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

MONITORAÇÃO DE TRÁFEGO PAR-A-PAR EM TEMPO REAL

TIAGO ALVES MACAMBIRA

Belo Horizonte

1 de Julho de 2005

TIAGO ALVES MACAMBIRA

MONITORAÇÃO DE TRÁFEGO PAR-A-PAR 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 Par-a-Par 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 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 à velocidade de 500 Mbps, considerada alta para os objetivos propostos, o sistema trabalha com uma taxa de perda de pacote da ordem de 3,3 % e um desempenho 99 % superior ao que poderia ser obtido utilizando abordagens tradicionais. Mesmo operando nessa condição de carga extrema, o sistema ainda é capaz de recuperar 70 % dos dados observados nos nossos experimentos, um valor que é mais de 100 vezes superior ao que pode ser obtido utilizando-se um sistema não otimizado. Além disso, identificamos que dos principais pontos de contenção no processo de coleta deve-se ao custo da passagem dos dados através da fronteira do sistema operacional entre o modo usuário e o modo protegido do *kernel*.

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. A análise dos resultados obtidos a partir desses esforços nos permitiu observar que existe uma significativa diferença tanto na natureza do tráfego gerado pelos usuários dos dois sistemas como dos recursos trocados nas duas redes.

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 system for real-time stateful monitoring of P2P traffic in high-speed networks, and discuss the requirements and challenges faced during its construction.

Even at 500 Mbps, a traffic rate higher than those commonly found at environments where *Palantír* was supposed to be employed, it archives a packet drop rate of approximately 3.3 % and performs 99 % better than what is possible with commonly used approaches. Even at those extreme load conditions *Palantír* is capable of reassembling as much as 70 % of the whole data available in our experiments, a 100 fold gain of what is possible with traditional systems.

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 10 days period. The results of this characterization suggest that there is a significant difference in the nature of both the resources traded by the users of the two P2P networks and of their traffic.

... o importante é o processo.

Agradecimentos

Foi uma longa caminhada. Tenho certeza de que muitos me ajudaram no decorrer desse percurso e que, no processo de lhes agradecer, alguns serão inevitavelmente esquecidos. A estes últimos peço desde já as minhas mais sinceras desculpas e dou meu muito obrigado.

Antes de mais nada gostaria de agradecer aos meus pais e à minha irmã, pessoas incríveis que a vida me deu a sorte e a honra de ter como familiares e amigos.

Aos professores Dorgival Olavo Guedes Neto e Wagner Meira Júnior. O percorrer desse caminho foi um processo longo e trabalhoso. Houve trechos mais atribulados do que o de costume, intercalados com léguas e léguas de árduas retas sem fim. O importante é que, findo esse processo, após ter desbravado todo esse percurso, fico feliz de ter tido vocês como guias e mentores. Além disso, gostaria de agradecer aos professores Creto Vidal, Javam Machado e Rossana Andrade por terem me preparado para essa jornada.

A todo o pessoal do POP-CE pela paciência, pela experiência, pela amizade, pelos ensinamentos e por todo o crescimento que eu tive quando trabalhei junto a vocês.

Aos camaradas João Caram de Oliveira e Roberto Jorge Dummar Filho. Os calos dessa caminhada foram todos meus mas vocês bem que compartilharam dos incômodos que vários deles causaram.

Aos “anfitriões” Camila Caram, Tia Jane e Tia Soraya, Pablo de Freitas Melo, Julia Selani Rodrigues Silva Melo, Valeria Goncalves Teixeira, Grace Carneiro Braz, Darwin Oliveira Teixeira Silveira e Rafael Takai, pela hospitalidade e por receberem com tanto carinho um cearense resmungão.

Aos “conterrâneos” e colegas Michelle Nogueira, André Luiz Lins de Aquino por me ajudarem a dar os primeiros passos dessa jornada.

Aos colegas Geraldo Antonio Ferreira, Fernando Almir Nascimento Jr, Patrícia Martins, Danielle Santos da Silva, Erikson Freitas de Morais, Wagner Ferreira de Barros, Marcelo Borghetti Soares, Ademir de Alvarenga Oliveira e Vilar Fiuza da Camara Neto pela camaradagem. Sei que compartilhamos várias das dificuldades passadas no inicio dos percursos que cada um de nós tomou e confesso que, se não fosse o apoio que demos uns aos outros

nesse início, o trajeto seria mais tumultuado e muito menos agradável.

Aos colegas Cristiano Maciel da Silva, Bruno Estolano Grossi, Bruno Rocha Coutinho e Walter dos Santos Filho pela ajuda e paciência.

Aos paraenses Daniel Moutinho de Moura, Bruno Brito, Márcio Sequeira, Rodrigo e Vânia Vale e ao quase-paraense Luciano Polisseni. Nem só de trabalho vive o homem e, "égua, muleques", vocês sabem como poucos fazer uma noite render muitas histórias.

A todo o pessoal do SPEED, em especial a André Assis Chaves George Luiz Medeiros Teodoro, Túlio Coelho Tavares, José Ismael Pereira Junior, e ao Beto e seu bar. "Tá ligado, neguim"!!!

Gostaria de agradecer também ao CNPq e FUNDEP pela ajuda financeira, indispensável nessa empreitada. Gostaria também de agradecer à WayBrasil, à Empresa de Infovias, e em especial a Ricardo Bastos, pelo suporte e pela sensibilidade à pesquisa acadêmica demonstrada ao colaborar com esta pesquisa.

Não poderia deixar de agradecer a velhos colegas e velhos amigos que, mesmo pertencendo a tempos mais remotos, mantiveram-se presentes de uma forma ou de outra e que foram de fundamental importância para que eu chegasse até aqui, quer pela simples e descompromissada amizade, que pelo exemplo de pessoa que foram para mim. Desta forma, gostaria de agradecer a Alison Sellaro Pelucio, Alex Sandro "Freud" Queiroz e Silva, Victor Campos, Viktor Saboia, Ana Shirley Ferreira da Silva, Windson Carvalho de Viana, Getúlio Fiúza e a Themístocles Mesquita.

Finalmente, jamais poderia deixar de agradecer veementemente aos todos os meus amigos e irmãos dos "Queridinhos do Vei Voyeur", aos associados e personalidades ilustres, em especial ao Rodrigo Santos Rocha e ao Daniel Pordeus Menezes, sem claro desmerecer a Flávia, Rommel, Clarissa, Alex José, Larissa, Jamille, Juliana, Nataly, Andrezinho, Marcus, Andréa, Samuel e tantos outros. Vocês sabem o tanto que eu prezo pela amizade de vocês, mesmo que eu não diga isso com a freqüência que eu deveria.

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, ocasionou

[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.

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 diminuí-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 e 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, a qualidade e relevância dos dados obtidos pela monitoração passiva está diretamente relacionada ao tipo de análise feita sobre o tráfego coletado. 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*), ou seja, limitaram-se a identificar as diversas conexões TCP existentes no tráfego coletado e a contabilizar o volume total de *bytes* trocados por essas conexões. Tal abordagem permite o fornecimento de indicativos da dinâmica dos fluxos e do volume do tráfego dos sistemas P2P, porém não possibilita a extração de 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. Para tanto, faz-se necessário realizar a recuperação do estado das conexões do tráfego monitorado, ou seja, é necessário recuperar do tráfego monitorado os dados tais como eles são entregues às aplicações para, finalmente, poder recuperar desses dados o seu valor semântico junto às aplicações.

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* oportunísticos", 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 subproblemas 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 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 de tráfego no nível da camada de aplicação com recuperação de estado em tempo real, mesmo

[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.

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 localidade de referência desses arquivos e a possibilidade de se obter ganhos através do emprego de *caches* (*cacheability*).

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 hardware 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 discussão 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, analisar-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, 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]

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

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.

  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

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

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 pode 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 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.

[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.

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 um 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 seqüência 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].

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

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

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 geral 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, nenhum 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 pacote, 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 de uso geral (ou COTS, *commodity, off-the-shelf hadware*) 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 (*stateful packet filtering*) mas também a NIDS (*stateful 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, 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 [Elson, 2005, Ethereal, 2005]. 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*) [Cisco Systems 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 [Song, 2005, flowgrep, 2005, Wojtczuk, 2005]. 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.

2.5 Sumário

Apresentou-se nesse capítulo um panorama do que compreende a realização de monitoração passiva de tráfego com recuperação de estado.

As duas formas de se realizar monitoração de tráfego, a monitoração ativa e a passiva, foram apresentadas, bem como o funcionamento, vantagens, desvantagens e limitações de cada uma delas.

Algumas atividades que possuem semelhanças com a monitoração passiva de tráfego foram analisadas. Além disso, comentou-se sobre suas similaridades, seus problemas e suas particularidades. Em especial, comentou-se sobre os problemas que tais atividades têm com o aumento da velocidade da rede e como elas os contornam.

Mecanismos de captura de pacotes são comuns a todas essas atividades, incluindo à monitoração passiva de tráfego. Por esse motivo, abordamos as soluções encontradas na literatura para implementar tais mecanismos e para torná-los eficientes, mesmo quanto tais mecanismos são utilizados em enlaces de alta velocidade.

Finalmente, abordamos o problema de realizar a recuperação de estado de tráfego monitorado, focando nas particularidades encontradas quando tal remontagem deve ocorrer em tempo real.

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 e alguns dos compromissos aos quais estávamos sujeitos durante a sua concepção.

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.

Para ser capaz de analisar o tráfego de rede a nível das aplicações envolvidas e recuperar as informações trocadas é necessário que o sistema (e em particular os mecanismos de captura adotados) tenha(m) um desempenho elevado. Taxas comuns em canais de saída de provedores de serviço banda larga, por exemplo, estão hoje na faixa de centenas de megabits por segundo ou mais. Por isso, o sistema deve ser capaz de capturar e tratar os pacotes a essa taxa, com todo o processamento necessário para recompor o comportamento dos usuários das aplicações de interesse.

Como mencionado no capítulo anterior, com as taxas atuais, a arquitetura dos computadores de uso geral tradicionais (usualmente identificados pela sigla COTS, *commodity, off-the-shelf hardware*) está perto do seu limite de capacidade [Cleary et al., 2000, Iannaccone et al., 2001]. Algumas soluções utilizando *hardware* especializado existem para o problema de captura de pacotes em enlaces de alta velocidade [Iannaccone et al., 2001, Endance Measurement Systems, 2005, Sen et al., 2004], mas com um custo elevado e limitadas à captura. 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. Essas soluções vão de encontro com os anseios inicias do projeto, pois desejava-se que a abordagem adotada fosse implementável em sistemas operacionais de código aberto e que tivesse um baixo custo.

É importante salientar que o fato de existirem soluções que possibilitam a captura de pacotes em enlaces de alta-velocidade com *hardware* comum não basta para que se possa construir sistemas que realizem recuperação de estado em tempo real nesse mesmo *hardware*. Há de se considerar que existe um custo não desprezível para processar o tráfego capturado para a fim de realizar a recuperação de estado. Além disso, tendo em mente que a tarefa de monitoração de tráfego é uma atividade onde não existe controle direto sobre a velocidade dos dados processados, se esse custo for muito alto, o sistema pode não ser capaz de realizar a sua função de maneira satisfatória.

Por esse motivo, nesse capítulo, apresentaremos e analisaremos o desempenho de diversas otimizações que podem ser aplicadas a um sistema operacional de uso geral bem como o desempenho de outras ferramentas que, quando reunidas, possibilitem a criação de um sistema como o discutido e com um desempenho satisfatório.

3.2 O processo de captura de pacotes

Para se entender os desafios de se implementar um sistema como proposto, é preciso identificar precisamente os passos envolvidos em sua operação. Inicialmente, um sistema de captura de pacotes é inserido na rede de forma a receber uma cópia de todo o tráfego de interesse. A maior parte das soluções baseadas em hardware oferecem o recurso de se interpor diretamente o sistema no canal de comunicação, enquanto que a maioria das soluções baseadas em *software*, como no caso presente, assumem que o tráfego seja replicado por um elemento de rede para um segundo canal onde o coletor é instalado.

Cada quadro a nível da rede que deve ser capturado chega à interface de rede do sistema coletor, onde é transferido para a memória da placa de rede. De lá o pacote deve ser transferido para a memória principal da máquina, onde deverá ser processado. Essa transferência é feita pelo acionador de dispositivo (*device driver*), usualmente ao ser acionado por uma interrupção gerada pela placa. Em sistemas de coleta mais simples, duas coisas podem acontecer: se o sistema tem por objetivo coletar estatísticas de tráfego por fluxo, o cabeçalho é inspecionado para identificar o fluxo adequado, onde o pacote é contabilizado; se o sistema é configurado para coleta apenas (para processamento *off-line*), o pacote termina armazenado em memória secundária para acesso posterior.

No caso de um sistema de análise de tráfego a nível da aplicação como o *Palantír*, é preciso recuperar dos pacotes o conteúdo semântico da comunicação. Um primeiro passo é realizar o processamento da pilha TCP/IP que ocorreria nas extremidades da conexão, a fim de identificar os dados da aplicação. Em uma máquina usual recebendo apenas o tráfego que lhe é direcionado isso é feito no *kernel* do sistema operacional; no caso da captura, entretanto, os pacotes vistos se referem à comunicação entre diversas máquinas distintas e devem ser recompostos de forma especial.

Uma vez recuperadas as conexões TCP com o seu conteúdo, o próximo passo é interpretar o protocolo de comunicação das aplicações envolvidas. Isso exige que seja replicado todo o processamento da aplicação, a fim de identificar requisições e suas respostas, por exemplo. Finalmente, uma vez que o protocolo da aplicação é reconhecido, é possível extrair da comunicação os dados trocados pelos usuários das aplicações. Em um sistema de troca de arquivos P2P, por exemplo, pode-se recuperar os arquivos transferidos entre os usuários.

Todo esse processo precisa ser implementado em um sistema de captura com recuperação do estado das aplicações e cada etapa acrescenta um processamento particular que precisa ser considerado. A transferência dos pacotes da placa de rede para a memória principal possui dois aspectos importantes: a utilização da banda do barramento interno da máquina e o

tratamento das interrupções. Devido à impossibilidade de se utilizar a pilha TCP existente no *kernel* padrão, o processamento desta pilha também exige atenção. O processamento do protocolo da aplicação obviamente é específico para cada caso e deve ser desenvolvido também de forma especial, pois precisa considerar que em um mesmo sistema será replicado o comportamento de diversos usuários. Finalmente, é preciso incluir também a lógica que fará uso da informação extraída — por exemplo, um mecanismo de *cache* passiva que disponibilize para todos os usuários locais o conteúdo dos arquivos observados, ou um mecanismo de controle de conteúdo, que pode optar por bloquear transferências de arquivos que tenham sido identificados como nocivos segundo alguma política do provedor (distribuição não autorizada de material protegido pelas leis de direito autoral, por exemplo)

3.3 A organização do *Palantír*

Considerando-se o processo de captura e análise de pacotes discutido, devemos então decidir como implementar a arquitetura de coleta para esse fim. Ao se considerar uma solução que concentra seus esforços quase que em sua totalidade em *software* ou, mais especificamente, na melhoria de um sistema operacional de propósito geral, nos defrontamos com a possibilidade de realizar cada parte do trabalho de análise de tráfego em modo protegido (*kernel-mode*) ou em modo de usuário (*user-level mode*).

Sistemas construídos em modo protegido podem obter ganhos significativos de desempenho por não estarem sujeitos ao custo de troca de contexto e de cópia de dados entre o modo protegido e o modo usuário que sistemas desenvolvidos como aplicações simples têm que pagar. Além disso, esses sistemas têm acesso direto aos dispositivos de rede, o que permite acessar recursos não disponíveis através das interfaces do sistema operacional para aplicativos que operam em modo de usuário.

Todavia, a criação 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 quanto à dificuldade de depurar tal código. Soluções que buscam eliminar parte dessas complicações ainda são incipientes ou não possuem a flexibilidade necessária para a construção de um sistema como o planejado.

Optamos então pelo desenvolvimento da arquitetura em sua maior parte no modo usuário. Isso levou à criação de um sistema dividido em camadas que assemelha-se bastante a soluções de servidores de rede em nível de usuário. Em síntese, realizamos a construção de uma pilha de protocolos que será executada em modo de usuário [Brustoloni and Steenkiste, 1998, Thekkath et al., 1993], como ilustrada na fi-

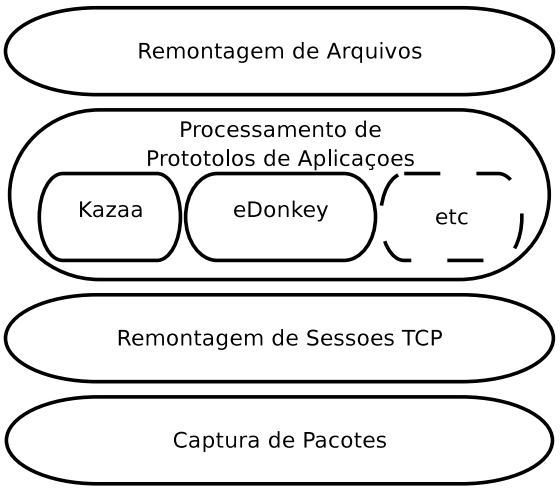

Figura 3.1: Organização em camadas do *Palantír*

gura 3.1.

Consideramos quatro camadas:

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 sistema, para reduzir o custo da transferência dos pacotes da placa de rede para a memória principal, utilizamos um *patch* para o *kernel* do Linux que implementa um *buffer* circular mapeado em memória do nível da aplicação para reduzir o custo das transferências entre o modo *kernel* e o modo usuário, denominado PF_RING [Deri, 2004].

2. **Remontagem de sessões TCP:** como discutido anteriormente, não é possível utilizar diretamente a pilha de protocolos do sistema operacional para fazer a análise dos pacotes. É necessário então um sistema que replique as máquinas de estados dos protocolos de forma eficiente a fim de garantir um desempenho suficiente para a operação

em tempo real. Como visto na seção 2.4, muitas das ferramentas existentes para esse propósito não são destinadas para recuperação de estado em tempo real. Optamos então pela utilização da libNIDS [Wojtczuk, 2005], 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 TCP em tempo real.

3. **Processamento dos protocolo das aplicações:** responsável pela análise dos protocolos das aplicações que se deseje monitorar. Para cada aplicação de interesse é preciso implementar as operações que seriam executadas na aplicação ao receber as mensagens da rede. A partir do uso da libNIDS foi definida uma interface através da qual *plugins* específicos para cada aplicação poderiam ser implementados e facilmente inseridos na arquitetura. Foram criados alguns desses *plugins*, para recuperar os dados das redes P2P eDonkey [Hoßfeld et al., 2004] e KaZaa [kaz, 2004]. Comentaremos mais sobre o desenvolvimento de tais elementos na seção 4.4.2.

4. **Remontagem de arquivos:** uma última camada da arquitetura foi concebida com o intuito de recuperar os arquivos transferidos pelos diversos protocolos de aplicação P2P monitorados, como base para uma *cache* passiva (ou oportunística) para os mesmos. Essa camada será detalhada futuramente, na seção 4.4.3.

Em suma, exceto pela aplicação do *patch* do sistema operacional para reduzir o custo da comunicação com a placa de rede, o sistema de coleta opera completamente em modo usuário. O *patch* contribui também para reduzir o custo da troca de contexto entre os modos *kernel* e usuário, utilizando o *buffer* circular mapeado para a memória da aplicação.

3.4 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 análise 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*?

3.4.1 Descrição dos experimentos

Para responder a essas perguntas, executamos dois experimentos onde a do sistema capacidade foi verificada sob condições de carga variável. Para isso, utilizamos um *trace* de tráfego coletado de um provedor de acesso à Internet banda larga. Além de observar o desempenho da arquitetura, pretendemos também caracterizar mais precisamente as limitações encontradas em um sistema de código aberto e *hardware* usual (COTS). Além disso, pretendemos observar o que pode ser feito para melhorar o desempenho desses sistemas e como tais melhorias afetam o desempenho do sistema original.

No primeiro experimento, variamos a taxa de envio dos pacotes a serem coletados para avaliar o comportamento do sistema sem otimizações. Com isso, pretendíamos avaliar o desempenho do sistema para entender as limitações da arquitetura original. Para tanto medimos a taxa de perda de pacotes. Essa taxa representa a incapacidade do sistema de lidar com o velocidade de envio de dados pela 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 sistema. No segundo experimento verificamos como a adição de cada nível da arquitetura alterava o desempenho da coleta, novamente observando a taxa de perda de pacotes do sistema.

Os experimentos utilizaram 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 Seagate modelo ST3120026AS e placa de rede PCI Intel PRO/1000. Todas as máquinas foram interligadas por um *switch* Intel Gigabit Ethernet. A maquina *A* executava o *kernel* do Linux na sua versão 2.6.11.12. As versões usadas do *patch* PF_RING e da libNIDS são, respectivamente, a 3.0 e a 1.20.

As características do tráfego coletado no provedor banda larga e usado como carga nos experimentos é apresentado na tabela 3.1. As máquinas *B* e *C* foram utilizadas para gerar a carga entregue à máquina *A*. Como cada uma continha uma cópia de todo o *trace*, o tráfego recebido pela máquina de coleta foi o dobro do conteúdo original coletado no provedor de banda larga.

Para que o tráfego pelas duas máquinas não fosse tratado como uma simples repetição dos mesmos pacotes utilizamos para o processo de envio o aplicativo tcpreplay [tcpreplay, 2005], capaz de modificar deterministicamente os cabeçalhos IP presentes no tráfego a ser enviado. Dessa forma, para o objetivo da coleta e análise, o tráfego proveniente das duas máquinas

Característica	No *trace*	Recebido
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*

geradoras de carga pode ser tratado como sendo independente pela máquina de coleta. A taxa de envio do tráfego gerado conjuntamente pelas duas máquinas e que era entregue à máquina *A* foi ajustada com velocidades que variaram de 100 Mbps a 500 Mbps no primeiro experimento e que foram mantidas em 500 Mbps no segundo.

Na máquina *A*, onde o sistema foi instalado, monitoramos a quantidade de pacotes capturados com sucesso, de onde podemos determinar o número de pacotes perdidos, e como esse valor reagiria tanto à variação da velocidade do tráfego como à adição de cada nível da arquitetura:

1. apenas com a captura de pacotes,
2. com o processo de remontagem de conexões TCP através da libNIDS,
3. com o processo de análise de protocolos,
4. com o processo de remontar totalmente arquivos.

Obviamente, cada novo nível adicionado pressupõe a manutenção dos níveis inferiores a ele.

Três dimensões de otimização do sistema foram consideradas:

- o uso ou não do *patch* PF_RING
- a adição ou não de suporte a *polling* da interface 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.

O recurso de *polling* é uma opção do *kernel* Linux que reduz a carga de interrupções geradas pela chegada de pacotes na interface. O recurso de HyperThreading da CPU Pentium 4 também é uma característica configurável no *kernel* Linux, cujo impacto também foi avaliado.

Foram consideradas as seguintes combinações das otimizações acima ou, como nos referiremos no restante do texto, configurações:

1. com PF_RING, NAPI e HT
2. com NAPI e HT
3. com PF_RING apenas
4. sem nenhuma das otimizações

Após os experimentos, verificamos que o uso de NAPI não causou nenhuma variação significativa nos resultados, mesmo com mudanças na taxa de envio da carga. Por esse motivo e para aumentar a clareza dos gráficos, retiramos dos gráficos e das estatísticas os dados referentes a configurações que diferiam apenas pelo uso ou não dessa otimização.

Para verificar etapa de captura utilizamos o programa *pcount*, disponível com o pacote do *PF_RING*. Para verificar o desempenho com as demais camadas utilizamos versões modificadas do nosso sistema, limitadas a operar até a camada selecionada para cada teste. Para cada combinação de etapa e configuração foram feitas três execuções do experimento e a média dessas foi utilizada.

3.4.2 Resultados

Como discutido anteriormente, avaliamos o impacto do aumento da taxa de envio dos pacotes e das mudanças de configuração do sistema.

Inicialmente observamos o comportamento do sistema sem nenhuma otimização, observando como a velocidade de transmissão dos pacotes a serem monitorados influencia o desempenho do processo de coleta. Para cada taxa de transferência, verificamos quantos pacotes foram realmente entregues à aplicação, quantos estabelecimentos de conexões TCP a libNIDS foi capaz de identificar a partir dos dados e quantos bytes de dados foram retirados da aplicação pela camada de remontagem de arquivos.

Os resultados desse experimento podem ser observados na figura 3.2, onde mostra-se o desempenho relativo de cada operação em relação ao total de pacotes, conexões e bytes de arquivos existentes no *trace* original. Como se pode ver, o desempenho da coleta de pacotes cai quase linearmente à medida que a velocidade aumenta. Mesmo com taxas de apenas 100 Mbps, um sistema sem otimizações já mostra perdas da ordem de 5 % na sua capacidade de capturar pacotes, atingindo 38 % a 500 Mbps.

Ainda na figura 3.2, pode-se observar o impacto que a perda de pacotes acarreta na capacidade do sistema de reportar o estabelecimento de conexões TCP. É interessante observar que a degradação na capacidade de identificar conexões TCP é muito mais pronunciado do que a que observamos na capacidade de captura de pacotes. Isso se deve ao fato de que um

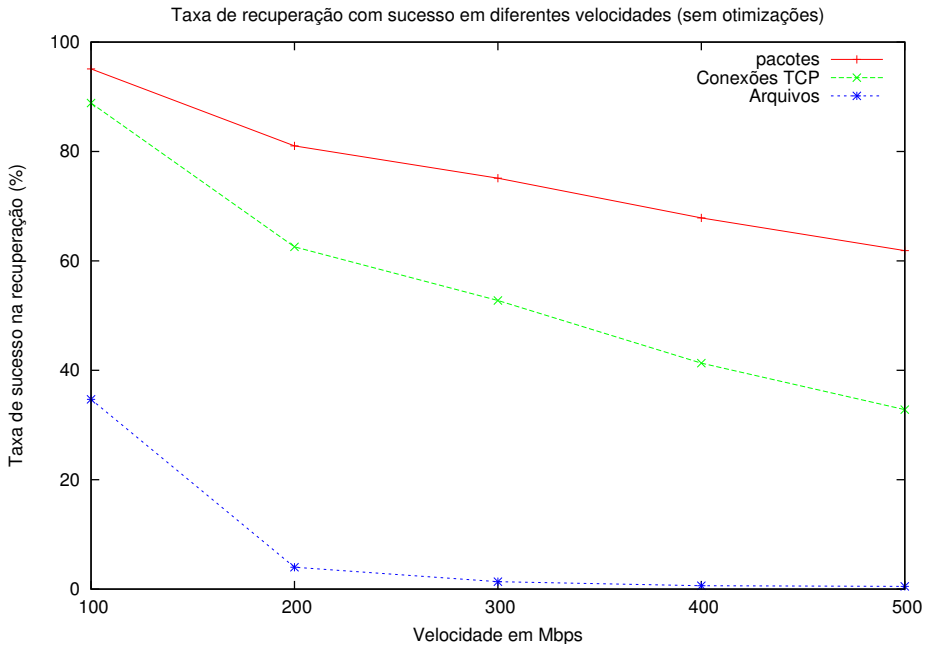

Figura 3.2: Impacto da taxa de chegada dos pacotes sobre a recuperação de pacotes, conexões TCP e arquivos

único pacote perdido durante o processo de estabelecimento da conexão implica na perda da conexão como um todo, logo uma perda tem impacto maior sobre a detecção da conexão, onde três pacotes TCP estão envolvidos (*three-way handshake*).

O efeito cumulativo das perdas se torna ainda mais notável no processo de remontagem de arquivos. Isso se deve ao fato de que o processo de recuperação de todos os de dados de uma conexão TCP exige que todos os pacotes da conexão sejam processados. Uma vez que uma perda ocorre no fluxo de dados, a libNIDS não é capaz de continuar o processo de remontagem, mesmo que outros pacotes sejam recebidos com sucesso para aquela conexão. Com isso, mesmo para velocidades acima de 200 Mbps o sistema já é incapaz de recuperar mais do que 1,35 % do volume de dados disponíveis em condições ideais.

A figura 3.3 mostra o total de pacotes capturados com sucesso para as quatro configurações testadas e para cada um dos quatro estágios. Nesse caso, como mencionado anteriormente, a carga foi gerada a 500 Mbps. Na legenda, *HT* se refere ao uso de *HyperThreading* e *NAPI* se refere ao uso ou não de *polling* de pacotes no *driver* da placa de rede. A numeração das etapas segue aquela usada na seção 3.4.1: (1) só captura de pacotes, (2) remontagem das conexões TCP, (3) processamento do protocolo da aplicação (eDonkey) e (4) remontagem dos arquivos trocados entre usuários.

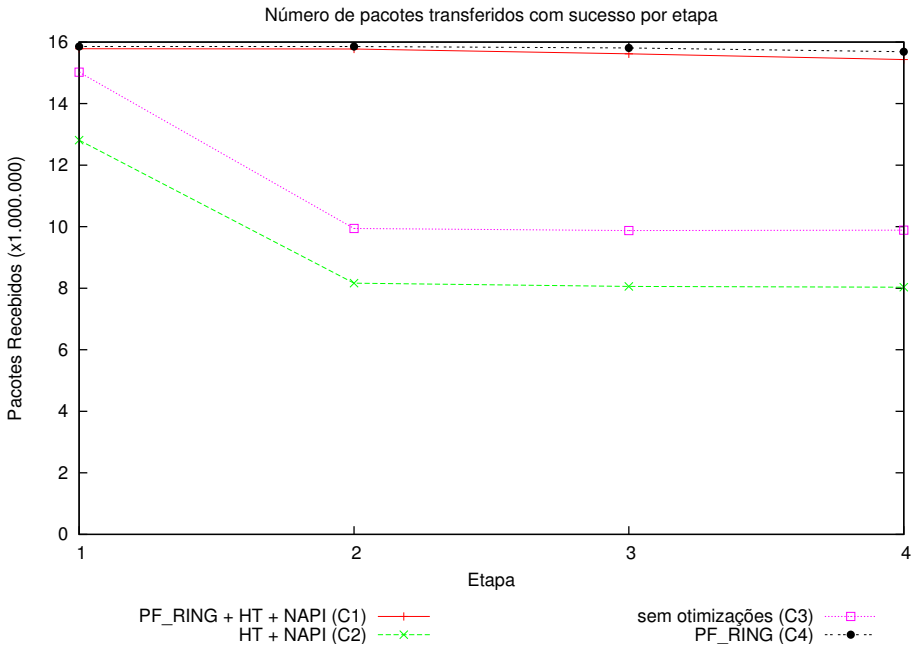

Figura 3.3: Quantidade de pacotes processados com sucesso

Primeiramente, é interessante notar o impacto negativo do uso do recurso de *HyperThreading* do Pentium 4: o desempenho de configurações com esse recurso habilitado são sempre inferiores àquelas sem o recurso. Acreditamos que o fato de não haver um uso intensivo nem de CPU, nem de *multithreading* e nem de paralelismo seja o motivo pelo qual o uso de HT não foi benéfico nesse caso.

Outro elemento interessante é que o processo de remontagem das conexões TCP pode ser apontado como a tarefa mais cara no sistema. Isso é visível pela queda acentuada ao se incluir o estágio da libNIDS, enquanto as curvas praticamente permanecem estáveis após esse ponto. Apesar de a tarefa de remontar as conexões e processar as máquinas de estado TCP ser realmente importante, ela não é em si a responsável pela queda de desempenho. O problema nesse caso deve ser atribuído ao baixo desempenho da interface do sistema operacional sem o *patch* PF_RING para a transmissão dos pacotes. Esse impacto se torna mais visível quando a libNIDS tenta processar o conteúdo de cada pacote, o que não ocorre no caso do programa de testes que apenas verifica se o pacote foi recebido.

Como pode ser observado, é nítida a diferença de desempenho entre as configurações com e sem o *patch* PF_RING. Na primeira etapa, onde é feita apenas a captura do pacote, sem processamento posterior, a pior configuração, sem o *patch* e com HT e NAPI (C2), chega a perder 19 % dos pacotes que chegam à interface. Mesmo a configuração sem nenhuma

Configuração	Conexões	Arquivos
PF_RING + HT + NAPI (C1)	97 %	57 %
HT + NAPI (C2)	21 %	0,4 %
sem otimizações (C3)	33 %	0,5 %
PF_RING (C4)	99 %	71 %

Tabela 3.2: Desempenho das várias configurações a 500 Mbps

otimização, ou seja, a configuração sem HT e sem o *patch* (C3), também já inicia com perdas significativas: 5 %. Na última etapa, a pior configuração sem o *patch* (C2) chega a perder 49 % dos pacotes, enquanto a pior configuração com o *patch* perde apenas 3 %.

Finalmente, na tabela 3.2, pode-se observar que as configurações com PF_RING são aquelas que conseguem os melhores valores para a capacidade de recuperar conexões e arquivos do tráfego monitorado. O melhor resultado na recuperação de arquivos com o PF_RING, 71 %, é mais de 100 vezes superior ao que é obtido pelo sistema sem otimizações.

3.4.3 Conclusões

Como pode ser visto, a arquitetura resultante atende aos objetivos especificados, tendo demonstrado a capacidade de recuperar o estado das aplicações P2P consideradas. Os resultados mostram que o uso de otimizações 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 nosso sistema de 4 camadas.

3.5 Sumário

É possível conceber um sistema capaz de realizar monitoração passiva de tráfego com recuperação de estado em tempo real e capaz de lidar com enlaces de alta velocidade apenas com *hardware* comum (COTS) e *software* de código aberto?

Nesse capítulo, salientamos alguns dos problemas que devem ser abordados para a concepção e operação de tal sistema. Fizemos considerações sobre as limitações das soluções baseadas em *software* e enumeramos os requisitos necessários para recuperar do tráfego coletado os estados das conexões nele existente.

Com base nessas considerações, apresentamos e detalhamos o *Palantír*, um sistema capaz de realizar a monitoração passiva de tráfego com recuperação de estado em tempo real.

Detalhamos os elementos utilizados em sua concepção e explicitamos as razões que nos levaram à escolha destes elementos.

Finalmente, realizamos dois experimentos no quais observamos o desempenho desse sistema e de sistemas de código aberto e *hardware* comum (COTS) sem otimizações. Como vimos, as ferramentas utilizadas para a construção do sistema se mostraram bastante satisfatórias e, através do seu uso, o sistema obteve um desempenho 99 % ao que poderia ser obtido utilizando abordagens tradicionais.

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., 2003]. 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.

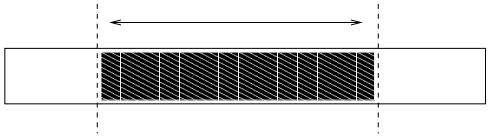

Figura 4.1: Modelo de transferência segmentada

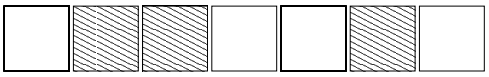

Figura 4.2: Modelo de transferência fragmentada

4.2.2.1 Transferência 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.

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

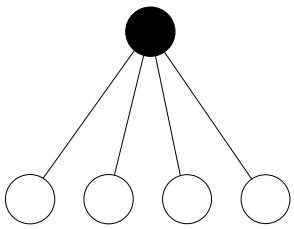

Figura 4.3: Topologia de um sistema centralizado

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 4.3.

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 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

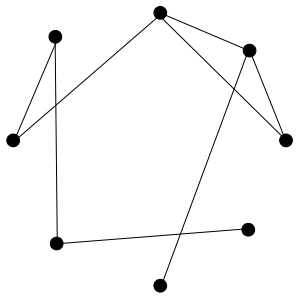

Figura 4.4: Topologia de um sistema descentralizado desestruturado

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 4.4. 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 as 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 descentralizadas 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 4.5:

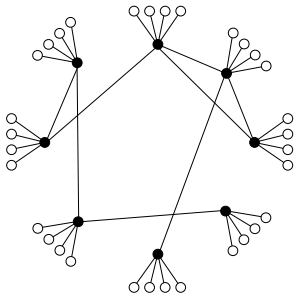

Figura 4.5: Topologia de um sistema descentralizado semi-estruturado

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 ar-

quivos 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 outros 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 subseções 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 como 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., 2005]. 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, a rede eDonkey possuía apenas um cliente oficial e utilizava um protocolo proprietário e fechado. Contudo, de maneira similar ao que o ocorreu com o KaZaa, o protocolo desta rede sofreu engenharia reversa e, 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. 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 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 nos 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 (28 %) ocorre no seu porto tradicional [Sen et al., 2004]. 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 esse monitor foi desenvolvido por Bruno Grossi [Grossi, 2005].

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 e depuração desse monitor relativamente mais complicada. Apesar da existência de documentos descrevendo o formato dessas estruturas, não foi raro encontrarmos estruturas cujos valores semânticos não eram conhecidos nem documentados. 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. Ela é 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 apresenta diversas funcionalidades que justificam o seu uso. Ela pode ser utilizada, por exemplo, para a averiguação de localidade de referencia entre os recursos trocados entre duas redes P2P distintas. Esse uso será melhor detalhado no capítulo seguinte. Além disso, a remontagem era peça fundamental para um sistema de *caching* oportunístico de tráfego P2P. Apesar de originalmente parte de um outro projeto, foi o desenvolvimento desse sistema de *cache* que acabou culminando no trabalho do qual surgiu o *Palantír*.

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.

À 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 arma-

zenados em disco. Isso é necessário tanto para melhorar o desempenho, evitando que octetos já vistos sejam regravados no disco, desperdiçando tempo e recursos, como para determinar se um determinado recurso já fora remontado completamente ou não.

Para que o controle de octetos já gravados fosse o mais eficiente possível, foi elaborada 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 $O(\log n)$.

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 necessário 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

Cada rede utiliza um mecanismo de identificação de usuários e arquivos distinto. 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].

4.5 Sumário

Nesse capítulo descrevemos os mecanismos de localização, identificação e transferência de arquivos utilizados pelas aplicações P2P mais comuns. Em especial, focamos em como as redes P2P monitoradas, a KaZaa e a eDonkey, utilizam tais mecanismos, expondo as particularidades de cada uma dessas redes e dos mecanismos por elas adotados.

Além disso, comentamos sobre como as peculiaridades de cada rede influíram no processo de elaboração de seus respectivos monitores, no processo de monitoração, de remontagem de arquivos e de identificação de usuários.

Como observamos, esses dois últimos pontos foram os mais afetados pelas diferenças entre as duas redes, o que acarreta em certos compromissos na forma com que se realizará a caracterização do tráfego obtido através da aplicação do *Palantír*.

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.

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

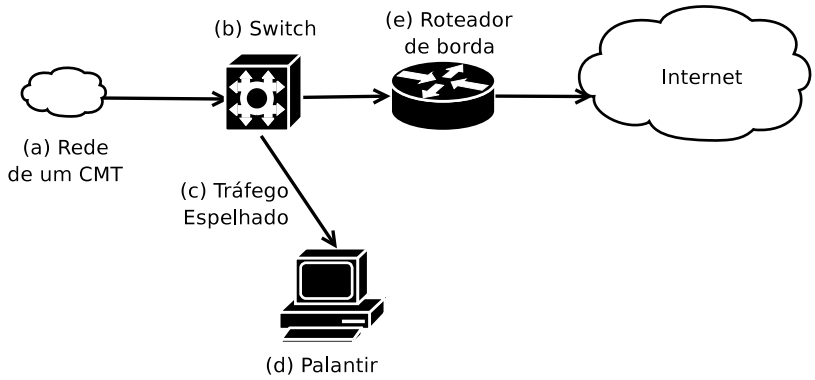

Figura 5.1: Diagrama do ambiente de coleta

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

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 A caracterização

Os dados obtidos através da coleta foram sumarizados na tabela 5.1. Deve-se observar que o intervalo de coleta para ambas as aplicações foi o mesmo, bem como a população potencial de usuários do provedor de cada aplicação.

Antes de prosseguirmos com a apresentação dos resultados da caracterização, algumas considerações sobre os dados obtidos na coleta apresentados na tabela 5.1 devem ser feitas. Como pode-se observar, o tráfego KaZaa monitorado é duas ordens de grandeza menor do que o total do tráfego eDonkey monitorado. Isso se deve principalmente ao fato de que, no processo de monitoração, coletou-se apenas tráfego nos portos tradicionais de ambas as aplicações. Devido ao fato do KaZaa utilizar portos aleatórios para as suas comunicações, parte do tráfego total desse protocolo não foi observado pelo nosso processo de monitoração. Todavia, existem trabalhos que afirmam que o tráfego KaZaa no seu porto tradicional, o porto TCP 1214, representa 28 % de todo o tráfego devido a essa aplicação [Sen et al., 2004].

Outro fator que deve ser levado em consideração é que o tamanho médio dos recursos na rede eDonkey é maior do que o encontrado na KaZaa, como veremos a seguir.

Além disso, e importante salientar que essa informação sobre o tamanho dos recursos trocados não está disponível da mesma forma para as duas aplicações monitoradas, como comentado na seção 4.4.3. Apesar do tamanho do arquivo ser informado em todas as

transferências KaZaa, o mesmo não ocorre no eDonkey. Nessa última aplicação P2P essa informação é obtida através de outros mecanismos, como através da busca junto aos servidores eDonkey. A extração desse tipo de informação diretamente dos servidores dessa aplicação através de um coletor automático (*crawler*) não é eficiente, devido ao uso, nos servidores, de políticas de racionamento de requisições por clientes. Nos nossos estudos, pra poder sanar essa deficiência, estimamos o tamanho dos arquivos eDonkey como a maior posição trocada de um dado arquivo.

Finalmente, deve-se observar que a quase totalidade dos gráficos apresentados a seguir está em escala logarítmica. Tal apresentação possibilita ressaltar algumas informações que, numa escala normal, não ficariam evidenciadas.

5.2.1 Métricas e metodologia

Através do uso do *Palantír* do provedor de banda Larga, obtivemos uma rica coleção de dados referentes ao tráfego das duas aplicações P2P de troca de arquivo monitoradas.

Como mencionado no capítulo anterior, as aplicações P2P observadas, KaZaa e eDonkey, possuem mecanismos de transferência de recursos que utilizam uma técnica conhecida como *swarming*. Essa técnica busca acelerar a transferência de recursos através do uso de um grande número de conexões independentes, possivelmente paralelas, para transferir diferentes partes (ou fragmentos) do recurso desejado. O seu uso, todavia, dificulta o processo de monitoração pois precisa-se monitorar várias conexões distintas para se observar a transferência de um único recurso por um único usuário. Por esse motivo, utilizou-se um processo hierárquico composto de quatro níveis para analisar e refinar os dados obtidos através da monitoração:

Fragmentos: cada conexão foi analisada e, naquelas onde ocorria a transferência de um recurso, quer em sua totalidade ou quer de seus fragmentos, contabilizava-se como essa conexão contribuía para a obtenção deste recurso pelo seu usuário, para o volume total de *bytes* trocados devido a este recurso e devido ao usuário solicitante deste recurso.

Recursos: A partir da consolidação dos dados referentes às diversas conexões pode-se obter informações referentes aos recursos trocados pelas aplicações P2P como, por exemplo, a sua popularidade.

Sessões: A transferência de diversos recursos para um único usuário pode ocorrer num mesmo intervalo de tempo. Para observar como se processam essas transferências

conjuntas, elas foram reunidas em sessões independentes, possuindo cada um um intervalo mínimo de 30 minutos entre si. Para cada uma dessas sessões analisou-se o número de recursos trocados e o volume de dados de cada uma delas, bem com a distribuição desses valores.

Usuários: Através da consolidação dos dados referentes à diversas sessões pode-se obter um conjunto de informações mais completo sobre o comportamento de um dado usuário: o volume de *bytes* que ele solicitou à rede, o número de recursos diferentes que trocou, etc. Os identificadores de usuários utilizados em cada rede P2P foram usados para diferenciar os usuários uns dos outros.

Através do processamento desses dados é possível relacionar dados sobre o comportamento de vários usuários e sobre como vários arquivos foram trocados através desses sistemas. Essa metodologia de caracterização também permite avaliar a carga dessas redes P2P sob diferentes perspectivas. Além disso disso, realizou-se uma análise da localidade de referência entre as duas redes P2P monitoradas.

5.2.2 Fragmentos

A caracterização dos fragmentos busca quantificar quais 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.

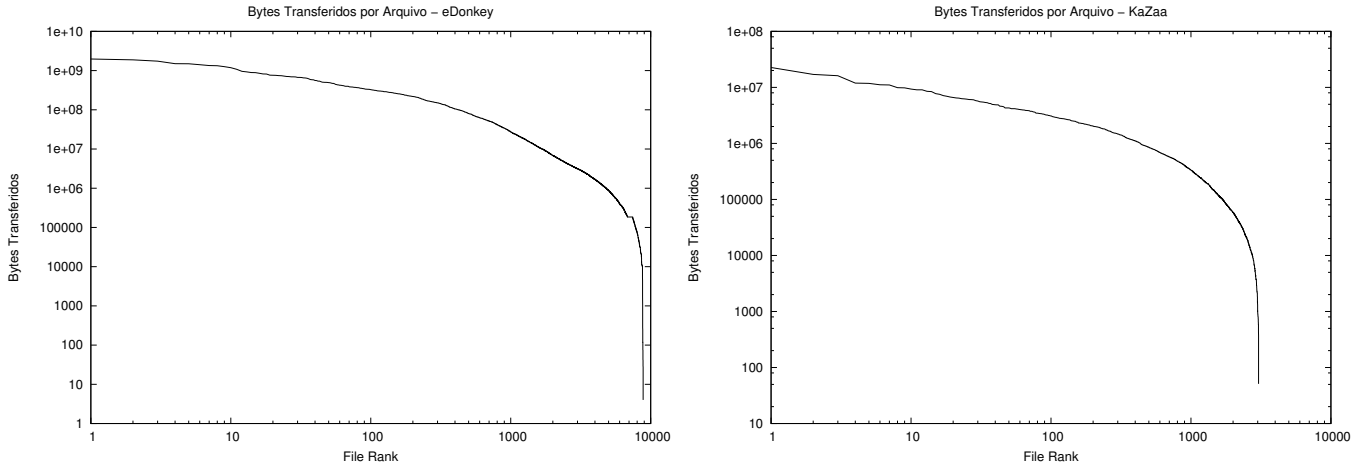

Figura 5.2: Distribuição de *bytes* transferidos por recurso.

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].

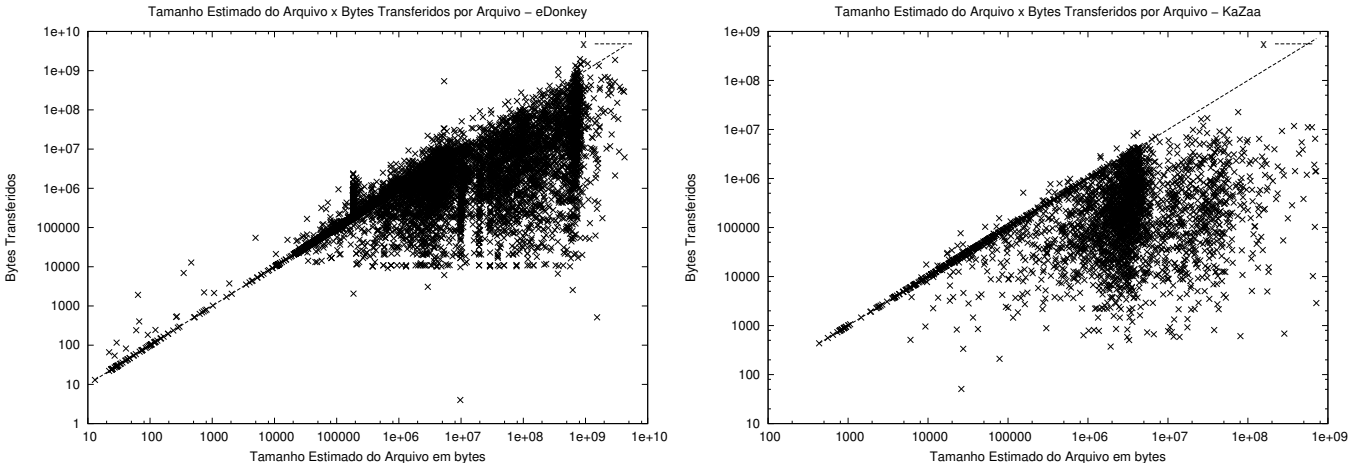

Figura 5.3: Correlação entre *bytes* transferidos e tamanho dos arquivos

A figura 5.3 apresenta a correlação entre os tamanhos estimados dos recursos em associação aos *bytes* transferidos associados desses mesmos recursos. Nessa figura, para cada recurso transferido, inserimos um ponto relacionando o seu tamanho (estimado) com o volume de dados em *bytes* observados devido à sua transferência. Para facilitar a observação dos valores apresentados, adicionou-se à figura uma linha tracejada indicando a área onde os valores para tamanho e *bytes* transferidos são os mesmos. Pode-se observar que eDonkey existem dois aglomerados, um próximo à faixa entre 1 e 10 MB e outro outro, mais disperso, entre a faixa de 10 MB e 1 GB. No KaZaa, o único aglomerado evidente está próximo à faixa entre 1 e 10 MB, exatamente a faixa de tamanho da maioria dos arquivos de MP3. Isso reforça as evidências de que a natureza dos recursos trocados na duas redes é diferente. Enquanto na rede eDonkey existe uma maior diversidade no tamanho dos recursos trocados, na rede KaZaa essa diversidade não está tão presente.

Pode-se observar também que, na maioria dos casos, os recursos não foram transferidos completamente, como conseqüência da forma fragmentada de transferência de arquivos.

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

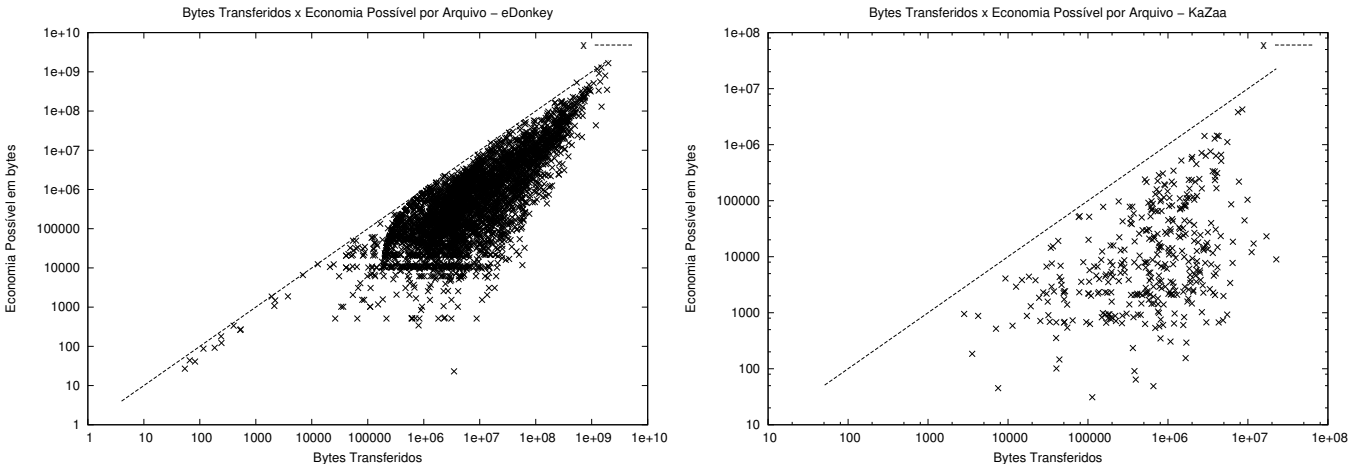

Figura 5.4: Correlação entre bytes transferidos e os ganhos potenciais de economia em largura de banda

tocolo, a correlação entre o número total de *bytes* transferidos para um determinado recurso e seu *bandwidth-savings* correspondente. Nessa figura, para cada recurso que permitia algum tipo de economia na largura de banda através do uso da suposta *cache* de tamanho infinito, inserimos um ponto relacionando o volume de dados em *bytes* observados devido à sua transferência com o volume possível de *bytes* que poderia-se economizar através do uso da *cache*. Para facilitar a observação dos valores apresentados adicionou-se à figura uma linha tracejada indicando a área onde os valores para *bytes* transferidos e *bytes* que podiam ser economizados são os mesmos. Pode-se observar que na rede eDonkey quanto mais dados de um arquivo forem transferidos, maiores são as chances de que transferências envolvendo esse arquivo gerem economia de largura de banda — quanto maior o volume transferido, maior é a concentração dos pontos próximo à linha tracejada. No KaZaa o mesmo não ocorre. Pode-se observar que os arquivos na figura 5.4 estão mais dispersos: mesmo arquivos cujas transferências geram um grande volume de dados não propiciam uma grande economia de largura de banda. Tais fatos corroboram o argumento de que que o mecanismo de transferência de recursos utilizados pela rede eDonkey facilita a disseminação de arquivos de forma mais eficiente do que o utilizado pelo KaZaa.

5.2.3 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

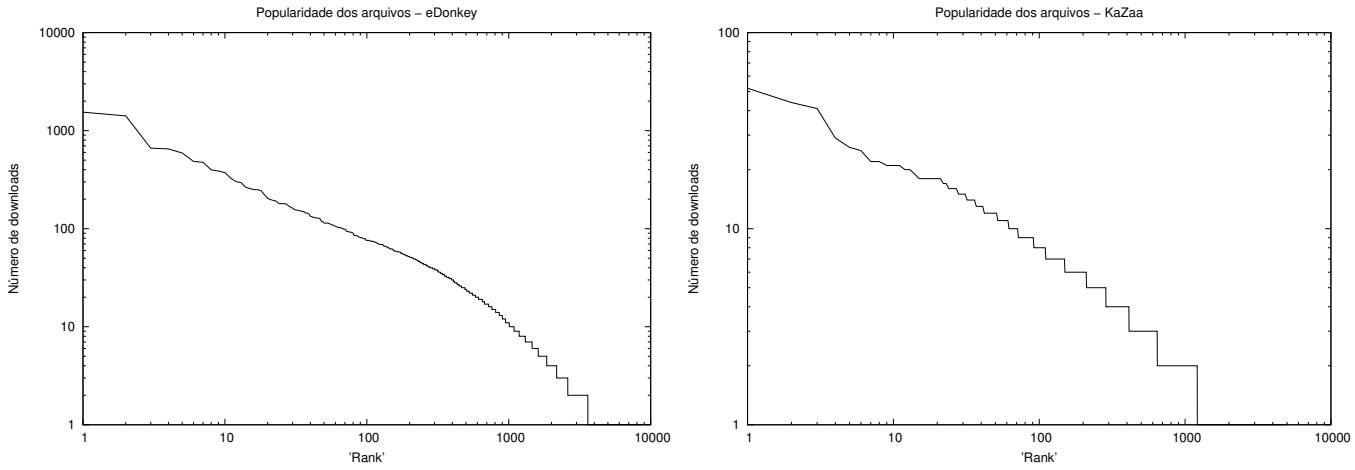

Figura 5.5: Popularidade dos Recursos

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.

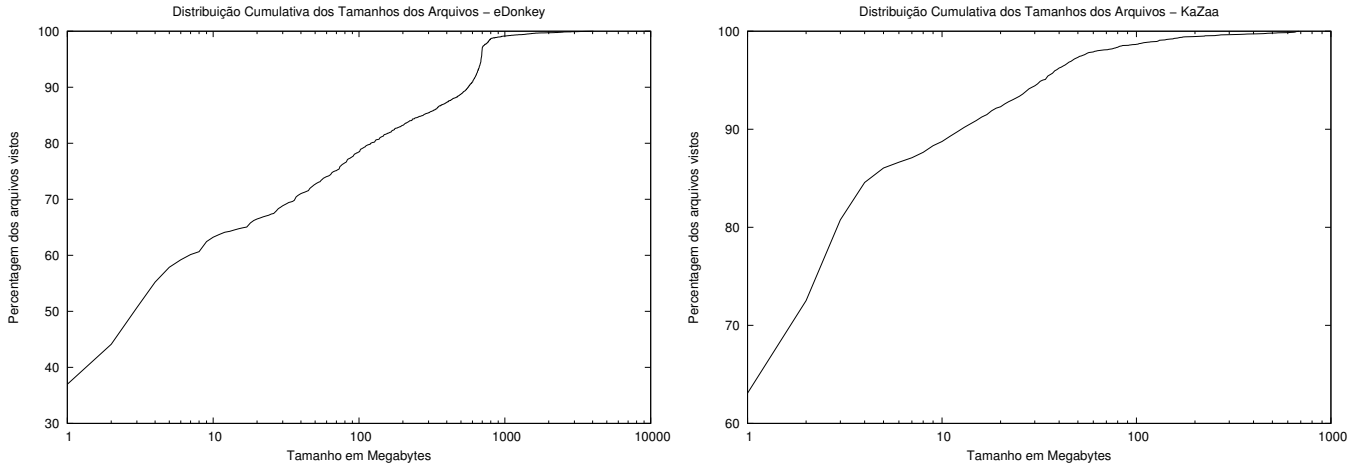

Figura 5.6: Distribuição cumulativa dos tamanhos dos arquivos

O resultado da análise da distribuição dos tamanhos dos arquivos no tráfego das duas aplicações pode ser observado na figura 5.6. Nela, pode-se observar a distribuição cumulativa dos tamanhos dos arquivos. Nessa distribuição os diferentes tamanhos foram separado em função do seu tamanho, com divisões (*bins*) de 1 MB. Vê-se claramente que pelo menos 80 % dos recursos trocados atavés do KaZaa possuem tamanhos inferiores a 10 MB. Em contraste, a mesma quantidade de recursos no eDoney é menor do que 100 MB, o que deixa bem claro a diferença 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, 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.

5.2.4 Sessões

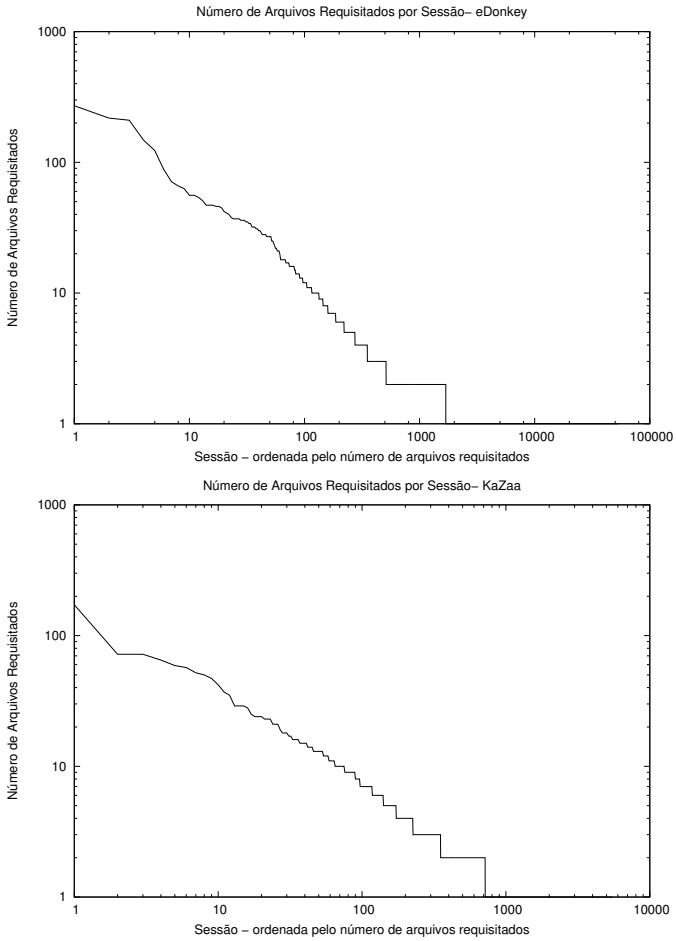

Figura 5.7: Distribuição do número de recursos transferidos por sessão

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.

5.2.5 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.

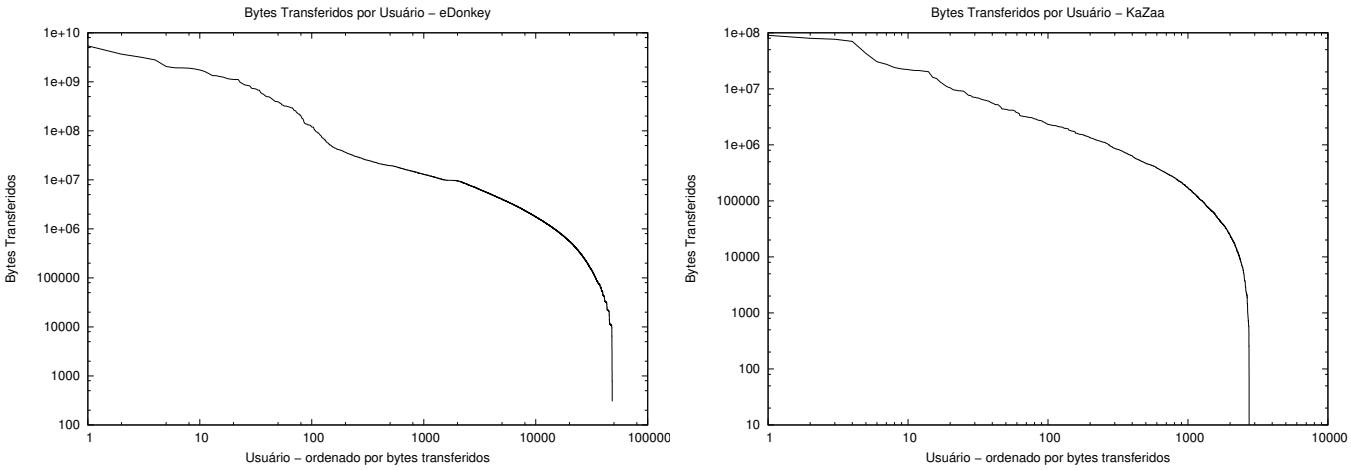

Figura 5.8: Distribuição do número de *bytes* transferidos por usuário

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.

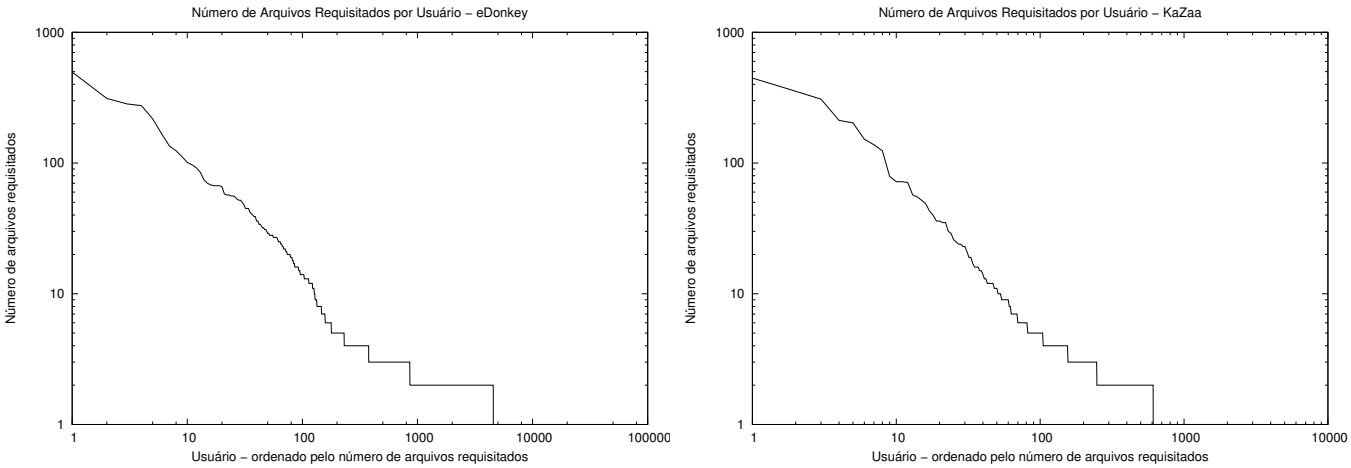

Figura 5.9: Distribuição do número de recursos solicitados por usuário

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

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

5.2.6 Localidade de referência inter-protocolo

As análises efetuadas anteriormente tanto da distribuição dos tamanhos dos recursos compartilhados como do volume de dados transferidos devido a recursos de diferentes tamanhos nos leva a crer que existe uma diferença na natureza dos recursos trocados em ambas as redes. Para verificar essa idéia realizamos uma análise da localidade de referência entre os recursos obtidos através das duas redes, 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 empregada 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.

5.3 Sumário

Nesse capítulo apresentamos alguns resultados gerados a partir do uso do *Palantír* em um provedor de banda larga local e descrevemos o ambiente onde o sistema foi instalado e algumas limitações do processo de coleta.

A análise dos resultados obtidos nos permitiu observar que existe uma significativa diferença tanto na natureza do tráfego gerado pelos usuários dos dois sistemas como dos recursos trocados nas duas redes. Essa diferença entre os recursos é re-enforçada através da análise de localidade de referência entre as duas redes.

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 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 apresentamos 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 desse sistema se mostraram bastante satisfatórias. Com a sua utilização, conseguimos construir um sistema capaz de, além da monitoração, realizar a remontagem dos arquivos trocados no tráfego monitorado. Mesmo à velocidade de 500 Mbps, considerada alta para os objetivos propostos, o sistema trabalha com uma taxa de perda de pacote da ordem de 3,3 % e um desempenho 99 % superior ao que poderia ser obtido utilizando abordagens tradicionais. Mesmo operando nessa condição de carga extrema, o sistema ainda é capaz de recuperar 70 % dos dados observados nos nossos experimentos. Apesar desse valor não ser o ideal, ele é mais de 100 vezes superior ao que pode ser obtido utilizando-se um sistema não otimizado.

Outra contribuição do trabalho foi a identificação dos principais pontos de contenção no processo de coleta. Verificamos que grande parte das perdas em um sistema de coleta desse tipo são devidas ao custo da passagem dos dados através da fronteira do sistema operacional entre o modo usuário e o modo protegido do *kernel*. Com a utilização de uma atualização do *kernel* Linux que reduz o custo da transferência de dados para o modo usuário o sistema

passou a apresentar um desempenho bastante superior ao original.

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. A análise dos resultados obtidos a partir desses esforços nos permitiu observar que existe uma significativa diferença tanto na natureza do tráfego gerado pelos usuários dos dois sistemas como dos recursos trocados nas duas redes. Essa diferença entre os recursos é re-enforçada através da análise de localidade de referência entre as duas redes, que se apresenta baixa mas significativa. Os valores encontrados para a popularidade de recursos na caracterização são condizentes com os apresentados em estudos anteriores.

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 Melhorias 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 para 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 para 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 capturados. 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.

Finalmente, pretendemos instrumentar o *kernel* para identificar os pontos que mais con-

somem recursos do sistema (*overhead*), comparar o desempenho desse tipo de tarefa entre sistemas operacionais diferentes e avaliar outras técnicas de implementação, como o uso de uma máquina de estados TCP simplificada ao invés da libNIDS e o aproveitamento de recursos de hardware disponíveis nas placas de rede modernas (TCP *offloading* e filtragem de pacotes diretamente na interface de rede, por exemplo).

6.2.2 Outras redes

Além 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 do desenvolvimento do trabalho 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]. Desta forma, vislumbramos como uma possível extensão deste trabalho a implementação de um processador do protocolo BitTorrent para o *Palantír* e a execução de trabalhos de caracterização do tráfego devido a esse sistema no provedor.

6.2.3 Análise do tráfego 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 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 interessante que poderia ser construída a partir da remontagem dos recursos presentes no tráfego monitorado é um "*cache* oportunístico". Apesar de possuir similaridades entre tal aplicação e entre *caches* transparentes, existem algumas diferenças fundamentais entre esses dois tipos de *caches*.

Sistemas de *caches* transparentes interceptam, com o auxílio de um roteador ou dispositivo similar, requisições de clientes de uma determinada rede a servidores e participam, mesmo que sem o conhecimento das partes envolvidas, do processo de obtenção

dos recursos. Desta forma, a *cache* tem acesso direto a recursos obtidos pelos clientes de uma dada rede e pode facilmente povoar a sua base de recursos. Quando uma requisição envolver um recurso que já disponível na base de recursos, a *cache* retornará diretamente uma cópia desse recurso ao cliente. Falhas de operação da *cache*, no entanto, podem atrapalhar ou impossibilitar a operação dos clientes que sequer sabem de sua operação e não tem como contorná-lo. Além disso, tais sistemas violam o argumento fim-a-fim [Barish and Obraczka, 2000, Saltzer et al., 1984].

Um "*cache* oportunístico", por outro lado, não intercepta as requisições vindas dos clientes nem participa do processo de obtenção de recursos. Ao invés disso, o sistema observa o tráfego de uma rede através de mecanismos de monitoração passiva de tráfego e recupera desse tráfego os recursos trocados. Esse tipo de *cache*, de maneira similar ao que ocorre com as *caches* transparentes, também povoará a sua base de recursos apenas com recursos obtidos pelos seus clientes. Por outro lado, falhas de operação dessas *caches* não causarão danos à operação dos clientes. Associado a algum mecanismo de divulgação de recursos, esse tipo de *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 o sistema tem 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 dessa dinâmica 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 transferência fragmentada vêm ganhando, é muito pouco. Desta forma, vislumbramos como uma possível extensão desse trabalho um estudo aprofundado dessa dinâmica.

Referências Bibliográficas

[edo, 2004] (2004). eDonkey home page. http://www.edonkey2000.com, último acesso em Abril de 2005.

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

[emu, 2005] (2005). eMule project home page. http://www.emule-project.net, último acesso em Abril de 2005.

[Androutsellis-Theotokis and Spinellis, 2004] Androutsellis-Theotokis, S. and Spinellis, D. (2004). A survey of peer-to-peer content distribution technologies. *ACM Computing Surveys*, 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 conference on Measurement and modeling of computer systems*, páginas 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. In *Proc. of the 1st Symposium on Operating System Design and Implementation*, páginas 115–123. 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 packet filter architecture. In *SIGCOMM '99: Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, páginas 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 sistemas 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 2003)*, Lecture Notes in Computer Science, páginas 256–267. Springer.

[Bos et al., 2004] Bos, H., de Bruijn, W., Cristea, M., Nguyen, T., and Portokalidis, G. (2004). FFPF: Fairly fast packet filters. In *Proceedings of 6th Symposium on Operating System Design and Implementation (OSDI 2004)*.

[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*, páginas 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*, páginas 463–471.

[Chankhunthod et al., 1996] Chankhunthod, A., Danzig, P. B., Neerdaels, C., Schwartz, M. F., and Worrell, K. J. (1996). A hierarchical internet object cache. In *USENIX Annual Technical Conference*, páginas 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*, páginas 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 Mainstream, 12th International Conference on Field-Programmable Logic and Applications*, páginas 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.

[Cisco Systems Inc., 2002] Cisco Systems Inc. (2002). NetFlow services and applications - white paper. http://www.cisco.com/warp/public/cc/pd/iosw/ioft/neflct/tech/napps_wp.htm, último acesso em Julho de 2005.

[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., McGregor, A., 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, Berkeley, CA, USA.

[Cáceres et al., 1998] Cáceres, R., Douglis, F., Feldmann, A., Glass, G., and Rabinovich, M. (1998). Web proxy caching: the devil is in the details. *SIGMETRICS Performance Evaluation Review*, 26(3):11–15.

[Degioanni et al., 2003] Degioanni, L., Baldi, M., Risso, F., and Varenni, G. (2003). Profiling and optimization of software-based network-analysis applications. In *SBAC-PAD '03: Proceedings of the 15th Symposium on Computer Architecture and High Performance Computing*, página 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 commodity hardware. In *IMC04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, páginas 233–238, New York, NY, USA. ACM Press.

[Deri, 2003] Deri, L. (2003). Passively monitoring networks at gigabit speeds using commodity hardware 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 Services (E2EMON)*.

[Desai, 2002] Desai, N. (2002). Increasing performance in high speed NIDS. http://www.linuxsecurety.com, último acesso em Abril de 2005.

[Elson, 2005] Elson, J. (2005). tcpflow: TCP flow recorder. http://www.circlemud.org/ jelson/software/tcpflow/, último acesso em Junho de 2005.

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

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

[Ethereal, 2005] Ethereal (2005). Ethereal home page. http://www.ethereal.com/, último acesso em Julho de 2005.

[flowgrep, 2005] flowgrep (2005). flowgrep's homepage. http://www.monkey.org/ jose/software/flowgrep/, último acesso em Junho de 2005.

[Grossi, 2005] Grossi, B. E. (2005). Estudo do modelo de computação orientada a serviços e sua aplicação a um sistema de mineração de dados. Master's thesis, Universidade Federal de Minas Gerais (UFMG), Departamento de Ciência da Computação. (em português).

[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 workload. In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, páginas 314–329. ACM Press.

[Harren et al., 2002] Harren, M., Hellerstein, J. M., Huebsch, R., 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 on Peer-to-Peer Systems*, páginas 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 Measurement and modeling of computer systems*, páginas 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*, páginas 13–18. USENIX.

[Hoßfeld et al., 2004] Hoßfeld, T., Leibnitz, K., Pries, R., Tutschku, K., Tran-Gia, P., and Pawlikowski, K. (2004). Information diffusion in eDonkey filesharing networks. In *Australian Telecommunication Networks and Applications Conference (ATNAC 2004)*, página 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*, páginas 267–271. ACM.

[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)*, páginas 121–126.

[Izal et al., 2004] Izal, M., Urvoy-Keller, G., Biersack, E. W., Felber, P., Hamra, A. A., and Garcés-Erice, L. (2004). Dissecting bittorrent: Five months in a torrent's lifetime. In *PAM'2004, 5th annual Passive & Active Measurement Workshop*, páginas 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 Waldhorst, O. P. (2004). Characterizing the query behavior in peer-to-peer file sharing systems. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, páginas 55–67. ACM Press.

[Kruegel et al., 2002] Kruegel, C., Valeur, F., Vigna, G., and Kemmerer, R. (2002). Stateful intrusion detection for high-speed networks. In *SP '02: Proceedings of the 2002 IEEE Symposium on Security and Privacy*, página 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 Distribution*.

[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*, página 112, Washington, DC, USA. IEEE Computer Society.

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

[Liang et al., 2005] Liang, J., Kumar, R., Xi, Y., and Ross, K. W. (2005). 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 *Third International Workshop Peer-to-Peer Systems Workshop (IPTPS'04)*, volume 3279 of *Lecture Notes in Computer Science*, páginas 141–150. Springer.

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

[Maymounkov and Mazières, 2002] Maymounkov, P. and Mazières, D. (2002). Kademlia: A peer-to-peer information system based on the xor metric. 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 capture. In *Proc. of the Winter 1993 USENIX Conference*, páginas 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*, páginas 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 (LISA 2001)*, páginas 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 analysis. In *4th International Workshop on Peer-to-Peer Systems (IPTPS)*. LNCS.

[Qiu and Srikant, 2004] Qiu, D. and Srikant, R. (2004). Modeling and performance analysis of bittorrent-like peer-to-peer networks. In *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, páginas 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*, páginas 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)*, páginas 99–100. IEEE Computer Society.

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

[Roos et al., 2003] Roos, M., Willemson, J., and Laud, P. (2003). Improving the gnutella protocol against poisoning. In *Proceedings of the Seventh Nordic Workshop on Secure IT Systems - NordSec 2003*, páginas 185–194. Disponível em 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 '02)*, 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 application signatures. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, páginas 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, last access on Abril de 2005.

[Song, 2005] Song, D. (2005). dsniff's homepage. http://monkey.org/ dugsong/dsniff/, último acesso em Junho de 2005.

[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 applications. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, páginas 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 libpcap's homepage. http://www.tcpdump.org, último acesso em Junho de 2005.

[tcpreplay, 2005] tcpreplay (2005). tcpreplay's homepage. http://tcpreplay.sourceforge.org, último acesso em Junho de 2005.

[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, páginas 20–23.

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

[van der Merwe et al., 2000] van der Merwe, J., Cáceres, R., hua Chu, 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, http://www.usenix.org/events/sec01/invitedtalks/rooij.pdf, último acesso em Junho de 2005.

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

[Wojtczuk, 2005] Wojtczuk, R. (2005). libNIDS homepage. http://libnids.sourceforge.net, último acesso em Junho de 2005.

[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 messages. In *USENIX Winter*, páginas 153–165.