UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS - ICEX
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

MONITORAÇÃO DE TRÁFEGO P2P EM TEMPO REAL

TIAGO ALVES MACAMBIRA

Belo Horizonte

1 de Julho de 2005

TIAGO ALVES MACAMBIRA

MONITORAÇÃO DE TRÁFEGO P2P EM TEMPO REAL

Dissertação apresentada ao Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Belo Horizonte

1 de Julho de 2005

UNIVERSIDADE FEDERAL DE MINAS GERAIS

FOLHA DE APROVAÇÃO

Monitoração de Tráfego P2P em Tempo Real

TIAGO ALVES MACAMBIRA

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

Prof. Dr. Dorgival Olavo Guedes Neto - Orientador
Departamento de Ciência da Computação – ICEx – UFMG

Prof. Dr. Wagner Meira Júnior
Departamento de Ciência da Computação – ICEx – UFMG

Prof. Dr. Carlos Frederico Marcelo da C. Cavalcanti
Departamento de Computação (DECOM) – UFOP

Belo Horizonte, 1 de Julho de 2005.

Resumo

O tráfego devido a aplicações P2P tem aumentado consideravelmente nos últimos anos e, apesar de ser hoje o responsável pela maior parte de todo o tráfego da Internet, não existem muitas ferramentas que auxiliem na monitoração e portanto no entendimento desse tráfego, tanto sob um ponto de vista acadêmico como sob um ponto de vista prático. Nesse trabalho abordamos o as dificuldades em se construir uma solução que viabilize a análise e a caracterização de tráfego de aplicações em tempo real, tendo como foco principal as aplicações P2P de troca de arquivo e propomos o *Palantír*, uma sistema de monitoração passiva com recuperação de estado em tempo real de tráfego em enlaces de alta-velocidade.

Esse sistema é capaz de, além da monitoração, realizar a remontagem dos arquivos trocados no tráfego monitorado, mesmo a velocidades de 500Mbps, apresentando uma taxa de perda de pacote da ordem de 3.3% e um desempenho 80% superior ao que poderia ser obtido utilizando abordagens tradicionais.

Apresentamos um estudo de caso onde analisou-se e caracterizou-se, usando o Palantír, o tráfego associado a dois sistemas P2P em um provedor de acesso a internet de banda larga por um período de aproximadamente 10 dias. Os valores encontrados na nossa caracterização são condizentes com os encontrados em estudos anteriores. Além disso pode-se observar que a localidade de referência entre as duas redes é pequena mas significativa.

Abstract

The traffic due to P2P traffic-swapping applications has grown considerably in the last years and, although it is responsible for the biggest portion of Internet traffic nowadays, there are not many tools that aid monitoring and thus understanding the said traffic, both from an academic and from a practical perspective. In this work we present *Palantír*, a framework for real-time statefull monitoring of P2P traffic in high-speed networks and discuss the requirements and challenges faced during its construction. Besides its monitoring capabilities this system is capable of performing on-the-fly reassembling files as they are transferred on the monitored traffic, even at speed close to 500Mbps, with a packet drop rate approximately 3.3% and with a performance gain of 80% compared to traditional approaches.

We also present a case study where we analysed and characterized, with the aid of *Palantír*, the traffic of two P2P networks in a local Broadband ISP during a period of 10 days. Our findings are are similar to those of previous work on the subject. Finally, we could observe that, with the use of caches, it could be possible to lower the traffic generated by such networks. Nevertheless, the reference locality found between those two networks was low.

... o importante é o processo.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Motivação

Existe uma constante preocupação no meio acadêmico e comercial por formas de economizar os recursos de rede de computadores em universidades, instituições, ISPs[1], órgãos governamentais, etc. Tradicionalmente, o foco dessa pesquisa é direcionado para as aplicações e formas de acesso que, em um dado momento, são tidas como responsáveis pela maior parte do tráfego observado na internet.

Até meados da década de 90, antes do surgimento da *Web*, grande parte das pesquisas nessa área não se concentravam nas aplicações, mas nos próprios mecanismos de transporte e roteamento da internet [Brakmo et al., 1994, Moy, 1991]. Naquela época *"pré-Web"*, a maior parte do tráfego era devido aos protocolos SMTP e FTP que, em conjunto, eram responsáveis por aproximadamente 80% de tudo que transitava no *backbone* da NSFNet, por exemplo [Cáceres, 1989, Heimlich, 1990].

O surgimento da *Web*, no entanto, modificou esse quadro profundamente. Enquanto em 1995 calculava-se que 21% de todo o tráfego na internet era devido a tráfego HTTP, em 1997 esse valor já oscilava entre 60% e 80% [Thompson et al., 1997]. Essa mudança nos padrões do tráfego na rede mundial motivou trabalhos de análise, caracterização, controle e redução desse tráfego [Chankhunthod et al., 1996, Cáceres et al., 1998, Barish and Obraczka, 2000, Arlitt and Jin, 2000, Arlitt and Williamson, 1996].

Ao final da década de 90, o aparecimento de aplicações *peer-to-peer*[2] (P2P) [Balakrishnan et al., 2003], em particular as de compartilhamento de arquivos, oca-

[1] *Internet Service Providers* - Provedores de Acesso à Internet

[2] A tradução do termo *peer-to-peer* ainda causa controvérsias entre "ponto-a-ponto", "fim-a-fim" e "par-a-par". Dessa forma, nesse trabalho, o termo será utilizado em inglês.

sionou uma nova mudança nos padrões de tráfego da internet.

Em 2000, 23% do tráfego de saída da Universidade de Wisconsin já era tráfego P2P, ao passo que o tráfego *web* representava então apenas 20% do total [Markatos, 2002]. Desde então, o tráfego P2P apenas aumentou, chegando a representar 45% de todo o tráfego daquela universidade. Relatos similares são bastante comuns [Gummadi et al., 2003], indicando que o tráfego P2P representa hoje a maior parcela de todo o tráfego trocado na internet e que ele só tende a aumentar [Karagiannis et al., 2004a, Sen et al., 2004].

Desta forma, similarmente ao que foi feito quando da explosão do tráfego *Web*, faz-se necessário analisar e caracterizar o tráfego devido a aplicações P2P. Tal esforço auxiliará na busca por formas de controlá-lo e diminui-lo, na criação de novas aplicações, no refinamento dos modelos existentes e num melhor entendimento da dinâmica das várias redes P2P de troca de arquivos atuais.

1.1.1 Caracterização e Monitoração de Tráfego P2P

Diversos trabalhos sobre caracterização de tráfego P2P podem ser encontrados na literatura, focando em diversos aspectos desses sistemas, tais como a disponibilidade dos recursos trocados nesses sistemas [Chu et al., 2002], o processo de busca e popularidade das palavras chaves [Klemm et al., 2004], seus padrões de tráfego [Sen and Wang, 2002, Tutschku, 2004], Alguns até propuseram a utilização de *caches* nesses sistemas, verificando antes como tal uso afetaria as redes o o tráfego delas [Markatos, 2002, Gummadi et al., 2003].

Para a realização desses trabalhos foi necessário monitorar, de alguma forma, o tráfego dessas redes para somente então realizar os trabalhos de caracterização. Uma parcela desses trabalhos concentrou-se no uso de técnicas de monitoração ativa para a obtenção de seus resultados. Nesse tipo de monitoração, coloca-se um ou vários nós "especiais" na rede que se deseja monitorar. A partir da interação desses nós "especiais" com nós reais, pode-se inferir e obter o comportamento dos nós da rede e dela própria. Entretanto, esse tipo de monitoração é limitado pelo próprio consumo de banda e de recursos que gera, não sendo capaz de analisar eficientemente redes com um grande número de nós. [Sen and Wang, 2002]

Como alternativa à monitoração ativa temos a monitoração passiva. A monitoração passiva consiste na coleta e análise do tráfego visto a partir de uma certa localidade (ISP, instituição, etc). Através da análise dessa parcela do tráfego pode-se inferir o comportamento dos nós internos e de alguns nós externos a essa localidade, além do

tráfego e o comportamento de todo o sistema. Essa abordagem além de não ser intrusiva é mais confiável, uma vez que toda a atividade dos nós internos a uma dada rede pode ser detectada e monitorada sem causar interferência. Além disso, pode-se observar a interação entre nós reais diretamente, algo que a análise ativa não permite [Sen and Wang, 2002, Gummadi et al., 2003].

Todavia, diversos trabalhos que utilizaram monitoração passiva para caracterizar o tráfego de redes P2P de troca de arquivo limitaram-se a fazer análises dos fluxos das redes monitoradas (*traffic flow analysis*). Tal abordagem permite o fornecimento de indicativos da dinâmica dos fluxos e do volume do tráfego dos sistemas P2P, porém não possibilita a extração informações mais detalhadas sobre a natureza dos recursos trocados.

Para ser eficaz, uma caracterização do tráfego P2P deve considerar não apenas informações em nível de rede, como quantidade de *bytes* trafegados, mas também informações semânticas da aplicação, como requisições efetuadas, os recursos transferidos e popularidade destes. Desta forma, torna-se mais fácil a criação de modelos mais realistas para o tráfego dessas redes pois pode-se compreender como funcionam as forças que atuam por trás da geração desse tráfego, ao invés de apenas gerar modelos baseados em estimativas e comportamentos observados e estimativas. Para tanto, faz-se necessário recuperar do tráfego monitorado os dados tais como eles são entregues às aplicações para poder recuperar desses dados a semântica delas, ou seja: é necessário realizar a recuperação do estado dessas conexões.

Esse requisito constitui um problema ao se considerar as técnicas tradicionais de monitoração passiva de tráfego com recuperação de estado, pois como essa se dá através da coleta do tráfego alvo para análise posterior, a coleta de longos períodos para análise torna-se proibitiva devido ao grande volume de dados. Além disso, análises *a posteriori* inviabilizam a tomada de decisões em tempo real para adequar o sistema a variações de comportamento dos clientes, o que pode ocorrer com freqüência.

Se a recuperação de estado *a posteriori* apresenta alguns problemas, a caracterização em tempo real do tráfego P2P também apresenta seus desafios. O alto volume de dados exige uma alta taxa de processamento de informações para evitar a perda de pacotes. Um complicador nesse caso é o fato das transferências de recursos em certas redes P2P poderem se estender por períodos de horas ou até mesmo dias, o que exige que um sistema de caracterização em tempo real mantenha informações sobre o estado das transferências em andamento por um longo tempo, aumentando o volume de informações que precisa ser gerenciado continuamente.

Ao problema de manutenção de estado, acrescenta-se o fato de os protocolos P2P serem usualmente complexos, dificultando a sua análise e de usarem *swarming*[3] para acelerar as transferências de arquivos. Isso exige que um sistema de caracterização em tempo real seja capaz de acompanhar e combinar diversas transferências em paralelo relacionadas a um mesmo objeto.

Por outro lado, crêmos que a recuperação de estado em tempo real permite a criação de uma maior gama de aplicações, tais como a criação de "*caches* oportunisticos", sistemas de policiamento de tráfego, entre outros.

1.2 Objetivos

Com base nessa discussão, este trabalho objetiva o desenvolvimento de uma solução que viabilize a análise e a caracterização de tráfego de aplicações em tempo real, tendo como foco principal as aplicações P2P de troca de arquivos mais populares.

É importante salientar que para atingir esse objetivo é necessário resolver dois sub-problemas independentes:

1. o problema de realizar de forma eficiente a monitoração em tempo real de tráfego no nível da camada de aplicação e

2. o problema de monitorar e analisar tráfego de aplicativos P2P.

A organização desse texto reflete essa a dualidade dos nossos esforços, podendo por isso ser dividido em duas partes relativamente distintas.

1.3 Contribuições

Como as três principais contribuições deste trabalho podemos listar um sistema capaz de realizar monitoração passiva de tráfego com recuperação de estado em tempo real, um trabalho de caracterização do tráfego de duas aplicações P2P num provedor local onde utilizou-se o sistema desenvolvido e um estudo sobre o volume de tráfego comum às duas redes.

O sistema desenvolvido, chamado de *Palantír* [4], é capaz de realizar a monitoração

[3] uso de um grande número de conexões independentes, possivelmente paralelas, para transferir diferentes partes do objeto desejado.

[4] Nos livros de J. R. R. Tolkien (trilogia do Senhor dos Anéis), um Palantír é um dispositivo que pode ser usado para coletar informação, permitindo à pessoa que o utiliza perceber eventos distantes no presente e no futuro.

de tráfego no nível da camada de aplicação com recuperação de estado em tempo real, mesmo em ambientes com grandes volumes de dados e altas taxas de transmissão. Sua arquitetura é baseada na monitoração passiva do tráfego da rede de interesse, fazendo a recomposição das conexões TCP observadas e permitindo que cada conexão entre clientes e servidores seja identificada e tratada de maneira independente. A partir dessa recomposição, os protocolos de diferentes redes P2P podem ser analisados e informações sobre cada requisição e sobre cada objeto transferido podem ser coletadas. O *Palantír* também permite a recuperação dos arquivos transferidos, mesmo que isso exija na sua recomposição a partir das diversas conexões de uma transferência que use *swarming*.

Através do uso do *Palantír*, realizamos a monitoração do tráfego associado a dois sistemas P2P de troca de arquivos em um provedor de acesso à Internet por um período de 10 dias. Analisamos diversas características desse tráfego, entre elas a popularidade dos recursos compartilhados, o processo de chegada das requisições e as características dos usuários dessas redes. Observamos comportamentos similares aos previamente relatados na literatura [Gummadi et al., 2003, Sen and Wang, 2002].

Além disso, sabendo que a localidade de referência encontrada no tráfego P2P o coloca como um bom candidato para o uso de soluções de economia de largura de banda tal como *caches* [Leibowitz et al., 2002], buscamos ver como essa localidade se manifesta entre duas redes P2P distintas e quais seriam os ganhos possíveis no caso da existência de um hipotético *cache* que operasse entre redes P2P, e descobrimos que poderia se conseguir ganhos da ordem de 6% com o seu uso.

Outro fator importante que diferencia nosso trabalho dos demais trabalhos nessa área é o nosso tratamento único da fragmentação de arquivos. Com a popularidade de redes que usam um esquema de transferência fragmentada, é importante observar como tal esquema de distribuição altera as características de *cacheabilidade* e de localidade de referencia desses arquivos.

1.4 Organização do texto

O restante desse texto encontra-se organizado da seguinte forma. O capítulo 2 apresenta o problema geral de monitoração de tráfego com recuperação de estado em tempo real.

O capítulo 3 apresenta a arquitetura e implementação do *Palantír*, discutindo como lidamos como problemas de captura de pacotes em redes de alta velocidade com hard-

ware comum e sistemas de código aberto, com o problema de manutenção de estado das várias conexões e como provemos mecanismos para tratamento de protocolos da camada de aplicação na arquitetura. Além disso, comenta-se também sobre o desempenho do sistema.

O capítulo 4 busca apresentar a estrutura e funcionamento de sistemas P2P, focando particularmente nas redes P2P monitoradas, discutindo como suas características e funcionamento. Discutimos também como as particularidades de cada rede foram consideradas durante a elaboração do *Palantír*.

O capítulo 5 discute os resultados obtidos com a aplicação do *Palantír* na análise do tráfego P2P em um provedor banda larga real, caracterizando o tráfego observado por um período de 10 dias. Comenta-se também sobre as métricas usadas na caracterização bem como sobre os detalhes de sua implantação.

Finalmente, no capítulo 6 apresentamos uma disscussão dos resultados, conclusões e dos trabalhos futuros.

Capítulo 2

Monitoração Passiva de Tráfego em tempo real com recuperação de estado

A monitoração passiva de tráfego em tempo real com recuperação de estado não pode ser encarada como uma simples especialização da atividade de monitoração de tráfego. Ao invés disso, ela deve ser encarada como uma conjunção de diversos aspectos diferentes que, apesar de não serem ortogonais entre si, apresentam peculiaridades e desafios próprios.

Por esse motivo, neste capítulo, tentaremos familiarizar o leitor com cada um desses diferentes aspectos individualmente, descrevendo-os e apresentando trabalhos existentes na literatura que os abordem, bem como as suas respectivas soluções propostas.

Buscaremos também observar as similaridades que cada uma dessas facetas apresenta com outros problemas encontrados na área de redes e quais são as medidas adotadas na literatura para contorná-los.

2.1 Formas de Monitoração

Em diversas áreas do conhecimento humano, a atividade de medir e aferir valores é um tanto quanto controversa, podendo às vezes apresentar-se simples e inquestionável e, em outras circunstâncias, complicada e imprecisa. Se em alguns casos a dificuldade reside em definir uma unidade de medida adequada, em outros o problema é definir e delimitar aquilo que se deseja medir ou de não modificar o objeto analisado pela sua medição.

No caso particular de monitoração de tráfego para a obtenção de métricas para caracterização de tráfego P2P, não estamos interessados somente em obter valores absolutos de, por exemplo, octetos trocados entre os nós da rede. Mais do que isso: interessa-nos qualificar ao que exatamente esses octetos se referem e até mesmo se eles sequer fazem parte do tráfego da rede P2P que nos interessa.

Existem duas abordagens para monitoração de tráfego de redes, aplicáveis também a redes P2P: a abordagem ativa e a abordagem passiva. Cada uma possui diferentes características, incorrem em diferentes custos e complicações e, por isso, analisá-las-emos separadamente.

2.1.1 Monitoração ativa

Para compreender como um dado programa ou algoritmo funciona, na maioria dos casos, a leitura do seu respectivo código-fonte ou pseudo-código é suficiente. No caso de um sistema distribuído, a pluralidade de situações possíveis torna as coisas um tanto quanto mais complicadas.

Para obter informações sobre o estado no qual se encontra um dado programa em execução, um programador poderia lançar mão de um depurador. Com tal ferramenta é possível, por exemplo, investigar a pilha de execução, descobrir em qual função ou procedimento o programa atualmente se encontra, os valores das suas variáveis globais e locais, dentre várias outras coisas. No caso de um sistema distribuído, mais precisamente de uma rede P2P, as coisas não são tão simples assim. Não é possível simplesmente usar um "depurador" em toda a rede P2P para obter informações sobre o seu estado global e o estado dos seus nós, sobre as suas conexões e outras informações pertinentes[1].

Todavia, se essa metodologia não pode ser diretamente aplicada à rede, é fácil ver que nada impede que ela seja aplicada a um único ou a vários nós da rede e que, a partir desses nós, seja inferido o comportamento da rede como um todo. Essa é a idéia por trás de *monitoração ativa*.

A monitoração ativa de redes consiste então no uso de um ou vários clientes modificados ou devidamente instrumentados que ingressarão na rede P2P que se deseja monitorar. Estes clientes estabelecerão contato com outros nós dessa rede, descobrirão novos nós, interagirão com eles, receberão e enviarão mensagens, etc. Através dessa interação dos clientes instrumentados com os demais nós, com as devidas ferramentas,

[1] Não estamos levando em conta algoritmos tais como *distributed snapshot* e similares por não as considerarmos soluções viáveis neste caso.

pode-se interpolar o comportamento de nós normais da rede e, num nível acima, da própria rede.

Diversos trabalhos de caracterização de redes P2P utilizaram essa metodologia para a obtenção de dados para os seus trabalhos. Alguns usaram-na para observar como nós de determinadas redes interagiam entre si, o que permitiu modelar o comportamento dos nós dessas rede. [Izal et al., 2004, Markatos, 2002]

Outros usaram-na para extrair diretamente dos nós informações sobre a natureza dos recursos por eles compartilhados, tempo durante o qual eles estiveram disponíveis para contato na rede e outras características dos nós de forma similar a um robô vasculhador para a *web*[2]. [Pouwelse et al., 2005, Chu et al., 2002, Saroiu et al., 2002]

Apesar de ser fácil de ser entendida e utilizada, a monitoração de forma ativa apresenta alguns problemas, entre os quais podemos citar os seguintes [Sen and Wang, 2002]:

- Grande consumo de recursos

  A monitoração ativa pode gerar um grande consumo de recursos, tanto de rede quanto computacionais. Primeiro pelo fato de que é necessário o estabelecimento de várias conexões com vários dos nós da rede alvo a fim de que se possa interagir com estes, o que já gera um consumo considerável de recursos computacionais, mesmo que tal seja distribuído por vários nós. Além disso, a interação com os vários nós da rede alvo implica na geração e consumo de tráfego na proporção do tamanho da rede desejada, o que torna essa forma de monitoração inerentemente inviável para redes que podem possuir uma quantidade de nós na ordem de milhares.

- Interferência

  A monitoração ativa permite **inferir** como se dá a interação de nós normais da rede, porém não permite **observar** como realmente ocorre essa interação, pois aquilo que é realmente observado é a interação de um nó "artificial" da rede com um nó comum desta rede.

  A carga gerada na rede bem como o comportamento de um nó comum é produto de como um usuário dessa rede P2P se comporta ao utilizá-la. Isso não acontece no caso de um cliente modificado para fazer monitoração ativa e, por isso, não pode-se esperar que o comportamento observado seja o mesmo.

[2] Comumente chamado de *web crawler* ou *spider*.

Além disso, quando se realiza monitoração ativa em uma rede, há que se ponderar até que ponto a atividade de monitoração não está interferindo com a atividade normal que se deseja monitorar. Devido à natureza intrusiva dessa forma de monitoração faz-se necessário quantificar até que ponto a atividade de monitoração está alterando ou afetando o ambiente sendo monitorado.

- Problemas com protocolos ou clientes fechados.

  A criação ou instrumentação de um cliente para monitoração ativa requer conhecimentos do protocolo da rede monitorada, acesso a um cliente que possa ser facilmente instrumentado ou ao código fonte desse último. Isso pode nem sempre ser o caso, dificultando ou impossibilitando o uso de monitoração ativa em algumas circunstâncias.

  Tome-se por exemplo as redes P2P KaZaa e Gnutella. Enquanto para essa última existem vários trabalhos que usam monitoração ativa, o mesmo não pode ser dito para o KaZaa. Isso ocorre exatamente por existirem clientes Gnutella com código aberto e facilmente modificáveis e pelo fato do protocolo dessa rede ser conhecido e documentado, enquanto que a rede KaZaa possui um protocolo fechado que utiliza criptografia em suas mensagens de controle e pelo fato de que o cliente oficial dessa rede não poder ser instrumentado com facilidade.

- NATs e *Firewalls*

  Com o crescimento do uso de NATs[3] e de *firewalls* até em ambientes residenciais, a premissa do argumento fim-a-fim [Saltzer et al., 1984] não é mais verdadeira: nem todos os nós que estão na rede são acessíveis diretamente. O impacto disso para a monitoração ativa é que uma parcela crescente dos nós não poderá ser monitorada, criando mais um inconveniente para esse tipo de monitoração. [Pouwelse et al., 2005]

2.1.2 Monitoração passiva

A atividade de monitoração lembra, em vários aspectos, a atividade de espionagem. Às vezes, é necessário entrar em território inimigo, aprender um idioma, um determinado sotaque e fazer perguntas às pessoas certas. Mas, ao fazê-lo, várias são as coisas que

[3] *Network Address Translation*, técnica empregada em alguns roteadores que, na sua forma mais comum, permite que vários endereços IPs de uma rede privada sejam mapeados em um único endereço IP de uma rede pública.

podem dar errado. O idioma pode ser completamente indecifrável aos serviços de inteligência, o sotaque pode possuir particularidades e idiossincrasias difíceis de serem imitadas, conseguir encontrar as pessoas-chaves para o sucesso da missão pode se tornar uma missão a parte, o sotaque ou até mesmo a forma de se portar e de se vestir podem comprometer o espião, a missão ou ambos.

Em certas circunstâncias, o melhor que um espião pode fazer é tentar passar completamente desapercebido pela multidão, plantar escutas telefônicas, tirar fotos à distância, procurar monitorar comunicações de rádio clandestinas, enfim: fazer tudo na surdina, sem entrar em contato direto com ninguém. Essa forma de ação pode não fornecer ao espião as respostas que ele deseja diretamente, mas ainda assim, com o tempo, ele obterá respostas para várias perguntas pertinentes sem que ele tenha que ir ativamente atrás deles.

De maneira similar à espionagem, na monitoração de redes existe uma forma de se conseguir informações sobre o tráfego e o comportamento de um particular sistema (que não precisa ser necessariamente uma rede P2P) sem que se interfira no seu tráfego. Esta forma de monitoração se chama *monitoração passiva*.

A monitoração passiva consiste na coleta do tráfego de uma certa localidade (ISP, instituição, etc) e na análise da parcela desse tráfego coletado que corresponda ao sistema desejado. Através da análise dessa parcela do tráfego pode-se inferir o comportamento dos nós internos e de alguns nós externos a essa localidade, além do tráfego e o comportamento de todo o sistema.

Analogamente à monitoração ativa, onde podemos distribuir clientes instrumentados em várias localidades para observar uma parcela maior do sistema em questão, a coleta para a monitoração passiva pode ocorrer em mais de um local, permitindo obter dados de várias localidades. O tráfego coletado pode ser posteriormente combinado para a obtenção de uma análise mais abrangente do sistema.

2.1.2.1 Formas de coleta

Essa coleta pode ser feita através do espelhamento e captura de pacotes associados a um dado tráfego ou pela interceptação do mesmo.

No primeiro caso o tráfego a ser monitorado é "espelhado" com o auxílio de um *hub* ou de algum dos dispositivos de rede especializado e é finalmente capturado e processado em um equipamento a parte, isolado dos equipamentos responsáveis pelo roteamento e filtragem desse tráfego. Dessa forma, o processo de coleta interfere o mínimo possível no fluxo normal do tráfego dessa localidade.

No segundo caso o processo de coleta ocorre nos mesmos equipamentos responsáveis pelo roteamento e filtragem desse tráfego. Nesse caso, o fluxo do tráfego é "interceptado" pelo equipamento de coleta e somente após ele ser totalmente processado pelo equipamento e pelo processo de coleta é que ele prossegue. Esse processo de coleta é mais intrusivo, pois a velocidade de toda a rede agora está submetida à velocidade de processamento do equipamento de coleta.

Observe que, em ambos os casos, não existe interação entre o mecanismo de coleta e nós no sistema que se está monitorando.

2.1.2.2 Vantagens e Desvantagens

Devido à sua natureza e à maneira pela qual se obtêm informações sobre o sistema monitorado, a monitoração passiva possui algumas vantagens em relação à monitoração ativa [Sen and Wang, 2002], dentre as quais podemos citar:

- Não causa interferência

  A coleta não gera nem insere nenhuma informação "artificial": tudo que é observado provem de nós reais e da interação entre eles.

- Todo o tráfego é monitorado

  Toda e qualquer atividade dos nós de uma rede que gerar tráfego que passe pelo mecanismo de coleta pode ser detectada e monitorada, mesmo aquelas advindas de nós que estejam atrás de NAT ou *firewalls*.

  Essa é uma grande diferença entre a monitoração ativa e a passiva: tudo aquilo que gerar tráfego pode ser monitorado.

- Pode ser usado com protocolos proprietários

  Como dito na seção 2.1.1, existem dificuldades para usar monitoração ativa num sistema com protocolos ou clientes fechados. Essas dificuldades não desaparecerem completamente quando usa-se monitoração passiva mas, através da coleta do tráfego gerado por aplicações desse sistema, pode-ser obter dados suficientes para caracterizar os padrões de tráfego dessas aplicações.

Essas vantagens, contudo, devem ser consideradas em relação às deficiências e os problemas que uma monitoração passiva possui, tais como:

- Sobrecarga nos equipamentos da rede

  Apesar de não ocorrer a geração de tráfego nem de processamento extra como ocorre na monitoração ativa, toda a infra-estrutura de rede que estiver relacionada com o processo de monitoração poderá sofrer uma carga adicional de processamento. Esse fato pode ser agravado se a análise do tráfego coletado ocorrer em paralelo com a coleta.

  Isto não é inesperado, mas um efeito colateral da própria atividade de coleta, e é tanto mais verdadeiro quanto mais envolvidos com a tarefa de coleta e análise estiverem os equipamentos de rede das localidades onde se realiza a coleta.

- Velocidade de captura e análise

  Dependendo da forma como a coleta for realizada, pode-se obter informações que vão desde os endereços IPs dos nós participantes de uma dada rede P2P, do número médio de conexões realizadas por nós do sistema e volume de dados trocados pelos nós, até o conteúdo dos recursos trocados no sistema. Ou seja: pode-se obter informações que vão desde a camada de rede até a camada de aplicação.Todavia, quanto maior o nível de detalhamento desejado, maior será o custo para obtê-las.

  Esse fato deve ser levado em conta especialmente no caso de monitoração passiva por "espelhamento" de tráfego. Nessa forma de monitoração não existe controle sobre a velocidade dos dados que chegam ao dispositivo de coleta, fazendo com que ele possa ser sobrecarregado se essa taxa ultrapassar a sua capacidade de processamento, que aumenta conforme o nível de detalhamento desejado aumente. Desta forma, faz-se necessário ponderar a qualidade ou nível de detalhamento dos dados que poderão ser obtidos através da monitoração passiva.

  Na monitoração ativa, é a capacidade de processamento do cliente instrumentado que determina o volume do tráfego gerado e consumido por ele e, portanto, o volume de informações que se pode obter a partir dessa forma de monitoração. O mesmo ocorre na monitoração passiva por interceptação de tráfego: todo o tráfego está submetido à velocidade de processamento do equipamento de monitoração. Nesse último caso, deve-se pesar até que ponto é aceitável a degradação dos serviços de rede devido à monitoração.

- Horizonte de monitoração limitado

A disponibilidade de recursos computacionais e de rede podem limitar o conjunto e a variedade de nós observados através de monitoração ativa. Apesar de que o mesmo ocorre, numa escala menor, para a monitoração passiva, o fator determinante para essa limitação na monitoração passiva é a própria rede monitorada.

Isso ocorre devido ao fato de que, na monitoração passiva, pode-se observar apenas o tráfego que passar pelo mecanismo de coleta. Se esse tráfego não for representativo o suficiente, a qualidade dos dados obtidos através dessa monitoração pode ser questionável.

2.2 Atividades Relacionadas

Muitos dos problemas encontrados na monitoração passiva de tráfego também são encontrados em outras áreas. Analisar como esses problemas são contornados em diferentes áreas pode nos ajudar a entender melhor o problema que está sendo tratado e nos fornecer indicativos de formas para lidar com tal problema bem como nos ajudar a delimitar os problemas que são únicos nossa área.

2.2.1 Identificação de Tráfego

A capacidade de associar o tráfego decorrente de diferentes aplicações em uma dada rede é útil para uma vasta gama de operações de gerência de redes de computadores, tais como engenharia de tráfego, planejamento de capacidade e diferenciação de serviços [Sen et al., 2004].

A alternativa tradicional para identificação de tráfego consiste em monitorar *fluxos*, ou seja, monitorar o volume de dados relativos a uma determinada conexão entre duas terminações da rede (*end-points*), geralmente efetuada através de uma conexão TCP. Após a monitoração de fluxos na rede realiza-se associação deles, através dos seus portos de origem e destino, a aplicações conhecidas. A partir da consolidação dos dados dos diferentes fluxos devidos a uma mesma aplicação, pode-se obter o volume de dados decorrentes das diversas aplicações em uso na rede. É importante salientar que todo esse processo ocorre sem que o conteúdo trocado por essas conexões seja observado.

Todavia, várias aplicações, em especial as mais recentes versões de aplicativos P2P de troca de arquivo, têm deixado de utilizar apenas portos fixos e bem conhecidas para as suas comunicações, passando a utilizar portos aleatórios e até mesmo portos tradicionalmente utilizados por aplicações bem conhecidas, tais como os portos 80

(HTTP), 25 (SMTP) e 143 (HTTP seguro). Isso tornou a alternativa tradicional de identificação de tráfego por portos de comunicação obsoleta e imprecisa. Esse fato tem motivado diversos estudos sobre formas de realizar identificação de tráfego em enlaces de alta velocidade que sejam capazes de lidar com tráfego P2P "camuflado" em portos não-convencionais [Karagiannis et al., 2004a].

A alternativa mais intuitiva, o que não significa que seja a mais trivial de ser implementada, para realizar identificação de tráfego nesse novo cenário consiste em analisar não mais os portos de origem e destino das conexões observadas no tráfego coletado, mas analisar o conteúdo dos dados trocados nessas conexões. Essa análise geralmente consiste na busca por *assinaturas* de aplicações conhecidas, ou seja, na busca por seqüências de caracteres ou de padrões que indicam ou evidenciam o uso de uma determinado protocolo ou aplicativo. Essa busca poder ser feita apenas apenas nos primeiros octetos trocados em uma conexão. Em alguns casos pode ser necessário reconstituir a seqüencia de dados trocados em uma conexão para somente então realizar a busca por assinaturas. [Sen et al., 2004].

Todavia, como veremos posteriormente, a captura de pacotes em enlaces de alta velocidade possui algumas complicações peculiares: em algumas infra-estruturas para coleta de pacotes em enlaces de altíssima velocidade (tais como OC-48, de 2488 MBps) não é possível recuperar mais do que partes do pacote, por vezes não conseguindo nem mesmo capturar mais do que os cabeçalhos dos protocolos da camada de transporte [Karagiannis et al., 2004a]. Mesmo nos casos onde a coleta de pacotes completos é possível, a reconstituição em tempo real dos dados trocados em um fluxos real pode não sê-lo, forçando a busca a ser feita diretamente nos pacotes capturados, o que resulta em identificações menos confiáveis.

Para contornar essas peculiaridades, alguns trabalhos aliaram a técnica de identificação por assinaturas com heurísticas, a fim de aumentar o desempenho, a eficácia e a confiabilidade de suas metodologias [Karagiannis et al., 2004a]. Outros trabalhos vão além e obtêm a identificação do tráfego através apenas nos padrões das conexões observadas no tráfego. [Karagiannis et al., 2004b].

2.2.2 Sistemas Detecção de Intrusão à Rede

Sistemas de Detecção de Intrusão à Rede ou NIDS (*Network Invasion Detection Systems*) são sistemas que monitoram o tráfego de uma rede, procurando por sinais de possíveis atividades ilícitas ou que possam vir a colocar em perigo a rede por eles monitorada, gerando alertas no caso de uma atividade dessas ser detectada.

A elaboração de tais sistemas apresenta problemas que são comuns aos de identificação de tráfego e aos de *firewalls*.

A concepção e funcionamento de um NIDS pode seguir dois modelos: um baseado em assinaturas e um baseado na análise de protocolos [Desai, 2002].

O primeiro é similar à identificação de tráfego: assinaturas, ou seja, seqüências de caracteres ou de padrões que indicam ou evidenciam a ocorrência de determinados ataques conhecidos são verificadas em pacotes de alguns fluxos. Todavia, no caso de NIDS, o problema de detectar um ataque pode requerer a agregação de informações não de um, mas de vários fluxos ao mesmo tempo. Além disso, para a correta identificação de certos ataques, pode ser necessário obter o conteúdo dos dados da camada de aplicação de alguns fluxos, o que pode requerer a remontagem de parte deles em tempo real.

No seguindo modelo, o conteúdo dos fluxos do tráfego monitorado é remontado e decodificado de acordo com a especificação de cada protocolo observado. Como o sistema possui informações sobre como um determinado protocolo deve funcionar e como deve ser a resposta a um determinado estimulo, tais sistemas conseguem decidir com mais precisão se um determinado ataque está em curso e se o mesmo obteve sucesso.

Esses modelos, mesmo que distintos, podem ser associados num só sistema, criando NIDSs mais robustos e mais difíceis de serem ludibriados.

É necessário considerar que, de maneira similar à identificação de tráfego, na medida que as velocidades de transferências de dados nas redes aumenta, corre-se o risco de se aproximar e até de se ultrapassar o limite da capacidade de processamento que os NIDS tradicionais podem lidar. Existem propostas que buscam contornar essas limitações, mesmo no caso onde é preciso realizar a recuperação de estado de um fluxo para a sua análise em tempo real pelo NIDS [Kruegel et al., 2002]. Todavia, a infra-estrutura necessária para implementar tais soluções são bastante caras.

2.2.3 Roteamento e Filtragem de pacotes

As funções desempenhadas por roteadores e por *firewalls* possuem algumas similaridades entre si e também com aquelas desempenhadas na monitoração passiva de pacotes. Afinal, todo pacote que passe por um deles será individualmente analisado e uma decisão deverá ser tomada com base no seu conteúdo – e possivelmente no conteúdo de pacotes anteriores [4].

[4] Excetuando-se os casos onde a velocidade do tráfego excede a capacidade desses equipamentos.

No caso de roteamento, a decisão, que depende das políticas configuradas no equipamento, consiste em determinar o destino que cada pacote terá, ou seja, para qual interface do dispositivo ele será repassado. Já para *firewalls*, a decisão consiste em determinar, com base num conjunto de regras configuradas no equipamento, se um determinado pacote pode seguir o seu caminho ou se se ele deverá ser descartado. Devido a essa similaridade, essas duas funcionalidades geralmente encontram-se associadas em um mesmo equipamento.

Ambas as atividades podem ser realizadas sem que seja necessário manter estado sobre o tráfego processado. Todavia, a necessidade de se proteger de ataques mais sofisticados fez com que diversas infra-estruturas de filtragem de pacotes e de firewalls passassem a guardar informações sobre os fluxos monitorados e usar tais informações como parte da do processo de decidir o destino de um pacote [Desai, 2002, van Rooij, 2001].

O mesmo aconteceu em roteadores. Devido à escassez de endereços IP e até por questões de segurança, vários roteadores passaram a disponibilizar recursos de tradução de endereços de rede ou NAT (*Network Address Translation*). Na sua forma mais comum, esse recurso permite que vários endereços IPs de uma rede privada sejam mapeados em um único endereço IP de uma rede pública. Antes de repassar qualquer pacote de uma a conexão originadas em um micro da rede privada, o NAT modificará o endereço e o porto de origem desse pacote de tal forma que aos micros da rede externa a conexão aparentará ter sido originada pelo próprio NAT. O processo inverso ocorre quando um pacote de rede externa, referente a uma conexão que sofreu mapeamento, que chegue ao NAT. Isso ocorre sem que seja necessária a intervenção e até mesmo o conhecimento dos micros da rede privada da existência de um dispositivo efetuando NAT em sua rede. Esse mapeamento, no entanto, requer a manutenção e o registro de cada conexão que sofrer mapeamento.

Nem todos os protocolos existes funcionam corretamente sob NAT. Dessa forma, para que o uso dessa técnica não interfira no uso da rede, pode ser necessário que, além de registro de estado sobre as conexões que sofreram tradução, o roteador precise monitorar conexões desses protocolo, remontar os seus fluxos, interpretar e até alterar o conteúdo de parte do tráfego de tal protocolo.

É importante observar que, diferentemente das atividades mencionadas nas subseções acima, o roteamento é uma atividade intrusiva por definição. Não é possível e nem faz sentido realizá-lo utilizando espelhamento de tráfego. É interessante que ele seja o mais eficiente e rápido possível mas, ao contrário das outras atividades, é a rede que se adequa à velocidade de processamento do roteador, e não o contrário.

2.3 Captura de pacotes

Como visto na seção anterior, são várias as atividades e os serviços que podem ser agregados a uma rede e que usam, de uma forma ou de outra, mecanismos de coleta de tráfego. Contudo, excetuando-se o roteamento, nenhum desses serviços pode ser considerado essencial à rede e, portanto, não é desejável que a utilização desses serviços degrade a rede ou debilite a capacidade de operação dos serviços essenciais. Por esse motivo é que, via de regra, tais serviços tidos como "secundários" ou "auxiliares" utilizam alguma forma de monitoramento passivo através de espelhamento de tráfego, como visto na seção 2.1.2.1, para coletar o tráfego necessário às suas operações. Contudo, não se tratou do processo de coleta em si. Nesta seção, abordaremos esse processo, apresentando os sistemas que são inevitavelmente utilizados para esse propósito: sistemas de *captura de pacotes*.

Sistemas para realizar captura de pacotes em sistemas operacionais de propósito gerais são antigos. O conceito e o nome aparecem já em meados de 76, apesar de que o primeiro trabalho publicamente disponível data de 87 [Mogul et al., 1987]. A proposta original desses sistemas era a de facilitar o desenvolvimento de protocolos e funcionalidades de rede através de aplicações rodando em modo usuário, onde existe uma maior variedade de ferramentas e de flexibilidade para o desenvolvimento. Para tanto, tais sistemas ofereciam uma interface pela qual aplicações poderiam registrar o interesse pela recepção dos pacotes capturados da rede e cadastrar filtros que limitariam os pacotes repassados apenas àqueles destinados à aplicação.

De forma geral, o processo de capturar pacotes da rede e entregá-los para as aplicações é composto dos seguintes passos:

1. Recepção do pacote pela interface de rede
2. Recuperação do pacote pelo *driver* da inteface de rede
3. Filtragem dos pacotes de interesse da aplicação
4. Cópia do pacote da área de memória do *kernel* para a área da aplicação
5. Processamento do pacote pela aplicação

Devido à versatilidade desses sistemas, eles acabaram ganhando uso em várias outras áreas e aplicações além daquelas para as quais foram originalmente concebidos. Tal

fato impulsionou o desenvolvimento de diversos trabalhos buscando aumentar o desempenho e a flexibilidade de tais sistemas, muitos atacando diversos aspectos particulares do processo de captura de pacotes.

Alguns propuseram novas arquiteturas ou arquiteturas aperfeiçoadas para captura [Mogul et al., 1987, McCanne and Jacobson, 1993, van der Merwe et al., 2000]. Outros buscaram otimizar o processo de filtragem [Yuhara et al., 1994, Bailey et al., 1994, Engler and Kaashoek, 1996, Begel et al., 1999], enquanto que alguns concentraram-se em diminuir o custo de recuperar o pacote da placa de rede para a área de memória do kernel e o custo da cópia de pacotes dessa área para a área de memória das aplicações [Rizzo, 2001, Deri, 2004], em trazer parte do processamento feito pelas aplicações para dentro do kernel [Bos et al., 2004, Ioannidis et al., 2002] e até em paralelizar o processo de captura para aumentar o seu desempenho [Varenni et al., 2003].

Apesar de todos os esforços para tornar a captura eficiente em sistemas comuns através de software, existem soluções que buscam melhorar o desempenho da coleta de pacotes otimizando por hardware alguns ou vários dos passos da captura [Cleary et al., 2000, Endance Measurement Systems, 2005, Degioanni et al., 2003, Degioanni and Varenni, 2004, Cho et al., 2002]. No entanto, nenhuma desses trabalhos apresenta uma solução definitiva. O uso de cada uma delas deve ser analisado frente às realidades de custo, desempenho e flexibilidade de cada projeto.

2.3.1 Captura de pacotes em enlaces de alta velocidade

A disponibilidade de interfaces para sistemas de captura de pacotes em vários sistemas operacionais de propósito geral bem como a flexibilidade desses sistemas de captura promoveram a sua disseminação e aplicação para os mais diversos fins, como visto acima. Entretanto, apesar de que várias melhorias no desempenho de sistemas de coleta podem ser obtidas pelo uso dos resultados dos trabalhos mencionados anteriormente, existe um limite para o que tais sistemas podem obter com a arquitetura atual de computadores pessoais e com hardware comum. Esse limite não se deve apenas a um mas a vários componentes da arquitetura atual [Cleary et al., 2000, Iannaccone et al., 2001]:

- Processador

  Em um estudo publicado em 2001, Iannaccone afirma que, com os processadores existentes na época, teria-se tempo para executar apenas 360 instruções por pa-

cote, caso a coleta ocorresse em um enlace OC-192 (10Gbps). Considerando-se que o custo de processamento de um segmento TCP, incluindo o custo do processamento do pacote IP que o encapsulasse e excluindo os custos e o tempo para levar tal pacote da interface de rede para o processador, é de aproximadamente 335 instruções, vê-se que mesmo os processadores já estão chegando perto do limite [Clark et al., 1989].

É bem verdade que de 2001 para os dias atuais houve progressos no que diz respeito à capacidade e à velocidade de processamento dos processadores. Ainda assim não podemos desconsiderar esses indícios.

- Barramento PCI

  Também não se pode esquecer que, a despeito de qualquer melhoria nos processadores, ainda existe um custo para levar os pacotes capturados da placa de rede ao processador e que esse custo deve-se, entre outras fatores, ao barramento usado na arquitetura dos computadores pessoais atuais.

  Além do custo de cópia para o processador o barramento também possui um limite para o volume de dados que pode atravessá-lo. No caso de barramentos PCI mais comuns (operando com 32 bits e a 33 Mhz), a taxa máxima de transferência teórica é de 132MBytes/s, enquanto que os valores mais realistas oscilam entre 40 e 50MBytes/s [Cleary et al., 2000]. Ou seja: as velocidades que podem ser encontradas hoje nos nós da rede em *datacenters* já atingem os limites da tecnologia de barramento dos computadores pessoais atuais.

- Memórias

  As memórias encontradas em computadores recentes são rápidas o suficiente para lidar com as velocidades das redes atuais. Todavia, a quantidade de memória geralmente disponível pode se mostrar um problema: 128 Mbytes podem ser completamente preenchidos em apenas 2,5 segundos de coleta de tráfego em enlaces com velocidades próximas a OC-12.

- Discos Rígidos

  Finalmente, em muitas atividades envolvendo captura de pacotes existe a necessidade de armazenar o tráfego coletado para uma análise posterior. Dessa forma, um outro componente dos PCs deve ser observado: os discos rígidos.

Os discos rígidos são e serão por um bom tempo o maior gargalo num sistema de coleta de tráfego operando em PCs comuns. O tráfego de enlaces Gigabit Ethernet não pode ser totalmente capturado por interfaces IDEs atuais. Isso pode ser contornado pelo uso sistemas como RAID, mas não eles ainda não são comuns.

É necessário considerar que atualmente o PC encontra-se em uma fase de transição: barramentos PCI estão sendo substituídos por barramentos PCI-Express, discos ATA por discos SATA; sistemas multiprocessados e processadores com vários núcleos estão ficando cada vez mas comum, etc. Essas mudanças provavelmente capacitarão PCs ordinários a capturar tráfego em velocidades cada vez maiores. Por outro lado, não existem indícios de que as velocidades das redes de computadores futuras diminuirão. Em suma: esses problemas persistirão por um bom tempo.

Para contornar essas limitações da arquitetura do PC, algumas medidas podem ser tomadas:

- Processar uma parte do todo ou o todo de uma parte

  Dependendo do tipo de informação que se deseja obter com a coleta de pacotes, pode chegar a conclusão de que não se deseja observar 100% do tráfego de uma rede. Até mesmo no caso onde deseja-se capturar todos os pacotes que passem no enlace monitorado, existem casos onde apenas os cabeçalhos iniciais dos pacotes interessam. Em ambos os casos, pode-se atenuar o impacto da captura em enlaces de alta velocidade diminuindo o volume efetivo de dados recuperados da rede. [Sen et al., 2004, Iannaccone et al., 2001]

- Unir software genérico e hardware especializado

  A união de hardware especializado com sistemas gerais de coleta via software pode mostrar-se como uma boa opção [Degioanni and Varenni, 2004]. O custo dessa solução excederá uma solução mais simples, mas obtêm-se a versatilidade de sistemas de software e a eficiência de hardwares dedicados.

- Dividir para conquistar

  Em algumas circunstâncias, a captura e processamento em uma única máquina pode se mostrar inviável ou fazê-lo pode ser muito caro computacionalmente. Nesse caso, uma possibilidade é de dividir o custo da monitoração por várias máquinas [Kruegel et al., 2002].

Além da arquitetura do PC, outros fatores podem contribuir para um desempenho aquém do esperado na captura de pacotes em enlaces de alta velocidade. Um desses fatores é a biblioteca mais comum para sistemas de captura de pacotes em sistemas operacionais abertos: a libpcap [tcpdump, 2005]. Sendo praticamente um padrão para a construção de aplicativos que usam captura de pacotes, essa biblioteca pode ser encontrada em vários sistemas operacionais e é bastante flexível. Entretanto, ao forçar o processamento serial dos pacotes capturados, essa biblioteca limita a sua usabilidade em redes mais rápidas por impedir o processamento paralelizado de pacotes [Desai, 2002]. Existem alternativas, mas que não tão populares quanto a libpcap e não que estão disponíveis para tantos sistemas operacionais quanto a libpcap [Moore et al., 2001].

2.4 Recuperação de Estado

O termo "recuperação de estado" é comumente associado a *firewalls* e à filtragem de pacotes (*statefull packet filtering*) mas também a NIDS (*statefull traffic inspection*). Esse mesmo termo, no entanto, quando aplicado no contexto de monitoramento passivo possui um caráter mais amplo que em ambos os casos anteriores. Em todos os casos entende-se que, de alguma forma, o processamento e análise de um pacote capturado se dará baseando-se no conteúdo desse pacote mas também no conteúdo de pacotes previamente observados. Além do propósito, a diferença entre os três casos reside basicamente na quantidade dos dados deste pacote e dos pacotes anteriores que será levada em consideração durante o processamento deste pacote. Na monitoração passiva com recuperação de estado, dependendo do tipo de métricas que se deseja obter com o seu uso, pode-se, por exemplo, dar-se ao "luxo" de observar apenas cabeçalhos de pacotes Ethernet presentes no tráfego ou, em outras circunstâncias, ter que recuperar todo o fluxo de dados bidirecional de uma conexão TCP.

Apesar dessa amplitude de situações, quando o termo "recuperação de estado" é aplicado à monitoração de tráfego, entende-se geralmente que o estado em questão refere-se ao protocolo TCP ou, mais precisamente, aos estados dos fluxos TCP existentes no tráfego monitorado, sem que se especifique exatamente até que ponto os estados desses fluxos serão recuperados.

2.4.1 Recuperação de estado em tempo real

Existem diversas ferramentas que permitem realizar a recuperação de estado no tráfego monitorado posteriormente à captura: tcpflow, ethereal, etc. Também existem ferramentas que permitem fazê-la em tempo real, em paralelo à captura. Todavia, o propósito e o desempenho delas variam consideravelmente.

Algumas dessas ferramentas destinam-se apenas à obtenção de métricas sobre os fluxos monitorados (*flow-level analysis*) [Inc., 2002, Sen and Wang, 2002]. Essas métricas limitam-se à volume de dados trocados, tempo de duração, etc. Como essas ferramentas obtêm esses dados apenas pela análise dos cabeçalhos dos pacotes e como algumas dessas ferramentas já vêm embutidas em roteadores, o desempenho delas é bastante satisfatório, sendo capazes de lidar com tráfegos até superiores a taxas obtidas em enlaces Gigabit EtherNet [Deri, 2003]. Usando hardware especializado e a mesma metodologia utilizada por essas ferramentas, pode-se obter métricas sobre os fluxos de enlaces com velocidades até OC-48 [Karagiannis et al., 2004a].

Outras soluções buscam ir além e recuperar todo o fluxo de dados bi-direcional de uma conexão TCP: dsniff, flowgrep, libnids. Todavia, como comentado na seção acima, quanto mais dados tiverem que ser processados e recuperados da rede, maior será o custo computacional de fazê-lo e mais complicado será fazê-lo com *harware* comum.

Capítulo 3

A arquitetura do sistema de monitoração

Tendo em vista a motivação apresentada na introdução e os trabalhos apresentados no capítulo anterior, como poderia se conceber um sistema capaz de realizar monitoração passiva de tráfego com recuperação de estado em tempo real? Como ele se adequaria às velocidades das redes atuais? Qual o custo e o desempenho de que tal sistema teria?

Nesse capítulo pretendemos responder essas questões, revisitando os objetivos e os requisitos desse trabalho. Comentaremos também sobre os compromissos e limitações encontradas durante a implementação do *Palantír*, o nosso sistema de monitoramento passivo com recuperação de estado em tempo real. Detalharemos como esses aspectos influenciaram o seu desenvolvimento e, finalmente, faremos uma análise do seu desempenho.

3.1 Considerações iniciais

Para que se possa compreender alguns dos aspectos envolvidos na forma com a qual propomos e implementamos o sistema de monitoração desenvolvido é importante que se saliente o que desejamos obter com tal sistema.

Conforme explicitado no capítulo 1, objetivamos implementar uma solução que viabilize a análise e a caracterização de tráfego de aplicações em tempo real, tendo como foco principal as aplicações P2P de troca de arquivos mais populares. Tal solução deve ser o menos intrusiva possível e, por esse motivo, optamos pelo uso de mecanismos de monitoração passiva. Além disso, para que se possa obter análises e caracterizações mais ricas através da solução, desejamos recuperar o tráfego tal como ele é entregue

às aplicações, o que implica na necessidade de realizarmos recuperação de estado no tráfego monitorado. Em suma, desejamos criar um sistema de monitoração passiva de tráfego com recuperação de estado em tempo real.

Todavia, é importante que observemos o cenário onde vislumbramos o uso de tal ferramenta: próximo aos roteadores de saída de instituições, ISPs e empresas que queiram realizar algum tipo de análise e monitoração em tempo real do seu tráfego. Nesses locais, em especial em ISPs, já é comum encontrar enlaces de alta velocidade, tais como Gigabit Ethernet e superiores.

Como vimos no capítulo anterior, existem diversas alternativas para lidarmos com o problema de captura de tráfego de maneira eficiente em enlaces de alta-velocidade. Desejávamos que a abordagem adotada fosse implementável em sistemas operacionais de código aberto e que tivesse um baixo custo. Alternativas baseadas em *hardware* são claramente mais caras e, como comentado no capítulo 2, existem soluções propostas que visam tornar o trabalho de captura viável e eficiente em sistemas comuns, mesmo em altas velocidades.

Ao optarmos por soluções que concentram seus esforços quase que em sua totalidade em *software*, ou mais especificamente em melhorias de sistemas operacionais de propósito geral, somos defrontados com a possibilidade de realizar parte do trabalho de análise de tráfego em modo protegido (*kernel-mode*) ou em modo de usuário (*user-level mode*). Todavia, a elaboração de sistemas que operem em modo protegido é mais laboriosa, tanto devido às limitações impostas pelos próprios sistemas operacionais à construção de código que execute em modo protegido como à dificuldade de depurar tal código. Soluções que buscam eliminar parte dessas complicações ainda são imaturas ou não possuem a flexibilidade necessária para a construção de um sistema como o *Palantír*.

Felizmente, como mencionado no capítulo anterior, diversos trabalhos melhoram o desempenho dos mecanismos de coleta em sistemas operacionais de propósito geral, eliminando os problemas comumente associados a sistemas que operem em modo de usuário e as razões que geralmente motivam a construção de sistemas em modo protegido.

3.2 A organização do *Palantír*

O processo de construir a arquitetura em modo de usuário nos levou à criação de um sistema dividido em camadas que assemelha-se bastante a soluções de servidores de rede

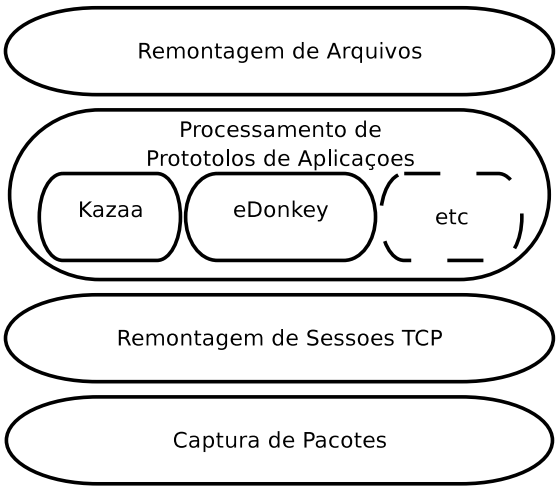

Figura 3.1: Organização em camadas do *Palantír*

em nível de usuário, ou seja, de soluções que propõem a construção de uma pilha de protocolos que será executada em modo de usuário [Brustoloni and Steenkiste, 1998, Thekkath et al., 1993].

No caso do *Palantír*, foram definidas 4 camadas, a saber:

1. Captura de pacotes
2. Remontagem de sessões TCP
3. Processamento dos protocolo das aplicações
4. Remontagem de arquivos

Essa organização pode ser vista na figura 3.1. Com exceção da última camada da arquitetura, todas as outras poderiam ser mapeadas para camadas do modelo OSI: a primeira para a camada física e de enlace, a segunda para a camada de rede e de transporte, e a terceira para as 3 últimas camadas do modelo OSI. A última camada, no entanto, é específica para o monitoração de aplicações P2P, e será comentada na seção 4.4.3.

3.2.1 Captura de Pacotes

Tradicionalmente, sistemas operacionais modernos tais como o Linux, FreeBSD e Windows 2000 não apresentam um desempenho aceitável para a coleta de pacotes em redes de alta velocidade. Usando PCs *low-end*, pode-se observar taxas consideráveis

de queda de pacotes até mesmo em redes mais lentas. Técnicas como *device pooling*, implementadas em versões mais recentes tanto do Linux como do FreeBSD, melhoram esse cenário consideravelmente, mas não eliminam o problema completamente [Ioannidis et al., 2002, Rizzo, 2001]. Outra forma de melhorar o desempenho da captura nesses sistemas é através de extensões que minimizem o custo da cópia de pacotes do *kernel* para os programas de análise de pacotes que rodam em modo usuário. Em nosso ambiente usamos um *patch* para o kernel do Linux criado por Luca Deri que implementa ambas as técnicas acima. Este *patch* é chamado de PF_RING e, com o seu uso, a perda de pacotes, mesmo em altas velocidades, é reduzida a níveis aceitáveis [Deri, 2004].

3.2.2 Remontagem das sessões TCP

Uma vez que se tenha recuperado os pacotes da rede, faz-se necessário realizar o trabalho de recuperar o estado das conexões existentes nesse tráfego. Isso significa analisar os quadros obtidos, recuperar destes pacotes IP, desfragmentando-os e validando-os conforme for o caso para finalmente iniciar o processo de remontagem das sessões ou fluxos TCP existentes — tudo isso em tempo real.

Como visto em 2.4, muitas das ferramentas existentes para esse propósito não são destinadas para recuperação de estado em tempo real. Uma das exceções é a libNIDS, uma biblioteca de código aberto que se mostrou bastante eficiente nos nossos testes e que é capaz de realizar tanto a validação e desfragmentação dos pacotes IPs quanto a remontagem dos fluxos em tempo real.

3.2.3 Processamento dos protocolos das aplicações

Finalmente, a última camada genérica da arquitetura: a responsável pela análise dos protocolos das aplicações que se deseje monitorar. Sua elaboração assemelha-se ao trabalho de conceber um NIDS baseado em análise de protocolo (seção 2.2.2) e, como pode-se ver na figura 3.1, para cada aplicação que se deseje monitorar um novo elemento deve ser adicionado a essa camada.

Criamos dois destes elementos, destinados à monitoração do tráfego de duas aplicações P2P de troca de arquivo. Comentaremos mais sobre o desenvolvimento de tais elementos na seção 4.4.2.

3.3 Avaliação de desempenho

Uma vez definidas a organização da arquitetura do *Palantír* e as ferramentas que serão utilizadas na sua implementação restam alguns questionamentos:

- As medidas tomadas para melhorar a capacidade e eficiência do processo de captura de pacotes no *Palantír* realmente surtiram efeito?
- Qual o impacto das camadas de remontagem de sessões, de analise de protocolos de aplicações e de remontagem de arquivos sobre o desempenho do processo de coleta?
- Ele é capaz de monitorar passivamente com recuperação de estado e em tempo real tráfego em enlaces de alta-velocidade com eficiência?

Em suma: como é o desempenho do *Palantír*?

Para responder a essas perguntas realizou-se um experimento onde verificou-se como a adição de cada elemento da arquitetura alterava o desempenho da coleta medindo-se a taxa de perda de pacotes. Essa taxa representa a incapacidade do sistema de lidar com o taxa de vazão de dados da rede. Ela é conseqüência direta das ineficiências do sistema e, quanto maior ela for, menos dados o sistema terá para processar e portanto pior será a qualidade dos resultados que poderão ser obtidos pelo uso do *Palantír*.

3.3.1 Descrição do experimento

O experimento utilizou três microcomputadores IBM-PC iguais, os quais chamaremos de computadores *A*, *B*, e *C*. Todos possuíam as mesmas especificações: processador Intel Pentium4 de 2.80GHz, 1 gigabyte de memória RAM DDR-400, placa-mãe Intel D865PERL, disco rígido SATA Seagete modelo ST3120026AS e placa de rede PCI Intel PRO/1000. Todas as máquinas foram interligadas por um *switch* Intel Gigabit Ethernet.

As máquinas *B* e *C* ficariam responsáveis por enviar à máquina *A* o conteúdo de uma coleta de tráfego (*trace*) realizada em um provedor de acesso à Internet de banda larga. Para que o tráfego enviado por *B* a *A* diferisse do tráfego enviado por *C*, utilizou-se para o processo de envio o aplicativo tcpreplay [tcpreplay, 2005], capaz de modificar deterministicamente os cabeçalhos IPs presentes no tráfego a ser enviado sem que isso acarrete no envio de pacotes inválidos. O tráfego gerado conjuntamente por essas máquinas chegava à máquina *A* a uma velocidade de aproximadamente 500Mbps.

Rede P2P	No *trace*	Total Observado
Número de Pacotes	7.928.526	15.857.052
Número de Conexões TCP	84.194	168.388
Número de Conexões eDonkey	18.781	37.562
Tamanho em *Mbytes*	2.921	5.842
Tempo de Coleta	816,49s	*N/A*

Tabela 3.1: Características do *trace* e do tráfego observado pela máquina A

Os dados originais da coleta bem como os dados do tráfego final recebido pela máquina A podem ser vistos na tabela 3.1.

Por sua vez, na máquina A monitoramos a quantidade de pacotes capturados com sucesso, o complemento da quantidade da quantidade de pacotes perdidos, e como esse valor reagiria à adição de cada elemento ou etapa arquitetura:

1. ao processo de captura de pacotes,
2. ao processo de remontagem de conexões TCP através da libNIDS,
3. ao processo de análise de protocolos,
4. ao processo de realizar uma remontagem parcial dos arquivos
5. ao processo de remontar totalmente arquivos

Observe que não existe um relação bijetiva entre as camadas da arquitetura e as etapas do experimento.

A quantidade de pacotes capturados com sucesso foi observada, para cada etapa, em quatro configurações distintas, onde os três seguintes aspectos eram variados:

- o uso ou não de PF_RING, o patch para captura de pacotes de Lucas Deri,
- a adição ou não de suporte a *pooling* de pacotes no *driver* da placa de rede Intel Pro/1000, recurso ao qual nos referiremos no restante do texto por NAPI,
- o uso ou não da capacidade de HyperThreading (HT) do processador Pentim 4.

Por julgamos desnecessário para os propósitos do trabalho testar todas as combinações dos aspectos acima, consideramos apenas as seguintes configurações:

1. com PF_RING, NAPI e HT

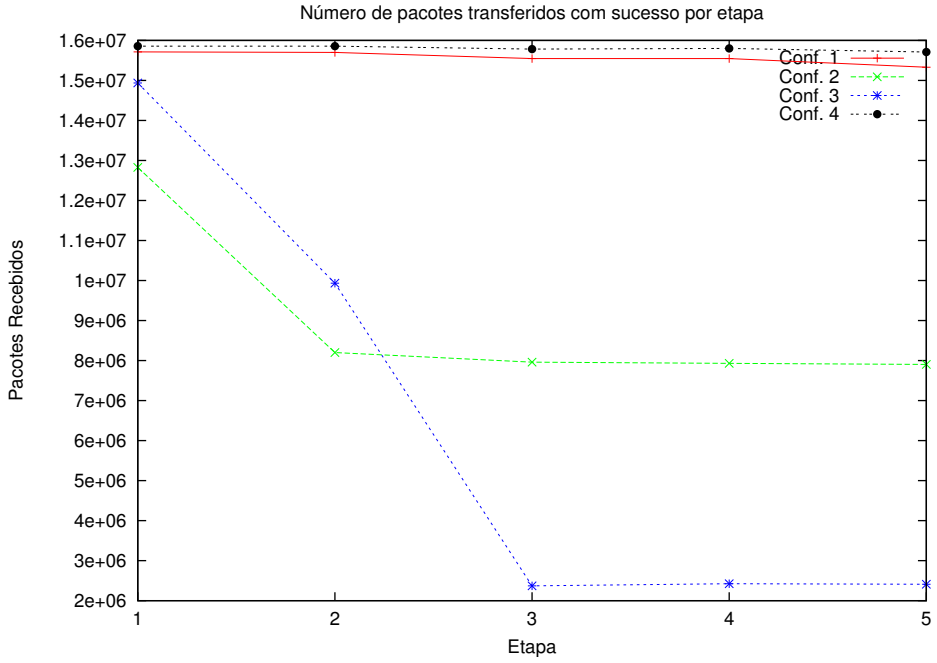

Figura 3.2: Quantidade de pacotes capturados

2. sem PF_RING, mas com NAPI e HT

3. sem PF_RING, mas sem NAPI e HT

4. com PF_RING, mas sem NAPI e HT

Para verificar etapa de de captura utilizamos o programa pcount, disponível com o código do PF_RING. Para todas as demais camadas utilizamos versões modificadas do *Palantír*, limitadas a fazer não mais do que o necessário para o teste. Em especial, o processo de análise de protocolos limitou-se ao protocolo eDonkey. Para cada combinação de etapa e configuração foram feitas três execuções do experimento e a média dessas foi utilizada.

3.3.2 Resultados e conclções

O gráfico 3.2 mostra o total de pacotes capturados com sucesso para as quatro configurações testadas e para cada um dos cinco estágios. Na legenda, HT se refere ao uso de HyperThreading no teste enquanto NAPI se refere ao uso ou não de *pooling* de pacotes no *driver* da placa de rede.

Como pode-ser observar, é nítida a diferença de desempenho entre as configurações com e sem PF_RING. Na primeira etapa, a diferença entre as configurações 2 e 4,

respectivamente a pior e a melhor na etapa de captura, chega a 19,1% da quantidade de pacotes no tráfego monitorado. Já na última etapa, a diferença entre as configurações 3 e 4, respectivamente a pior e a melhor na etapa de captura, chega a 83,8% da quantidade de pacotes no tráfego monitorado. Na pior combinação de etapa e configuração dos sistemas com PF_RING a taxa de perda de pacote não ultrapassou 3,32%, quanto na pior combinação sem PF_RING ela atinge 84% e 5,79% na melhor.

Ainda sobre configurações com e sem PF_RING, podemos observar como a quantidade de pacotes capturados decresce abruptamente em todos as configurações sem PF_RING, enquanto que o mesmo não ocorre com o sistemas com PF_RING que tem uma degradação suave à entrada de cada um das etapas. Como pode-se ver, o uso de soluções que minimizem o custo de cópia de pacotes do kernel do sistema operacional para o modo usuário podem oferecer grandes ganhos a sistemas de captura, mesmo que nestes ocorra processamento dos dados capturados, como ocorre no *Palantír*.

Um aspecto interessante aparece analisando o desempenho das configurações com e sem HyperThreading. Nas configurações com PF_RING, a configuração 4, sem HT se, mostra melhor do que a configuração 1, que possui HT. O mesmo comportamento pode ser observado nas duas primeiras etapas das configurações sem PF_RING. Nestas, a configuração 3, sem HT, se mantêm melhor nas fases iniciais. Todavia, a medida que aumenta a taxa de processamento feito com os pacotes (etapas 3–5) a situação se inverte e o rendimento da configuração 3 HT cai abruptamente, enquanto que na configuração 2, com HT, se mantêm melhor. Isso explica-se devido ao fato de que o nas últimas etapas existe um acréscimo do uso de CPU, tanto pelo processo do *Palantír* quanto pelo próprio sistema operacional é exatamente nessas circunstâncias que sistemas com HT demonstram ganhos de desempenho. Nas primeiras etapas, contudo, o que predomína não é o uso de CPU, mas a cópia de memória do dispositivo de rede para a memória principal. Nessas circunstâncias o uso de HT pode degradar o desempenho do sistema.

De qualquer forma, o que podemos concluir com os resultados do experimento é que pode-se sim criar um sistema eficiente de monitoramento passivo de tráfego e com recuperação de estado em tempo real que opere em redes de alta-velocidades. Particularmente o nosso sistema é capaz de monitorar passivamente e recuperar arquivos trocados por sistemas P2P de monitoração de arquivos mesmo em enlaces que cheguem até 500Mbps com uma taxa de perda de pacote da ordem de 3,32%.

Capítulo 4

Monitoração de sistemas P2P de troca de arquivos

Para seja possível realizar um trabalho de análise e de monitoração de uma determinada aplicação é importante que, antes de mais nada, compreendamos essa determinada aplicação, conheçamos ao que ela se destina e obtenhamos alguma informação sobre o seu funcionamento. Nesse capítulo, faremos um rápido apanhado sobre os sistemas P2P existentes. Focaremos as redes P2P de troca de arquivos e seus aspectos mais relevantes ao trabalho desenvolvido. Comentaremos sobre as as especificidades das redes monitoradas e abordaremos o problema particular de monitorar e caracterizar essas redes, apresentando os trabalhos já existentes na literatura. Finalmente, discutiremos as dificuldades que foram enfrentadas durante esse trabalho para tratar os dados obtidos com a monitoração.

4.1 Aspectos gerais de redes P2P

Apesar da falta de consenso sobre o que realmente caracterizaria um "sistema distribuído", muitos concordam que um sistema organizado segundo o modelo cliente-servidor possa ser de fato rotulado como um sistema distribuído [Tanenbaum and Steen, 2001].

Existe uma outra forma de organização de sistemas distribuídos que tem atraído bastante atenção nos últimos anos: os sistemas *peer-to-peer* (P2P). Contudo, similarmente à própria área de sistemas distribuídos, não existe um consenso sobre o que torna ou deixa de ser um sistema P2P. Vários sistemas, até mesmo alguns que claramente seguem o modelo cliente-servidor, foram rotulados como P2P no decorrer dos últimos

anos [Androutsellis-Theotokis and Spinellis, 2004]:

- Sistemas de comunicação e colaboração, tais como ICQ, MSN Messenser, IRC, etc.
- Sistemas de computação distribuída, tais como Seti@Home e Genome@Home.
- Sistemas de Banco de dados distribuídos.
- Sistemas de compartilhamento e de distribuição de arquivos, tais como Gnutella, KaZaa, FreeNet, eDonkey, BitTorrent e outros.
- Sistemas distribuídos para localização e roteamento de informação, tais como CAN, Chord, Kademlia, Pastry, etc.
- Sistemas para criação e manutenção de redes *overlay*.

Conforme a prática comum nessa área, não buscaremos definir formalmente o que caracteriza um sistema como P2P. Basta-nos, para o propósito desse texto, salientar duas características marcantes que esperamos encontrar em sistemas rotulados como P2P:

1. A troca de recursos ocorre fundamentalmente diretamente entre nós similares do sistema.
2. Os nós no sistema possuem uma conectividade bastante instável e variável.

Também não entraremos no mérito de enumerar todos os aspectos envolvidos na concepção de um sistema P2P nem de apresentar detalhadamente cada tipo de sistema P2P existente. No contexto desse texto, interessa-nos apenas observar as redes P2P de troca de arquivos. Em especial, salientaremos apenas os aspectos dessas redes pertinentes aos nossos trabalhos.

4.2 Aspectos de redes P2P de troca de arquivo

Existem diversas redes P2P de troca de arquivo, cada uma com particularidades e peculiaridades próprias. Nessa seção abordaremos os aspectos mais importantes que as diferem ou que essas redes possuem em comum.

No restante desse texto, nos referiremos à essas redes apenas por redes P2P, excetuando-se casos onde seja clara a distinção.

4.2.1 Identificação de recursos

Em um sistema destinado a facilitar a troca de recursos entre os seus usuários, a capacidade de identificar um determinado recurso que se deseja trocar é fundamental.

Na *Web*, usa-se "localizadores de recursos uniformes", também conhecidas como URLs (*Uniform Resourse Locators*), para a identificação de recursos [Berners-Lee, 1994, Berners-Lee et al., 1994]. No caso mais comum, no qual utiliza-se o protocolo HTTP para a recuperação dos recursos, a URL não apenas identifica o recurso, mas também informa a sua localização. Mecanismos similares de identificação de recursos, que atrelam a sua localização ao seu identificador, facilitam o processo obtenção do arquivo pelo cliente e por isso foram bastante utilizados nas redes P2P mais antigas.

Entretanto, essa forma de identificação de recursos é inadequada para sistemas P2P de troca de arquivos. Como a conectividade dos nós nesses sistemas é instável e variável, ao vincular a identificação de um recurso a um determinado nó e ao seu endereço de rede, acaba-se vinculando o tempo durante o qual aquele identificador será válido à disponibilidade do nó.

Além disso, esses mecanismos dificultam ou até mesmo impossibilitam a localização e o uso de réplicas de um recurso. Através do uso de réplicas, um cliente de um sistema P2P pode contornar a instabilidade dos nós que lhe fornecem recursos obtendo o que desejar de nós contendo réplicas.

Para contornar essas limitações, sistemas mais recentes utilizam mecanismos de identificação de recursos que desatrelam a localização de um recurso do seu identificador. Muitos desses mecanismos ou utilizam processos de geração de assinaturas baseadas no conteúdo do recurso para criar identificadores. Como exemplo de tais processos podemos citar algoritmos como CRC32 e algoritmos de *hash* criptográficos como MD4, MD5, SHA-1 e RIPEMD-160, por vezes associados a mecanismos de verificação de conteúdo como *Merkle Trees* [Roos et al.,]. Identificadores criados dessa forma facilitam a descoberta de réplicas de um recurso existentes em diferentes nós do sistema e, para aqueles que usam algoritmos de *hash* criptográfico, ainda permitem verificar a integridade dos recursos obtidos. Todavia, o uso de funções de *hash* criptográfico para esses propósitos é controverso [Henson, 2003].

4.2.2 Modelos de Transferência de Arquivos

Uma vez que a identificação de réplicas de um dado recurso se tornou mais fácil, permitiu-se não somente retomar processos de obtenções de recursos que foram outrora interrompidos devido à indisponibilidade de nós mas também o desenvolvimento de técnicas mais arrojadas e eficientes para obtenção de recursos. Uma dessas técnicas, conhecida por *swarming*, busca acelerar a transferência de recursos através do uso de um grande número de conexões independentes, possivelmente paralelas, para obter partes distintas do recurso a partir de diferentes nós (ou fontes) do sistema.

Nem todas as redes P2P de troca de arquivo existentes usam *swarming*, mas todas as redes mais populares atualmente o fazem, inclusive as duas redes monitoradas no nosso experimento. Por esse motivo, detalharemos o funcionamento, vantagens e desvantagens dos dois modelos mais usados para a realização de *swarming*: o de transferências segmentadas e o de transferências fragmentadas.

4.2.2.1 Tranferência Segmentada

O modelo de transferência de recursos mais comum entre as redes P2P que realizam *swarming* é o de transferência segmentada.

Nesse modelo, um nó cliente especifica que segmento, ou seja, uma uma faixa contínua de dados, de um determinado recurso ele deseja. Em resposta, o nó servidor entrega apenas o segmento solicitado e nada mais. Com essa semântica, fica fácil dotar sistemas que usem o modelo de transferências segmentada de mecanismos de recuperação de transferências interrompidas e da capacidade de realizar *swarming*.

Um dos motivos pelos quais é fácil encontrar sistemas P2P de troca de arquivo que usem esse modelo de transferência é que ele não é usado somente por esses sistemas. Protocolos como o HTTP (através do comando `REST`) e até mesmo o FTP (através da diretriz `Range`) implementavam esse modelo de transferência de arquivos e já existiam ferramentas que realizavam *swarming* antes de que essa capacidade fosse adicionada às primeiras redes P2P de troca de arquivo.

Há que se salientar que, apesar da possibilidade de se solicitar qualquer parte de um recurso, presume-se, nesse modelo, que o nó que o disponibilizará já o obteve em sua totalidade antes de anunciá-lo.

4.2.2.2 Transferências Fragmentadas

O suporte a transferências segmentada é relativamente simples de ser implementado. Todavia, o fato de um determinado nó não poder compartilhar um recurso até que ele o tenha em sua totalidade constitui uma grande limitação desse modelo, pois priva nós que não possuam o recurso completo de participarem do processo de disseminação do conteúdo de um recurso. Para contornar esta limitação, outras redes P2P tais como a eDonkey [edo, 2004] e BitTorrent [Cohen, 2003] usam o modelo de transferência fragmentada.

Nesse modelo, um recurso é dividido em *fragmentos*, segmentos consecutivos de igual tamanho. Tão logo um nó tenha obtido um desses fragmentos, ele poderá disponibilizá-lo para outros nós. Para saber quais fragmentos de um recurso um nó pode solicitar a outro, quando do estabelecimento de cada nova conexão, os nós trocam entre si listas informando quais fragmentos cada um possui. Cabe ao nó cliente escolher quais fragmentos de cada um do nós aos quais ele se conectar lhe interessam.

Por diminuir o tempo necessário para que um nó possa participar da rede de disseminação de um determinado recurso, esse modelo de transferência, de uma perspectiva global, aumenta a quantidade de nós que podem servir um recurso, diminuindo assim o tempo necessário para a sua disseminação pela rede [Qiu and Srikant, 2004]. Por outro lado, incorre-se num número elevado de conexões estabelecidas entre nós, sendo que muitas dessas conexões ocorrem apenas para troca da lista de fragmentos possuídos. Essa última característica é um dos motivos pelos quais é mais difícil monitorar redes que usem esse modelo de transferência de arquivos.

4.2.3 Organização da rede e localização de recursos

Central a praticamente todas as redes P2P de troca de arquivo está o problema de localização de recursos. Nesse contexto, compreende-se por "localização de recursos" todo o processo que antecede a transferência do mesmo e que, de forma simplificada, pode ser entendido como possuindo duas fases distintas:

- a localização *fontes*, nós no sistema que possuem um determinado recurso e
- o processo de descoberta de recursos que possam interessar ao usuário, através da busca por meta-informações desses recursos, tais como seus nomes, seus tamanhos, fragmentos disponíveis, etc.

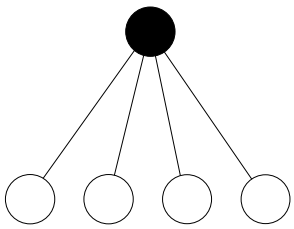

Figura 4.1: Topologia de um sistema centralizado

Apesar de distintas, ambas as fases podem ocorrer simultaneamente em redes onde não existe uma desvinculação entre a localização de um recurso e a sua identificação. Nessas redes, a busca por meta-dados já retorna, para cada recurso encontrado, o endereço de suas fontes.

De qualquer forma, juntas ou separadas, ambas as fases do problema enfrentam o mesmo desafio: localizar dados em um sistema distribuído onde os nós possuem uma conectividade bastante variável e instável. Pode-se argumentar que a evolução das redes P2P de troca de arquivos deve-se a uma busca por maneiras mais eficientes de lidar com esse desafio.

4.2.3.1 Sistemas centralizados

A forma mais intuitiva de abordar esse problema é contornando-o, mudando seu paradigma para outro no qual esse problema já é bem conhecido, ou seja, transformar o problema de localizar dados em um sistema distribuído no problema de localizar dados em um sistema centralizado. Sistemas construídos utilizando essa abordagem são conhecidos como *sistemas centralizados* e tem a rede Napster como o seu representante mais conhecido. [Balakrishnan et al., 2003]. Sua estrutura pode ser observada na figura 4.1.

Nesses sistemas, servidores mantêm bancos de dados com índices dos meta-dados dos recursos disponibilizados pelos seus clientes. Ao entrarem na rede, nós clientes organizam uma lista contendo informações sobre todos os arquivos que desejam compartilhar e a repassam para o servidor ao qual se conectaram. Esse servidor, por sua vez, atualiza então seus índices de meta-dados e de localização de recursos. O problema de busca na rede transforma-se, para os clientes na rede, no envio de uma requisição de busca para o servidor e, para este, numa simples consulta a um banco de dados.

Como se pode observar, esses sistemas são claramente moldados em torno do modelo cliente-servidor. Contudo, é importante salientar que a única função dos servidores é de agir como diretórios centrais que auxiliam clientes na localização de recursos. A

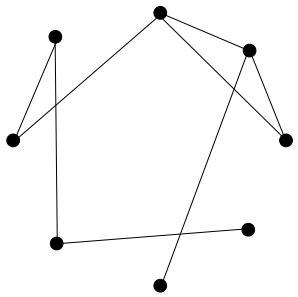

Figura 4.2: Topologia de um sistema descentralizado desestruturado

troca destes recursos ocorre apenas entre nós clientes do sistema, um marco que ajuda a caracterizar esses sistemas como P2P.

Também há que se ressaltar que, apesar da facilidade com a qual pode-se conceber sistemas centralizados, esses sistemas são vulneráveis à censura, ações legais, ataques maliciosos e falhas técnicas, além de serem inerentemente não-escaláveis. Em suma, a centralização, apesar de resolver o problema de localização, cria um ponto único de falha: o servidor [Androutsellis-Theotokis and Spinellis, 2004].

4.2.3.2 Sistemas decentralizados desestruturados

A abordagem tradicional para melhorar a escalabilidade de um sistema cliente-servidor é através da sua hierarquização. No contexto do problema de localização, podemos citar o DNS como exemplo de um sistema que adota essa tática [Balakrishnan et al., 2003].

Todavia, na evolução das redes P2P de troca de arquivo, pode-se observar que a abordagem preferencialmente adotada para lidar com os problemas existentes com o modelo centralizado consistiu em operar um rompimento radical com o paradigma cliente-servidor e tornar o processo de localização de recursos completamente descentralizado. Ao invés de construir um sistema composto de clientes e servidores, optou-se pela elaboração de sistemas onde todos os nós seriam potencialmente tanto clientes como servidores no processo de localização de recursos, às vezes realizando ambas as funções ao mesmo tempo. A organização da estrutura da rede também não seria coordenada por elementos centralizadores. Cada *servent*, como são conhecidos os nós em tais sistemas, ao entrarem na rede, estabelecem conexões com outros nós previamente conhecidos ou descobertos através do sistema, formando uma malha auto-organizada mas desestruturada de nós [Ripeanu, 2001]. Sistemas construídos usando essa abordagem ficaram conhecidos como *sistemas descentralizados desestruturados*, e pode-se obter uma idéia de como é a topologia de uma rede que segue esse modelo observando

a figura 4.2. Como representante maior de tais sistemas podemos citar a rede Gnutella.

Nesses sistemas, os mecanismos de busca são não-determinísticos e operam basicamente através de força-bruta: inundação, busca em largura e busca em profundidade. A rede Gnutella, por exemplo, utiliza inundação: um nó envia uma mensagem de busca para os seus vizinhos que, após procurarem nos seus próprios índices de meta-dados, repassam essa mensagem para os seus vizinhos imediatos, e assim por diante. Respostas são roteadas de volta pelos mesmos caminhos através dos quais as suas respectivas mensagens de busca viajaram. Para evitar que mensagens já vistas sejam repassadas à rede novamente, cada mensagem possui um identificador global (GUID) que será único com grande probabilidade. Além disso, para limitar o número de nós para os quais uma mensagem é repassada, cada mensagem é dotada de um campo contendo seu tempo de vida ou TTL (*time-to-live*). Cada vez que uma mensagem é repassada de um nó para outro, esse TTL é decrementado e quando esse alcançar 0 a mensagem é descartada [Markatos, 2002].

A completa descentralização tanto da topologia da rede como dos mecanismos de localização desses sistemas lhes trouxe várias vantagens frente aos centralizados. Em especial podemos citar a remoção de um ponto único de falhas, a sua estrutura altamente escalável em termos de topologia e a sua capacidade de suportar uma grande variabilidade nas taxas de entrada e saída de nós do sistema.

A descentralização, contudo, não vem sem efeitos colaterais. Se por um lado a topologia da rede estruturalmente escala, seu mecanismo de busca por inundação claramente apresenta problemas de escalabilidade, por tender a gerar um número exponencial de mensagens e, portanto, um grande volume de tráfego devido apenas à "sinalização" entre nós. Apesar desse problema ser amenizado pelo adoção de TTLs e GUIDs nas mensagens, esses recursos acabam acarretando a segmentação da rede, impondo para cada usuário um "horizonte de busca" a partir do qual suas buscas não ultrapassariam. Por outro lado, sem o uso de TTLs a rede sucumbiria devido ao número exponencial de mensagens geradas pela busca. [Androutsellis-Theotokis and Spinellis, 2004].

4.2.3.3 Sistemas descentralizados semi-estruturados

Para contornar essas limitações sem perder todos os ganhos obtidos pela descentralização, uma alternativa bastante popular entre a redes P2P de troca de arquivo foi a de lançar mão daquela que, como já mencionado, é tida como a alternativa mais conhecida para se lidar com problemas de escalabilidade em sistemas distribuídos: hierarquização. A idéia é a de construir uma rede que em sua essência seria similar às redes descentra-

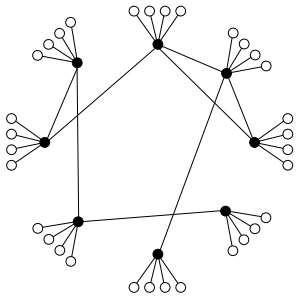

Figura 4.3: Topologia de um sistema descentralizado semi-estruturado

lizadas e desestruturadas mas que possuísse um mecanismo de busca mais escalável e que tirasse proveito da heterogeneidade existente entre os nós [Chawathe et al., 2003].

Esses sistemas, que podem ser entendidos como híbridos entre sistemas centralizados e sistemas descentralizados desestruturados, organizam a rede em uma hierarquia de dois níveis, como pode ser observado na figura 4.3:

Super-nós: nós mais poderosos e com maiores responsabilidades para com a rede.

Nós ordinários: nós comuns, que não participam diretamente do processo de organização da estrutura e do roteamento de mensagens da rede.

A interação entre nós ordinários e super-nós é bastante similar à interação entre um cliente e um servidor em uma rede centralizada. Ao entrar na rede, todo nó ordinário estabelece uma conexão com um super-nó e lhe entrega uma lista com meta-dados dos recursos que ele deseja compartilhar. Todas as buscas e demais interações que um nó ordinário desejar realizar na rede são feitas através de solicitações ao seu super-nó. Nós ordinários que possuírem bastantes recursos computacionais e de rede disponíveis podem ser promovidos a super-nós. Entre os super-nós, no entanto, as interações transcorrem de maneira similar àquelas de uma rede descentralizada e desestruturada como a Gnutella.

Sistemas elaborados dessa forma são conhecidos como *sistemas descentralizados e semi-estruturados*. Como representante mais conhecido desse modelo de rede P2P de troca de arquivo temos a rede KaZaa [kaz, 2004], onde os super-nós são chamados de *super-nodes*. Existe também uma extensão da rede Gnutella que a torna uma rede descentralizados semi-estruturada. Nessa rede, os super-nós são chamados de *ultra-peers* [Singla et al., 2003].

Ao organizar a rede dessa forma, os sistemas descentralizados semi-estruturados conseguiram manter várias das vantagens das redes descentralizadas e ainda remover

alguns dos seus problemas. Primeiro, a hierarquização permitiu retirar do interior da rede nós que possam ser possíveis gargalos e permitiu fazer de forma mais transparente e eficiente o *caching* de buscas anteriores. Além disso, reduziu-se o tempo necessário para a descoberta de recursos e o custo de fazê-lo, sem que haja uma degradação no desempenho da mesma [Androutsellis-Theotokis and Spinellis, 2004, Benevenuto et al., 2005]. Apesar disso, um problema ainda persiste nessas redes: a busca continua sendo não-determinística, ou seja, não existem garantias de encontrar um determinado recurso no sistema, mesmo que ele sabidamente exista. Esse problema manifesta-se de forma mais evidente para arquivos raros, tendo efeito mínimo em arquivos bastante populares [Chawathe et al., 2003, Loo et al., 2004].

4.2.3.4 Sistemas descentralizados estruturados

Para que fiquem mais claras as implicações do não-determinismo das buscas em redes descentralizadas e destruturadas, presentes também nas redes descentralizadas semi-estruturadas, pode-se dizer que para se realizar uma busca de maneira confiável com o seu mecanismo de localização de recursos, seriam necessárias $O(n)$ mensagens — um custo exorbitante para uma busca em uma rede P2P com milhões de nós [Chawathe et al., 2003]. O fato é que mesmo que se desejasse pagar por esse custo ainda assim não seria possível fazê-lo nas redes atuais devido ao uso de recursos de prevenção contra o uso excessivo de largura de banda tais como TTL.

Todavia, existem sistemas P2P descentralizados capazes de lidar com o problema de localização de recursos de maneira determinística e com custos na ordem de $O(\log n)$ mensagens. Esses sistemas conseguem dar essas garantias de custo e de confiabilidade através da manutenção de uma topologia lógica formada pelos seus nós e são, devido a isso, chamadas de *redes descentralizadas estruturadas*.

Para seus usuários, esses sistemas apresentam-se como tendo uma interface similar àquela de uma tabela de dispersão (*hash table*) [Szwarcfiter and Markenzon, 1999, Cormen et al., 1989]. De forma simplificada, eles permitem o mapeamento de identificadores de recursos a nós no sistema que ficariam responsáveis por armazenar dados sobre esses recursos. Devido à sua natureza distribuída, esses sistemas são conhecidos como "tabelas de dispersão distribuídas" ou pela sigla DHTs, do inglês *Distributed Hash Tables*. Como exemplo de tais sistemas podemos citar o Chord [Stoica et al., 2001], o CAN [Ratnasamy et al., 2001] e o Kademlia [Maymounkov and Mazières, 2002]. As estratégias usadas para estruturar a rede e garantir os custos esperados e a confiabilidade em cada uma dessas redes varia muito, sendo comentada de forma apropriada em ou-

tros trabalhos [Balakrishnan et al., 2003, Androutsellis-Theotokis and Spinellis, 2004]. A despeito das diferenças em suas estratégias, todos apresentam a abstração de uma DHT.

A forma de anunciar recursos e de procurar por estes em sistemas P2P de troca de arquivo baseados em DHTs é relativamente diferente daquela dos sistemas anteriores. Quando um nó deseja anunciar algum recurso ele precisa localizar, utilizando o identificador do recurso, o nó responsável pelo armazenamento deste. Determinado esse nó, solicita-se que então que este armazene a informação de que o nó de origem da requisição está compartilhando aquele dado recurso, ou seja, que ele é uma fonte desse recurso. A busca transcorre de forma similar: localizado o nó responsável por armazenar dados do identificador em questão, solicita-se a ele uma lista de fontes do recurso com tal identificador.

Apesar de resolverem o problema da confiabilidade de busca presentes em sistemas descentralizados desestruturados, nos semi-estruturados e em sistemas distribuídos em larga escala de maneira geral, muitos argumentam que sistemas baseados em DHTs não são uma panacéia. Apesar de serem uma solução escalável para o problema de busca exata, DHTs não são apropriadas para as buscas por palavras-chaves, tão freqüentes em sistemas P2P de troca de arquivo [Chawathe et al., 2003]. A construção de índices invertidos de palavras-chaves sobre uma DHT também é controversa, apesar de já ser utilizada em sistemas de troca de arquivo tais como o Kad [Loo et al., 2004, emu, 2005]. Todavia, já existem propostas pra construção de sistemas capazes de lidar com buscas complexas construídas sobre a abstração fornecida por DHTs [Harren et al., 2002]. A volubilidade da conectividade de nós de clientes dessas redes P2P também é encarada como um problema, devido ao custo que sistemas que usassem DHTs teriam que pagar para manter as estruturas da topologia dessa redes, requeridas para o bom funcionamento delas. Existem trabalhos que argumentam a favor da construção de sistemas híbridos, obtendo o bom desempenho de redes descentralizadas desestruturadas para buscas por recursos populares e o bom desempenho de sistemas baseados em DHTs para busca pelos pouco populares [Loo et al., 2004].

4.2.3.5 Outros sistemas e abordagens

Nas subsessões anteriores apresentamos várias abordagens para organizar a estrutura de uma rede P2P de troca de arquivo para lidar com o problema de localização de recursos. Apesar de termos visto todos os modelos mais relevantes, ainda existem outros modelos dignos de menção. Todavia, esse texto não tem como propósito fazer um

estudo aprofundado sobre todas as redes P2P de troca de arquivo existentes e suas respectivas abordagens para lidar com o problema de localização de recursos. Existem bons trabalhos que, além desses aspectos, discorrem sobre muitos outros tópicos referentes a sistemas P2P no geral [Androutsellis-Theotokis and Spinellis, 2004].

4.3 As redes monitoradas

Para poder validar o nosso ambiente de monitoração, era necessário colocá-lo em execução em um ambiente real, capturando e analisando o tráfego de redes P2P reais e significativas. Observou-se que, no provedor de acesso à Internet de banda larga onde instalamos o *Palantír*, as duas redes P2P mais populares e que mais geravam tráfego eram a KaZaa e a eDonkey.

Há de se observar que esses dois sistemas P2P são bem diferentes entre si, tanto no que se refere às estruturas de suas respectivas redes bem como à forma com que arquivos são localizados e transferidos. A seguir, detalharemos cada uma dessas redes, observando as suas peculiaridades e o impacto destas no processo de elaboração e de execução da arquitetura de monitoração. A necessidade de suportar ambas as redes na arquitetura fez com que a deixássemos flexível o suficiente para acomodar várias outras redes além dessas duas.

4.3.1 Kazaa

Após o fechamento dos servidores da rede Napster devido a uma ação legal, o Kazaa [kaz, 2004] se tornou o sistema P2P de troca de arquivos mais popular. Apesar de ser apenas o cliente mais conhecido da rede FastTrack, da qual também fazem parte clientes como o iMesh e Grokster, a popularidade daquele programa fez com que seu nome se tornasse sinônimo para a própria rede. Essa rede também despertou bastante interesse na comunidade científica tanto pelas proporções que a sua base de usuários atingiu com por ter sido a primeira a aproveitar a heterogeneidade de seus clientes em proveito próprio, sendo pioneira na utilização o modelo de sistemas descentralizados semi-estruturados.

A rede FastTrack utiliza protocolos proprietários cujas licenças de uso são controladas pela Sharman Networks. Nesta rede o protocolo usado para buscas, sinalização e comunicação entre super-nós e entre nós e seus super-nós é cifrado. Apesar disso, a partir dos esforços de vários grupos que realizaram engenharia reversa desse protocolo e dos

formatos dos arquivos utilizados pelo cliente KaZaa, foi possível adquirir informações mais precisas sobre as estruturas e funcionamento desse protocolo, bem como a criação de ferramentas auxiliares ao uso da rede e a criação de clientes não oficiais. Além disso, esses esforços possibilitaram a realização de trabalhos de buscava por informações mais detalhadas sobre a estrutura e o funcionamento dessa rede, tanto no que diz respeito ao processamento de buscas quanto à organização e a interação de seus nós e super-nós. Dentre esses trabalhos podemos destacar o de Liang [Liang et al., 2004], que também fornece boas referências sobre vários outros aspectos não documentados dessa rede.

Apesar das dificuldades que o protocolo de busca e sinalização apresenta para a sua monitoração, as transferências nessa rede não são cifradas e ocorrem utilizando uma especialização do protocolo HTTP. Nesse protocolo, cabeçalhos especiais são utilizados para repassar meta-dados sobre o recurso trocado. Tais cabeçalhos são comumente utilizados para distingüir tráfego KaZaa de tráfego *Web*. Além desses cabeçalhos, esse protocolo adiciona ao protocolo HTTP o comando `GIVE`, utilizado para permitir a obtenção de arquivos de nós que estejam atrás de *firewalls*.

O uso do protocolo HTTP e a existência dos cabeçalhos específicos poderiam levar alguém a pensar que a monitoração do tráfego dessa rede é fácil. Todavia, esse não é *mais* o caso. Clientes dessa rede, na tentativa de burlar *firewalls*, passaram a utilizar portos de comunicação diferentes daquele que era tradicionalmente usado para transferências KaZaa, o porto 1214 [Karagiannis et al., 2004a, Sen et al., 2004]. Em 2003, pelo menos 38% do tráfego Kazaa de um grande provedor israelense ocorria em portos diferentes do tradicional [Leibowitz et al., 2003], o que indica que análises de tráfego baseadas apenas em portos de comunicação teriam problemas para caracterizar tráfego dessa rede. A despeito disso, podemos encontrar na literatura alguns trabalhos de caracterização da carga gerada por essa rede [Gummadi et al., 2003, Leibowitz et al., 2003].

O KaZaa usa o modelo de transferência segmentada. Um recurso nessa rede é identificado através do seu *ContentHash*, um identificador que facilita a localização e identificação de suas cópias, criado através de um processo para geração de assinatura de recursos baseada no conteúdo destes denominado de *UUHash*. Este processo usa apenas algumas amostras do conteúdo do recurso para gerar seu identificador. Apesar de ser bastante rápido, esse mecanismo facilita a criação e distribuição de cópias falsas ou "poluídas" de um arquivo. A indústria de música e vídeos tem se aproveitado bastante dessa fraqueza para sabotar a rede: mais de 50% das cópias das músicas mais populares nessa rede são "poluídas" [Liang et al.,]. A identificação de usuários usa

um esquema ainda mais precário, consistindo tão somente de *usernames*, o que torna difícil distinguir com precisão usuários caso seus *usernames* sejam idênticos.

4.3.2 eDonkey

A rede eDonkey vem se popularizando nos últimos tempos e já é responsável por uma boa parcela do tráfego P2P não somente do provedor que monitoramos como em vários outros, como comentado por Tutschku e por Sen [Tutschku, 2004, Sen et al., 2004]. Ela destaca-se por ser a primeira rede que emprega o modelo de transferências fragmentadas de recursos a ser largamente utilizada.

Originalmente uma rede com apenas um cliente oficial e com um protocolo proprietário e fechado, de maneira similar ao que o ocorreu com o KaZaa, o protocolo da rede sofreu engenharia reversa. Ironicamente, o cliente mais popular para essa rede atualmente, o eMule [emu, 2005], é fruto desse trabalho de engenharia reversa e possui seu código aberto e coberto por uma licença *copyleft*. Muitos outros clientes foram construídos com base no que foi descoberto do protocolo e no código do eMule, entre os quais podemos citar o xMule, aMule, Shareazaa, lphant e mldonkey.

A rede eDonkey segue o modelo centralizado, similar ao do Napster. Todavia, diferentemente deste, qualquer pessoa pode criar um novo servidor. Além disso, apesar de cada cliente estar conectado a apenas um servidor através de uma conexão TCP, as buscas nessa rede podem ser feitas a todos os servidores conhecidos através de UDP, o que possibilita a ampliação do horizonte de busca disponível para cada cliente. Além disso, alguns clientes eDonkey atuais, como o eMule e o cliente eDonkey oficial, também possuem redes descentralizadas estruturadas, conhecidas respectivamente como Kad e Overnet, ambas baseadas na DHT Kademlia [Bhagwan et al., 2003, Maymounkov and Mazières, 2002]. Estas redes estruturadas, todavia, não são utilizadas atualmente como substitutas à rede centralizada, mas como redes complementares para a realização de buscas e de divulgação de recursos disponíveis para troca.

Arquivos nessa rede são identificados através dos seus *FileIDs*, identificadores de arquivo criados utilizando-se um esquema baseado no algoritmo de *hash* criptográfico MD4 e que leva em consideração todo o conteúdo do recurso. Apesar de existirem ataques conhecidos para o algoritmo MD4, em comparação com o esquema empregado pelo KaZaa, esse mecanismo possibilita a criação de identificadores que tornam a dissipação de cópias-falsas muito menos provável [Henson, 2003]. Além de identificar unicamente recursos na rede, *FileIDs* também são utilizados para a averiguação da

integridade do recurso que eles identificam. Além dele, um identificador secundário também é utilizado para permitir a recuperação de partes de um recurso que, durante o processo de obtenção, acabaram mostrando-se corrompidas [Hoßfeld et al., 2004]. Os identificadores de usuários são criados de maneira similar aos GUIDs utilizados pelo Gnutella, diminuindo consideravelmente as chances de colisões de identificadores e facilitando a distinção de suas conexões e portanto a identificação e monitoração de tráfego eDonkey específico de um dado usuário.

Como já dito, essa rede foi uma das pioneiras a adotar o modelo de transferência fragmentada de recursos, o que, juntamente com o uso de identificadores desvinculados à localização dos seus respectivos recursos, dá a todo o processo composto das etapas de busca, localização de fontes e obtenção de recursos um aspecto bem particular. Enquanto em outras rede todo o processo pode ser bem caracterizado nessas três etapas, nessa rede ele se desenvolve em cinco etapas distintas [Hoßfeld et al., 2004]:

1. Busca a partir de meta-dados

   No eDonkey, o resultado de uma busca é apenas uma lista de identificadores (*FileIDs*) de arquivos que atendem os critérios dessa busca.

2. Localização de fontes do recurso escolhido

   Escolhido um arquivo dessa lista, é necessário realizar uma segunda busca, desta vez por fontes desse arquivo.

3. Descoberta de fragmentos disponíveis nas fontes

   Devido ao emprego de transferências fragmentadas, cada nó descoberto na etapa anterior deverá ser contactado a fim de que se possa ter uma idéia da disponibilidade de cada fragmento desse arquivo. Durante essa etapa pode-se descobrir outros nós que também estejam obtendo esse recurso, até mesmo nós de servidores desconhecidos e de outras redes (Overnet, por exemplo), através de um recurso de troca de nós chamado *SourceExchange*.

4. Entrada nas filas de espera

   Após descobrir os fragmentos cada fonte encontrada dispõe, é preciso solicitar os fragmentos desejados àquelas que o possuam. Contudo, esses fragmentos não serão entregues imediatamente: cada solicitação recebida por uma fonte é enfileirada e será atendida quando esta chegar ao topo dessa fila de espera (*upload queue*). O tempo de espera, transações anteriores realizadas entre essa fonte o

nó requisitante e outros fatores influenciam como cada requisição progride nessa fila.

5. Solicitação por fragmentos de interesse

   Finalmente, a medida que atinge-se o topo da fila de espera de diferentes fontes obtêm-se de cada um delas a permissão para se obter imediatamente partes dos fragmentos desejados. Ao final do processo, o conjunto dos diferentes fragmentos obtidos permitirá a obtenção do recurso completo.

Esse processo difere até mesmo da rede BitTorrent, o único outro sistema P2P de troca de arquivos popular a utilizar o modelo de transferência fragmentada. Outra peculiaridade dessa rede é que, durante a transferência, partes de cada fragmento podem ser compactadas em tempo real pela fonte do recurso, com o intuito de diminuir a quantidade da dados que será transmitida (e recebida). Apesar de útil, há de se considerar que, uma vez que vasta maioria dos recursos trocados nessas redes consiste em músicas no formato MP3 e em vídeos, os parcos ganhos obtidos com esse mecanismo podem não justificar o custo em CPU para realizar tal compressão. Além disso, esse mecanismo pode ser encarado como um complicador para sistemas de monitoramento com recuperação de estado para esse tipo de rede.

Ao contrário do KaZaa e de outros protocolos P2P, o tráfego eDonkey ocorre, em sua maior parte, nas suas portas conhecidas [Sen et al., 2004]. Isso facilita o processo de monitoramento do seu tráfego. Por outro lado, o monitoramento dessa rede apresenta alguns desafios. Primeiro, transferências completas de recursos nessa rede são relativamente longas se comparadas ao tempo necessário para realizar o mesmo em outras redes, o que pode ser atribuído parcialmente ao o enfileiramento dos pedidos por fragmentos. A associação disso ao *swarming* realizado para obter os arquivos constitui um bom desafio para aqueles interessados em monitorar transferências entre nós dessa rede.

4.4 Monitoração de Redes P2P com o *Palantír*

No capítulo 3 descrevemos a arquitetura proposta para a monitoração passiva de tráfego em tempo real com recuperação de estado: o *Palantír*. Comentamos também sobre o custo de se realizar o monitoramento de tráfego P2P sobre essa arquitetura. Como explicado, tal feito foi possível através da adição de duas novas camadas à arquitetura,

responsáveis respectivamente pela análise e decodificação do tráfego de cada rede monitorada e pela remontagem dos arquivos trocados nessa rede a partir das conexões monitoradas.

Nessa seção final comentaremos sobre as dificuldades e surpresas encontradas durante a concepção dessas camadas, como que as particularidades de cada uma das duas redes rede influenciou no desenvolvimento de seus respectivos monitores e sobre compromissos aos quais ficamos sujeitos devido às abordagens adotadas.

4.4.1 Identificação de tráfego

Nosso trabalho não objetiva a identificação de tráfego, comentada na seção 2.2.1. Por essa razão e por motivos de eficiência, optamos por monitorar apenas o tráfego de cada uma das redes P2P escolhidas que ocorresse no seus portos tradicionais.

Essa abordagem sabidamente limita o volume de tráfego KaZaa que poderá ser monitorado mas ainda assim permitirá verificar a eficácia da ferramenta na monitoração do tráfego dessa rede uma vez que uma parcela representativa dele ocorre no seu porto tradicional. Já no caso da rede eDonkey, como a maior parcela de seu tráfego ocorre no seu porto tradicional, essa abordagem ainda permitirá o monitoramento da maior parte do tráfego dessa rede.

4.4.2 Interpretação do protocolo

No início da elaboração da arquitetura cogitou-se a viabilidade de monitorar tanto o tráfego de sinalização das redes escolhidas quanto o tráfego devido a transferências. Devido à dificuldade em encontrar na época documentação suficiente sobre o protocolo de sinalização do KaZaa e sobre o seu mecanismo de cifragem, optou-se por monitorar apenas o tráfego devido à transferência de recursos em ambas as redes.

Desta forma, a implementação de um monitor para transferências de recursos na rede KaZaa limitou-se a especialização de um monitor de transferências HTTP que contivessem os cabeçalhos especiais usados nessa rede. O desenvolvimento desse monitor ficou a cargo do então aluno de mestrado Bruno Grossi.

Enquanto que o HTTP usado pelo KaZaa possui uma natureza textual, o protocolo da rede eDonkey, usado tanto para sinalização e para transferências, é basicamente composto de várias estruturas C orientadas à arquitetura Intel 32 bits. Esse fato tornou a concepção desse monitor relativamente mais complicada. Apesar da existência de documentos descrevendo o formato dessas estruturas, não foi raro encontrarmos

estruturas das quais não era conhecido nem documentado seus valores semântico no protocolo. A leitura do código fonte de clientes eDonkey, em especial o do eMule, apesar de laboriosa devido a tais códigos serem pouco comentados, sanava essas deficiências das documentações.

4.4.3 Remontagem de arquivos

Acima dos monitores de tráfego das redes P2P escolhidas está a última camada da arquitetura, responsável pela remontagem de arquivos, ou seja, responsável por extrair, com auxílio dos monitores, pedaços dos arquivos que estivessem sendo trocados no tráfego monitorado com o intuito de, ao juntá-los, obter cópias completas desses recursos.

Apesar de não ser estritamente necessária para uma arquitetura de monitoramento de tráfego de redes P2P de troca de arquivos, a remontagem de arquivos era fundamental para a averiguação de localidade de referencia entre os recursos trocados nas duas redes. Além isso, a remontagem era peça fundamental para um sistema de *caching* de tráfego P2P que, parte de um outro trabalho que acabou culminando nesse.

É interessante observar que, a despeito da divergência de modelo de transferência de recursos entre as duas redes, no que diz respeito à remontagem de arquivos, ambas as redes e seus respectivos monitores funcionam de maneira similar sob o ponto de vista da camada de remontagem: ambas lhe passam um conjunto de octetos e lhe informam de qual recurso esses octetos fazem parte e a posição deles no mesmo.

A medida que esses octetos são repassados para a camada de remontagem, é necessário determinar se eles já foram armazenados para que, no caso negativo, tais octetos sejam armazenados em disco. Isso é necessário tanto para melhorar o desempenho, evitando que octetos já vistos sejam regravados no disco, desperdiçando tempo e recursos, como pra determinar se um determinado recurso já fora remontado completamente ou não.

Para que o controle de octetos já gravados fosse o mais eficiente possível, elaboramos uma estrutura chamada "mapa de intervalos", implementada utilizando uma árvore rubro-negra com costuras [Szwarcfiter and Markenzon, 1999] e que permite verificar se um determinada faixa de dados no recurso fora gravada anteriormente ou não a um custo $O(\log n)$. *Carece definir formalmente um mapa de intervalos?*

Determinar quando um certo recurso já fora remontado completamente é mais problemático. Apesar do tamanho do recurso trocado ser informado em transferências entre nós da rede KaZaa, o mesmo não ocorre na rede eDonkey. Esse fato torna ne-

cessário validar toda e qualquer requisição de adição de octetos a um recursos no mapa de intervalos desse último, mesmo para arquivos que na prática já estão completos.

4.4.4 Identificação de arquivos e usuários

A falta de um identificador comum inter-redes faz com que adote-se, como identificar de um dado recurso para a camada de remontagem, o seu identificador na rede da qual um dado segmento seu foi recuperado. Na prática, isso significa que um dado recurso sofrendo remontagem não poderá sofrer acréscimos de octetos das duas redes. O uso do nome do recurso ao invés de seu identificador nas várias redes poderia ser visto como uma forma de contornar essa limitação, uma vez que ele é informado nas transferências de ambas as redes. Entretanto, o uso de nome de recursos isoladamente não possibilita distinguir recursos com precisão suficiente.

A identificação de usuários pode ser feita tanto através de seus endereços IPs como dos identificadores usados duas redes.

Apesar de servir à ambas as redes, o uso de endereços IPs com o propósito de identificar usuários é falho pois, isoladamente, não permite a identificação de usuários que mudem de endereços IPs ou que estejam atrás de *firewalls* com NAT.

Como mencionado anteriormente, ambas as redes usam algum mecanismo para identificar seus usuários. No caso da KaZaa, esses identificadores são criados manualmente pelo usuário e podem gerar bastante colisão. No entanto, utilizamo-os mesmo assim, de maneira similar a outros trabalhos [Gummadi et al., 2003, Leibowitz et al., 2003] Já na rede eDonkey com seus identificadores gerados de maneira similar à dos GUIDs do gnutella, a probabilidade de colisão de identificadores é bem reduzida, o que torna esses identificadores perfeitos para identificar usuários, permitindo distinguir até aqueles a que estiverem atrás de um *firewall* com NAT [Bhagwan et al., 2003].

Capítulo 5

Caracterização do Tráfego P2P

Como mencionado na introdução, utilizando-se do *Palantír*, realizou-se uma caracterização do tráfego KaZaa e eDonkey de um provedor de acesso a Internet de banda larga.

Esse trabalho de caracterização serviu a três propósitos: verificar se as características da carga desses sistemas P2P condiziam com o encontrado em trabalhos anteriores, validar o *Palantír* como plataforma e a sua implementação e observar a localidade de referência entre diferentes redes P2P.

5.1 O ambiente e a coleta

A coleta foi realizada por por um período de 10 dias. O provedor monitorado possuía na época da coleta 6000 clientes sendo que monitorou-se o tráfego de apenas um quarto deles.

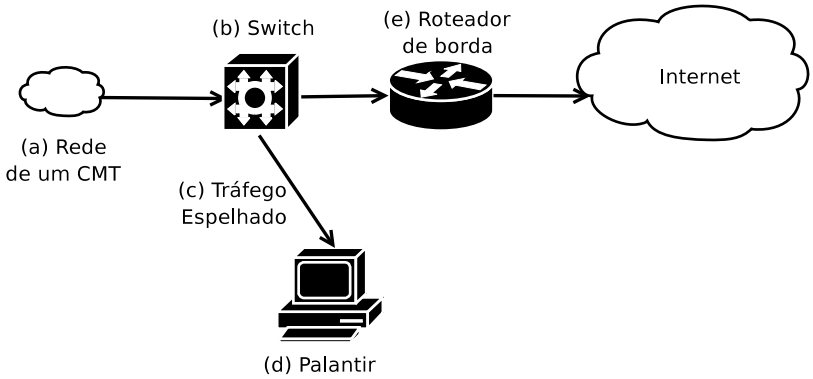

Figura 5.1: Diagrama do ambiente de coleta

Rede P2P	Kazaa	Edonkey
Intervalo	2004/10/18-28	2004/10/18-28
Bytes transferidos	1.644.589.908	175.419.189.687
Requisições	8.490	59.324
Recursos únicos	3.042	8.835
Sessões	5.512	53.020
Usuários únicos	4.388	48.206

Tabela 5.1: Estatísticas gerais observadas

Nesse estudo de caso, o tráfego monitorado era espelhado para uma máquina onde o *Palantír* fora instalado e onde esse tráfego era analisado e caracterizado em tempo real, de maneira similar ao descrito na seção 2.1.2.1, também ilustrado pela figura 5.1.

A máquina de coleta consistia em um microcomputador PC com processador AMD Athlon XP 1500+, *chipset* nVidia, 768 Mbytes de memória RAM. Foram utilizadas duas placas de redes ethernet, sendo uma para gerência da máquina e outra, uma Intel e1000, para o recebimento do tráfego espelhado. O sistema operacional utilizado era o GNU/Debian 3.0 rodando uma versão modificada do Linux 2.4.26.

5.2 Estatísticas Gerais

Nessa seção apresentamos algumas estatísticas gerais obtidas para ambos os protocolos de forma a quantificarmos as suas diferenças. Deve-se observar que o intervalo de coleta para ambas as redes foi o mesmo, bem como a população potencial de usuários do provedor em ambas as redes. Analisando a tabela 5.1, observa-se que o tráfego eDonkey é significativamente maior do que o tráfego KaZaa. Uma das razões por trás disso deve-se ao fato de que o tráfego KaZaa encontra-se mais distribuído em vários portos, ao contrário do tráfego eDonkey, que encontra-se bastante concentrado no seu porto tradicional, o 4662. Outro fator de deve ser levedo em consideração é que o tamanho médio dos recursos na rede eDonkey é maior do que o encontrado na KaZaa, como veremos a seguir.

5.3 Métricas e Metodologia de Caracterização

Empregou-se uma metodologia de caracterização hierárquica, onde os relatórios resultantes do *Palantír* foram analisados em quatro níveis:

Fragmentos: Analisa-se os efeitos de segmentação e/ou fragmentação no processo de obtenção dos arquivos.

Recursos: Nesse nível caracteriza-se os recursos solicitados no tráfego monitorado, desconsiderando efeitos de fragmentação.

Sessão: Definimos sessão como uma seqüência consecutiva de solicitações por recursos iniciadas por um mesmo usuário. O intervalo mínimo entre cada sessão é de 30 minutos.

Usuários: Usando os identificadores de usuário existentes em cada rede P2P, caracterizamos o tráfego gerado por eles.

Essa metodologia de caracterização nos permite avaliar a carga dessas redes P2P sob diferentes perspectivas. Além disso, realizou-se uma análise da localidade de referência entre as duas redes P2P monitoradas.

5.4 A caracterização

Observe que a quase totalidade dos gráficos está em escala logarítmica.

5.4.1 Fragmentos

A caracterização dos fragmentos busca quantificar quais e fragmentos de um dado recurso foram transferidos, a localidade de referência desses fragmentos e qual poderia ser a economia de largura de banda caso fosse usado um *cache* infinito. Nesta seção, não fazemos distinção entre segmentos e fragmentos.

Inicialmente analisa-se o número de *bytes* transferidos associados a cada recurso. Observando a Figura 5.2 pode-se notar que a distribuição desses recursos assemelha-se a uma função exponencial. Isso é condizente com trabalhos anteriores, que refutam uma distribuição Zipf para a popularidade de arquivos em redes P2P. [Gummadi et al., 2003]

A figura 5.3 apresenta a correlação entre os tamanhos estimados dos recursos em associação aos *bytes* transferidos associados desses mesmos recursos. Pode-se observar que, na maioria dos casos, os recursos não foram transferidos completamente, como conseqüência da forma fragmentada de transferência de arquivos.

Todavia, existe uma referência de localidade significativa para os fragmentos transferidos. Mediu-se tal referência de localidade através da economia possível de largura de

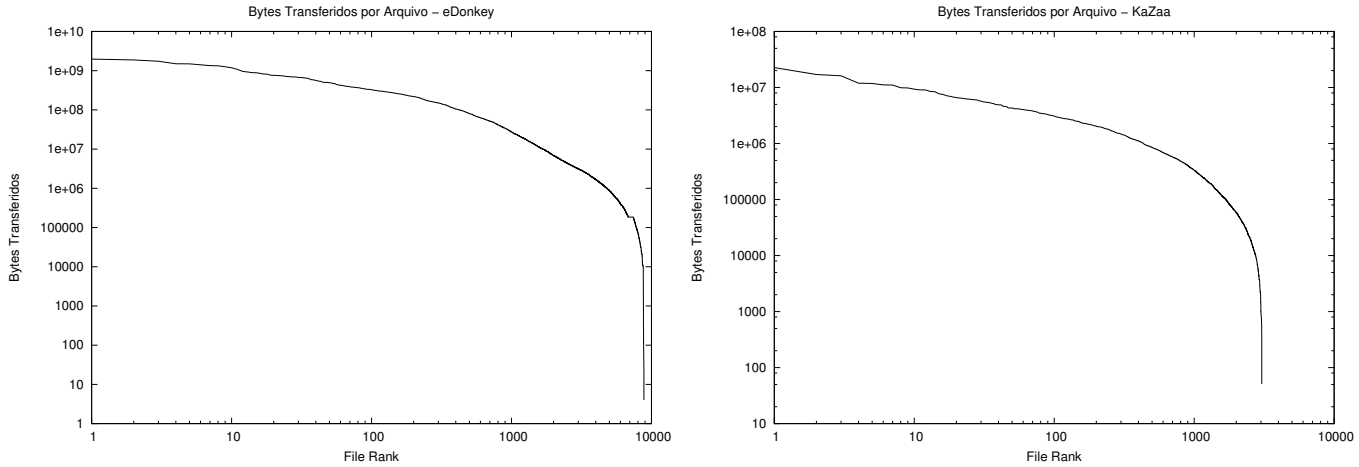

Figura 5.2: Distribuição de *bytes* transferidos por recurso.

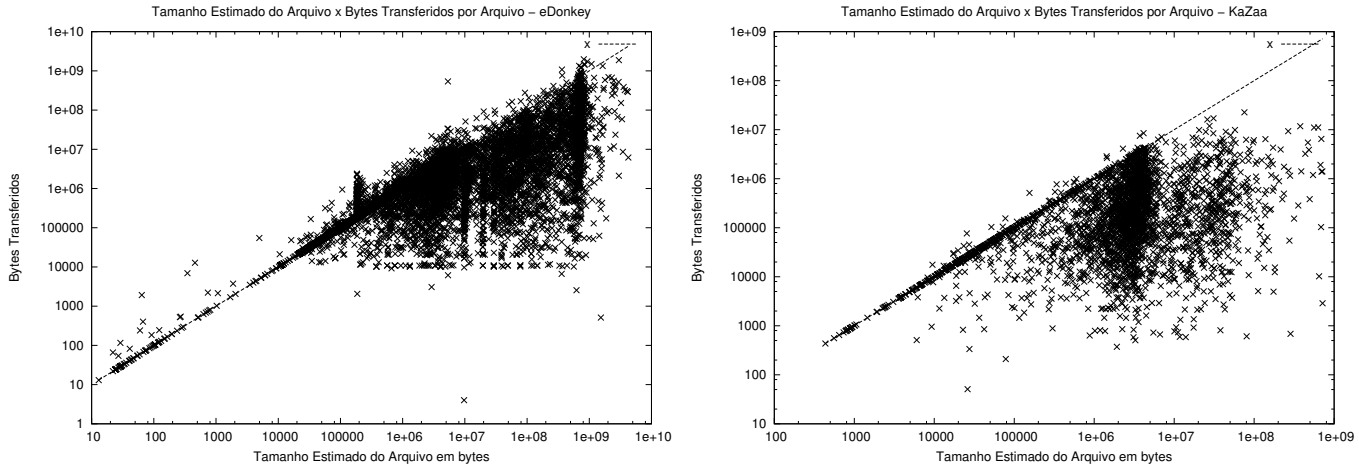

Figura 5.3: Correlação entre *bytes* transferidos e tamanho dos arquivos

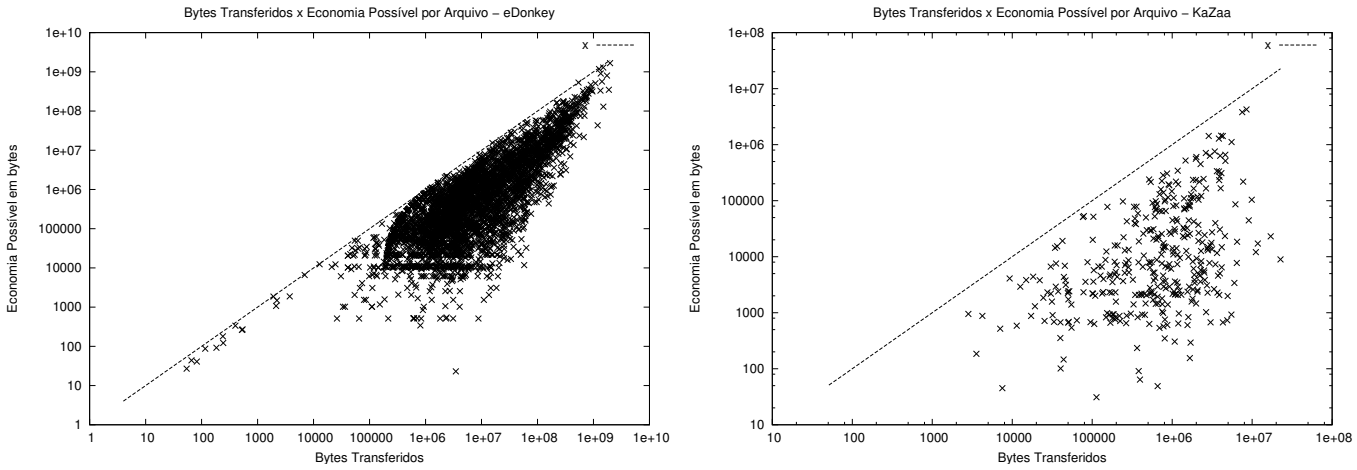

Figura 5.4: Correlação entre bytes transferidos e os ganhos potenciais de economia em largura de banda

banda, ou *bandwidth savings*, que representa o número de *bytes* que não seriam transmitidos caso tivéssemos um *cache* infinito armazenando todos os *bytes* que passassem pelo roteador de borda do provedor. Esse valor é calculado *byte*-a-*byte*. A figura 5.4 mostra, para cada protocolo, a correlação entre o número total de *bytes* transferidos para um determinado recurso e seu *bandwidth-savings* correspondente. Pode-se ver que existem vários recursos que foram transferidos diversas vezes, representando uma boa oportunidade para o emprego de *cache*.

5.4.2 Recursos

Nossa caracterização de recursos leva em conta quatro critérios: a popularidade de um recurso, seu tamanho e o processo de chegada de requisições por recursos.

A figura 5.5 mostra a popularidade de recursos para ambos os protocolos, onde pode-se ver que a distribuição é claramente *skewed* e altamente concentrada. Além disso, ele possui uma longa calda, composta de arquivos que foram requisitados apenas uma única vez.

Considerando o tamanho dos recursos, novamente encontramos uma grande variabilidade (Figura 5.6), com 80% dos recursos do KaZaa sendo menores que 10 Mb. Em contraste, a mesma quantidade de recursos no eDoney é menor do que 100 Mb, o que deixa bem claro a difernença do tipo de recursos transmitido em ambas as redes.

O processo de chegada de requisições é bem caracterizado por rajadas. A maioria das requisições por recursos chegou em intervalos bem próximos. Para para o KaZaa,

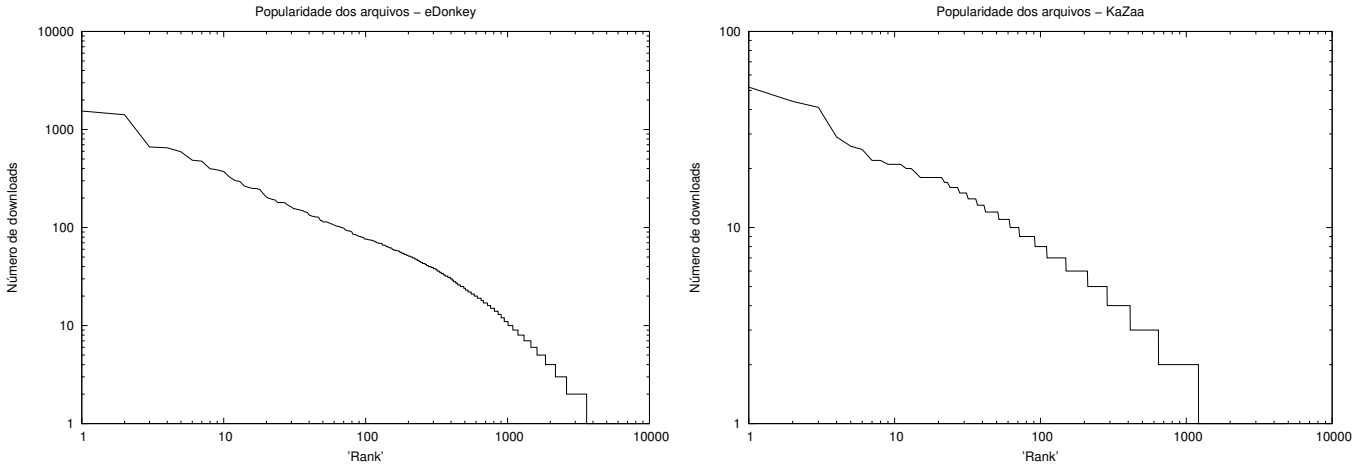

Figura 5.5: Popularidade dos Recursos

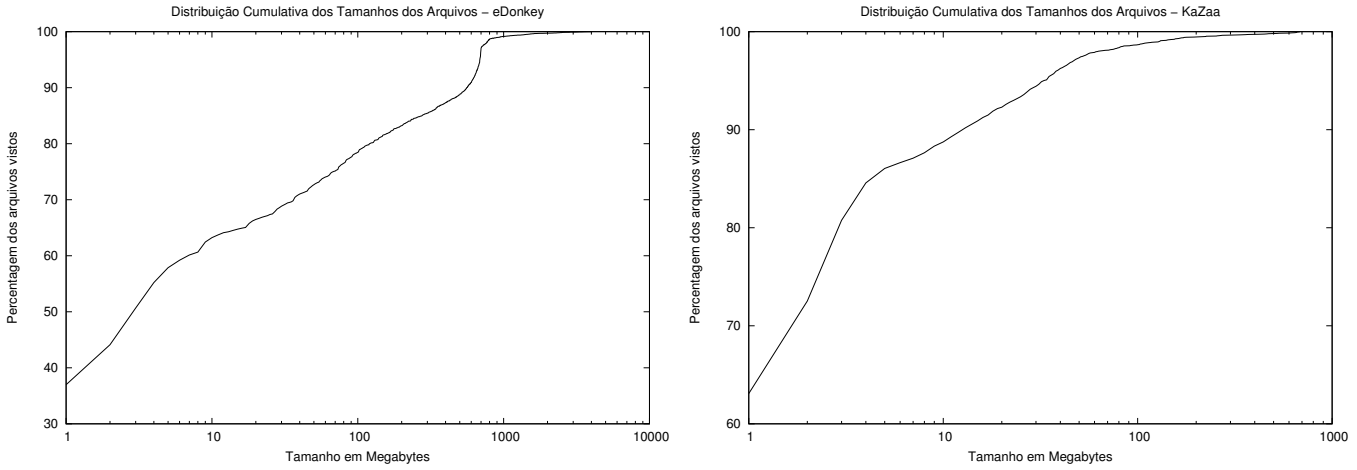

Figura 5.6: Distribuição cumulativa dos tamanhos dos arquivos

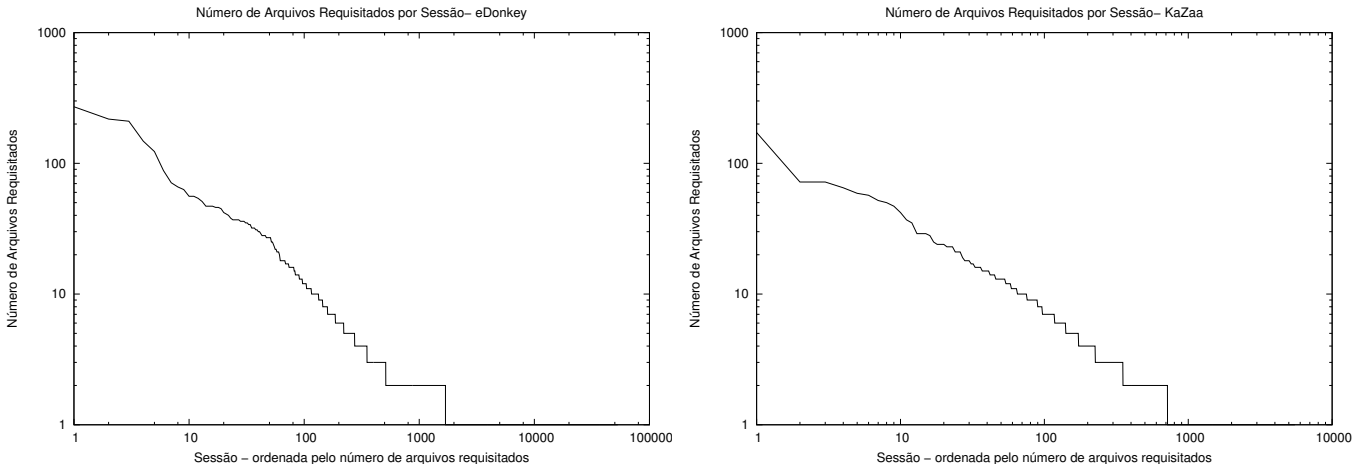

Figura 5.7: Distribuição do número de recursos transferidos por sessão

aproximadamente 100% das requisições estão separadas uma das outras por menos de 1000 segundos. No eDonkey, 100% das requisições estão separadas por menos de 100 segundos.

5.4.3 Sessões

Uma vez que a maioria das sessões solicitam apenas um recurso, observa-se que o processo de chegada e duração das sessões são similares àqueles encontrados para as requisições. Todavia, para uma parcela das sessões, existe uma grande variabilidade no número de recursos solicitados, como pode ser visto na figura 5.7.

5.4.4 Usuários

Analisamos a carga sob a perspectiva do usuário através da análise do tráfego, do número de recursos solicitados e do número de sessões por por usuários únicos.

A quantidade de *bytes* transferidos por usuário claramente mostra que existe uma grande variabilidade entre a carga gerada por cada usuário. Em particular, pode-se dizer que a distribuição segue uma distribuição exponencial, como pode ser observado na figura 5.8.

A quantidade de recursos solicitados por usuários também apresenta grande variabilidade, mas sua distribuição assemelha-se uma lei de potêncua, como pode ser visto na figura 5.9.

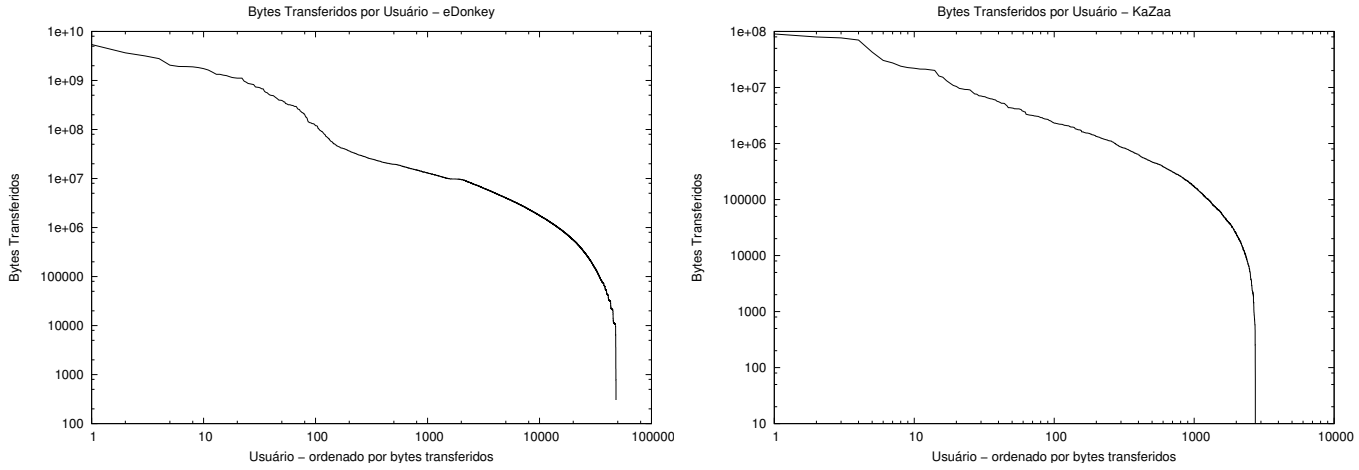

Figura 5.8: Distribuição do número de *bytes* transferidos por usuário

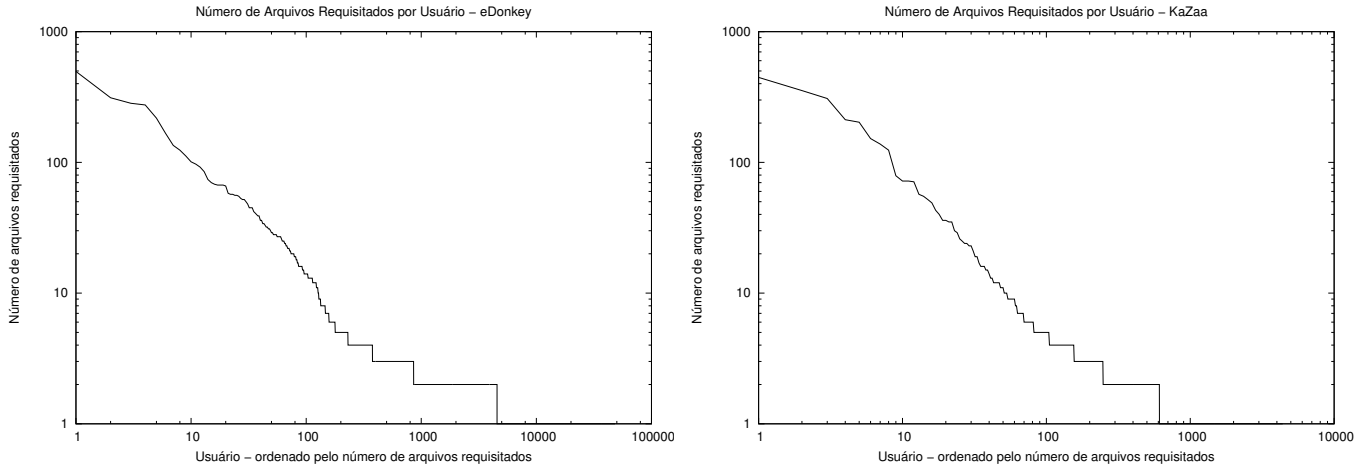

Figura 5.9: Distribuição do número de recursos solicitados por usuário

P2P	Recursos	Sig1	Sig2	Sig3	Recursos Comuns	Bytes Comuns
Kazaa	3042	1481	288	54	8.20%	12.63%
Edonkey	8835	6211	2118	829	5.25%	7.04%

Tabela 5.2: Localidade de referência entre o KaZaa e o eDonkey

5.4.5 Localidade de Referencia inter-protocolo

Nessa seção caracteriza-se a referencia de localidade entre os arquivos obtidos pelo KaZaa e pelo eDonkey, i.e., a quantidade do tráfego comum aos dois protocolos. Essa caracterização dará uma estimativa da quantidade de recursos que são comuns a ambas as redes e que poderiam ser obtidos mais facilmente caso houvesse uma mecanismo de interligação entre esses protocolos.

Durante o período de coleta, observou-se um grande número de fragmentos mas, na maioria dos casos, não foi possível recuperar os recursos inteiros, como conseqüência da fragmentação gerada pelos protocolos de transferência de arquivos e outros motivos. Foi empregado uma estratégia baseada no uso de algoritmos de *hashing* criptográfico para a geração de assinaturas dos recursos. Foram escolhidas três faixas e, para cada uma delas, para cada recurso recuperado, uma assinatura foi gerada. Através do casamento dessas assinaturas entre os vários arquivos foi possível determinar que recursos provavelmente estariam disponíveis em ambas as redes.

As mesmas faixas foram escolhidas para a geração das assinaturas em ambos os protocolos. Sempre que não foi possível recuperar uma dessas faixas completamente a geração da da assinatura dessa faixa não era feita.

A tabela 5.2 mostra o número de assinaturas que foi possível gerar para cada uma das faixas (Sig1, Sig2, and Sig3). Como era de se esperar, o número de assinaturas feitas próximas ao início dos recursos era maior do que os demais. Também estimou-se o número de recursos que seriam comuns ambas as redes e quanto dos *bytes* transmitidos seriam devido a esses arquivos. Em ambos os casos, o número de *bytes* é mais significativo que o número de recursos, confirmando que esses recursos comuns são mais populares. Em ambos os casos os números são pequenos, tanto como conseqüência do pequeno período de observação mas também devido ao grande número de assinaturas que não puderam ser geradas.

Capítulo 6

Conclusões e Trabalhos Futuros

6.1 Conclusões

O tráfego devido a aplicações P2P tem aumentado consideravelmente nos últimos anos e, apesar de ser hoje o responsável pela maior parte de todo o tráfego da Internet, não existem muitas ferramentas que auxiliem na monitoração e portanto no entendimento desse tráfego, tanto sob um ponto de vista acadêmico como sob um ponto de vista prático. Nesse trabalho abordamos o as dificuldades em se construir uma solução que viabilize a análise e a caracterização de tráfego de aplicações em tempo real, tendo como foco principal as aplicações P2P de troca de arquivo e propusemos o *Palantír*, uma sistema de monitoração passiva com recuperação de estado em tempo real de tráfego em enlaces de alta-velocidade.

Como vimos, as ferramentas utilizadas para a construção dessa ferramenta se mostraram bastante satisfatórias e, devido ao seu uso, conseguimos construir um sistema capaz de, além da monitoração, realizar a remontagem dos arquivos trocados no tráfego monitorado, mesmo a velocidades de 500Mbps com uma taxa de perda de pacote da ordem de 3.3% e um desempenho 80% superior ao que poderia ser obtido utilizando abordagens tradicionais.

Apresentamos um estudo de caso onde analisou-se e caracterizou-se, usando o *Palantír*, o tráfego associado a dois sistemas P2P em um provedor de acesso a Internet de banda larga por um período de aproximadamente 10 dias. Os valores encontrados na nossa caracterização são condizentes com os encontrados em estudos anteriores. Além disso pode-se observar que a localidade de referência entre as duas redes é pequena mas significativa.

6.2 Trabalhos Futuros

Durante a elaboração da arquitetura, da sua implementação e da realização da caracterização do tráfego do provedor, observamos várias oportunidades para realizar melhorias não só na arquitetura mas em abordagens adotadas na sua elaboração e uso.

6.2.1 Melhorarias de desempenho

Um dos fatores decisivos para o desempenho da estrutura de coleta é o processo de captura de pacotes. Estudos recentes mostram que ainda é possível conseguir mais do que se supunha possível com equipamentos convencionais [Deri, 2005]. Resta testar o desempenho da plataforma tirando proveito dos resultados desses trabalhos.

Outro processo fundamental para a arquitetura de monitoramento passivo com recuperação de estado em tempo real é a infra-estrutura utilizada pra realizar a remontagem dos fluxos TCP em tempo real. Decidimo-nos por utilizar a libNIDS devido a sua flexibilidade e por outros motivos, como comentado no capítulo 3, mas não realizamos um estudo do seu desempenho nem buscamos formas pra otimizá-la. Uma das formas que vislumbramos para melhorar o desempenho dessa biblioteca é retirar dela a responsabilidade de executar alguns testes de verificação de sanidade nos pacotes capturadas. Várias interfaces de rede atuais dispõem de recursos para realizar tais verificações internamente, retirando do processador custo desses testes e descartando pacotes mal-formados. Aproveitar esses recursos sem diminuir a funcionalidade da libNIDS seria uma boa alternativa para melhorar seu desempenho.

6.2.2 Outras redes

Além dessa da Kazaa e eDonkey, outras redes poderiam ter sido monitoradas. Todavia, excetuando-se o BitTorrent, os demais sistemas P2P de troca de arquivo, como o Gnutella, DirectConnect e SoulSeek não apresentam tráfego representativo no provedor em questão. No decorrer dos nossos trabalhos, a rede BitTorrent, passou a ser responsável por uma parcela extremamente significativa do trafego P2P do provedor, bem como em outras redes [Sen et al., 2004].

6.2.3 Análise do trágefo de sinalização

Como argumentado na seção 4.4.2, decidimos logo no início da elaboração dos monitores por não monitorar o tráfego de sinalização das redes escolhidas. Todavia, uma análise

desse tráfego poderia nos auxiliar a criar um modelo para a carga de busca das redes P2P atuais. Apesar de existirem trabalhos nessa área, tais trabalhos observam apenas tráfego de redes que não são mais populares nem e relevantes [Klemm et al., 2004]. Uma análise nova renderia bons indicativos de como desenvolver futuros sistemas P2P visando melhorar o processo de localização de recursos com base na carga dos sistemas atuais.

6.2.4 *Cache* oportunístico e controle de tráfego

Uma aplicação trivial que poderia ser construída a partir da remontagem dos recursos presentes no tráfego monitorado é um "*cache* oportunístico". Em *proxies* transparentes, para popular seu *cache*, o sistema intercepta transferências e participa, mesmo que sem o conhecimento dos clientes, do processo de obtenção dos recursos.

Um "*cache* oportunístico", por outro lado, é populado sem interagir com nenhuma das partes envolvidas no processo de obtenção de recursos, sendo "sorrateiramente" populado a medida que os clientes de uma rede transferem recursos com outros clientes. Associado a algum mecanismo de divulgação de recursos, esse *cache* poderia diminuir o volume de tráfego devido a aplicações P2P em uma rede, como observado em outros trabalhos [Leibowitz et al., 2002, Gummadi et al., 2003].

Outra opção é a construção de um sistema de controle de infrações de direitos autorais e de propriedade. Isso é possível visto que temos conhecimento de quais recursos estão sendo trocados e por quem. O sistema poderia apenas gerar relatórios ou até mesmo impedir que tais violações se concretizassem enviando pacotes adulterados de fim de conexão para ambas as partes.

6.2.5 Dinâmica dos fragmentos

Um das características do nosso trabalho é o enfoque que fazemos na dinâmica de fragmentos e segmentos. Todavia, o estudo que fazemos sobre a dinâmica desses fragmentos ainda é muito tímido.

Existem na literatura trabalhos que modelam ou simulam essa dinâmica tanto para a rede BitTorrent quanto para a rede eDonnkey [Hoßfeld et al., 2004, Qiu and Srikant, 2004]. Excetuando-se um trabalho de Izal, não conhecemos mais nenhum outro trabalho que realize uma análise baseada em trafego real da dinâmica de fragmentos [Izal et al., 2004], o que, visto a popularidade que sistemas que adotam o modelo de download fragmentado vêm ganhando, é muito pouco.

Referências Bibliográficas

[edo, 2004] (2004). eDonkey home page. http://www.edonkey2000.com.

[kaz, 2004] (2004). KaZaa home page. http://www.kazaa.com.

[emu, 2005] (2005). eMule project home page. http://www.emule-project.net.

[Androutsellis-Theotokis and Spinellis, 2004] Androutsellis-Theotokis, S. and Spinellis, D. (2004). A survey of peer-to-peer content distribution technologies. *ACM Comput. Surv.*, 36(4):335–371.

[Arlitt and Jin, 2000] Arlitt, M. and Jin, T. (2000). A workload characterization study of the 1998 world cup web site. In *Network*. IEEE.

[Arlitt and Williamson, 1996] Arlitt, M. F. and Williamson, C. L. (1996). Web server workload characterization: the search for invariants. In *Proceedings of the 1996 ACM SIGMETRICS international conferenc e on Measurement and modeling of computer systems*, pages 126–137. ACM Press.

[Bailey et al., 1994] Bailey, M. L., Gopal, B., Pagels, M. A., Peterson, L. L., and Sarkar, P. (1994). PathFinder: A pattern-based packet classifier. pages 115–123. Proc. of the 1st Symposium on Operating System Design and Im plementation. USENIX Association.

[Balakrishnan et al., 2003] Balakrishnan, H., Kaashoek, M. F., Karger, D., Morris, R., and Stoica, I. (2003). Looking up data in P2P systems. *Communications of the ACM*, 46(2):43–48.

[Barish and Obraczka, 2000] Barish, G. and Obraczka, K. (2000). World wide web caching: Trends and techniques. In *IEEE Communications Magazine - Internet Technology Series*.

[Begel et al., 1999] Begel, A., McCanne, S., and Graham, S. L. (1999). BPF+: exploiting global data-flow optimization in a generalized pac ket filter architecture. In *SIGCOMM '99: Proceedings of the conference on Applications, techn ologies, architectures, and protocols for computer communication*, pages 123–134, New York, NY, USA. ACM Press.

[Benevenuto et al., 2005] Benevenuto, F., Júnior, J. I., and Almeida, J. (2005). Avaliação de mecanismos avançados de recuperação de conteúdo em sist emas P2P. In *Anais do 23 Simpósio Brasileiro de Redes de Computadores, SBRC2005*.

[Berners-Lee, 1994] Berners-Lee, T. (1994). RFC 1630: Universal Resource Identifiers in WWW.

[Berners-Lee et al., 1994] Berners-Lee, T., Masinter, L., and McCahill, M. (1994). RFC 1738: Uniform Resource Locators (URL).

[Bhagwan et al., 2003] Bhagwan, R., Savage, S., and Voelker, G. M. (2003). Understanding availability. In *Second International Workshop on Peer-to-Peer Systems (IPTPS 2 003)*, Lecture Notes in Computer Science, pages 256–267. Springer.

[Bos et al., 2004] Bos, H., de Bruijn, W., Cristea, M., n, T. N., and Portokalidis, G. (2004). FFPF: Fairly fast packet filters. In *Proceedings of OSDI'04*.

[Brakmo et al., 1994] Brakmo, L. S., O'Malley, S. W., and Peterson, L. L. (1994). TCP vegas: New techniques for congestion detection and avoidance. In *SIGCOMM*, pages 24–35.

[Brustoloni and Steenkiste, 1998] Brustoloni, J. C. and Steenkiste, P. (1998). User-level protocol servers with kernel-level performance. In *Proceedings of the INFOCOM'98*, pages 463–471.

[Chankhunthod et al., 1996] Chankhunthod, A., Danzig, P. B., Neerdaels, C., ichael F. Schwartz, M., and Worrell, K. J. (1996). A hierarchical internet object cache. In *USENIX Annual Technical Conference*, pages 153–164.

[Chawathe et al., 2003] Chawathe, Y., Ratnasamy, S., Breslau, L., Lanham, N., and Shenker, S. (2003). Making gnutella-like P2P systems scalable. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications , technologies, architectures, and protocols for computer communications*, pages 407–418, New York, NY, USA. ACM Press.

[Cho et al., 2002] Cho, Y. H., Navab, S., and Mangione-Smith, W. H. (2002). Specialized hardware for deep network packet filtering. In *FPL '02: Proceedings of the Reconfigurable Computing Is Going Mai nstream, 12th International Conference on Field-Programmable Logic and Applicat ions*, pages 452–461, London, UK. Springer-Verlag.

[Chu et al., 2002] Chu, J., Labonte, K., and Levine, B. N. (2002). Availability and locality measurements of peer-to-peer file systems.

[Clark et al., 1989] Clark, D., Jacobson, V., Romkey, J., and Salwen, H. (1989). An analysis of tcp processing overhead. *IEEE Communications*, 27:23–29.

[Cleary et al., 2000] Cleary, J., Donnelly, S., Graham, I., egor, A. M., and Pearson, M. (2000). Design principles for accurate passive measurement. In *Passive and Active Measurement Workshop*.

[Cohen, 2003] Cohen, B. (2003). Incentives Build Robustness in BitTorrent. In *Workshop on Economics of Peer-to-Peer Systems*, Berkeley, CA, USA.

[Cormen et al., 1989] Cormen, T. H., Rivest, R. L., and Leiserson, C. E. (1989). *Introduction to Algorithms*. McGraw-Hill, Inc., New York, NY, USA.

[Cáceres, 1989] Cáceres, R. (1989). Measurements of wide area internet traffic. Technical report, Berkeley.

[Cáceres et al., 1998] Cáceres, R., Douglis, F., Feldmann, A., Glass, G., and binovich, M. R. (1998). Web proxy caching: the devil is in the details.

[Degioanni et al., 2003] Degioanni, L., Baldi, M., Risso, F., and i, G. V. (2003). Profiling and optimization of software-based network-analysis applica tions. In *SBAC-PAD '03: Proceedings of the 15th Symposium on Computer Archi tecture and High Performance Computing*, page 226, Washington, DC, USA. IEEE Computer Society.

[Degioanni and Varenni, 2004] Degioanni, L. and Varenni, G. (2004). Introducing scalability in network measurement: toward 10 gbps with c ommodity hardware. In *IMC04: Proceedings of the 4th ACM SIGCOMM conference on Interne t measurement*, pages 233–238, New York, NY, USA. ACM Press.

[Deri, 2003] Deri, L. (2003). Passively monitoring networks at gigabit speeds using commodity hard ware and open source software. In *In Passive and Active Measurement Workshop 2003*.

[Deri, 2004] Deri, L. (2004). Improving passive packet capture: Beyond device polling. In *4th International System Administration and Network Engineering Conference*.

[Deri, 2005] Deri, L. (2005). nCap: Wire-speed packet capture and transmission. In *IEEE/IFIP Workshop on End-to-End Monitoring Techniques and Servi ces (E2EMON)*. available at http://luca.ntop.org/nCap/.

[Desai, 2002] Desai, N. (2002). Increasing performance in high speed NIDS. Retrived in April 2005 from www.linuxsecurety.com.

[Endance Measurement Systems, 2005] Endance Measurement Systems (2005). The DAG project. http://dag.cs.waikato.ac.nz.

[Engler and Kaashoek, 1996] Engler, D. R. and Kaashoek, M. F. (1996). Dpf: fast, flexible message demultiplexing using dynamic code generat ion. In *SIGCOMM '96: Conference proceedings on Applications, technologies , architectures, and protocols for computer communications*, pages 53–59, New York, NY, USA. ACM Press.

[Gummadi et al., 2003] Gummadi, K. P., Dunn, R. J., Saroiu, S., Gribble, S. D., Levy, H. M., and Zahorjan, J. (2003). Measurement, modeling, and analysis of a peer-to-peer file-sharing wo rkload. In *Proceedings of the nineteenth ACM symposium on Operating system s principles*, pages 314–329. ACM Press.

[Harren et al., 2002] Harren, M., Hellerstein, J. M., Huebsch, R., hau Loo, B. T., Shenker, S., and Stoica, I. (2002). Complex queries in dht-based peer-to-peer networks. In *IPTPS '01: Revised Papers from the First International Workshop o n Peer-to-Peer Systems*, pages 242–259, London, UK. Springer-Verlag.

[Heimlich, 1990] Heimlich, S. A. (1990). Traffic characterization of the NSFNET national backbone. In *Proceedings of the 1990 ACM SIGMETRICS conference on Measurem ent and modeling of computer systems*, pages 257–258. ACM Press.

[Henson, 2003] Henson, V. (2003). An analysis of compare-by-hash. In *Proceedings of HotOS'03: 9th Workshop on Hot Topics in Operating Systems, May 18-21, 2003, Lihue (Kauai), Hawaii, USA*, pages 13–18. USENIX.

[Hoßfeld et al., 2004] Hoßfeld, T., Leibnitz, K., Pries, R., tschku, K. T., Tran-Gia, P., and Pawlikowski, K. (2004). Information diffusion in edonkey filesharing networks. In *ATNAC 2004*, page 8, Sydney, Australia.

[Iannaccone et al., 2001] Iannaccone, G., Diot, C., Graham, I., and McKeown, N. (2001). Monitoring very high speed links. In *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement 2001, San Francisco, California, USA, November 1-2, 2001*, pages 267–271. ACM.

[Inc., 2002] Inc., C. S. (2002). NetFlow services and applications - white paper. http://www.cisco.com/warp/public/cc/pd/iosw/ioft/neflct/tech/nap ps_wp.htm, último acesso em julho de 2005.

[Ioannidis et al., 2002] Ioannidis, S., Anagnostakis, K., Ioannidis, J., and Keromytis, A. (2002). xPF: packet filtering for lowcost network monitoring. In *IEEE Workshop on High-Performance Switching and Routing (HPSR)*, pages 121–126.

[Izal et al., 2004] Izal, M., Urvoy-Keller, G., Biersack, E. W., cal Felber, P., Hamra, A. A., and Garcés-Erice, L. (2004). Dissecting bittorrent: Five months in a torrent's lifetime. In *PAM*, pages 1–11.

[Karagiannis et al., 2004a] Karagiannis, T., Faloutso, M., Broido, A., Brownlee, N., and Claffy, K. C. (2004a). Is P2P dying or just hiding. In *Globecom 2004*.

[Karagiannis et al., 2004b] Karagiannis, T., Faloutso, M., Broido, A., Brownlee, N., and Claffy, K. C. (2004b). Transport layer identification of P2P traffic. In *Internet Measurement Conference 2004*.

[Klemm et al., 2004] Klemm, A., Lindemann, C., Vernon, M. K., and r P. Waldhorst, O. (2004). Characterizing the query behavior in peer-to-peer file sharing system s. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measure ment*, pages 55–67. ACM Press.

[Kruegel et al., 2002] Kruegel, C., Valeur, F., Vigna, G., and d Kemmerer, R. (2002). Stateful intrusion detection for high-speed networks. In *SP '02: Proceedings of the 2002 IEEE Symposium on Security and Pr ivacy*, page 285, Washington, DC, USA. IEEE Computer Society.

[Leibowitz et al., 2002] Leibowitz, N., Bergman, A., Ben-Shaul, R., and Shavit, A. (2002). Are file swapping networks cacheable? characterizing p2p traffic. In *7th International Workshop onWeb Content Caching and Distribut ion*.

[Leibowitz et al., 2003] Leibowitz, N., Ripeanu, M., and Wierzbicki, A. (2003). Deconstructing the kazaa network. In *WIAPP '03: Proceedings of the The Third IEEE Workshop on Internet Applications*, page 112, Washington, DC, USA. IEEE Computer Society.

[Liang et al., 2004] Liang, J., Kumar, R., and Ross, K. (2004). Understanding KaZaA. submitted, 2004.

[Liang et al.,] Liang, J., Kumar, R., Xi, Y., and Ross, K. W. Pollution in P2P file sharing systems. In *Proceedings of IEEE Infocom 2005*.

[Loo et al., 2004] Loo, B. T., Huebsch, R., Stoica, I., and Hellerstein, J. M. (2004). The case for a hybrid P2P search infrastructure. In *IPTPS*, pages 141–150.

[Markatos, 2002] Markatos, E. P. (2002). Tracing a large-scale peer to peer system: an hour in the life of gn utella. In *2nd IEEE/ACM International Symposium on Cluster Computing and th e Grid*.

[Maymounkov and Mazières, 2002] Maymounkov, P. and Mazières, D. (2002). Kademlia: A peer-to-peer information system based on the xor met ric. In *IPTPS'02 - 1st International Peer To Peer Systems Workshop*.

[McCanne and Jacobson, 1993] McCanne, S. and Jacobson, V. (1993). The BSD packet filter: A new architecture for usert-level packet c apture. In *Proc. of the Winter 1993 USENIX Conference*, pages 259–270, San Diego, California.

[Mogul et al., 1987] Mogul, J., Rashid, R., and Accetta, M. (1987). The packer filter: an efficient mechanism for user-level network code. In *SOSP '87: Proceedings of the eleventh ACM Symposium on Operating systems principles*, pages 39–51, New York, NY, USA. ACM Press.

[Moore et al., 2001] Moore, D., Keys, K., Koga, R., Lagache, E., and Claffy, K. C. (2001). The CoralReef software suite as a tool for system and network administrators.

In *Proceedings of the 15th Conference on Systems Administration (L ISA 2001)*, pages 133–144. USENIX.

[Moy, 1991] Moy, J. (1991). RFC 1247: OSPF version 2.

[Pouwelse et al., 2005] Pouwelse, J., Garbacki, P., Epema, D., and Sips, H. (2005). The bittorrent P2P file-sharing system: Measurements and anal ysis. In *4th International Workshop on Peer-to-Peer Systems (IPTPS)*. LNCS (to appear).

[Qiu and Srikant, 2004] Qiu, D. and Srikant, R. (2004). Modeling and performance analysis of bittorrent-like peer-to-peer net works. In *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 367–378. ACM Press.

[Ratnasamy et al., 2001] Ratnasamy, S., Francis, P., Handley, M., Karp, R., and Schenker, S. (2001). A scalable content-addressable network. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 161–172. ACM Press.

[Ripeanu, 2001] Ripeanu, M. (2001). Peer-to-peer architecture case study: Gnutella network. In *1st International Conference on Peer-to-Peer Computing (P2P 2001)*, pages 99–100. IEEE Computer Society.

[Rizzo, 2001] Rizzo, L. (2001). Device polling support for freebsd. In *BSDConEurope Conference*.

[Roos et al.,] Roos, M., Willemson, J., and Laud, P. Improving the gnutella protocol against poisoning. In *Proceedings of the Seventh Nordic Workshop on Secure IT Syste ms NordSec 2003*, pages 185–194. Available at http://home.cyber.ee/jan/gnutella.ps.

[Saltzer et al., 1984] Saltzer, J. H., Reed, D. P., and Clark, D. D. (1984). End-to-end arguments in system design. *ACM Transactions on Computer Systems*, 2(4):277–288.

[Saroiu et al., 2002] Saroiu, S., Gummadi, P. K., and Gribble, S. D. (2002). A measurement study of peer-to-peer file sharing systems. In *Proceedings of Multimedia Computing and Networking 2002 (MMCN '0 2)*, San Jose, CA, USA.

[Sen et al., 2004] Sen, S., Spatscheck, O., and Wang, D. (2004). Accurate, scalable in-network identification of P2P traffic using a pplication signatures. In *WWW '04: Proceedings of the 13th international conference on Worl d Wide Web*, pages 512–521. ACM Press.

[Sen and Wang, 2002] Sen, S. and Wang, J. (2002). Analyzing peer-to-peer traffic across large networks. In *Second Annual ACM Internet Measurement Workshop*.

[Singla et al., 2003] Singla, A., Rohrs, C., and LLC, L. W. (2003). Ultrapeers: Another step towards gnutella scalability. http://rfc-gnutella.sourceforge.net/src/Ultrapeers_1.0.html.

[Stoica et al., 2001] Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., and Balakrishnan, H. (2001). Chord: A scalable peer-to-peer lookup service for internet applicatio ns. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 149–160. ACM Press.

[Szwarcfiter and Markenzon, 1999] Szwarcfiter, J. L. and Markenzon, L. (1999). *Estrutura de dados e seus algoritmos*. Livros Técnicos e Científicos Ltda., Rio de Janeiro, RJ, Brasil.

[Tanenbaum and Steen, 2001] Tanenbaum, A. S. and Steen, M. V. (2001). *Distributed Systems: Principles and Paradigms*. Prentice Hall PTR, Upper Saddle River, NJ, USA.

[tcpdump, 2005] tcpdump (2005). tcpdump and libpcaṕs homepage. http://www.tcpdump.org.

[tcpreplay, 2005] tcpreplay (2005). tcpreplayś homepage. http://tcpreplay.sourceforge.org.

[Thekkath et al., 1993] Thekkath, C. A., Nguyen, T. D., Moy, E., and Lazowska, E. D. (1993). Implementing network protocols at user level. *IEEE/ACM Trans. Netw.*, 1(5):554–565.

[Thompson et al., 1997] Thompson, K., Miller, G., and Wilder, R. (1997). Wide-area internet traffic patterns and characteristics. In *IEEE Network*, volume 11, pages 20–23.

[Tutschku, 2004] Tutschku, K. (2004). A measurement-based traffic profile of the edonkey filesharing servic e. In *5th Passive and Active Measurement Workshop (PAM2004)*, volume 3015 of *Lecture Notes in Computer Science*, Antibes Juan-les-Pins, France. April 19-20. Springer.

[van der Merwe et al., 2000] van der Merwe, J., Cáceres, R., hua C hu, Y., and Sreenan, C. (2000). mmdump: a tool for monitoring internet multimedia traffic. *SIGCOMM Comput. Commun. Rev.*, 30(5):48–59.

[van Rooij, 2001] van Rooij, G. (2001). Real stateful TCP packet filtering in Ip Filter. Unpublished invited talk, Tenth USENIX Security Symposium, August 13–17, 2001, Washington, DC, USA.

[Varenni et al., 2003] Varenni, G., Baldi, M., Degioanni, L., and o, F. R. (2003). Optimizing packet capture on symmetric multiprocessing machines. In *SBAC-PAD '03: Proceedings of the 15th Symposium on Computer Archi tecture and High Performance Computing*, page 108, Washington, DC, USA. IEEE Computer Society.

[Yuhara et al., 1994] Yuhara, M., Bershad, B. N., Maeda, C., and Moss, J. E. B. (1994). Efficient packet demultiplexing for multiple endpoints and large m essages. In *USENIX Winter*, pages 153–165.