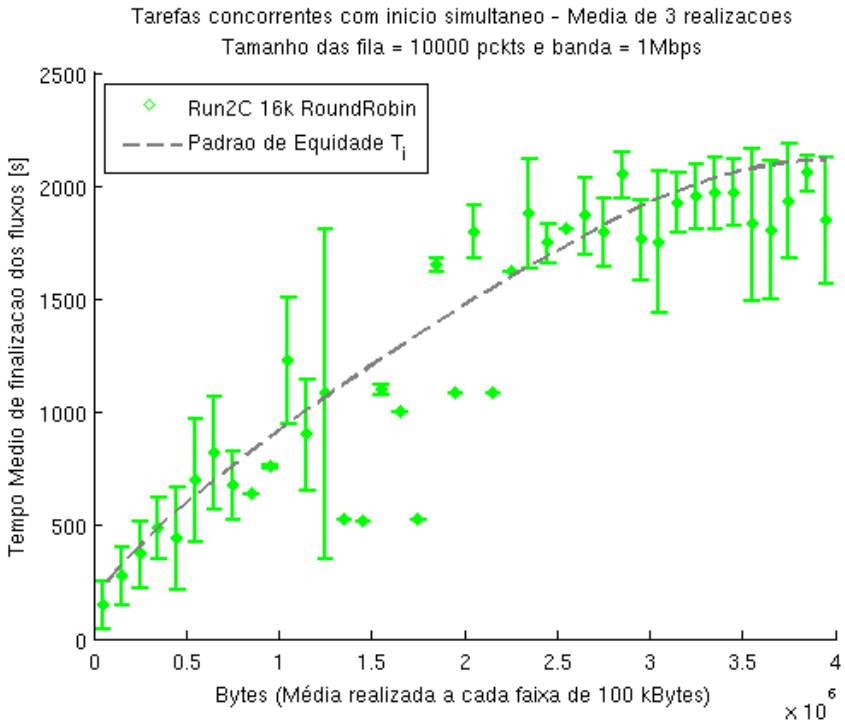


Figura 4.7: Tempos médios de finalização de fluxos e desvio padrão para o classificador RuN2C com escalonador *round robin*

Tabela 4.3: Tempos médios de finalização e desvio padrão para o RuN2C com limiar de 16KB

Faixa	Padrão (T_i)		*Drop Tail*		RuN2C PrioSched		RuN2C RR	
[KB]	$\bar{t}$ [s]	σ [s]	$\bar{t}$ [s]	σ [s]	$\bar{t}$ [s]	σ [s]	$\bar{t}$[s]	σ [s]
0 à 100	225,600	0	168,267	304,951	157,072	107,247	146,344	107,868
100 à 200	326,400	0	242,231	117,352	281,258	127,090	277,436	130,317
200 à 300	410,400	0	320,642	126,085	411,895	139,599	370,537	149,912
300 à 400	488,000	0	453,563	169,347	509,720	145,537	487,665	133,801
400 à 500	562,400	0	450,703	182,594	500,218	192,643	440,305	226,393

Também, obtém-se ganhos de $184,34\%$ e $182,70\%$ para os desvios padrão destes tempos, apresentados na mesma ordem, PrioSched e RR. Quando comparados aos ganhos efetuados pelos classificadores ideais, nota-se que os obtidos com o RuN2C RR são aproximadamente 2/3 dos ganhos para os tempos médios com classificador Ideal RR. Quando são comparados os classificadores com disciplinas PrioSched, os resultados obtidos pelo RuN2C são menores que 1/5 quando o classificador ideal é utilizado. Entretanto, com relação ao ganho na diminuição da variabilidade, os valores se aproximam bastante dos obtidos com o classificador ideal RR, como se esperava pelas análises gráficas efetuadas.

Os resultados obtidos pelo RuN2C se aproximam do padrão de equidade. Neste sentido, seus resultados alcançam desempenho próximo ao obtido pelo RCP (DUKKIPATI, 2007). O ganho ocorre de forma mais efetiva na implementação, uma vez que o RCP requer alteração completa do protocolo de transporte, enquanto no RuN2C a exigência é apenas a alteração do processo de geração dos números de sequência iniciais.

No entanto, apesar das vantagens obtidas com a implementação do classificador RuN2C, sua exigência por adequar a geração dos ISNs ainda pode criar resistência para sua utilização em ambientes comerciais. Muitos esforços podem ser necessários para convencer as empresas desenvolvedoras de sistemas operacionais a realizar a adequação, já que outros mecanismos de utilização consagrada podem utilizar também do campo de número de sequência, como é o caso do SYN *cookie* (HARDENING..., 2010). Este mecanismo oferece proteção contra esgotamento de memória no caso de um ataque SYN.

Um pacote com a *flag* SYN marcada inicia o processo de *three-way handsake* do TCP para abertura de uma sessão. Cada pacote deste tipo gera uma conexão semi aberta que é estocada numa região de memória específica alocada pelo SO. Durante ataque SYN, incontáveis pacotes SYN são enviados ao alvo por máquinas que camuflam seu IP de origem via IP *spoofing*, impossibilitando a utilização de um filtro que boqueie o ataque. São tantas conexões semi abertas que a memória alocada é esgotada não permitindo que novas conexões sejam abertas.

O SYN *cookie* é o resultado de uma função *hash* com o IPs e portas de origem e destino de um pacote com TCP com a *flag* SYN marcada, além de outros valores secretos. Este *cookie* possue o mesmo tamanho que um número de sequência e é utilizado como número de sequência inicial na máquina que recebe a tentativa de inicio de conexão. Com a utilização do *cookie* não é necessário guardar as conexões semi abertas na memória. A máquina consegue identificar uma conexão que passou pelo processo inicial do *three-way handsake* comparando o resultado da função *hash* do último pacote do processo de abertura de conexão, um pacote ACK, com o número de confirmação (*acknowledgement number*) deste pacote.

Portanto, um mecanismo que não dependa de alteração de mecanismos que se encontram operacionalizados e alcancem resultados tão satisfatórios quanto os produzidos pelo RuN2C tornaria-se mais interessante. Esta é a proposta do RAFLE e seus resultados serão apresentados na próxima seção.

4.4 RAFLE

Seguindo a linha de classificação de fluxos em duas categorias, longos e curtos, proposta no RuN2C, propomos o método *Random Assorter of Flow LEngths* (RAFLE). Seu algoritmos de classificação derivam da técnica CHOKe, utilizada para oferecer proteção de fluxos TCP contra fluxos UDP que inundem uma rede (WANG; TANG; LOW, 2003).

O RAFLE também não necessita da manutenção de estados dos fluxos ativos no roteador. Realiza a inferência através de uma pequena memória da identificação de fluxo dos últimos pacotes encaminhados por uma interface do roteador. Realiza-se um resgate teórico do RAFLE na Seção 4.4.1, que é seguido da determinação do parâmetro que define o tamanho da memória recente, Seção 4.4.2.

4.4.1 Introdução

Conexões TCP realizam a transmissão de informação através de rajadas de pacotes. O número de pacotes de uma rajada é determinado pela janela de transmissão da conexão. No princípio evitar a possibilidade de enviar muitos datagramas em uma rede congestionada, piorando a situação, o crescimento da janela entra na fase de *slow start*. Nela a janela começa em 1MSS e cresce quase que exponencialmente a cada RTT. Quando se detecta a perda de pelo menos um pacote, a sessão entra em fase de congestionamento e o crescimento é de 1MSS por RTT, caso a perda seja detectada por três ACKs duplicados, ou se ocorrer *timeout* a janela cai

para 1MSS o *slow start* é reiniciado.

Devido à característica transmissão em rajadas do TCP é provável que se receba vários pacotes em sequência pertencentes a um mesmo fluxo. Armazenando o ID do fluxo dos últimos pacotes encaminhados em uma memória é possível tentar inferir se um fluxo é longo ou curto, uma vez que quanto maior for a duração do fluxo, maior se espera que seja sua janela de transmissão e mais provável do ID deste fluxo se encontrar na memória.

No RAFLE, a cada pacote recebido uma posição desta memória é sorteada e comparada com o ID de fluxo do pacote atual. Se houver coincidência, o pacote é considerado como pertencente a um fluxo longo, caso contrário a um fluxo curto. Quanto mais pacotes de uma rajada são tratados, mais provável é de o ID sorteado na memória coincidir com um novo pacote que entra no sistema, e maior a probabilidade deste pacote ser inserido na fila de fluxos longos.

Desta forma, o RAFLE segrega pacotes de fluxos que ocupam muitos recursos da rede, isto é, possuam alta taxa de transmissão de pacotes. Já o RuN2C, onde a identificação de fluxos longos é realizada pelo volume total de dados transmitidos, penaliza, também, fluxos que utilizem uma fatia muito pequena de banda, mas tenham grande tempo de duração. Porém, para que o algoritmo de classificação do RAFLE seja eficiente é necessário determinar o tamanho ideal de memória recente a ser utilizada.

4.4.2 Metodologia

A determinação do tamanho da memória recente utilizada no RAFLE (parâmetro N) deve resultar na priorização de fluxos curtos. É importante perceber que quanto menor for esta memória mais chance existe de um pacote pertencente a um fluxo curto ser direcionado para a fila de fluxos longos. Por outro lado, quanto maior esta memória for menor se torna a probabilidade do ID de fluxo de um pacote pertencente a uma rajada de um fluxo longo ser sorteado dentro das N posições desta memória. Assim, a determinação de N deve encontrar um valor intermediário as duas situações.

Desta forma, realizou-se um experimento com um roteador com classificadores ideais cascateados às filas classificadas pelo RAFLE submetido à transmissão simultânea dos arquivos da base. A exemplo do experimento realizado na Seção 4.3.2 calculou-se as probabilidades PA_c, PE_c, PA_l, PE_l e o índice de acerto $PA_c \cdot PA_l$, para diferentes valores de N. Quanto maior for o índice de acerto, mais adequada é a identificação. São apresentados na Tabela 4.4 os resultados destes cálculos.

Analisando o índice de acerto, obtém-se que o melhor valor de N para se utilizar como

limiar é $N = 40$. No entanto, percebe-se que para $N = 10$ o valor encontrado para o índice de acerto é próximo do índice de $N = 40$. O principal problema em utilizar este valor baixo de N é que a aumentar a probabilidade de deslocar um dos últimos pacotes de um fluxo curto para a fila de longos, fazendo com que haja um atraso que pode ser responsável pelo não atendimento dos requisitos de latência de uma aplicação. Sendo assim, deve-se escolher um N que possua um valor não tão pequeno quanto 10 e que ocasione o maior índice de acerto. Desta forma, opta-se por utilizar o valor de 40 posições de memória para armazenagem dos identificadores de fluxo dos últimos pacotes encaminhados, $N = 40$.

Ressalta-se que experimentos empíricos realizados com a variação do parâmetro N corroboram com as indicações apontadas anteriormente de que o parâmetro $N = 40$ é o mais adequado. Prossegue-se então com a análise temporal da finalização dos fluxos iniciados simultaneamente, dos tempos médios de transmissão e respectivo desvio padrão, comparando os resultados obtidos aqui com o classificador de tráfego RAFLE, com os alcançados na Seção 4.2.

Tabela 4.4: Definição experimental do tamanho da memória recente (N) para o RAFLE

Limiar	Fluxos Curtos		Fluxos Longos		Índice
N	PA_c	PE_c	PA_l	PE_l	$PA_c \cdot PA_l$
10	0,937	0,063	0,140	0,860	0,131
30	0,976	0,024	0,096	0,904	0,094
40	0,979	0,021	0,139	0,861	0,136
50	0,983	0,017	0,094	0,906	0,092
60	0,986	0,014	0,079	0,921	0,078
100	0,989	0,011	0,087	0,913	0,086

$PA_c \rightarrow$ Probabilidade de acerto para pacotes de fluxos curtos
$PE_c \rightarrow$ Probabilidade de erro para pacotes de fluxos curtos
$PA_l \rightarrow$ Probabilidade de acerto para pacotes de fluxos longos
$PE_l \rightarrow$ Probabilidade de erro para pacotes de fluxos longos

4.4.3 Resultados: Tarefas Concorrentes com Início Simultâneo

Após o estabelecimento do tamanho da memória recente ($N = 40$) a ser utilizado nos classificadores de tráfego RAFLE, inciaram-se os experimentos de tarefas concorrentes com início simultâneo. Os resultados obtidos para este classificador serão comparados com os obtidos na Seção 4.2 e com o padrão de equidade analítico proposto.

Análise Temporal de Número de Fluxos Ativos

A finalização de fluxos ao longo do tempo efetuada pelos classificadores RAFLE é apresentada na Figura 4.8. As curvas são apresentadas juntamente com o padrão de equidade analítico e os resultados obtidos para os classificadores ideais.

Inicialmente, observa-se a grande semelhança entre o comportamento, ao longo de todo experimento, das finalizações alcançadas pelo RAFLE e pelo classificador ideal, quando são utilizadas as mesmas disciplinas de serviço para as filas. Na faixa entre $t = 200\ s$ e $t = 500\ s$ o ocorre o espelhamento da divergência entre os classificadores RAFLE PrioSched e RR quando comparado com os seus respectivos classificadores ideais.

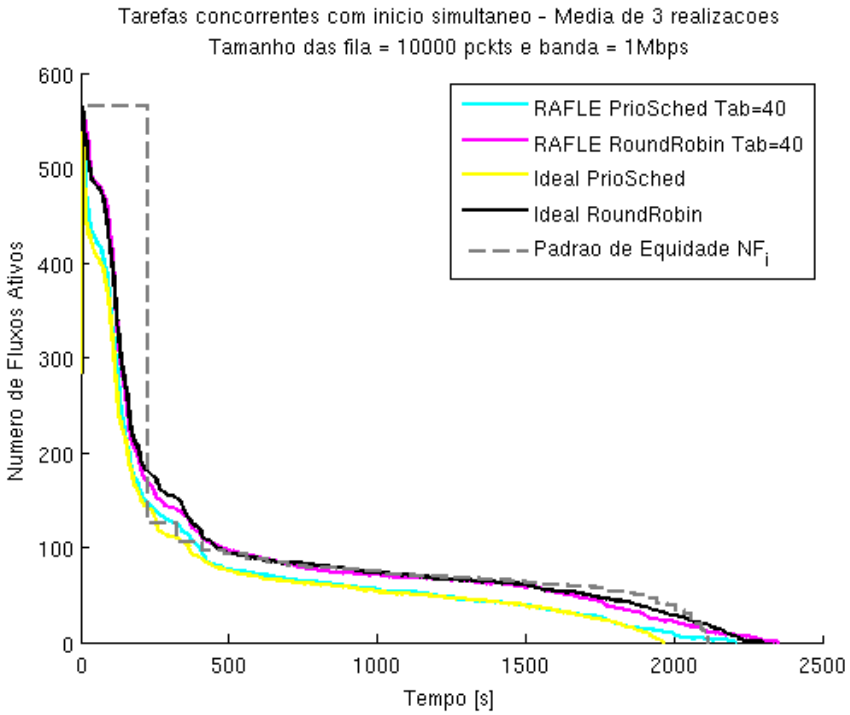


Figura 4.8: Diagrama comparativo de fluxos ativos por tempo para o classificador RAFLE

A partir de $t = 1900\ s$ a finalização dos fluxos do RAFLE PrioSched se distancia do ideal PrioSched e vai ao encontro do tempo de finalização global dos fluxos para os classificadores RAFLE e ideal com disciplina RR. Salienta-se por fim, que quando analisamos as curvas em relação ao padrão de equidade, os classificadores evidenciados nesta seção que utilizam RR conseguem maior aproximação. Apenas para o intervalo entre $t = 225\ s$ e $t = 410\ s$ a utilização da disciplina de serviço PrioSched alcança maior proximidade ao padrão de equidade. Para melhor aferir a proximidade do padrão de equidade passa-se a análise do tempo médio de finalização dos fluxos e seus respectivos desvios padrão.

Tempo Médio de Finalização dos Fluxos

São apresentados nas Figuras 4.9 e 4.10 os tempos médios de finalização de fluxos, juntamento com o intervalo relativo aos desvios padrão das médias efetuadas para o RAFLE. Para que se possa analisar a tendência em relação a justiça na alocação de recursos, mostra-se, também, o padrão de equidade analítico.

Assim como na análise temporal, o comportamento global dos tempos médios se aproxima aos resultados obtidos pelos classificadores ideais, quando as mesmas disciplinas de serviço para as classes diferenciadas são utilizadas. A procedência deste fato era esperada, devido a proximidade das curvas de número de fluxos ativos ao longo do tempo.

Do ponto de vista da tendência dos gráficos, observa-se que quando se utiliza a disciplina RR os tempos médios aproximam-se de forma mais justa ao padrão analítico. Em oposição, é notado maior espalhamento e aleatoriedade para os tempos médios para as faixas de arquivos acima de 1MB no classificador RAFLE PrioSched, sendo que em sua maior parte, os tempos médios estão abaixo da linha do padrão de equidade. Apesar disto, a penalização dos fluxos deste intervalo ocorre no sentido do desvio padrão, que cresce aumentando as barras associadas aos pontos médios.

Na Tabela 4.5 são apresentados valores dos tempos médios e desvios padrão, para que se possam calcular os ganhos alcançados pelo classificador RAFLE. Nesta tabela, também se encontram os valores obtidos nos experimentos com o roteador sem priorização de fluxos, *Drop Tail*.

Os ganhos em relação ao *Drop Tail* para os tempos médios, apresentados na Seção 4.3, quando os classificadores RuN2C foram utilizados, corresponderam a 2/3 dos obtidos com o classificador ideal, com a disciplina RR e a menos que 1/5 quando comparados utilizando as disciplina PrioSched. Utilizando o RAFLE conseguiu-se ganho de $22,32\%$ para a disciplina de serviço RR e de $31,43\%$ para a PrioSched. Isto equivale a 2/3 do valor alcançado pelo classificador ideal, comparando os resultados com a disciplina PrioSched. Analisando os resultados dos classificadores com a disciplina RR, notamos que o RAFLE obtém um ganho $0,45\%$ acima do classificador ideal.

Quanto aos desvios padrões os ganhos experimentados foram de $200,07\%$ e $209,33\%$, para o RAFLE RR e PrioSched, respectivamente, em relação a não utilização de priorização de tráfego. O classificador ideal PrioSched obtém ganho $33,20\%$ acima do RAFLE PrioSched. Enquanto o valor alcançado pelo RAFLE RR é $10,89\%$ maior que o ideal RR.

Os ganhos obtidos sobre o valor delimitado pelo classificador ideal decorrem de questões

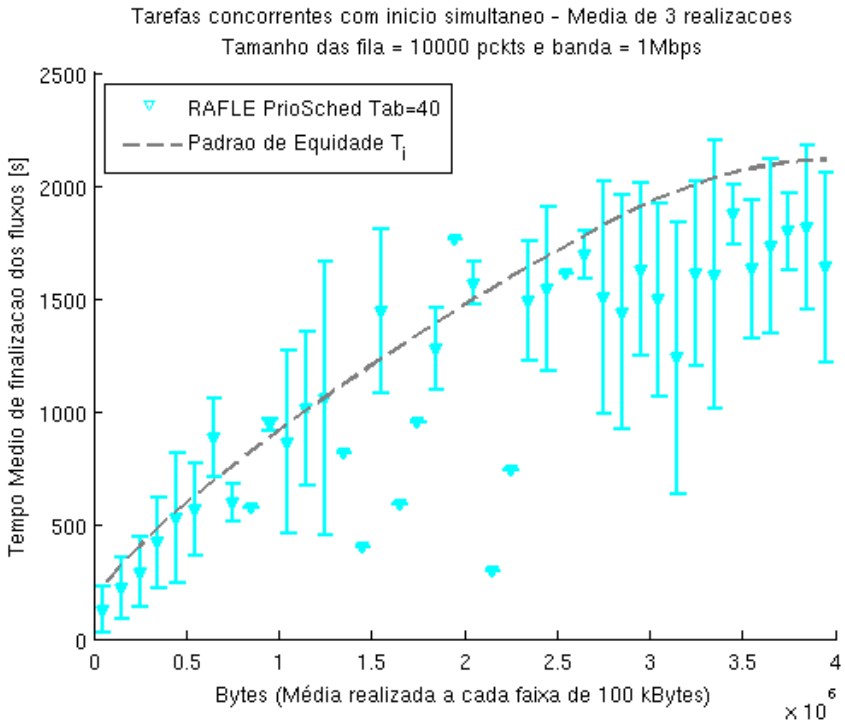


Figura 4.9: Tempos médios de finalização de fluxos e desvio padrão para o classificador RAFLE com escalonador de prioridade estrita

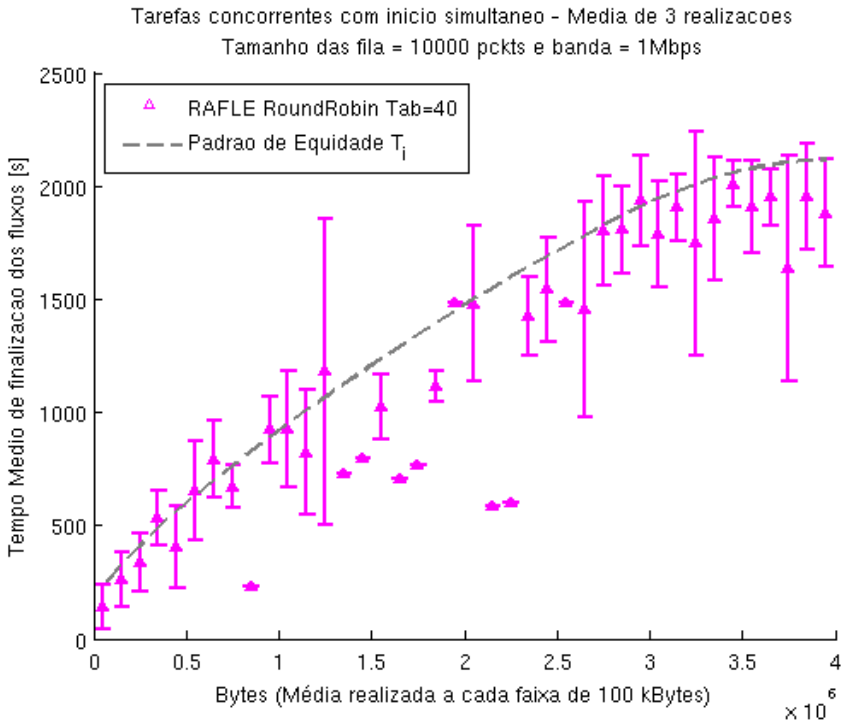


Figura 4.10: Tempos médios de finalização de fluxos e desvio padrão para o classificador RAFLE com escalonador *round robin*

Tabela 4.5: Tempos médios de finalização e desvio padrão para o RAFLE com $N = 40$

Faixa	Padrão (T_i)		*Drop Tail*		RAFLE PrioSched		RAFLE RR	
[KB]	$\bar{t}$ [s]	σ [s]	$\bar{t}$ [s]	σ [s]	$\bar{t}$ [s]	σ [s]	$\bar{t}$[s]	σ [s]
0 à 100	225,600	0	168,267	304,951	128,028	101,626	137,564	98,583
100 à 200	326,400	0	242,231	117,352	226,004	135,493	258,138	120,287
200 à 300	410,400	0	320,642	126,085	293,833	154,693	335,265	127,435
300 à 400	488,000	0	453,563	169,347	423,693	201,833	530,535	121,696
400 à 500	562,400	0	450,703	182,594	532,777	288,520	404,530	182,580

estatísticas. Ressalta-se que as médias encontradas estão dentro da faixa do desvio padrão e consequentemente do intervalo de confiança. Considerado isto, pode-se concluir que o RAFLE utilizando o escalonador RR alcança o objetivo de se aproximar dos resultados proporcionados pelo classificador ideal.

Entende-se, então, que com relação aos experimentos de tarefas concorrentes com início simultâneo, o RAFLE ultrapassa o objetivo de conseguir ganhos semelhantes ao RuN2C, sem a necessidade de qualquer adaptação nas estações das pontas. É necessária apenas a implementação dos classificadores RAFLE nos roteadores, o que facilita a implantação em ambientes operacionais.

4.5 Análise comparativa dos classificadores

Até agora apenas os resultados para os experimentos de tarefas concorrentes com início simultâneo para cada classificador foram apresentados. Para uma melhor visualização dos ganhos obtidos quando o tráfego é tratado por uma técnica de classificação de tráfego, visualizam-se, nesta seção, os resultados obtidos pelos diversos classificadores de forma conjunta. Evidenciam-se, aqui, os experimentos para tarefas concorrentes com início não simultâneo com a proposta de análise em regime permanente dos roteadores implementados. Por fim, apresentamos os resultados dos mecanismos de diferenciação de tráfego normalizados em relação ao *Drop Tail*.

4.5.1 Tarefas Concorrentes com Início não Simultâneo

É comum em uma rede operacional que novas conexões sejam iniciadas em instantes distintos. Solicitações são enviadas entre diferentes máquinas a todo tempo. Para analisar os benefícios que o tratamento diferenciado de fluxos de curta duração pode oferecer são realizados os experimentos de transmissão de arquivos de forma não simultânea, gerando um outro cenário

para a concorrência pelos recursos disponíveis. Este tipo de teste está associado à situação prática, onde um servidor possui capacidade de serviço limitado a um número de requisições (i.e. conexões) efetuadas por diversos tipos de clientes, ora requisitando transmissão de grande volume de dados, fluxos longos, ora requisitando poucos recursos, fluxos curtos. Cada requisição será realizada sorteando-se com igual probabilidade um arquivo dentro da base amostrada apresentada na Figura 3.4. Portanto, a análise dos arquivos transmitidos em um intervalo temporal se aproximou da distribuição da base original, utilizada nos demais experimentos.

Na Seção 3.4 definiu-se que a tarefa padrão engloba a conexão de controle e a de dados necessárias para a transmissão de um arquivo da base. Dentro do cenário definido, quanto mais rápido o servidor conseguir tratar as solicitações de serviço necessárias para o cumprimento de uma tarefa, mais clientes poderão ser atendidos. Desta forma, as métricas associadas aos experimentos anteriores, de análise temporal na finalização de fluxos, tempo médio e desvio padrão de duração das conexões, não fazem sentido nos experimentos de tarefas concorrentes com início não simultâneo, já que a cada requisição a rede se encontra em estado diferente de utilização. Utilizaremos a métrica de número de requisições atendidas em um determinado tempo, porém, para isto, deve-se garantir que o sistema se encontre em regime permanente de funcionamento.

Na Seção 3.4.2 vimos que um processo aleatório é estacionário, encontra-se em regime permanente, quando a natureza da aleatoriedade do mesmo não muda com o tempo (LEON-GARCIA, 1994). Assim, se o sistema estiver em regime permanente, devemos observar que a distribuição do número de arquivos com relação ao tamanho destes em processo de transmissão ou com transmissão concluída (i.e. conexões que foram ou estão sendo servidas neste período de tempo), em um intervalo (t_i, t_{i+1}), deve ser igual a do instante $(t_{i+\tau}, t_{i+1+\tau})$.

Neste experimento, o servidor pode servir até 300 requisições simultâneas. A cada tarefa cumprida, a máquina cliente inicia uma nova conexão FTP de controle e, logo após, a conexão para transferência dos dados. O processo de abertura das conexões pode demorar algum tempo em uma rede congestionada, no entanto, isto deve ocorrer de forma menos demorada quando se atribui prioridade a conexões de curta duração. No classificador ideal toda a tarefa, tanto a conexão de controle, como a de dados, de um fluxo longo é tratada como tal. No RuN2C e no RAFLE isto não ocorre, as conexões de controle por transferirem pequeno volume de dados são tratadas isoladamente como fluxos curtos, podendo agilizar o inicio da transferência de uma tarefa que caracterizaria um fluxo longo.

Na Figura 4.11 são apresentados os diagramas de fluxos ativos ao longo do tempo para os diversos classificadores implementados no intervalo de $[1000, 4000]$ segundos dos experimentos,

desprezando-se, assim, a fase transitória do início do experimento. Nota-se, antecipadamente, que o classificador RAFLE com *Round Robin* consegue manter um número mais constante de tarefas ativas, demonstrando algum benefício no tratamento da conexão de controle que faz parte das tarefas. Para possibilitar uma análise visual de estacionariedade e de semelhança com a base original de arquivos, a cada 1000 segundos foi contabilizado o histograma dos arquivos servidos utilizando cada classificador. Optou-se por traçar apenas os contornos dos histogramas obtidos, para que se pudesse comparar o número tarefas cumpridas a cada faixa destes.

Ao se comparar os histogramas do número de arquivos servidos (tarefas cumpridas) a cada intervalo considerado, nota-se muita semelhança entre eles. Uma pequena diferença ocorre entre os arquivos da primeira faixa dos histogramas (fluxos curtos) e um pequeno acumulo ao longo do tempo de tarefas nas últimas faixas dos histogramas. Este acumulo acontece porque quando uma conexão curtas é finalizada e sequencialmente uma de longa duração assume seu lugar, esta permanece no sistema por mais tempo. Ainda, de acordo com o classificador utilizado, a priorização de fluxos curtos irá favorecer de forma distinta o cumprimento das tarefas.

Assim, para propósitos práticos assumimos o sistema como estacionário. Os histogramas, de forma geral, possuem o mesmo comportamento. Passa-se, então, à análise de número de arquivos servidos durante este intervalo.

Apresenta-se na Figura 4.12 o histograma das médias de arquivos servidos (tarefas cumpridas) em todo o intervalo analisado, entre $[1000, 4000]$ segundos. Observa-se que, para as faixas seguintes a primeira, o número de arquivos servidos é praticamente o mesmo para todos os classificadores implementados. Já na primeira faixa, onde se localizam os fluxos curtos, podemos o efeito quantitativo de ganho no número de arquivos servidos para cada classificar testado.

Primeiramente, é importante salientar o melhor desempenho na finalização de fluxos curtos dos classificadores que utilizam a disciplina *Round Robin* para serviços das filas. Este resultado é diferente do obtido nos experimentos com tarefas concorrentes iniciadas simultaneamente. O principal fator que influi para este resultado é que se um fluxo curto possuir um único ou poucos pacotes que ultrapasse o limiar determinado, sendo classificado para a fila de fluxos longos, em um sistema com carga alta previamente instalada, este pacote levará mais tempo para ser servido se uma política de prioridade estrita for utilizada.

Quando comparamos os roteadores com a disciplina RR, o menor desempenho é do RuN2C. Isso é decorrente da classificação de todos os fluxos que iniciam no sistema como curtos misturando o início da conexão de dados de fluxos curtos e longos. Ainda, nota-se o desempenho do RAFLE ligeiramente superior ao do classificador ideal. Este comportamento decorre de apesar dos erros de classificação que o RAFLE pode cometer ao longo de toda a transmissão de

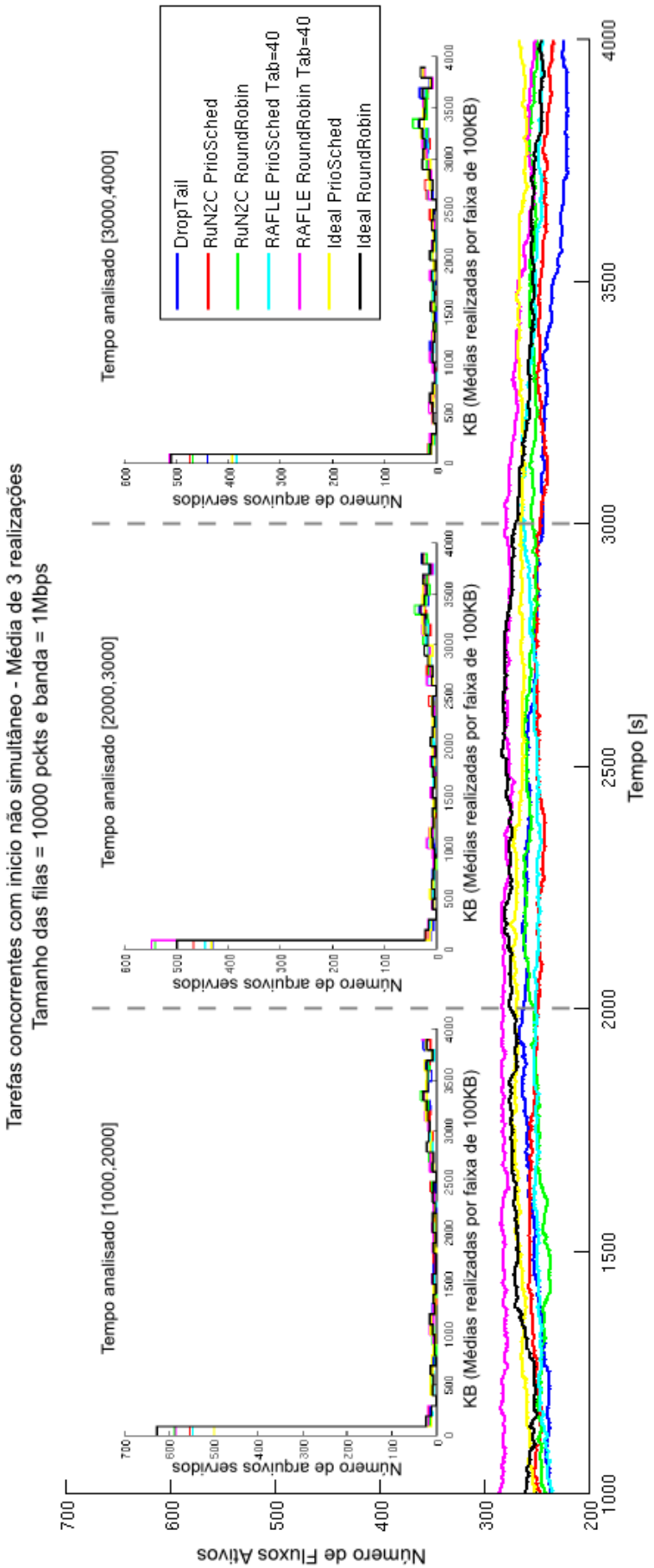


Figura 4.11: Diagrama comparativo de fluxos ativos com início não simultâneo ao longo do tempo

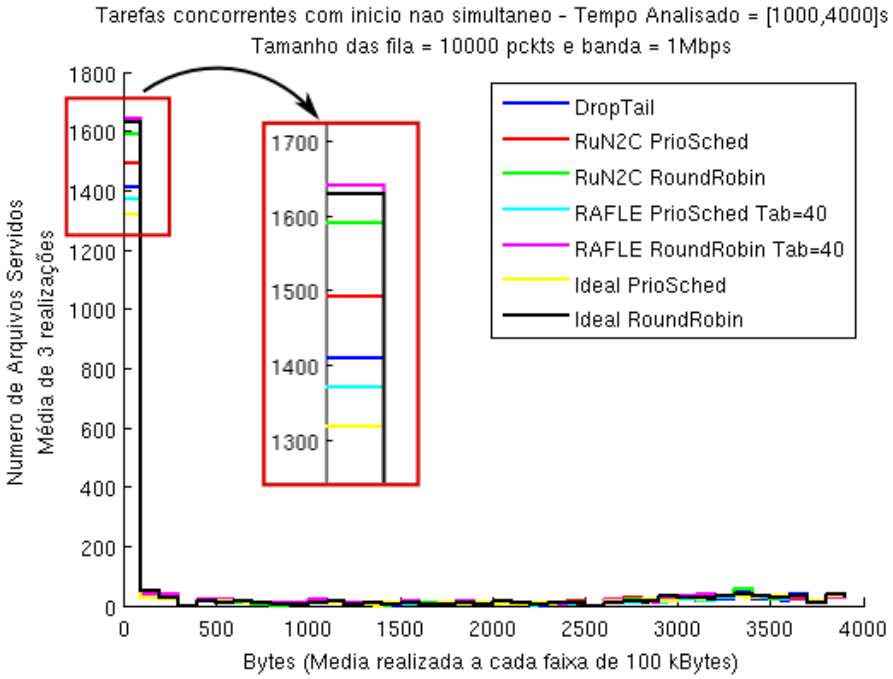


Figura 4.12: Histograma dos arquivos servidos no experimento de tarefas não simultâneas

um fluxo, ele consegue aumentar a equidade de taxas de transmissão alcançadas, diminuindo o tempo para cumprimento de uma tarefa entendida como um fluxo curto, já que penaliza com o atraso de pacotes os fluxos que já obtiveram maiores janelas transmissão, o que não ocorre com o classificador ideal.

Com a disciplina de prioridade estrita obtém-se, neste experimento, desempenho melhor que o *Drop Tail* somente quando o classificador RuN2C é empregado. O classificador denominado ideal alcança os piores resultados, já que a prioridade estrita para fluxos curtos dificulta o término das conexões longas que ocupam o sistema por mais tempo. Somado a este motivo, este classificador prejudica o início das tarefas consideradas como fluxos longos, já que suas conexões de controle também não recebem priorização. Isto aumenta o tempo necessário para cumprimento da tarefa. Assim, neste regime de funcionamento, a priorização estrita dos fluxos curtos diminui o desempenho total do sistema.

4.5.2 Ganhos Relativos ao Roteador Convencional (*Drop Tail*)

O desempenho dos classificadores de tráfego é evidenciada através dos ganhos contabilizados em relação ao roteador sem priorização de tráfego, o *Drop Tail*. Nesta seção será compilado os resultados de todos os classificadores utilizados através da normalização das curvas obtidas em relação aos valores alcançados pelo *Drop Tail*.

Tarefas Concorrentes com Início Simultâneo

Apresentamos a compilação dos resultados para os experimentos de tarefas concorrentes com início simultâneo. Na Figura 4.13 é apresentado o diagrama de finalização de fluxos ativos ao longo do tempo, normalizado em relação a curva obtida pelo *Drop Tail*. Nela observa-se nos segundos iniciais o RAFLE PrioSched serve quase três vezes o número de arquivos se comparado ao roteador convencional sem priorização de fluxos, DT. Este resultado é seguido pelo RAFLE RR e se aproximam dos classificadores ideais.

Em $t = 400s$ os ganhos se juntam em duas faixas. O RAFLE PrioSched segue a tendência do classificador Ideal PrioSched, finalizando as conexões mais rapidamente, o que era desejado. As demais curvas seguem a tendência o classificador Ideal RR. Essa aproximação das curvas aponta que a banda está praticamente dividida entre os fluxos longos. A maior finalização de tarefas inicialmente faz com que as curvas decaiam mais rapidamente quando se aproxima do final do experimento.

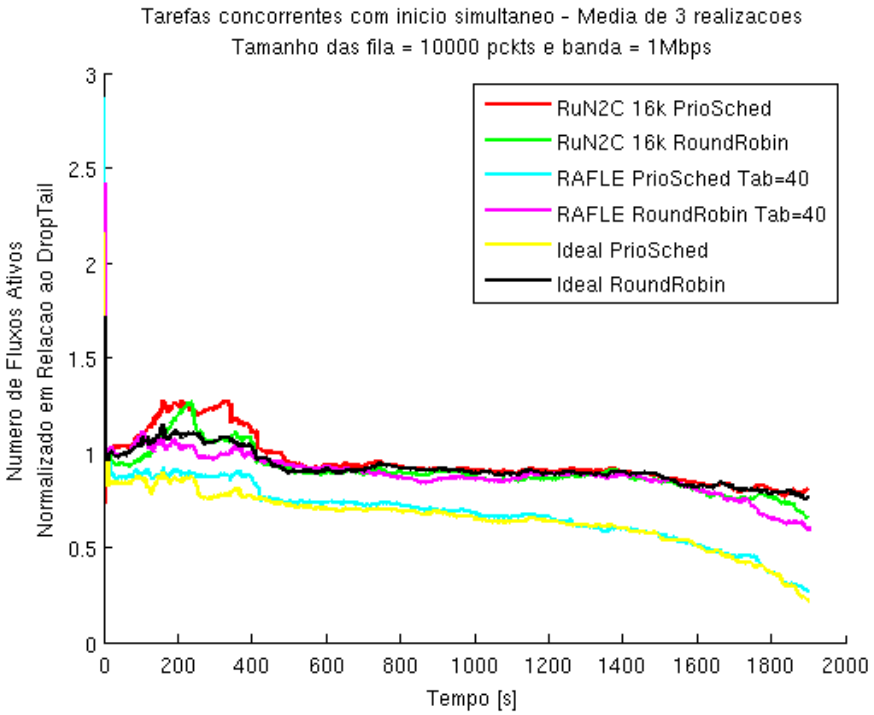


Figura 4.13: Diagrama de finalização de fluxos em relação ao tempo para tarefas simultâneas normalizado em função da curva do *Drop Tail*

Para a evidenciação dos ganhos em relação aos tempos médios de finalização dos arquivos localizados na primeira faixa do histograma (i.e. fluxos curtos) e o respectivo desvio padrão, mostra-se na Tabela 4.6 a contabilização dos ganhos percentuais para cada classificador implementado. Observa-se por esta tabela que, ao contrário do esperado, com o classificador RuN2C os maiores ganhos são conseguidos com o uso da disciplina de serviço *Round Robin* ao invés da

disciplina de priorização estrita, sugerido no artigo onde a técnica é proposta. No entanto, tanto com o RuN2C, quanto com o RAFLE os resultados obtidos são positivos e consideráveis. Isto indica, que a metodologia adotada para escolha dos parâmetros de controle de funcionamento dos classificadores RAFLE e RuN2C devem estar próximos do ponto ótimo para a distribuição de arquivos empregada.

Na Tabela 4.6, nota-se ainda, o melhor desempenho do classificador RAFLE em relação ao RuN2C. Mais ainda, sem exigir nenhuma alteração do funcionamento do protocolo de transporte os classificadores RAFLE é capaz de se aproximam dos ganhos obtidos pelo classificador denominado ideal, nas duas disciplina de serviço utilizadas.

Tabela 4.6: Ganhos percentuais para os arquivos da primeira faixa nos experimentos de tarefas simultâneas em relação ao *Drop Tail*

Escalonador	Disciplina de Serviço das Filas			
	Priority Scheduling		*Round Robin*	
	$\bar{t}$ [%]	σ [%]	$\bar{t}$ [%]	σ [%]
Ideal	44,69	198,44	21,87	233,27
RuN2C 16k	7,13	184,34	14,98	182,70
RAFLE N=40	31,43	209,33	22,32	200,07

Ainda, é importante salientar que os maiores ganhos são em relação ao desvio padrão, indicando o maior grau de confiança que a transmissão dos fluxos curtos ocorre sempre em menor tempo. Isto, porém, não traz prejuízos dignos de nota à finalização dos fluxos longos, uma vez que o tempo de finalização de todas as tarefas se aproxima do valor apontado pelo roteador *Drop Tail*, como mostrado nas seções anteriores.

Tarefas Concorrentes com Início não Simultâneo

Na Tabela 4.7 são mostrados os ganhos percentuais em relação do número de arquivos servidos, tarefas cumpridas, no experimento de tarefas concorrentes com início não simultâneo. O servidor suporta servir até o limite 300 conexões, e a cada requisição atendida, uma nova é solicitada. Neste experimento o sistema entra em processo estacionário.

Observa-se o atendimento de mais requisições quando a disciplina *Round Robin* é utilizada, contrariando a intuição de que um escalonamento de prioridade estrita seria o melhor nesta situação. Ainda no que diz respeito à disciplina RR, nota-se a grande aproximação do RAFLE ao classificador ideal, como era desejado. Quando a prioridade estrita é utilizada, apenas o classificador RuN2C obteve ganhos positivos. Isto se deve a priorização estrita dificulta o termino das conexões longas, ocupando recursos do sistema por mais tempo sob a restrição do número

máximo de sessões simultâneas.

Tabela 4.7: Ganhos percentuais para o número de arquivos servidos da primeira faixa relação ao *Drop Tail* em regime estacionário

Escalonador	Ganhos percentuais em relação ao *Drop Tail* para as disciplina de serviço das filas	
	Priority Scheduling	*Round Robin*
Ideal	$-6,83\%$	$13,57\%$
RuN2C 16k	$5,57\%$	$11,45\%$
RAFLE N=40	$-2,27\%$	$14,15\%$

É importante salientar que a hipótese do classificador ideal ter um *overhead* de processamento para classificação é descartada em função de estarmos usando o roteador para tratar 1Mbps, bem abaixo de sua capacidade máxima, 73Mbps.

Ao contrario da sugestão do classificador denominado ideal estabelecer os limites máximos a poderem ser alcançados pelos demais classificadores desenvolvidos, notou-se a superação de seus resultados pelo classificador RAFLE. O primeiro ponto a ser levantado é que os resultados foram obtidos por estatística, o que introduz um certo grau de incerteza. Evidencia-se que as tarefas a serem cumpridas eram compostas pelo conjunto conexão FTP de controle e de dados, identificadas e tratadas pelo RAFLE de forma distinta, porém tratadas pelo classificador ideal de forma igual. Por último, quando se trata de equidade em redes de pacotes, os resultados obtidos podem não ser os intuitivos, nos quais os fluxos são imaginados como tarefas distintas contínuas e não como uma sequência de pacotes embaralhados no tempo. Ainda, é difícil prever intuitivamente os efeitos do descarte de pacotes no limitador de banda inserido nas interfaces de entrada.

5 *Conclusão e Trabalhos Futuros*

Neste trabalho, esquemas de diferenciação de fluxos sem manutenção de estados foram propostos e implementados sobre PCs, utilizando a arquitetura CLICK, submetidos a tráfego real. Desta forma, os resultados não estão atrelados à qualidade de modelos de simuladores, frequentemente muito simples para traduzir o comportamento real das redes de computadores. A metodologia foi cuidadosamente pensada para que outros fatores fossem eliminados durante os experimentos.

Mostrou-se que a divisão dos recursos pode ser realizada de forma mais equânime através da segregação do tráfego em duas classes distintas, fluxos longos e curtos, e atuando no atraso relativo ao enfileiramento dos datagramas. Com estas medidas atrasos no tempo de finalização das sessões tornam-se mais proporcionais a duração e ao volume de dados trafegado individualmente por cada uma das sessões.

Dois mecanismos, o RuN2C e o RAFLE, foram investigados neste trabalho, utilizando-se variações no serviço das classes de tráfego, fluxos longos e curtos, classificadas por estes mecanismos. Dois tipos de disciplinas de serviço de filas, escalonamento de prioridade estrita (PrioSched) e *Round Robin* (RR) foram testados. Os resultados foram ainda comparados aos obtidos por um classificador de fluxos ideal, por um roteador sem a capacidade de diferenciação de tráfego (*Drop Tail*), e ainda, com uma metologia experimental, também proposta no presente trabalho, que utiliza o conceito de *Processor Sharing* para análise da equidade entre eles no acesso à banda.

É importante salientar, que todos os resultados foram obtidos através de roteadores implementados sobre PCs, utilizando a arquitetura CLICK, submetidos a tráfego real. Desta forma, estes não estão atrelados a qualidade de modelos de simuladores, frequentemente muito simples para traduzir o comportamento real das redes de computadores. A metodologia foi cuidadosamente pensada para que outros fatores fossem eliminados durante os experimentos.

Quanto aos cenários de teste, foram realizados dois tipos de experimento, 1) o de tarefas concorrentes com início simultâneo, a fim de se analisar um pior caso de competição de recursos

de rede, além de aferir-se os ganhos em relação aos tempos médios de finalização de fluxos, 2) e o de tarefas concorrentes com início não simultâneo, onde novas sessões eram iniciadas com o sistema sob carga, exprimindo o comportamento dos mecanismos em regime permanente, avaliado com relação ao número de fluxos atendidos por um servidor ao longo do tempo.

No primeiro tipo de experimento, o mecanismo RuN2C, que não possui a necessidade de manutenção de estados dos fluxos ativos, faz com que o tempo médio de finalização dos fluxos curtos diminua, tornando a competição destes mais justa em relação a fluxos longos, que já alcançaram grandes janelas de transmissão. Os ganhos giram em torno de de 7,13% quando se utiliza a disciplina PrioSched para serviço das filas, e 14,98% quando a disciplina RR comparados ao roteador sem priorização de tráfego. Os ganhos são melhor quantificados quando comparados aos alcançados com os classificadores ideais de tráfego, onde determinou-se os limites dos ganhos possíveis em 44,69% e 21,87%, apresentados na mesma ordem, aproximando-se, assim, em 15,95% e 68,50% dos limites ideais. No entanto, a relevância dos resultados se dá, principalmente com relação ao desvio padrão, métrica que evidencia de melhor forma o tratamento igualitário entre os fluxos da mesma classe. Os ganhos alcançados para esta métrica foram de 184,34% e 182,70%, para as disciplinas PrioSched e RR respectivamente, aproximando-se 92,89% e 78,32% dos limites ideais. Os resultados aqui apresentados, apontam a disciplina RR como mais favorável ao serviço das filas classificadas, ao contrário da proposta original do RuN2C (AVRACHENKOV; BROWN; NYBERG, 2004), onde não é avaliada outra alternativa a disciplina PrioSched.

Utilizando-se o esquema RAFLE, proposto neste trabalho, o ganho nos tempos médios para este tipo de experimento é ainda maior, 31,43% e 22,32%, para as disciplinas PrioSched e RR respectivamente. Quando comparado ao limite obtido pelo classificador ideal, nota-se que o RAFLE alcança 70,33% do limite com a disciplina PrioSched e 102,06% com a RR, evidenciando no último caso o alcance máximo do ganho máximo esperado. Vale aqui ressaltar que os tempos médios obtidos então dentro das faixas delimitadas pelo desvio padrão e, assim, os resultados acima dos valores delimitados pelo escalonador ideal são derivados de questões estatísticas. A melhora dos resultados em relação ao RuN2C é mantida na análise dos ganhos relativos ao desvio padrão, sendo de 209,33% e 200,07%, em mesma ordem, totalizando 105,44% e 85,76%, o que indica o alcance do limite da métrica, agora, pela disciplina PrioSched. É importante resaltar, que há ainda mais uma vantagem do RAFLE em relação ao RuN2C, a de que os melhores resultados foram alcançados também sem a manutenção de estados dos fluxos ativos, porém sem a necessidade de qualquer no funcionamento do protocolo nas pontas. O RuN2C exige a alteração da geração dos números de sequência iniciais, o que implica na exigência de modificações nas implementações do TCP nos diversos sistemas ope-

racionais. Apesar de ser corriqueiro o uso de atualizações automáticas nos SOs, as alterações em sistemas proprietários dependem das empresas que os mantém, podendo este ser um fator limitante durante a implantação.

Nos experimentos de tarefas concorrentes com início não simultâneo, onde novos fluxos eram iniciados quando o roteador já estava sob carga, é importante observar que o regime permanente é alcançado, caracterizando a métrica de número de arquivos servidos como adequada. Os limites de ganhos obtidos com o classificador ideal de tráfego apontam que, ao contrário da intuição, a disciplina de serviço das classes mais adequada é a RR e não a de prioridades estritas, tornando possíveis ganhos em torno de 13,57%. Quando a disciplina PrioSched foi utilizada, nota-se que -6,83% de arquivos que caracterizavam fluxos curtos foram servidos em relação ao roteador sem nenhuma capacidade de diferenciação de fluxos. Mesmo assim, os mecanismos implementados alcançaram ganhos sobre estes valores por fundamentalmente não se basearem em mecanismos estritos. Utilizando-se a PrioSched, o RAFLE obteve ganho de -2,27% e o RuN2C de 5,57%. Todavia, nota-se neste experimento, quando a RR é aplicado, ganhos de 11,45% e 14,15%, para o RuN2C e RAFLE, respectivamente, o que corresponde a 84,38% e 104,27% do ganho esperado.

Observa-se que em todos os casos não são notados prejuízos aos fluxos longos dignos de nota, uma vez que as curvas de tempo de finalização de fluxos e de tempos médios seguem próximas ou abaixo das definidas pelo padrão analítico de equidade, evidenciando o provimento de justiça na alocação de recursos em regime de banda escassa. Decorre-se do exposto até o momento, a escolha próxima da ideal para os parâmetros de ajuste de funcionamento dos mecanismos utilizados, 16KB de transmissão para o RuN2C e memória dos 40 últimos *flowIDs* no RAFLE, nestes experimentos.

Evidencia-se agora de forma global que o escalonador RAFLE proposto e desenvolvido neste trabalho obteve ganhos significativos em relação ao RuN2C com a grande vantagem de que ele só necessita de implementação nos roteadores da rede, não necessitando de alteração nos protocolos e funcionalidades dos sistemas operacionais hoje existentes. Nos dois casos, a implantação pode ser gradativa, trazendo benefícios a cada salto da rede onde estiverem implementados.

Aponta-se ainda que o RAFLE não depende, de forma estrita, de nenhum campo específico do cabeçalho TCP (O RuN2C utiliza do Número de Sequência dos Pacotes) tornando-se possível, em trabalhos futuros, sua implementação e aplicação em outras camadas diferentes da de transporte. Desta forma, o RAFLE se adequar as novas definições de fluxos apontadas pelo OpenFlow e poderia atuar de forma a aliviar a demanda para o controlador OpenFlow

aumentando a escalabilidade desta tecnologia. Isto viabilizaria maior desempenho de serviços provindos do amadurecimento desta tecnologia.

É importante mencionar que o classificador ideal utilizado neste trabalho não foi tão eficiente ao tratar as tarefas definidas, transmissões de arquivos via FTP. Seu funcionamento deixa a cargo do transmissor a marcação dos fluxos longos e curtos, pela definição do IP de destino. Desta forma, não foi possível o tratamento distinto entre a conexão de controle e a conexão de dados presentes em cada tarefa. O desenvolvimento de um método de classificação mais eficiente colaboraria para a melhor análise dos resultados.

Ainda, dando seguimento a este trabalho, sugere-se a realização de novos experimentos para o RAFLE e RuN2C utilizando outras métricas e de estudos para estes classificadores no sentido de economia de *buffers*, como indica ser possível com a utilização da técnica SIFT, que também utiliza a classificação de fluxos longos e curtos em filas distintas (PSOUNIS et al., 2005).

Neste sentido, sugere-se como alvo de estudos futuros, uma definição formal de um modelo analítico para RAFLE que viabilizará o estudo de suas adequações do parâmetro de tamanho da memória recente para diferentes cenários de tráfego.

A utilização de outros escalonadores, que apesar de mais custosos, poderiam aumentar o desempenho dos esquemas de diferenciação de fluxos de forma compensatória, gerando benefícios maiores que custos. A evolução da eletrônica corrobora em amenizar a limitação que o processamento eletrônico.

Também, a utilização de cenários com diferentes protocolos de rede são interessantes, pois ampliariam as possibilidades de utilização dos mecanismos de diferenciação tráfego discutidos ou propostos neste trabalho.

Referências

ALIZADEH, M. et al. DCTCP : Efficient Packet Transport for the Commoditized Data Center. *SIGCOMM*, 2010. Disponível em: <http://research.microsoft.com/pubs/121386/dctcp-public.pdf>.

ALTMAN, E.; AVRACHENKOV, K.; BARAKAT, C. A stochastic model of TCP/IP with stationary random losses. *IEEE/ACM Transactions on Networking*, IEEE Press, v. 13, n. 2, p. 356–369, 2005. ISSN 10636692.

ANELLI, P.; LOCHIN, E.; DIANA, R. Favourqueue: a stateless active queue management to speed up short tcp flows (and others too!). *Cornell University Library*, 2011. Disponível em: <http://arxiv.org/abs/1103.2303v2>.

AVRACHENKOV, K.; BROWN, P.; NYBERG, E. Differentiation between short and long TCP flows: Predictability of the response time. In: *In Proc IEEE INFOCOM*. [S.l.: s.n.], 2004.

BENSON, T.; AKELLA, A.; MALTZ, D. A. Network Traffic Characteristics of Data Centers in the Wild. *Traffic*, ACM, p. 267–280, 2010. Disponível em: <http://pages.cs.wisc.edu/ akella/papers/dc-meas-imc10.pdf>.

BLACK, S. et al. An architecture for differentiated services. *RFC2475*, agosto 1998.

BRADEL, R. Requirements for internet hosts – communication layers. *RFC2988*, October 1989.

BRADEN, R.; CLARK, D.; SHENKER, S. Integrated services in the internet architecture: an overview. *RFC1633*, junho 1994.

BRADEN, R. et al. Resource reservation protocol (rsvp). *RFC2205*, Setembro 1997.

CISCO Systems Inc. Quality of Service for Voice over IP Solution Guide. Disponível em: http://www.cisco.com/, acessado em: 27/07/2010. [S.l.], julho 2002.

CISCO Systems Inc. Quality of Service Solutions Configuration Guide, Congestion Management Overview. Disponível em: http://www.cisco.com/, acessado em: 27/07/2010. [S.l.], abril 2001.

CIULLO, D.; MELLIA, M.; MEO, M. Two schemes to reduce latency in short lived TCP flows. *IEEE Communications Letters*, v. 13, n. 10, p. 806–808, 2009. ISSN 10897798.

CLAFFY, K. C.; BRAUN, H.-W.; POLYZOS, G. C. A parameterizable methodology for internet traffic flow profiling. *IEEE Journal on Selected Areas in Communications*, v. 13, n. 8, p. 1481–1494, 1995. ISSN 07338716.

CLARK, D.; WROCLAWSKI, J. An approach to service allocation in the internet. *IETF, Disponível em: http://tools.ietf.org/id/draft-clarkk-diff-svc-alloc-00.txt, visitado em: 26/07/2011*, julho 1997.

CURTIS, A. R. et al. DevoFlow : Scaling Flow Management for High-Performance Networks. *ACM SIGCOMM*, p. 254–265, 2011. Disponível em: <http://www.hpl.hp.com/personal/Praveen_Yalagandula/papers/SIGCOMM2011-DevoFlow.pdf>.

DUKKIPATI, N. *Rate Control Protocol (RCP): Congestion control to make flows complete quickly*. Tese (PhD) — Stanford University, outubro 2007.

DUKKIPATI, N.; MCKEOWN, N. Why flow-completion time is the right metric for congestion control. *ACM SIGCOMM Computer Communication Review*, v. 36, n. 1, p. 59, 2006. ISSN 01464833. Disponível em: <http://portal.acm.org/citation.cfm?doid=1111322.1111336>.

FALL, K.; FLOYD, S. Simulation-based comparisons of Tahoe, Reno and SACK TCP. *ACM SIGCOMM Computer Communication Review*, ACM Press, v. 26, n. 3, p. 5–21, 1996. ISSN 01464833. Disponível em: <http://portal.acm.org/citation.cfm?doid=235160.235162>.

FREDJ, S. B. et al. Statistical Bandwidth Sharing : A Study of Congestion at Flow Level. *Simulation*, v. 31, n. 4, p. 111–122, 2001. ISSN 01464833. Disponível em: <http://portal.acm.org/citation.cfm?doid=964723.383068>.

GETTYS, J. Red in a different light. *jg's Ramblings, Disponível em: http://gettys.wordpress.com/2010/12/17/red-in-a-different-light/, visitado em: 25/07/2011*, dezembro 2010.

GREENBERG, A. et al. VL2 : A Scalable and Flexible Data Center Network. *In Proceedings of the ACM SIGCOMM 2009*, ACM, v. 39, n. 4, p. 51–62, 2009. ISSN 01464833. Disponível em: <http://portal.acm.org/citation.cfm?id=1594977.1592576>.

HANDLEY, M.; HODSON, O.; KOHLER, E. XORP: An open platform for network research. *portalacmorg*, ACM, v. 33, n. 1, p. 57, 2003. Disponível em: <http://portal.acm.org/citation.cfm?doid=774763.774771>.

HARDENING the TCP/IP stack to SYN attacks. Disponível em: http://www.symantec.com/connect/articles/hardening-tcpip-stack-syn-attacks, visitado em: 18/08/2011. [S.l.], novembro 2010.

HP ProCurve 5400 zl switch series. Disponível em: http://h17007.www1.hp.com/us/en/products/switches/HP_E5400_zl_Switch_Series/index.aspx, visitado em: 27/06/2011. [S.l.], janeiro 2011.

IP - Internet Protocol. *RFC791*, Setembro 1981.

JAIN, R.; ROUTHIER, S. A. Packet Trains-Measurements and a New Model for Computer Network Traffic. *IEEE Journal of Selected Areas in Communications*, SAC-4, n. 6, p. 986–995, 1986.

JOHNS, M. S. Identification protocol. *RFC1413*, fevereiro 1993.

JR., F. L. T.; RIBEIRO, M. R. N.; WALDMAN, H. On the benefits of traffic segregation in wdm networks. *XXIII SImpósio Brasileiro de Rede de Computadores*, v. 2, p. 869–882, 2005.

KOHLE, E. *The click modular router*. Tese (PhD) — MIT, agosto 2000.

KOZIEROK, C. M. *The TCP/IP Guide, Version 3.0*. [S.l.], Setembro 2005. Disponível em: <http://www.tcpipguide.com>.

LABOVITZ, C. et al. Atlas internet observatory 2009 annual report. *47th NANOG*, 2009.

LEON-GARCIA, A. *Probability and Random Processes for Electrical Engineering*. [S.l.]: Addison-Wesley, 1994. 372 p. ISSN 00401706. ISBN 020150037X.

LOCKWOOD, J. W. et al. NetFPGA–An Open Platform for Gigabit-Rate Network Switching and Routing. *2007 IEEE International Conference on Microelectronic Systems Education MSE07*, Ieee, p. 160–161, 2007.

MARINE, A.; REYNOLDS, J.; MALKIN, G. Answers to commonly asked "new internet user"questions. *RFC1594*, março 1994.

MATA, R. S. da. *Dimensionamento de Enlaces em Redes com Integração de Serviços*. Dissertação (Mestrado) — Universidade Estatual de Campinas, abril 2002.

MCKEOWN, N. et al. OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, ACM, v. 38, n. 2, p. 69–74, 2008. ISSN 01464833. Disponível em: <http://portal.acm.org/citation.cfm?id=1355746>.

MUSSI, S. S.; RIBEIRO, M. R. N. Análise experimental de equidade em roteador com diferenciação stateless de fluxos. *XXVII Simpósio Brasileiro de Telecomunicações - SBrT*, 2009.

NAOUS, J.; GIBB, G.; BOLOUKI, S. Netfpga: reusable router architecture for experimental research. *Proceedings of the ACM workshop on Programmable routers for extensible services of tomorrow*, 2008. Disponível em: <http://portal.acm.org/citation.cfm?id=1397720>.

NASCIMENTO, M. R. et al. Virtual routers as a service: The routeflow approach leveraging software-defined networks. *6th International Conference on Future Internet Technologies 2011 (CFI 11))*, junho 2011.

NETFPGA. 2011. Disponível em: <http://netfpga.org>.

NOUREDDINE, W.; TOBAGI, F. Improving the performance of interactive tcp applications using service differentiation. *Computer Networks*, IEEE, v. 40, n. 1, p. 31–40, 2002. ISSN 13891286.

PAXSON, V.; ALLMAN, M. Computing tcp's retransmission time. *RFC2988*, novembro 2000.

PEE-WEE OSPF Protocols Details. Disponível em http://yuba.stanford.edu/cs344/admin/pwospf/. Visitado em 09/05/2011. [S.l.], 2011.

POSTEL, J.; REYNOLDS, J. File transfer protocol (ftp). *RFC959*, Outubro 1985.

PROFTPD. 2011. Disponível em: <http://www.proftpd.org/>.

PSOUNIS, K. et al. SIFT: A simple algorithm for tracking elephant flows, and taking advantage of power laws. 2005. Disponível em: <http://www.stanford.edu/ balaji/papers/05sifta.pdf>.

QUAGGA Routing Suite. Disponível em http://www.quagga.net. Visitado em 09/05/2011. [S.l.], 2006.

SHREEDHAR, M.; VARGHESE, G. Efficient fair queueing using deficit round robin. In: *Proceedings of SIGCOMM 95*. [S.l.]: IEEE Press, 1995. v. 4, n. 3, p. 375–385. ISBN 0897917111. ISSN 01464833.

STEVENS, W. Tcp slow start, congestion avoidance, fast retransmit, and fast recovery algorithms. *RFC2001*, janeiro 1997.

SUN, C. et al. Drr-sff: A practical scheduling algorithm to improve the peformance of short flows. *Third International Conference on Networking and Service, IEEE*, 2007.

TANENBAUM, A. S. *Redes de Computadores*. [S.l.]: Nova Terra, 2003. 832 p. ISBN 9788561893057.

TCP - Transmission Control Protocol. *RFC793*, Setembro 1981.

THOMPSOM, K.; MILLER, G. J.; WILDER, R. Idle-area internet traffic patterns and characteristics. *IEEE Network*, v. 11, n. 6, 2004.

THOMPSON, K.; MILLER, G. J.; WILDER, R. Wide-Area Internet Traffic Patterns and Characteristics (Extended Version). *Engineering*, v. 11, n. 6, p. 10–23, 1997. ISSN 08908044.

TSAI, T.-Y.; CHUNG, Y.-L.; TSAI, Z. Introduction to Packet Scheduling Algorithms for Communication Networks. In: PENG, J. (Ed.). *Communications and Networking*. InTech, 2011. p. 263–288. ISBN 9789533071145. Disponível em: <http://www.intechopen.com/articles/show/title/introduction-to-packet-scheduling-algorithms-for-communication-networks>.

WANG, J.; TANG, A.; LOW, S. H. Maximum and asymptotic udp troughput under choke. *ACM Sigmetrics*, junho 2003.