UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

IDENTIFICAÇÃO DE TRÁFEGO *BITTORRENT* COM FINS PERICIAIS UTILIZANDO WEKA

GERSON LUIZ HAUS

ORIENTADOR: DÍBIO LEANDRO BORGES

DISSERTAÇÃO DE MESTRADO EM ENGENHARIA ELÉTRICA
ÁREA DE CONCENTRAÇÃO INFORMÁTICA FORENSE E SEGURANÇA DA INFORMAÇÃO

PUBLICAÇÃO: PPGENE.DM - 110/2012

BRASÍLIA / DF: JUNHO/2012

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

IDENTIFICAÇÃO DE TRÁFEGO *BITTORRENT* COM FINS PERICIAIS UTILIZANDO WEKA

GERSON LUIZ HAUS

DISSERTAÇÃO DE MESTRADO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA, COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE PROFISSIONAL EM INFORMÁTICA FORENSE E SEGURANÇA DA INFORMAÇÃO.

APROVADA POR:

DÍBIO LEANDRO BORGES, Doutor, UnB
(ORIENTADOR)

FLAVIO ELIAS GOMES DE DEUS, Doutor, UnB
(EXAMINADOR SUPLENTE)

ANDERSON CLAYTON ALVES NASCIMENTO, Doutor, UnB
(EXAMINADOR INTERNO)

DATA: BRASÍLIA/DF, 6 DE JUNHO DE 2012.

FICHA CATALOGRÁFICA

HAUS, GERSON LUIZ
Identificação De Tráfego *BitTorrent* com Fins Periciais Utilizando WEKA [Distrito Federal] 2012. xxiv, 57p., 210 x 297 mm (ENE/FT/UnB, Mestre, Engenharia Elétrica, 2012).

Dissertação de Mestrado – Universidade de Brasília, Faculdade de Tecnologia. Departamento de Engenharia Elétrica.

1. Informática forense 2. *BitTorrent*
3. Identificação de fluxo 4. WEKA

I. ENE/FT/UnB. II. Título (Série)

REFERÊNCIA BIBLIOGRÁFICA

HAUS, G. L. (2011). Identificação de Tráfego *BitTorrent* com Fins Periciais Utilizando WEKA. Dissertação de Mestrado, Publicação PPGENE.DM - 110/2012, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 57p.

CESSÃO DE DIREITOS

NOME DO AUTOR: Gerson Luiz Haus
TÍTULO DA DISSERTAÇÃO: Identificação de Tráfego *BitTorrent* com Fins Periciais Utilizando WEKA.
GRAU/ANO: Mestre/2012.

Gerson Luiz Haus
Universidade de Brasília
Campus Universitário Darcy Ribeiro – Asa Norte
CEP 70.910-900 – Brasília – DF - Brasil

DEDICATÓRIA

A minha esposa, pela compreensão.

AGRADECIMENTOS

Ao meu orientador Prof. Dr. DÍBIO LEANDRO BORGES, pelo apoio como orientador deste trabalho e pelo constante apoio, incentivo, dedicação e amizade essenciais para o desenvolvimento deste trabalho e para o meu desenvolvimento como pesquisador.

Aos Professores das disciplinas e a todos que se dedicaram a apoiar o curso.

A todos os colegas do Mestrado em Informática Forense, pela amizade.

Ao colega Perito Criminal Federal HELVIO PEREIRA PEIXOTO pela iniciativa e desdobramento na viabilização do Mestrado em Informática Forense, sem o qual nada seria possível.

Aos membros da Coordenação do Curso de Mestrado Profissionalizante em Engenharia Elétrica com Ênfase em Informática Forense e Segurança da Informação, que possibilitaram essa parceria entre poder público e academia, permitindo um melhor combate à criminalidade em nossa era digital.

A todos, os meus sinceros agradecimentos.

O presente trabalho foi realizado com o apoio do Departamento Polícia Federal – DPF, com recursos do Programa Nacional de Segurança Pública com Cidadania – PRONASCI, do Ministério da Justiça.

RESUMO

IDENTIFICAÇÃO DE TRÁFEGO *BITTORRENT* COM FINS PERICIAIS UTILIZANDO WEKA

Autor: Gerson Luiz Haus

Orientador: Díbio Leandro Borges

Programa de Pós-graduação em Engenharia Elétrica

Brasília, junho de 2012

O trabalho descrito nesta dissertação objetiva identificação de tráfego de rede proveniente de interceptação telemática judicialmente autorizada utilizando WEKA (*Waikato Environment for Knowledge Analysis*).
Propõe-se o desenvolvimento de um método de identificação do tráfego de rede gerado pelo aplicativo P2P (*peer-to-peer) BitTorrent*. Com a identificação do fluxo de rede do *BitTorrent* podem ser obtidas informações periciais importantes tais como: provas de materialidade, delimitação geográfica dos locais para onde foram transferidos arquivos, entre outras informações. A proposta deste trabalho emprega o conjunto de ferramentas WEKA, com o uso do algoritmo J.48 (baseado no C4.5) e SVM (*Support Vector Machine*), para classificar o fluxo de dados que utilizou criptografia. Como resultado experimental, foram detectados até 97,03% do tráfego criptografado. Os resultados experimentais alcançados demonstram a viabilidade da utilização do WEKA para a identificação de tráfego do *BitTorrent*.

ABSTRACT

IDENTIFYING *BITTORRENT* TRAFFIC WITH EXPERT PURPOSES USING WEKA

Author: Gerson Luiz Haus

Supervisor: Díbio Leandro Borges

Programa de Pós-graduação em Engenharia Elétrica

Brasília, June of 2012

The work described in this thesis aims at identifying network traffic from interception telematics judicially authorized using WEKA (Waikato Environment for Knowledge Analysis).
It is proposed the development of a method for the identification of network traffic generated by P2P (peer-to-peer) application *BitTorrent*. With the identification of the *BitTorrent* network flow information can be obtained important expert such as: proofs of materiality, geographical boundaries of sites for which they have been transferred files, among other information. The proposal of this work employs the WEKA toolset, using J.48 (based on C4.5) and SVM (Support Vector Machine)algorithms, to sort the data flow that uses encryption. As a result, 97.03% of the encrypted traffic were detectThe experimental results achieved demonstrate the feasibility of using WEKA for identifying *BitTorrent* traffic.

Sumário

Índice de Tabelas

LISTA DE FIGURAS

LISTA DE SÍMBOLOS, NOMENCLATURA E ABREVIAÇÕES

API	A*pplication Programming Interface*
ARFF	*Attribute-Relation File Format*, ou Formato de Arquico Atributo-Relação
CTI	Coordenação de Tecnologia da Informação (do DPF)
CLIT	Cliente de Interceptação Telemática
DFI	*Deep Flow Inspection*, ou Inspeção Profunda em Fluxos
DHT	*Distributed hash table*, ou Tabelas hash distribuídas
DNS	*Domain Name System*
DPF	Departamento de Polícia Federal do Brasil
DPI	*Deep Packet Inspection*, ou Inspeção Profunda em Pacotes
eD2k	*eDonkey2000 network*
FTP	*File Transfer Protocol*
HTTP	*Hypertext Transfer Protocol*
HTTPS	*Hypertext Transfer Protocol Secure*
IDS	*Intrusion Detection Systems*, ou Sistema de Detecção de Intrusão
IP	*Internet Protocol*
ISP	*Internet Service Provider*, ou Provedor de Acesso à Internet
KDD	knowledge discovery in databases (Descoberta de Conhecimento em Banco de Dados
MLP	*Multilayer Perceptron*
NAT	N*etwork Address Translation*
P2P	*Peer-to-Peer*

PCAP	*Packet Capture*
RBF	R*adial Basis Function* (função de base radial)
RFC	*Request for Comments*, ou Pedidos de Comentários
SIT	Servidor de Interceptação Telemática
SMTP	*Simple Mail Transfer Protocol*
SVM	*Support Vector Machine*
TCP	*Transmission Control Protocol*
TI	Tecnologia da Informação
WEKA	Waikato Environment for Knowledge Analysis
WWW	*World Wide Web*

1. INTRODUÇÃO

A expansão da facilidade de acesso às tecnologias de informação e internet também trouxe o aumento dos crimes com utilização dessas tecnologias. O crescimento continua conforme (figura 1.1) o relatório da *International High Tech Crime Investigation Association* – HTCIA (HTCIA, 2010).

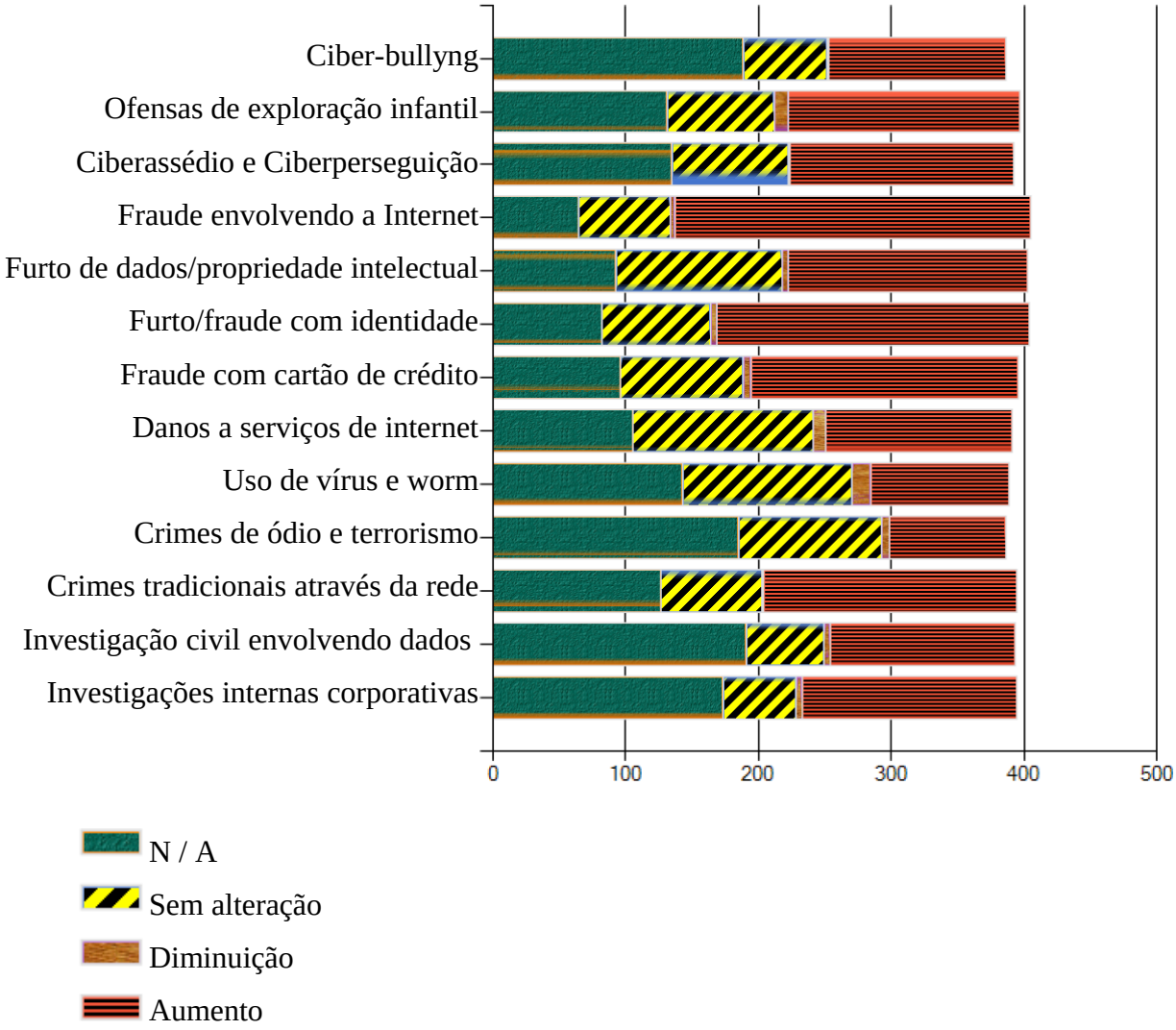

Figura 1.1: alterações nas categorias de ciber-crime em 5 anos (adaptado de HTCIA, 2010)

De acordo com Tanenbaum e Wetherall (2010), a utilização da Internet como ferramenta para compartilhamento ilícito de arquivos é um crime realizado principalmente por meio da comunicação não-herárquica do tipo par a par ou ponto a ponto (P2P).

Em uma pesquisa textual no Sistema de Criminalística do DPF realizada em Maio de 2012 observamos a ocorrência de 860 documentos com o termo "eMule" e 89 com o termo "*BitTorrent*", demonstrando que foram investigados quase 10 vezes mais casos com a utilização do primeiro em relação ao segundo.

Porém, de acordo com o *Internet Study* 2008/2009 (Ipoque, 2009) o *BitTorrent* é o protocolo número um em popularidade no mundo. A América do Sul, segundo o citado estudo, foi a única região que o *BitTorrent* não ficou em em primeiro lugar, tendo ficado em segundo lugar, com 20%, atrás do Ares, com 28% e à frente do eDonkey e do HTTP com 17% e 10% respectivamente (figura 1.2). O eDonkey é a rede (protocolo) utilizado pelo eMule.

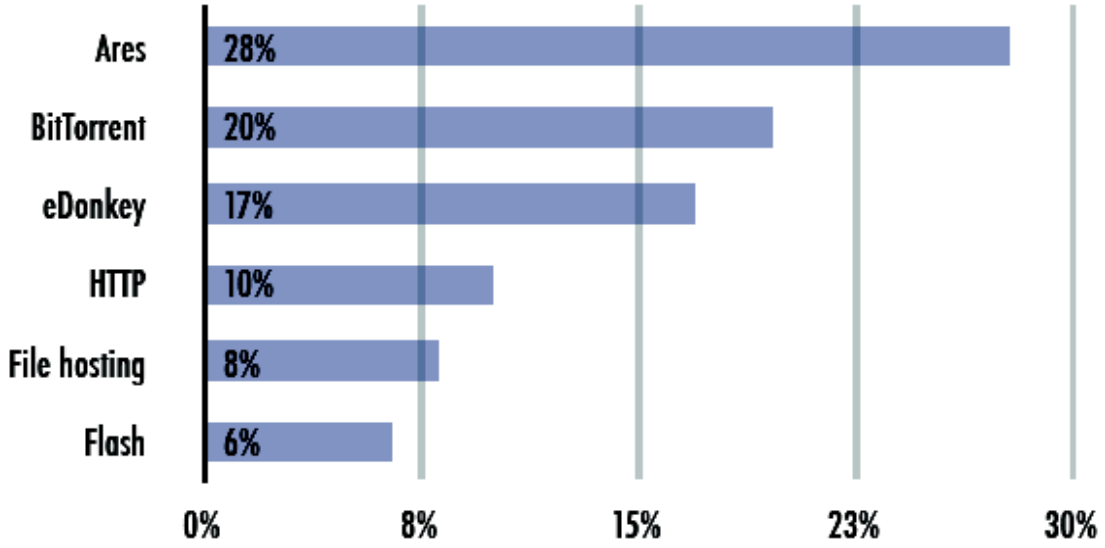

Figura 1.2: popularidade do tráfego de internet na América do Sul (Ipoque, 2009)

Uma das causas é o sucesso da Ferramenta Para Monitoramento de Redes P2P-EspiaMule (Dalpian & Benites, 2007) no monitoramento de arquivos compartilhados em redes peer-to-peer, ou ponto a ponto, principalmente sobre as redes utilizadas pelo eMule.

Não há uma ferramenta semelhante ao EspiaMule para utilização sobre as redes *BitTorrent*. Portanto, apesar da quantidade muito maior dos casos investigados referentes ao eMule em relação ao *BitTorrent* resta a dúvida se este segundo não é tão utilizado para o tráfego criminoso de arquivos ou se é por causa da falta de um monitoramento e investigação eficaz para este tipo de tráfego.

(Peron, 2012) desenvolveu e propôs uma infraestrutura central para recebimento de tráfego interceptado , autorizado judicialmente, juntamente com uma ferramenta para obtenção, tratamento, importação e análise deste tráfego com o intuito de automatizar o processo, facilitando o uso da interceptação de internet nas investigações. O SIT (Servidor de Interceptação Telemática) é a infraestrutura central para recebimento do tráfego e o CLIT (Cliente de Interceptação Telemática) é a ferramenta de obtenção, tratamento, importação e análise de tráfego.

A ferramenta CLIT dispõe dos seguintes filtros:

- Filtro WEB
- Filtro POP
- Filtro SMTP
- Filtro RTP
- Filtro MSN
- Filtro YMSG
- Filtro ICQ

O sistema está aberto para filtros externos, de acordo com as especificações em (Peron, 2012).

Lange (2011) propôs um método de identificação do tráfego de rede gerado pelo aplicativo peer-to-peer eMule empregando Redes Neurais Artificiais (RNA) para classificar o fluxo de dados que utilizou criptografia e heurística em caso contrário. Em suas experiências detectou 100% dos pacotes do conjunto de teste não criptografado do eMule e 86,03% do tráfego criptografado foi identificado pela RNA. Desta forma (Lange, 2011) demonstra a viabilidade da utilização de RNA para a identificação de tráfego do eMule, com a finalidade de servir como filtro externo para a ferramenta CLIT de (Peron, 2012).

Este trabalho pretende analisar a viabilidade da identificação de tráfego *BitTorrent* para fins de proporcionar a separação do mesmo no processo de interceptação de dados judicialmente autorizados, através da utilização da ferramenta WEKA.

1.1. ÁREAS DE APLICAÇÃO DOS RESULTADOS

O resultado deste trabalho possui aplicação específica dentro da área da informática forense, pois objetiva a obtenção de informação para análise de como foi utilizado o tráfego de rede em uma interceptação telemática autorizada judicialmente.

1.2. HIPÓTESE DE TRABALHO E METODOLOGIA

A Hipótese deste trabalho é que se pode obter informação que está oculta em uma grande massa de dados através da aplicação de metodologias e algorítimos de mineração de dados ou seja, a descoberta de conhecimento.

1.3. RESULTADOS ESPERADOS

Espera-se, a partir da hipótese aplicada, conseguir uma identificação eficaz do tráfego de rede não identificado pelo número da porta com características do protocolo *BitTorrent*.

Como objetivo específico este trabalho pretende avaliar algoritmos de classificação e mineração de dados na identificação de tráfego *BitTorrent*. Essa separação é importante para fins de proporcionar a identificação e análise do mesmo no processo de interceptação de dados judicialmente autorizados. A ferramenta WEKA será utilizada para testar os algoritmos de uma forma sistemática e publicamente testadas, tendo em vista que esta ferramenta além de possuir implementados os principais algoritmos de mineração de dados, é de domínio público e amplamente aceita pela comunidade de pesquisa em mineração de dados, propiciando para nós uma base segura e confiável para testes.

1.4. ORGANIZAÇÃO DA DISSERTAÇÃO

O restante deste trabalho está organizado em três partes: o Capítulo 2 contempla os trabalhos correlatos encontrados. A segunda parte, composta pelos Capítulos 3 e 4 apresenta as técnicas e os conceitos utilizados, obtidos através de pesquisa bibliográfica. Finalmente, nos dois últimos capítulos (5 e 6), são apresentados os experimentos feitos, com os resultados obtidos nos testes realizados e as conclusões decorrentes da análise destes experimentos.

Mais detalhadamente os capítulos restantes possuem os seguintes conteúdos:

- O Capítulo 2 relata breves resumos dos trabalhos onde pesquisou-se assuntos semelhantes ou de interesse ao trabalho presente;
- No Capítulo 3 são apresentadas informações referentes às redes P2P e mais especificamente o protocolo *BitTorrent*;
- O Capítulo 4 contém informações básicas sobre o conjunto de ferramentas WEKA;
- O Capítulo 5 apresenta os experimentos utilizados e os resultados obtidos;

- Finalmente, no capítulo 6 faz-se a comparação dos resultados e apresenta-se a conclusão do trabalho.

2. TRABALHOS CORRELATOS

Uma identificação dos artefatos forenses produzidos pelo sistema de compartilhamento de arquivos *BitTorrent* foi estabelecida por (Acorn, 2008).Este trabalho foca no processo de descoberta e análise dos vestígios deixados pela utilização dos aplicativos chamados clientes *BitTorrent*, enquanto que o trabalho desta dissertação pretende colaborar na descoberta e análise dos vestígios do tráfego *BitTorrent* durante a investigação do possível delito, quando se tem a oportunidade de arrecadar mais informações.

Em 2004 (Sen, Spatscheck, & Wang, 2004) mostrou resultados de que a identificação do protocolo *BitTorrent* naquela época era possível de ser feita tanto pela assinatura quanto pela porta padrão que era utilizada na época, o que não ocorre mais hoje em dia. Em virtude de que quase a totalidade do tráfego *BitTorrent* estar circulando encriptada atualmente (Markey, 2011; Yang, Li, Ji, & Zhu, 2012) esta abordagem perde sua eficácia. Por este motivo o trabalho ora apresentado toma por base as abordagens de descoberta do conhecimento nos elementos estatísticos dos pacotes do fluxo de dados (Yang et al., 2012).

(Pouwelse, Garbacki, Epema, & Sips, 2005) apresentou um estudo de medição de *BitTorrent* concentrado na disponibilidade, integridade e desempenho do download. Este trabalho auxiliou na compreensão de um sistema real de P2P ressaltando que tal sistema teve os mecanismos para atrair uma comunidade grande de usuários.

Em (D. Erman, Ilie, & Popescu, 2006) temos um estudo de modelagem e medição das características de sessão e mensagem do tráfego *BitTorrent*. Os resultados da medição são relatados na modelagem e análise de vestígios coletados na camada de rede de aplicação em um link da Blekinge Institute of Technology (BTH) e em um provedor de internet (ISP) local na Suécia. Vestígios de camada de ligação e *logs* (registros) dos aplicativo foram coletados para modelagem e análise usando uma infra-estrutura dedicada de medição desenvolvida no BTH para coletar tráfego P2P. Os resultados mostram que características da sessão podem ser modelados por uma distribuição hiper exponencial de segunda ordem, enquanto tamanhos e durações de sessão podem ser razoavelmente bem modelados por várias misturas da

distribuição Log-normal. Também foram observados tempos de resposta para ser modelado por uma mistura normal de Log dual, enquanto taxas de solicitação são modeladas como duas distribuições gaussianas.

(Bindal et al., 2006) examina uma nova abordagem para incentivar com que o tráfego de *BitTorrent* seja o mais local possível, em que um cliente escolhe a maioria de seus pares dentro do mesmo provedor (ISP). Usando simulações, mostram que a “seleção preferencial do vizinho” mantém o desempenho quase ideal para velocidade do tráfego *BitTorrent* em uma variedade de ambientes e, fundamentalmente, reduz o tráfego de inter-ISP diminuindo seu crescimento com o aumento do número de pares.

No trabalho de (Cuevas, Laoutaris, Yang, Siganos, & Rodriguez, 2009) eles tentam aprofundar e expandir a compreensão da localidade idealizada por (Bindal et al., 2006) reafirmando o seu potencial. (Cuevas et al., 2009) desenvolveu metodologias dimensionáveis que nos permitem processar um conjunto de dados enorme e responder perguntas como: "Qual é o mínimo e a máxima redução do tráfego através das centenas dos ISPs?", "quais são os limites de ganha-ganha para ISPs e seus usuários?" , entre outras.

Há grande quantidade de trabalhos focando a identificação de disseminação de conteúdo ilegal e/ou com violação de direitos autorais. (Bauer, McCoy, Grunwald, & Sicker, 2009) desenvolveu um *framework* ativo chamado *BitStalker* que identifica pares ativos e coleta evidências forenses concretas do envolvimento no compartilhamento de um arquivo específico. A eficácia desta abordagem foi avaliada através de um estudo de medição com torrents reais, grandes, consistindo de mais de 186.000 pares. Foi descoberto que enquanto os métodos de investigação da época produziam, pelo menos, 11% de falsos positivos, os mesmos são raros com sua abordagem ativa. (Bauer, McCoy, Grunwald, & Sicker, 2009) inicia sua identificação a partir de listas de usuários do tracker controlador. Cada ponto listado pelo servidor tracker é analisado ativamente para confirmar a sua participação no compartilhamento do conteúdo examinado com a finalidade de coletar provas forense concretas. A ferramenta BitStalker emite uma série de sondas leves que fornecem evidências cada vez mais determinante da participação ativa dos pares no compartilhamento de arquivos.

Nesta proposta ele foca-se em reduzir a identificação potencial de falsos positivos no monitoramento grandes enxames de *BitTorrent*. Enquanto o "sujeito" do trabalho de (Bauer et al., 2009) é o arquivo sendo compartilhado, pois a partir dele são localizados os usuário que o compartilham, nosso foco está no usuário que já foi identificado em uma parte anterior da investigação, sendo necessária a busca das evidências incriminatórias ou de inocência referente ao fato investigado.

(Dischinger et al., 2010) desenvolveu e implantou um sistema que melhora a transparência de rede denominado Glasnost, permitindo que os usuários de Internet comuns detectar se seus ISPs interferem ilegalmente no seu tráfego com diferenciação entre fluxos de aplicativos específicos. (Dischinger et al., 2010) identificou três principais desafios na criação de um sistema deste tipo:

(a) para atrair muitos usuários, o sistema deve ter baixa barreira de uso e gerar resultados em tempo hábil;

(b) os resultados devem ser resistentes ao ruído de medição e evitar falsas acusações de interferência, que pode afetar negativamente os negócios e a reputação dos ISPs, e

(c) o sistema deve incluir mecanismos para mantê-lo atualizado com as políticas de interferências dinâmicas nos ISPs em todo o mundo.

(Yang, Li, Ji, & Zhu, 2010) apresentaram uma abordagem global para identificar o tráfego de *BitTorrent* em tempo real. Aplicaram assinaturas de aplicativo para identificar o tráfego não criptografado. E para tráfego criptografado, propuseram um modelo de fluxo de mensagens de acordo com os *handshakes* do protocolo de encriptação (MSE) de fluxo de mensagem que é usado pelo *BitTorrent* para ofuscar o tráfego. Finalmente, propôs um método de pré identificação baseado na análise de sinalização do *BitTorrent*. Ele pode prever fluxos de *BitTorrent* e distingui-los, através do primeiro pacote de cada fluxo TCP com o sinalizador SYN. Os resultados indicam que a abordagem pode identificar o tráfego de *BitTorrent* no início do fluxo TCP.

(Yang et al., 2012) no mais recente trabalho correlato encontrado, relata as tendências mais atuais, confirmando que o método mais simples de identificação de tráfego, o baseado em mapeamento de porta é ineficaz diante da técnica de porta dinâmica. A abordagem baseada em assinatura é inútil quando enfrenta tráfego criptografado. No seu trabalho comenta que algumas abordagens que usam algoritmos de aprendizado de máquina e algoritmos de mineração de dados baseando-se em comportamentos de estatísticas ou host de fluxo precisam um processo demorado para formação ou cálculo e dificilmente podem ser usados na identificação em tempo real. (Yang et al., 2012) propõe uma abordagem de técnicas conjuntas consistindo de três sub métodos para identificar o tráfego de *BitTorrent*: assinaturas de aplicativo para identificar o tráfego não criptografado; e método baseado em mensagem de acordo com as características do protocolo de criptografia (MSE) do fluxo de mensagem. Também propõe um método pré-identificação baseado na análise de sinalização. Ele pode prever fluxos de *BitTorrent* e diferenciá-los mesmo com o primeiro pacote com flag SYN somente. Foi utilizado o cliente Vuze para rotular o tráfego *BitTorrent* em rastreamentos de tráfego real, que proporcionaram condições de fazer conjuntos de dados de valor de referência de alta precisão para avaliar a abordagem. Os resultados ilustraram a eficácia da abordagem, especialmente para aqueles fluxos que não têm assinaturas óbvias ou estatísticas de fluxo. Outro trabalho anterior que apresentou a utilização de abordagem com métodos mistos é (Dai, Yang, & Lin, 2010).

A classificação de tráfego de internet mais generalizada utilizando pela primeira vez o aprendizado de máquina foi encontrada no estudo de (J. Erman, Mahanti, & Arlitt, 2006). Porém o trabalho não foca nas especifidades do tráfego *BitTorrent.*

A identificação do protocolo *BitTorrent* na classificação de tráfego de rede tem sido estudada principalmente com a finalidade de redução forçada da utilização de banda por este. (Le & But, 2009) justifica sua pesquisa citando auxiliar os administradores de rede, pois o *BitTorrent* tem problemas na legalidade do seu conteúdo. Em seu artigo intitulado "*BitTorrent* traffic classification" quatro principais atributos foram usados no processo de classificação: payload mínimo, relação de pacotes pequenos, razão de pacotes grandes e

menor payload padrão .Os resultados quando usando todos os quatro atributos em um subfluxo do tamanho de 200 pacotes foram:

- 95,3% *BitTorrent* corretamente classificados;
- 99,1% FTP corretamente classificadas; e
- 97% de outros protocolos.

Neste ponto temos uma boa base do que já foi pesquisado e números para comparação com os resultados que apresentaremos no decorrer deste trabalho.

Passaremos, a seguir, no Capítulo 3, a um estudo mais específico do comportamento do protocolo BitTorrent.

3. *BITTORRENT*

As redes de computadores integraram a tecnologia da informação para possibilitar o compartilhamento de diversos tipos de recursos (Tanenbaun, 2003). Entre os recursos compartilhados os arquivos tem um local de destaque, pois eles podem possuir os mais diversos tipos de conteúdos: aplicativos, textos, áudio, vídeo etc.

Considerando que várias pessoas podem desejar determinado arquivo, o recurso da banda de rede do ponto que disponibiliza tal arquivo passa a ser um recurso restritivo. Quanto mais pessoas estiverem acessando o mesmo ponto, menor a eficiência da transferência.

Genericamente poderíamos considerar a equação 3.1:

$$BI = \frac{BD}{NU} \qquad \text{EQ (3.1)}$$

onde,

BI = Banda utilizada por cada usuário que acessa o arquivo;

BD = Banda total do ponto que disponibiliza o arquivo;

NU = Número de usuários que acessam o arquivo.

Dentre os aplicativos que utilizam a tecnologia P2P o *BitTorrent* destaca-se pela integridade do conteúdo compartilhado de acordo com (Pouwelse et al., 2005) que realizou um estudo considerando também as questões de disponibilidade, *flashcrowd* e desempenho de *download*.

Segundo Cohen, (2003) a finalidade do protocolo de compartilhamento de arquivos *BitTorrent* é possibilitar a transferência de arquivos de forma eficiente e sem sobrecarregar a conexão do ponto do que o disponibiliza. O arquivo a ser transferido é dividido em pequenas partes. Assim que os primeiros a requisitar o arquivo completam a transferência destas partes eles tornam-se fonte para os próximos que requisitam a transferência, retirando a carga que

seria do ponto que inicializou a disponibilização (Cohen, 2008). Desta forma, quantos mais nós estão baixando o arquivo mais nós também estarão servindo de fonte, aumentando a eficiência geral da transferência sem onerar o inicializador do compartilhamento, ao contrário da transferência via FTP (*File Transfer Protocol* – Protocolo de transferência de Arquivo) ou HTTP (*Hypertext Transfer Protocol* – Protocolo de transferência de Hipertexto) (Qiu & Srikant, 2004). Nos compartilhamentos via FTP ou HTTP quanto mais pontos realizando a transferência ao mesmo tempo pior o desempenho, tendo em vista que ocorre a divisão do recurso de capacidade de transferência (banda de *upload*). Esta importante vantagem é denominada escalabilidade (Cohen, 2003).

A seguir destaca-se os principais termos utilizados especificamente para compreensão da operação do protocolo *BitTorrent*:

- sanguessugas (*leeches*) - clientes que baixam arquivos;
- semeador (*seed* ou *seeder*) - um computador com uma cópia completa de um arquivo BitTorrent (é necessário pelo menos um computador "*seed*" para que o download de um BitTorrent funcione);
- enxame (*swarm*) - um grupo de computadores enviando (*upload*) ou recebendo (*download*) o mesmo arquivo simultaneamente;
- .torrent - um arquivo indicador que direciona seu computador para o arquivo que você deseja baixar;
- rastreador (*tracker*) - um servidor que gerencia o processo de transferência de arquivos *BitTorrent*.

3.1. O ARQUIVO TORRENT

Antes de iniciar um compartilhamento via *BitTorrent* há necessidade da criação de um arquivo que conterá determinadas informações do(s) arquivo(s) que se pretende compartilhar

(Fernandes & Fernandes, 2009). Este novo arquivo possui a extensão ".torrent" e daqui por diante será denominado apenas por torrent para diferenciar do arquivo alvo que se deseja compartilhar.

As especificações do torrent, de acordo com o site oficial (Cohen, 2008) , revelam que ele deve possuir, pelo menos, as seguintes informações codificadas em bencode[1]:

- *announce*: o endereço (URL[2]) de localização do tracker (servidor que manterá as informações sobre os clientes atualizadas);
- *announce-list*: é uma extensão opcional à especificação oficial que serve para manter a compatibilidade com versões anteriores.
- *creation date*: (opcional) a data de criação do torrent no formato padrão data/hora do UNIX;
- *comment*: comentários livres do autor (opcional);
- *created by*: (opcional) Nome e versão do programa utilizado na criação do torrent;
- *encoding*: o formato de codificação utilizado para gerar as partes do dicionário *info* no *torrent* (opcional);
- *info* (dicionário): Campo mais importante, pois possui todas as informações referentes ao arquivo alvo (que será baixado), entre elas:
    - comprimento do pedaço: numero de bytes em de cada pedaço;
    - pedaços: sequência de caracteres composta dos 20 bytes do hashe SHA1, um por pedaço;

1 Bencoding é um modo de especificar e organizar os dados em um formato conciso. Suporta os tipos sequências de caracteres, números inteiros, listas e dicionários ("Bittorrent protocol specification v1. 0" 2002)

2 URL (de Uniform Resource Locator), em português Localizador de Recursos Padrão, é o endereço de um recurso (um arquivo, uma impressora etc.), disponível em uma rede; seja a Internet, ou uma rede corporativa, uma intranet. Uma URL tem a seguinte estrutura: protocolo://máquina/caminho/recurso (fonte: http://pt.wikipedia.org/wiki/URL, acessado em 25/12/2011)

- privativo: 0 ou 1 dependendo da possibilidade ou não de fontes externas de clientes (DHT).

3.2. TRACKER

Um serviço facilitador para o funcionamento da rede *BitTorrent* são os repositórios que mantem as listas com informações dos pontos que possuem o arquivo, ou partes deles, disponíveis. Estes locais de armazenamento de informações são chamados de *Trackers*, pois possuem os caminhos dos arquivo que se deseja transferir (Cohen, 2003). As informações referentes ao endereço do *tracker* (rastreador) constam no *torrent*.

Para iniciar a disponibilização de um arquivo na pelo protocolo *BitTorrent* o usuário deverá criar o arquivo ".torrent" contendo as informações necessárias sobre o arquivo disponibilizado e onde estarão as informações dos pontos com os quais se fará a conexão para baixar o arquivo (Pouwelse, Garbacki, Epema, & Sips, 2004). Como existe a necessidade deste arquivo ser facilmente encontrado e indexado eles são colocados em sites que os organizam em categorias e oferecem mecanismos de buscas. Alguns dos sites mais conhecidos são: http://www.demonoid.me/, http://www.isohunt.com/ e http://thepiratebay.org/. Normalmente eles acumulam a função de repositório/indexador de *torrents* e com o serviço de *tracker*.

3.3. REPOSITÓRIOS

Repositórios nada mais são do que sites web que possuem armazenados e organizados os *torrents*. Eles possuem mecanismos de de busca e catalogação conforme o tipo de arquivo alvo (áudio, vídeo, aplicativos, livros etc).

O cliente que disponibilizará o arquivo alvo fará a criação e a carga do *torrent* no repositório, especificando e detalhando seu conteúdo. Os clientes que buscam aquele conteúdo procurarão no repositório e baixarão o *torrent*, o qual conterá informações que possibilitará que seu cliente *torrent* localize o *tracker*.

3.4. O PROCESSO

A partir da busca e *download* do torrent e sua execução em um cliente Bittorrent esse cliente fará o contato com o *tracker* que fornecerá informações de onde o cliente localizará os pedaços dos arquivos para baixar (endereço de outros clientes que já possuem pedaços do arquivo disponíveis para compartilhamento). A partir daí iniciará a comunicação ponto a ponto, sendo que o de temos em tempos o *tracker* atualiza as informações da disponibilidade o arquivo. Passo a passo, o processo ocorre conforme detalhado a seguir:

1. O cliente localiza em um repositório (página WEB) o arquivo desejado e clica em um link para o arquivo torrent que contêm as informações do arquivo alvo (aquele que se deseja);
2. O arquivo torrent deve ser aberto por um software cliente *BitTorrent*. Os mais populares, segundo (Acorn, 2008) são o BitComet[3], o uTorrent[4], o Azureus[5] (atualmente chamado Vuze), o ABC[6] e o BitTornado[7].
3. o cliente *BitTorrent* comunica-se com o *tracker* especificado no torrent para encontrar outros computadores que possuam o arquivo completo (semeadores) ou partes do arquivo (clientes que normalmente também se encontram no processo de download e possuem partes inteiras para compartilhar);
4. o *tracker* identifica o enxame (*swarm*), que é o conjunto dos computadores conectados à rede que possuem todo ou parte do arquivo e estão em processo de enviar ou receber;
5. o *tracker* passa as informações necessárias para o cliente iniciar a troca de partes do arquivo alvo desejado com outros computadores no enxame (cadeia de compartilhamento), conforme visualizamos na Figura 3.1. O processo é paralelizado

3 http://www.bitcomet.com/
4 http://www.utorrent.com/
5 http://azureus.sourceforge.net/
6 http://pingpong-abc.sourceforge.net/
7 http://www.bittornado.com/

de forma a possibilitar o recebimento de diversas partes do arquivo simultaneamente, o que otimiza a velocidade do *download* ;

6. ao continuar executando o cliente *BitTorrent* depois que o seu *download* for concluído, outras pessoas continuam a receber partes do arquivos do seu computador, e sua posição no enxame passa a ser a de semeador (*seeder*).

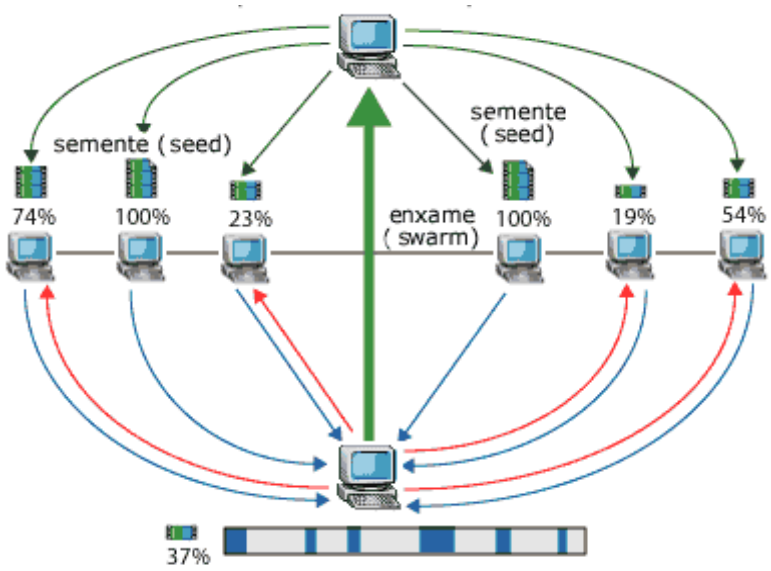

Figura 3.1: esquema de funcionamento do *BitTorrent* (adaptado de http://www.howstuffworks.com)

O sucesso do *BitTorrent* vem principalmente do atributo escalabilidade (D. Erman, 2005; Izal, Urvoy-Keller, Biersack, & Felber, 2004; Luan & Tsang, 2006; Qiu & Srikant, 2004; Wu, Dhungel, Hei, Zhang, & Ross, 2010). Sua utilização viabiliza que se compartilhe arquivos grandes sem maior comprometimento dos recursos de banda de rede do semeador inicial (Izal et al., 2004; Pouwelse et al., 2005; Qiu & Srikant, 2004; Wu et al., 2010). Teoricamente, basta que ele transfira uma vez cada parte do arquivo que o mesmo poderá ser recebido (baixado) por centenas ou milhares de outros usuários.

Na figura 3.2 visualizamos um exemplo do compartilhamento de um arquivo que foi dividido em seis partes nominadas 1, 2, 3, 4, 5, e 6, representadas por pontos abaixo das máquinas e colocados na respectiva ordem. A seguir explica-se detalhadamente o que representa cada quadro da figura:

(a) Na situação inicial, apresentada na figura 3.2(a), apenas o semeador original (representado pelo desenho de um computador maior) possui todas as partes do arquivo e disponibiliza as partes 2 e 6 (representadas pelos pontos da segunda e sexta posições, respectivamente) para outros dois clientes.

(b) A seguir, na figura 3.2(b), enquanto o semeador original distribui as partes 3 e 5, os clientes que tinham recebido as partes 2 e 6 anteriormente às redistribuem para outros clientes.

(c) Em seguida o semeador distribui as partes 1 e 4, completando a transferência inicial do arquivo inteiro. Os outros clientes continuam a redistribuição conforme representado na figura 3.2(c)

(d) Na figura 3.2(d) temos um fato interessante que é: nenhum dos clientes possui o arquivo completo, porém o enxame (conjunto dos computadores que está compartilhando o arquivo) possui todas as partes do arquivo, possibilitando o prosseguimento da redistribuição do arquivo completo para todos no enxame, mesmo, a partir de agora, sem a participação do semeador original.

(e) A figura 3.2(e) mostra o surgimento dos primeiros clientes que completam o recebimento do arquivo, passando, a partir deste ponto, a serem denominados também de semeadores, pois, apesar de não serem o semeador original, agora também possuem todas as partes do arquivo para redistribuir.

(f) Finalmente, na figura 3.2(f) vemos as últimas redistribuições, e o estado em que todos os clientes possuem todas as partes do arquivo, ou seja, o arquivo completo. Este exemplo mostra que o o arquivo disponibilizado pelo semeador original foi transferido

completamente para outros 7 (sete) clientes, tendo transferido apenas uma vez cada parte do arquivo, sem congestionar sua banda de rede.

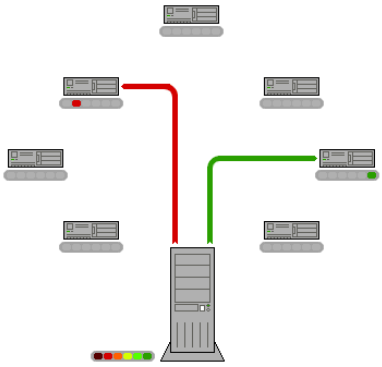

Figura 3.2(a): semeando 2 e 6

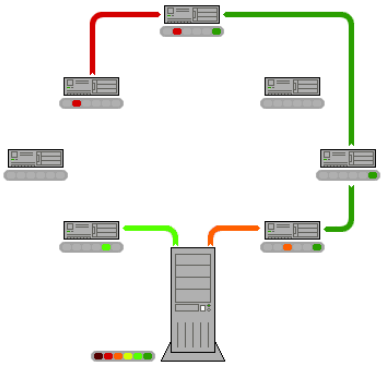

Figura 3.2(b): semeando 3 e 5 e redistribuição de 2 e 6

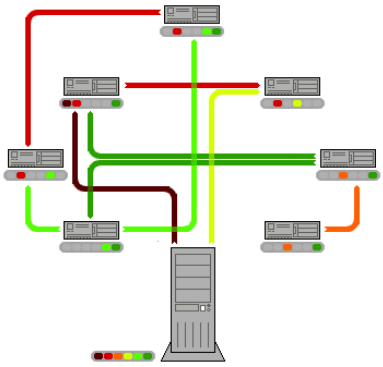

Figura 3.2(c): semeando 1 e 4 e redistribuição de 2, 3, 5 e 6

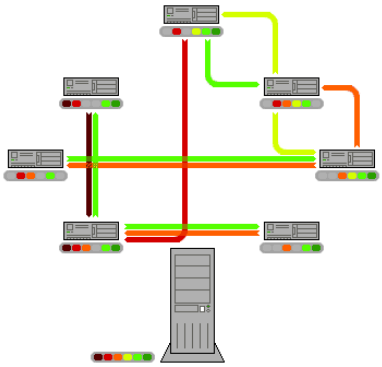

Figura 3.2(d): redistribuição sem participação do semeador

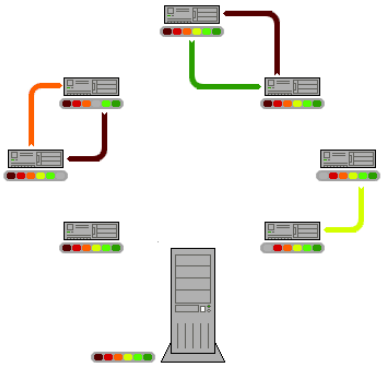

Figura 3.2(e): redistribuição com surgimento de novos semeadores

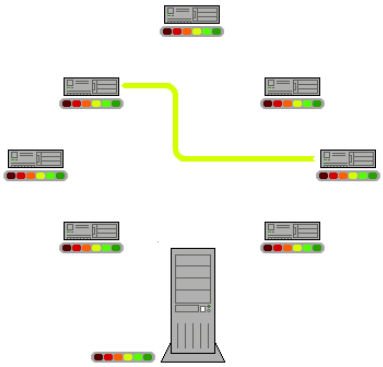

Figura 3.2(f): redistribuição final

Figura 3.2: exemplo de um processo de transferência (compartilhamento) via *BitTorrent* (adaptado de http://en.wikipedia.org)

Observa-se que a velocidade geral é muito melhorada quando baixa-se várias partes do arquivo ao mesmo tempo (Mansilha, Mezzomo, Facchini, & Gaspary, 2010). Por conseguinte, quanto mais computadores participam do enxame, é mais rápida a transferência do arquivo, visto que há mais fontes para cada parte do arquivo. Por essa razão, o *BitTorrent* é extremamente útil para arquivos grandes e muito procurados (Roberto Santos, da Costa Cordeiro, Gaspary, & Barcellos, 2010; Yu & Chen, 2010).

A seguir, na figura 3.3, é apresentado um diagrama de estados auto explicativo referente às conexões entre clientes e *tracker*. Nele podemos observar a sistemática do comportamento do protocolo *BitTorrent (Fernandes & Fernandes, 2009)*.

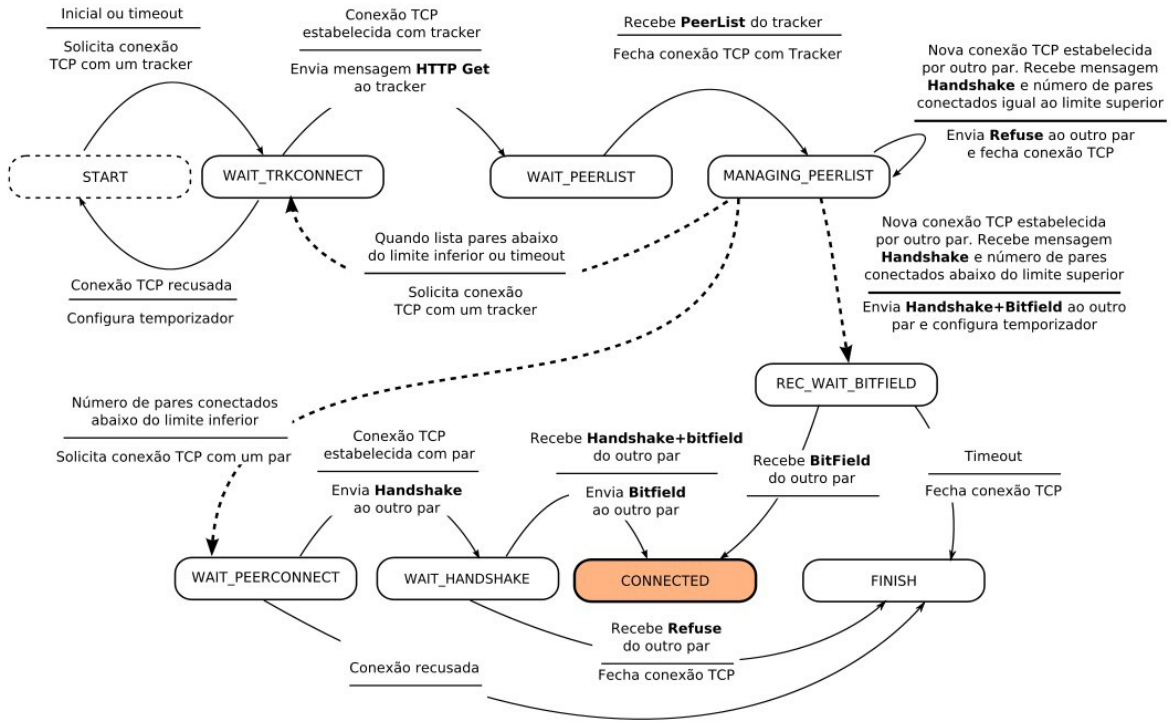

Figura 3.3: Diagrama de estados do *BitTorrent* (Konrath, Barcellos, Silva, Gaspary, & Dreher, 2007)

A eficiência do protocolo foi estudada e comprovada por (Izal et al., 2004) quando ainda era uma nova mas já muito popular aplicação P2P. Sua análise baseou-se em medidas coletadas em um período de cinco meses de duração que envolveu milhares de pares.

A evolução do protocolo *BitTorrent*, de acordo com (Doumen, Roozenburg, & Brinke, 2009) é o lançamento de novos recursos tais como as seguintes extensões para a plataforma:

- *Live streaming* (suporte para vídeo sob demanda);
- Relacionamentos (criação de redes sociais);
- Monitoramento remoto (permitindo a medição remota do comportamento e estatísticas do sistema);
- Detecção do tipo de NAT (*Network Address Translator* – Tradutor de Endereço de Rede); e
- Políticas de distribuição (semear) conectáveis para incentivar que o cliente permaneça semeando (fornecendo) os arquivos.

4. MINERAÇÃO DE DADOS E O WEKA

A sobrecarga de dados que sofremos a partir da popularização da facilidade do armazenamento em meio eletrônico (bancos de dados) trouxe, agregadamente, a oportunidade da descoberta de conhecimento nessa quantidade enorme de *bytes (Fayyad, Piatetsky-Shapiro, & Smyth, 1996)* .

Segundo (Frawley, Piatetsky-Shapiro, & Matheus, 1992) Descoberta de Conhecimento é a extração não-trivial de informação implícita, previamente desconhecida e potencialmente útil de dados.

Percebemos que descoberta de conhecimento é um conceito do campo da ciência da computação que detalha o os passos para procurar, automatizadamente, em grandes volumes de dados, padrões que podem ser considerados conhecimentos.

Para chegarmos a informações úteis (conhecimentos) a partir de uma grande massa de dados devemos seguir determinados passos, conforme ilustra a figura 4.1 (Fayyad et al., 1996).

Dentro destes passos, o mais importante e complexo é a Mineração de Dados. Nela é que reside a inteligência maior do processo.

Mineração de dados (do inglês *data mining*) é definido por Ian Hitten como o processo de descobrir padrões nos dados. Ainda, segundo (Witten, Frank, & Hall, 2011) :

> O processo deve ser automático ou (mais comumente) semiautomático. Os padrões descobertos devem ser significativos, de forma que eles levam a alguma vantagem, geralmente uma vantagem econômica. Os dados estão invariavelmente presentes em quantidades substanciais.

Para (Fayyad et al., 1996) , a definição de Mineração de Dados é:

> "o processo não-trivial de identificar padrões válidos, novos, potencialmente úteis e compreensíveis nos dados"

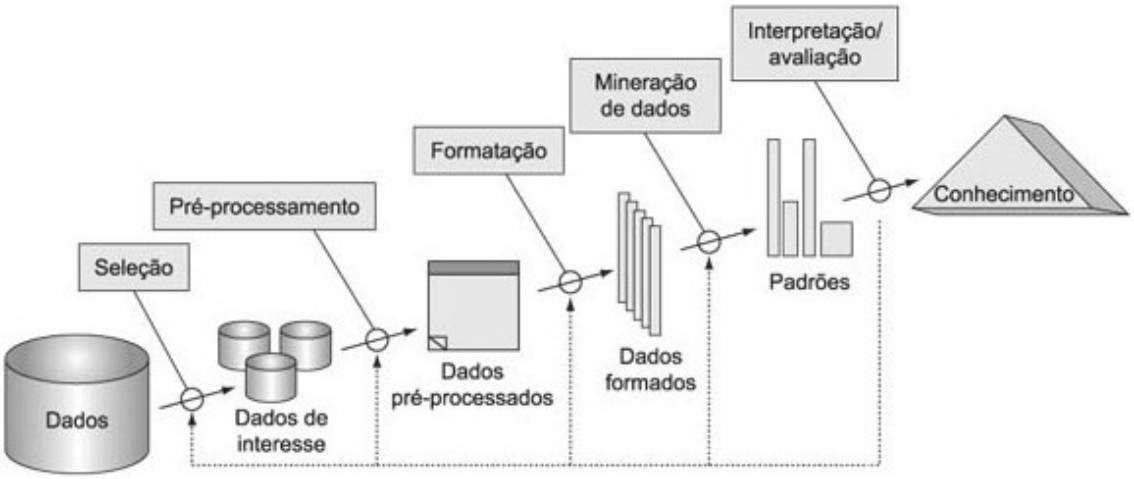

Figura 4.1: Passos do processo que compões a descoberta do conhecimento (adaptado de (Fayyad et al. 1996))

Detalhando o processo para se alcançar o conhecimento, ou seja, a sua descoberta, vamos passar pelas etapas ilustradas na figura 4.1.

A etapa de Seleção separa do conjunto de Dados os dados que representam o domínio do conjunto, focando no subconjunto de dados no qual o processo de descoberta será executado, que são os Dados de interesse.

Na etapa do Pré-processamento ocorre uma remoção dos ruídos (informações desnecessárias) para limpeza dos Dados de interesse. Seu objetivo é a consolidação dos dados permitindo o trabalho com o que é relevante.

Para superar possíveis limitações existentes nos algorítimos empregados na fase de Mineração de Dados é executada uma transformação chamada, na figura, de Formatação, deixando tudo pronto para a fase de Mineração de dados.

A Mineração de Dados é a fase principal processo. Neste ponto ocorre a descoberta dos padrões e a sua respectiva extração, através de um conjunto de ferramentas computacionais

com técnicas para identificação dos conhecimentos (padrões) contidos nos dados transformados.

Finalmente a interpretação e avaliação dos padrões tem por objetivo tornar compreensível o resultado para os tomadores de decisão.

4.1. MINERAÇÃO DE DADOS

A fase de mineração de dados consiste no aspecto principal da exploração do conjunto de dados para a procura dos padrões.

Segundo (Witten et al., 2011) existem quatro tipos diferentes de aprendizagem nas aplicações de mineração de dados. São elas:

- Aprendizagem por classificação, onde o regime de aprendizagem é apresentado com um conjunto de exemplos classificados em que se espera para aprender uma forma de classificar novos exemplos;
- Aprendizagem por associação, onde é procurada qualquer associação entre características, e não apenas aqueles que preveem um valor de classe especial;
- Aprendizagem por *clustering*, é aquela em que são procurados grupos de exemplos com características similares (Bishop, 2006); e
- Finalmente, no tipo previsão numérica, o resultado no qual se deseja chegar não é uma classe discreta, mas uma quantidade numérica.

Destes tipos, verificamos que o que melhor se enquadra nos objetivos deste trabalho é a aprendizagem por classificação, pois desejamos verificar a possibilidade de classificação dos tipos de protocolos, sendo que temos como gerar um conjunto de exemplos já classificados para serem utilizados na aprendizagem.

Para ajudar na compreensão da variedade de métodos de mineração de dados, suas inter-relações e agrupamento apresenta-se na figura 4.2 sua taxionomia. Visualizando a figura citada é possível distinguir entre dois tipos principais de mineração de dados: orientado a

verificação (o sistema verifica a hipótese do usuário) e orientado ao descobrimento (o sistema encontra novas regras e padrões de forma autônoma).

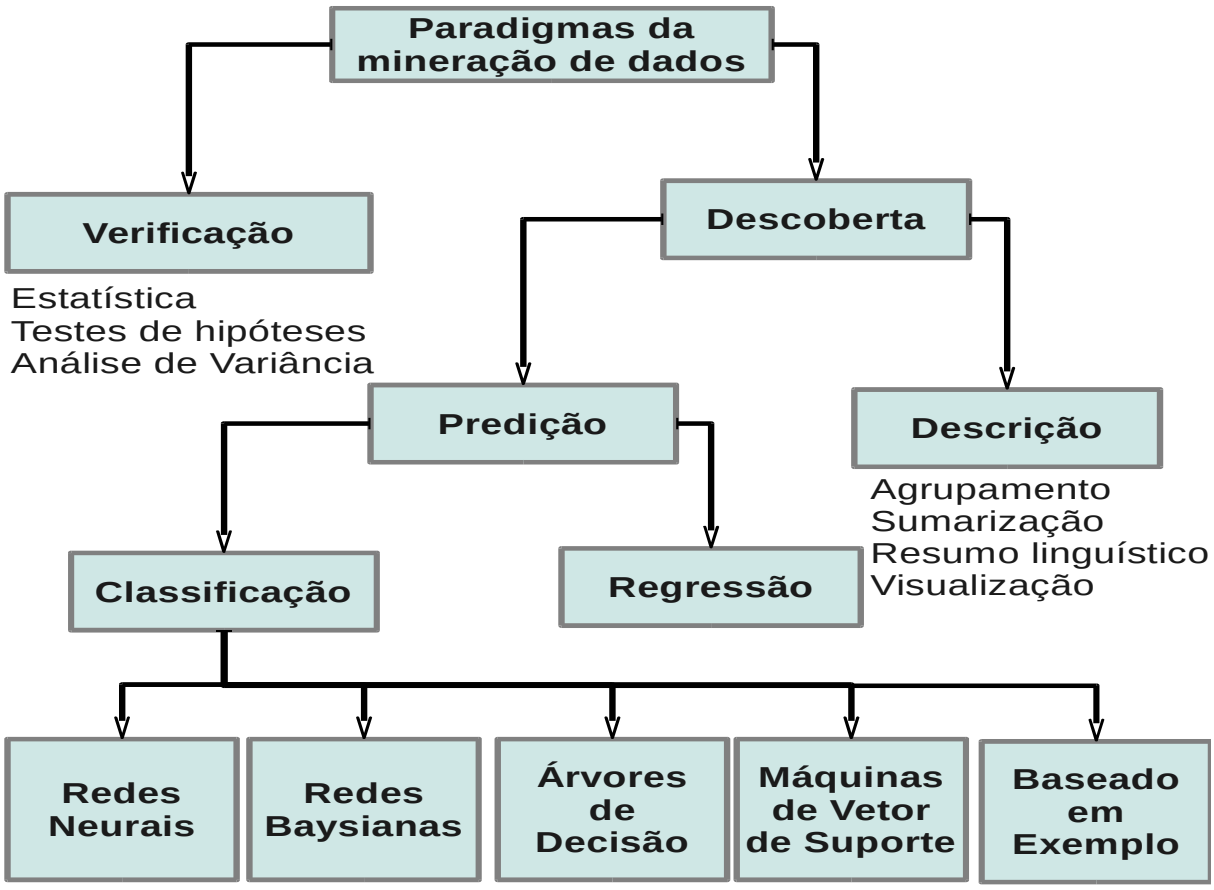

Figura 4.2: taxionomia da mineração de dados – adaptado de (Maimon & Rokach, 2010)

Desta ampla gama de possibilidades estudaremos neste trabalho algoritmos de Classificação de Árvores de Decisão e Máquinas de Vestor de Suporte (SVM, do inglês *Support Vector Machine*).

4.2. TIPOS DE ALGORITMOS

4.2.1. ÁRVORES DE DECISÃO

As Árvores de Decisão ou árvores de Classificação são um meio eficiente de construir classificadores que estabelecem classes baseadas nos atributos de um conjunto de dados, sendo representações simples do conhecimento.

Os principais algoritmos de árvores de decisão são, segundo (Quinlan, 1986, 1993; Quinlan & Kohavi, 1999) o ID3 (*Iterative Dichotomizer 3*), o C4.5, o CHAID (*Chi-square Automatic Interaction Detection*) o CART (*Classification and Regression Trees*), e o QUEST (*Quick, Unbiasied, Efficient Statistical Tree*).

O C4.5 inicia-se com grandes conjuntos de dados pertencentes a classes já conhecidas. Os dados (descritos por qualquer mistura de propriedades nominais e numéricas) são examinados para verificar a existência de padrões que permitam que classificá-los confiavelmente. Esses padrões, em seguida, são expressos como modelos em forma de árvores de decisão, com a finalidade de poderem ser utilizadas para classificar novos casos, com a fim de tornar os modelos tanto compreensíveis assim como precisos. Este algoritmo foi aplicado com sucesso em casos envolvendo dezenas de milhares de casos descritos por centenas de propriedades (Quinlan, 1993).

4.2.2. SVM (*Support Vector Machines*)

As Máquinas de Vetores de Suporte (SVM, do Inglês *Support Vector Machines*) são embasadas pela teoria de aprendizado estatístico (Vapnik, 1995). Uma série de princípios são estabelecidos por essa teoria, os quais que devem ser seguidos na obtenção de classificadores com boa generalização (Burges, 1998).

As SVMs podem ser lineares (com margens rígidas e margens suaves) ou não lineares.

Utilizando o princípio de minimização do risco estrutural, as SVMs obtêm resultados com uma alta capacidade de generalização, mesmo que os dados utilizados no conjunto de

treinamento não sejam muito representativos. Esses algoritmos possuem várias características, que justificam a denominação de estado da arte em métodos de reconhecimento de padrões (Penna & Carvalho, 1995).

4.3. VALIDAÇÃO CRUZADA

Para avaliar os experimentos deste trabalho, utilizamos a técnica de validação cruzada. A validação cruzada é uma técnica para aferir como os resultados de uma análise estatística vão ser generalizados para um conjunto de dados independente (Geisser, 1993). Ela é principalmente usada em configurações onde o objetivo é a predição, e alguém deseja estimar o quão correto um modelo preditivo irá ser executado na prática (Schaffer, 1993). Uma rodada da validação cruzada envolve o particionamento de uma amostra de dados em subconjuntos complementares, executando a análise de um subconjunto (chamado de conjunto de treinamento), e validando a análise em outro subconjunto (chamado de conjunto de validação ou teste) (Diamantidis, Karlis, & Giakoumakis, 2000). Na figura 4.3 podemos visualizar a representação gráfica de um conjunto de dados do qual o subconjunto de teste (*Test*) vai sendo retirado de partes diferentes, permanecendo o restante para traino (*Train*).

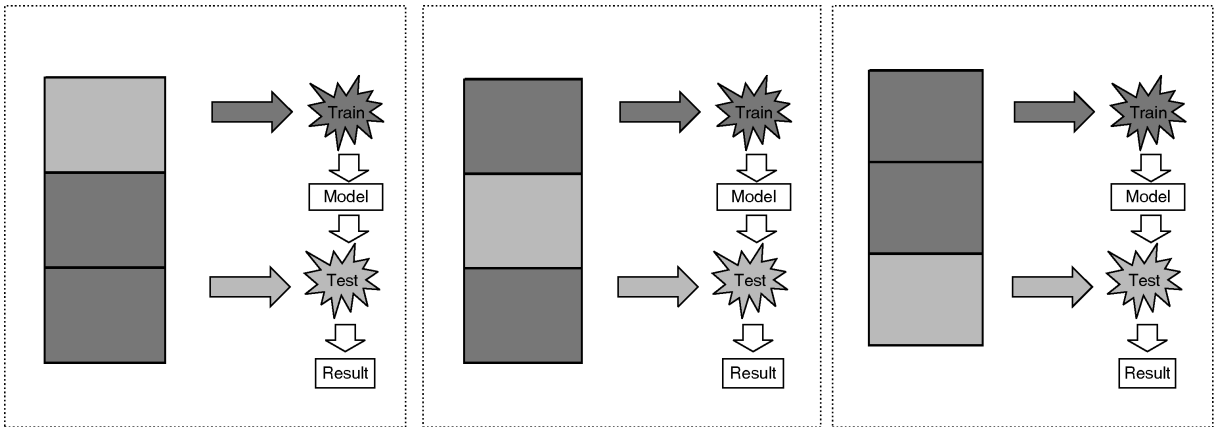

Figura 4.3: divisão dos modelos e métodos de classificação ((Rodrigues, 2005)

4.4. WEKA

O pacote de software WEKA (*Waikato Environment for Knowledge Analysis*) é um ambiente de análise de conhecimento cujo projeto teve início em 1992, na Universidade de Waikato, Nova Zelândia (Hall et al., 2009). O pacote contempla um abrangente conjunto de implementações de algoritmos de diversas técnicas de Mineração de Dados.

O WEKA tem o código fonte disponível para estudos e modificações de acordo com a *General Public License* (GPL).

Historicamente, o objetivo do projeto, de acordo com (Hall et al., 2009) era:

> *"O programa visa desenvolver uma ferramenta facilitadora no estado da arte para o desenvolvimento de técnicas de aprendizado de máquina e investigar sua aplicação em áreas-chave da economia da Nova Zelândia. Especificamente, vamos criar uma bancada para a aprendizagem de máquina, determinar os fatores que contribuem para seu sucesso aplicação na indústria agrícola, e desenvolver novas métodos de aprendizado de máquina e formas de avaliar a sua eficácia."*

O WEKA recorre a técnicas de mineração de dados para proceder à análise computacional e estatística dos dados fornecidos tentando, indutivamente, a partir dos padrões encontrados gerar hipóteses para soluções e teorias sobre os dados em questão.

4.4.1. Conceitos Básicos do WEKA

De acordo com o manual do WEKA (Bouckaert et al., 2008) são apresentados a seguir os conceitos básicos para entendimento da ferramenta.

4.4.1.1. Dataset

Dataset é um conjunto de dados, com ligeira semelhança a uma planilha bidimensional ou uma tabela de um banco de dados (Silva, 2004). São representadas externamente pelos arquivos ARRF (*Attribute-Relation File Format*). Apesar de aceitar entrada de dados em outros padrões, como, por exemplo, o simples CVS (*Comma Separated Values* – Valores separados por vírgulas), o método preferido do WEKA para carregar dados é no Formato de Arquivo de Atributo-Relação (ARFF). Nesse formato é possível definir os tipos de dados que estão sendo carregados, e então fornecer seus dados propriamente ditos (Melo, Marcelo Damasceno de, 2010) . É definido no arquivo cada atributo (coluna) e o que o mesmo conterá. Cada linha de dados é fornecida em um formato delimitado por vírgulas (Bouckaert et al., 2008; Hall et al., 2009). A seguir, na figura 4.4, é apresentado um exemplo de um Dataset em formato ARFF, dividido em cabeçalho e dados pela declaração "`@DATA`".

```
% 1. Title: Iris Plants Database
%
% 2. Sources:
%  (a) Creator: R.A. Fisher
%  (b) Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
%  (c) Date: July, 1988
%
@RELATION iris

@ATTRIBUTE sepallength  NUMERIC
@ATTRIBUTE sepalwidth   NUMERIC
@ATTRIBUTE petallength  NUMERIC
@ATTRIBUTE petalwidth   NUMERIC
@ATTRIBUTE class  {Iris-setosa,Iris-versicolor,Iris-virginica}
```

```
@DATA
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
```

Figura 4.4: exemplo de *dataset* em formato ARFF (fonte: http://weka.wikispaces.com/ARFF+%28stable+version%29, acessado em 12/12/2011)

Dentro do arquivo no formato ARFF as linhas que possuem o símbolo % no início são comentários.

4.4.1.2. Classificadores

Todo e qualquer algoritmo de aprendizagem é considerado um classificador no WEKA. Um modelo de classificador é um mapeamento arbitrário complexo de todos-exceto-um conjunto de dados de atributos para o atributo de classe.

4.4.1.3. Filtros

Os filtros referem-se às classes que transformam os conjuntos de dados, removendo ou adicionando atributos, reamostrando o conjunto de dados, removendo exemplos e assim por diante. Este pacote oferece suporte útil para pré-processamento de dados, que é um passo importante no aprendizado de máquina.

4.4.2. Telas de execução

O WEKA possui uma interface gráfica como ponto de partida para utilização da ferramenta. Ela é a classe *weka.gui.GUIChooser* e consiste de 4 (quatro) botões para as principais funções além do menu: *Explorer*, *Experimenter*, *KnowledgeFlow* e *Simple CLI*, conforme podemos visualizar na figura 4.5.

Figura 4.5: janela inicial da interface gráfica do WEKA (*WEKA GUI Chooser*)

Os botões são utilizados para executar as seguintes funções:

- *Explorer*: Um ambiente para explorar a execução dos algoritmos via interface gráfica, conforme figura 4.6.

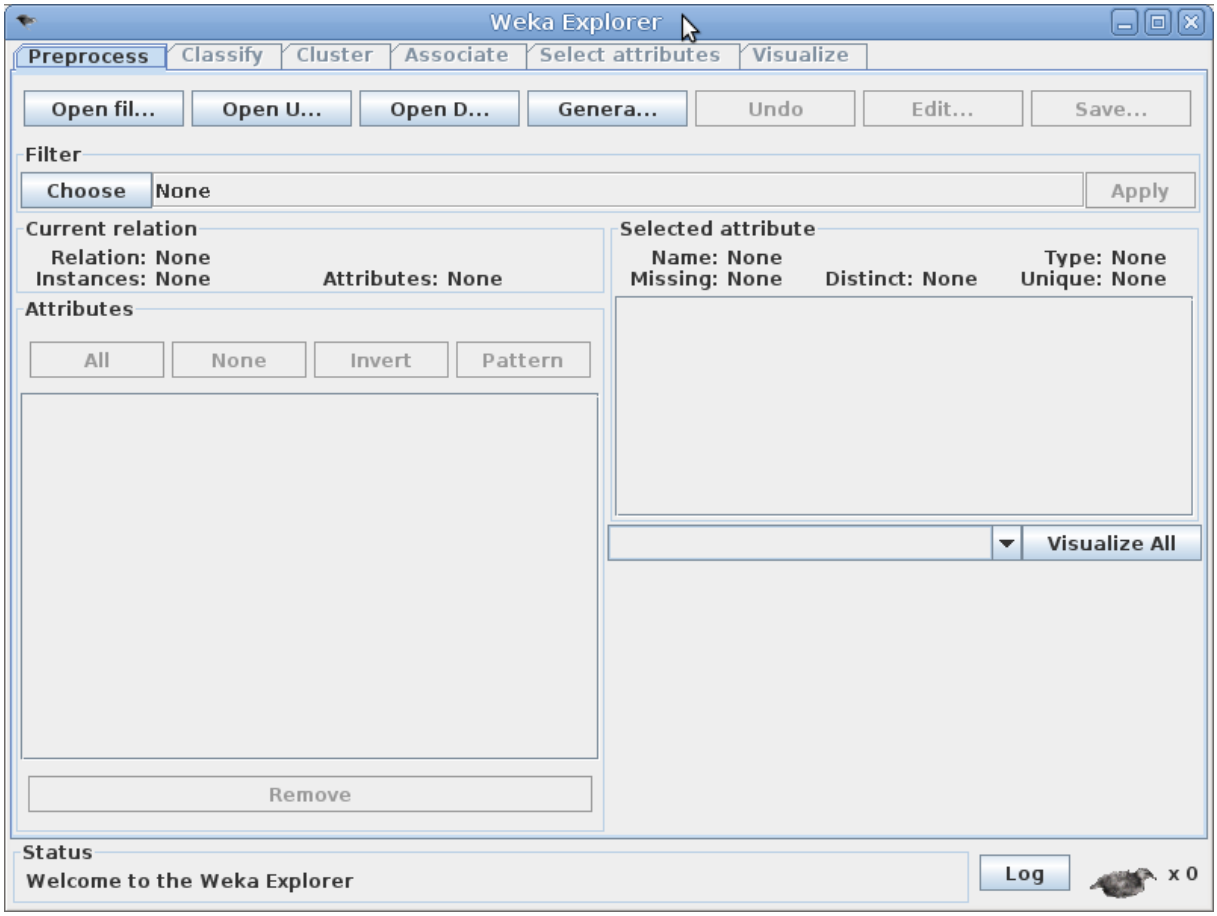

Figura 4.6: janela que permite a execução dos algoritmos via interface gráfica

- *Experimenter*: Um ambiente para a realização de experiências e realização de testes estatísticos entre os sistemas de aprendizagem, conforme figura 4.7.

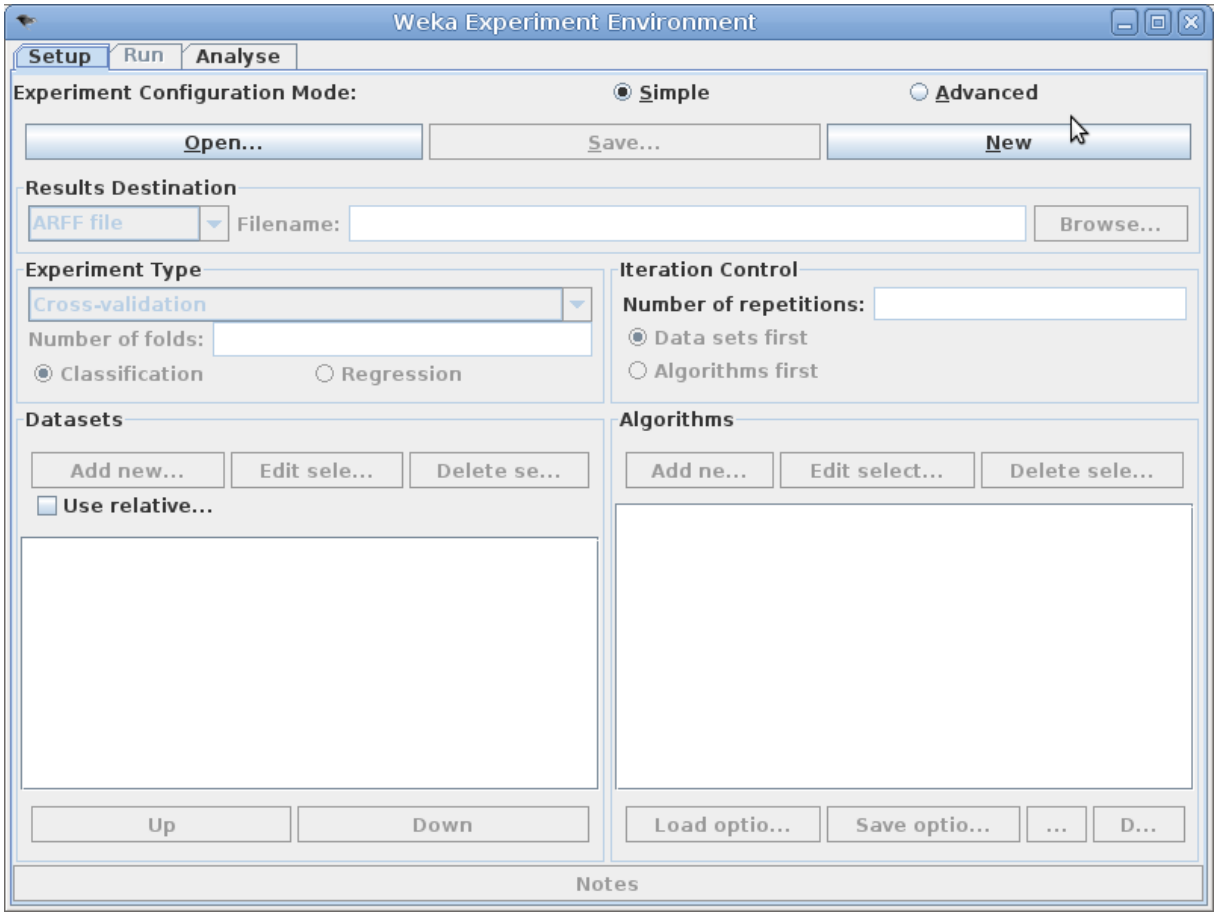

Figura 4.7: janela para a realização de experiências e realização de testes estatísticos entre os sistemas de aprendizagem

- *KnowledgeFlow* (fluxo de conhecimento): Provê um ambiente que suporta essencialmente as mesmas funções que o *Explorer*, mas com uma interface *drag-and-drop* (arraste-e-solte), conforme visualizado na figura 4.8. O diferencial é que ele possui a vantagem de suportar aprendizagem incremental.

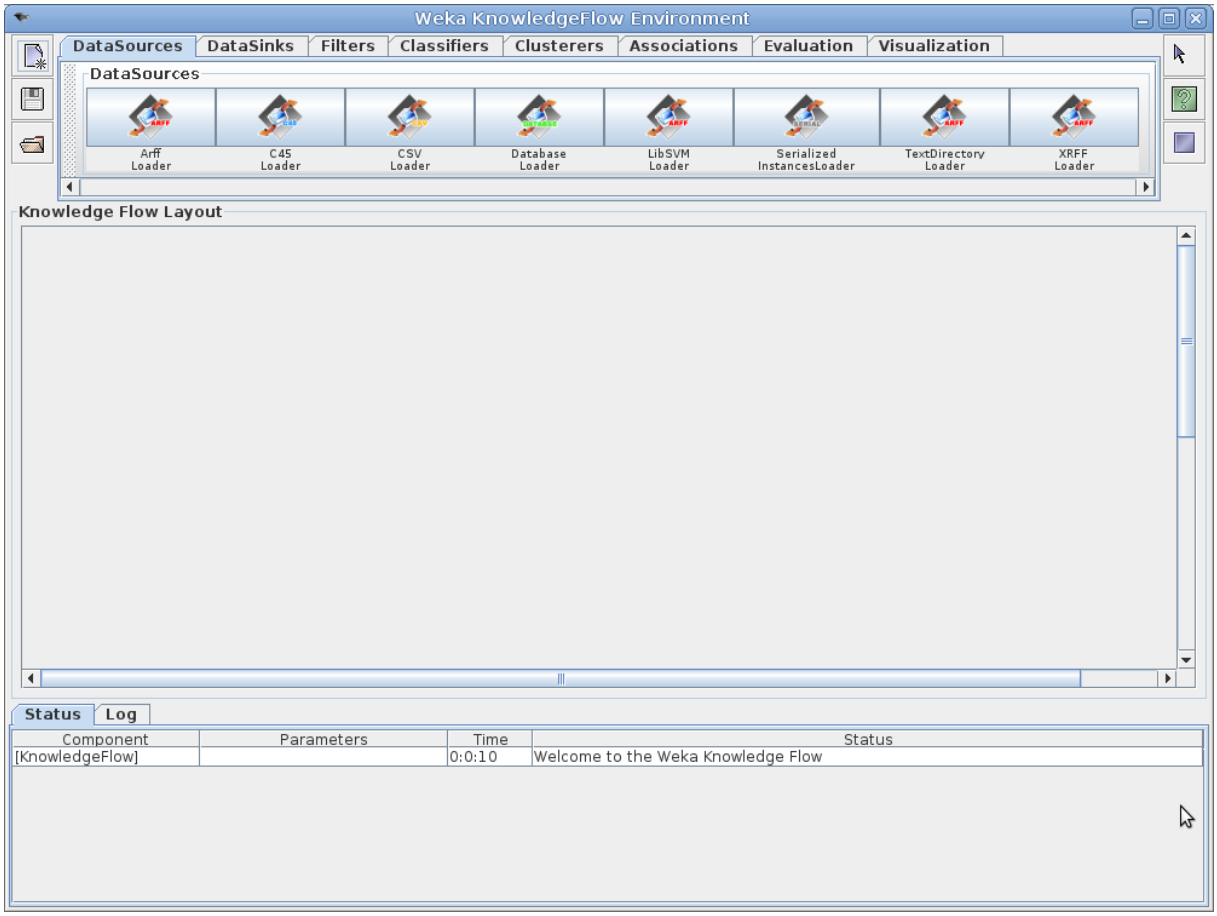

Figura 4.8: janela do *KnowledgeFlow* apresentando uma interface arraste-e-solte (*drag-and-drop)*

- *SimpleCLI*: Fornece uma interface de linha de comando simples que permite a execução direta de comandos WEKA para sistemas operacionais que não fornecem sua própria interface de linha de comando, conforme figura 4.9.

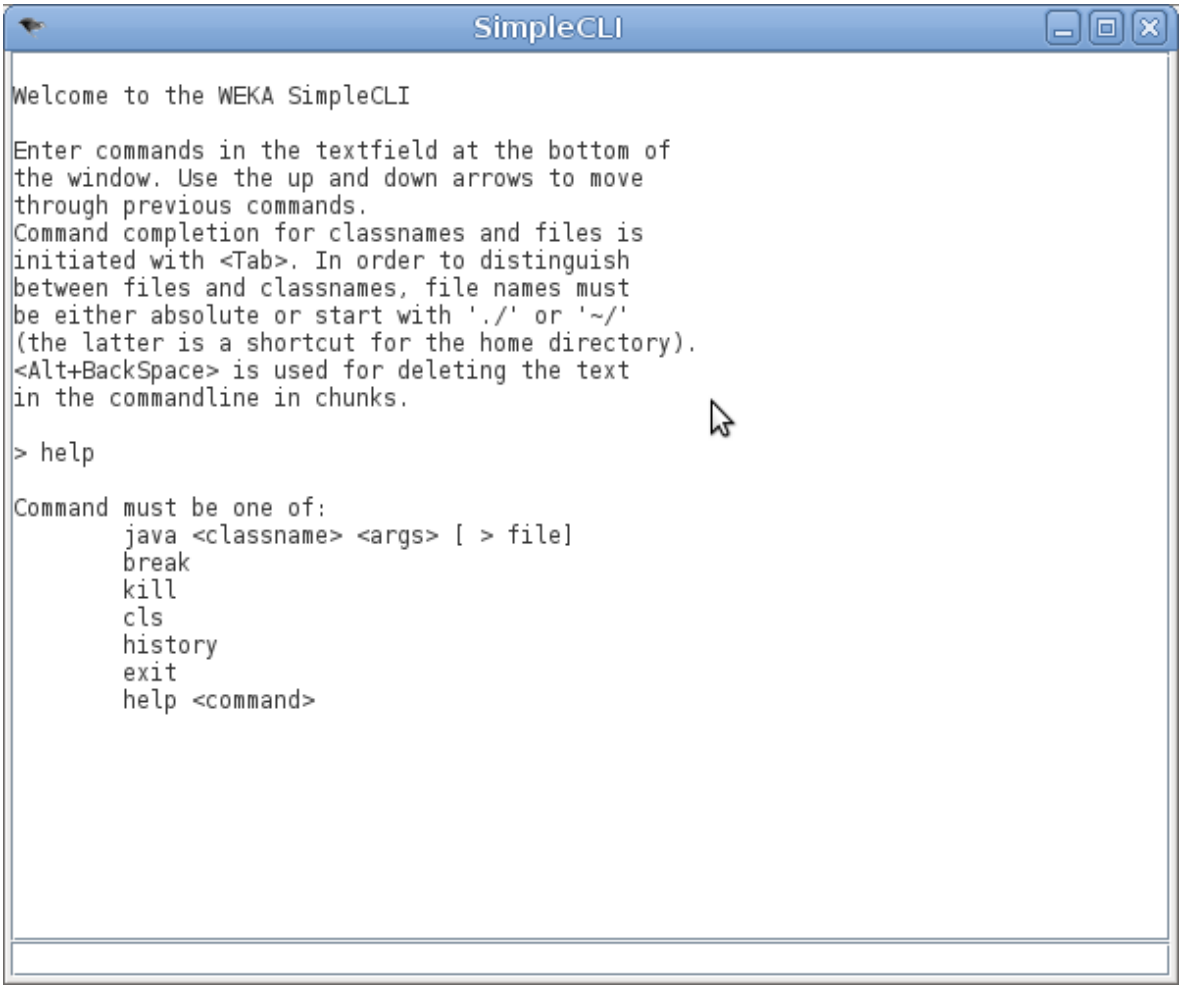

Figura 4.9: janela da interface *SimpleCLI* (linha de comando simples)

Para utilização mais completa dos recursos do WEKA é recomendada a interface de linha de comando, pois através dela tem-se acesso a algumas funcionalidades que não são oferecidas através da interface gráfica, além de consumir menos recursos de memória do sistema (Bouckaert et al., 2008).

Após terminada esta revisão em que passamos pelos detalhes do processo de aquisição de conhecimento em bases de dados (KDF), os algoritmos de mineração de dados e termos

conhecido o WEKA, no próximo capítulo será mostrado os experimentos feitos com o mesmo e os resultados alcançados.

5. EXPERIMENTOS E ANÁLISE DOS RESULTADOS

Neste capítulo será detalhado como foram feitos os testes e seus resultados. Para isso o capítulo está dividido em:

- Ambiente de operação;
- Captura dos pacotes de tráfego de rede;
- Extração das informações referentes aos pacotes capturados;
- Conversão das informações extraídas para o formato aceito pelo WEKA;
- Divisão dos dados em treino e teste;
- execução dos algoritmos de classificação;
- Comparação dos resultados.

5.1. AMBIENTE DE OPERAÇÃO

A etapa de captura dos pacotes de rede para análise foi executada inicialmente em máquinas virtuais (VirtualBox[8]) com a distribuição linux BackTrack[9] versão 5r2 rodando em um host linux Debian Wheezy. A distribuição BackTrack foi escolhida em virtude de sua característica de ser "silenciosa" no tráfego de rede e também por já possuir diversas ferramentas de análise de tráfego integradas. Foi procurado trabalhar com um sistema operacional "silencioso" para conseguir uma maior garantia de que o tráfego que está sendo capturado é apenas referente ao protocolo/aplicativo desejado. Outros sistemas operacionais e distribuições rodam processos em *background* que podem gerar tráfego de tipos indesejados atrapalhando a fidelidade do resultado da captura.

8 https://www.virtualbox.org/
9 http://www.backtrack-linux.org/

5.2. CAPTURA DO TRÁFEGO DE REDE

A captura dos pacotes foi realizada através do software Wireshark[10] (figura 5.1). O Wireshark executa a captura dos pacotes da interface de rede selecionada, salvando no formato padrão PCAP (captura de pacotes, do inglês *Packet CAPture*). O PCAP consiste de uma interface de programação de aplicativo (API - A*pplication Programming Interface*) para capturar o tráfego da rede, implementado na biblioteca libpcap.

Assim que ocorre a captura o programa analisa o tráfico de pacotes recebidos e enviados organizando-os em protocolos (uma convenção ou padrão que controla e possibilita uma conexão, comunicação ou transferência de dados entre dois sistemas computacionais). Foi utilizada a versão 1.6.8 do Wireshark.

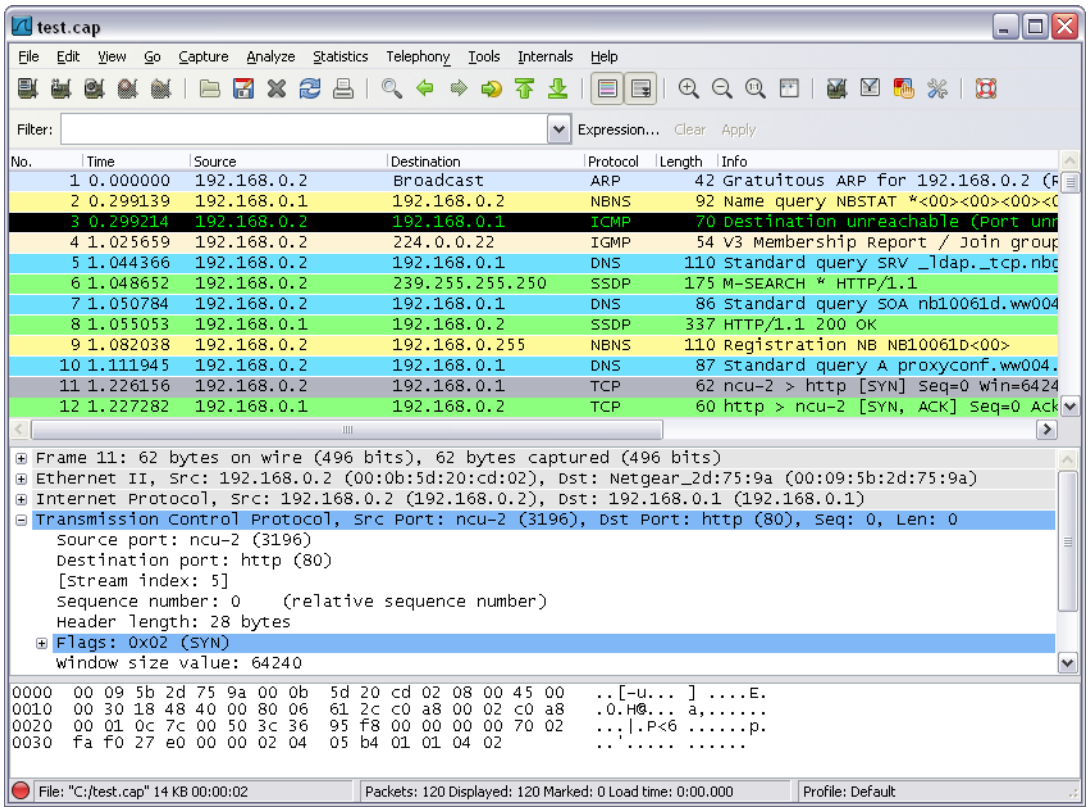

Figura 5.1: tela de captura de pacotes do aplicativo Wireshark

10 http://www.wireshark.org (Copyright 1998-2010 Gerald Combs <gerald@wireshark.org> and contributors)

5.2.1. Formato de arquivo de captura

O arquivo tem um cabeçalho global contendo alguma informação global seguida de zero ou mais registros para cada pacote capturado, conforme a figura 5.2:

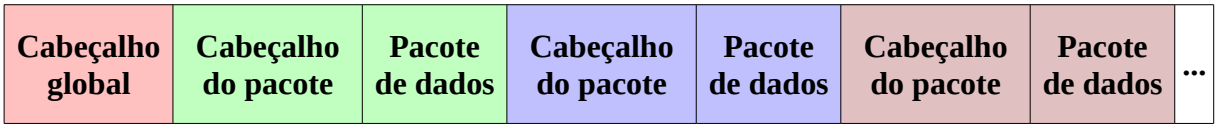

Figura 5.2: estrutura dos pacotes

Um pacote capturado em um arquivo de captura PCAP[11] não necessariamente contêm todos os dados no pacote exatamente como apareceu na rede; o arquivo de captura pode conter, no máximo, os primeiros N bytes de cada pacote, para algum valor de N.

5.3. EXTRAÇÃO DAS INFORMAÇÕES REFERENTES AOS PACOTES CAPTURADOS;

Os pacotes de dados PCAP possuem diversas informações úteis que podem ser utilizadas na descoberta do tipo de tráfego. Para extração destas informações foi utilizado o aplicativo tcptrace[12].

A versão utilizada do aplicativo informava: “Version: Ostermann's tcptrace -- version 6.6.7 -- Thu Nov 4, 2004 ; Compiled by 'buildd' at 'Tue Jan 3 06:54:50 UTC 2012' on machine 'barber' ”.

No comando do aplicativo tcptrace foram utilizadas as opções de formato de saída longo, reportar as estatísticas RTT (*Round-Trip Time*), que são as estatísticas de tempo para enviar um pacote de requisição de eco e o tempo para tê-lo de volta.

11 http://wiki.wireshark.org/Development/LibpcapFileFormat
12 http://www.tcptrace.org

5.4. CONVERSÃO DAS INFORMAÇÕES EXTRAÍDAS PARA O FORMATO ACEITO PELO WEKA

Foi desenvolvido para o WEKA pelo *Machine Learning Project* do Departamento de ciências da Computação da universidade de Weikato, um formato de arquivo de nome e extensão ARFF, sigla que representa *Attribute - Relation File Format*, ou Formato de Arquivo Relação - Atributo. Este arquivo descreve uma lista de instâncias compartilhando um conjunto de atributos, no formato de texto ASCII (Paynter, 2010)

Para conversão do formato CSV (*Comma - Separeted Values*, em português, Valores Separados por Vírgula) gerado pelo tcptrace foi utilizada uma rotina própria desenvolvida dentro do WEKA, com a seguinte sintaxe:

```
java weka.core.converters.CSVLoader filename.csv > filename.arff
```

5.5. DIVISÃO DOS DADOS EM TREINO E TESTE

O algoritmo usará uma parte dos dados para treinamento, no momento em que ele fará o processo para descobrir a forma de chegar ao resultado que ele possui; e outra parte para o teste de verificação, quando ele utilizará a o processo descoberto e verificará se é compatível com a resposta, indicando a eficiência do modelo. O conjunto foi separado em uma proporção de 90 % para treino e 10% para teste, conforme orienta o estudo publicado em forma de Nota Técnica por (Schaffer, 1993). Para executar esta separação foi utilizada a função interna *StratifiedRemoveFolds* do WEKA.

Para uma estimativa de acurácia dos resultados ainda mais apurada, conforme (Diamantidis et al., 2000), também foi feita a verificação pelo método de Validação Cruzada, já explicada no item 4.3 deste trabalho.

5.6. EXECUÇÃO DOS ALGORITMOS DE CLASSIFICAÇÃO

Foram executados vários testes com os algoritmos de classificação. A seguir apresenta-se os algoritmos que apresentaram melhores resultados.

5.6.1. ÁRVORE DE DECISÃO – ALGORITMO J.48 com modo de teste por arquivo externo

O algoritmo J.48 (Quinlan, 1986) é uma atualização do C4.5 (Quinlan, 1993) que permite a criação de modelos de decisão em árvore utilizando uma tecnologia onde, em todas as decisões que ele toma, visa àquela com a mais imediata e óbvia vantagem (Markey, 2011; Quinlan & Kohavi, 1999). O modelo de árvore de decisão é elaborado pela análise dos dados de treino e posteriormente utilizado para classificar os dados de teste. O J48 gera árvores de decisão avaliando a significância ou existência de cada atributo individual em cada nó. As árvores de decisão são construídas de forma hierárquica (do topo para a base), através da seleção do atributo mais apropriado para cada situação (Bouckaert et al., 2008) .

A classificação foi executada com o seguinte comando e parâmetros, sugeridos pela configuração padrão da aplicação *explorer* do WEKA:

weka.classifiers.trees.J48 -C 0.25 -M 2

Onde:

- O parâmetro -C é o fator confiança usado para limite de confiança (Bouckaert et al., 2008), sendo que valores menores incorrem em mais poda.

- O parâmetro -M define o número mínimo de instâncias, ou número mínimo de ocorrências por folha (Bouckaert et al., 2008).

A seguir, nas tabelas 5.1 e 5.2, verificamos os resultado da classificação através do algoritmo J.48, utilizando, para levantamento dos valores atributos o teste com arquivo externo criado com a função *StratifiedRemoveFolds* do WEKA..

Tabela 5.1: Resumo do resultado do teste na ÁRVORE DE DECISÃO – ALGORITMO J.48 – teste com arquivo externo

Atributo	**Valor**
Instâncias corretamente classificadas	97,8541%
Instâncias incorretamente classificadas	2,1459%
Erro médio absoluto	0,0288

Tabela 5.2: Precisão detalhada por classe na ÁRVORE DE DECISÃO – ALGORITMO J.48 – teste com arquivo externo

	Taxa VP	Taxa FP	Precisão	Cobertura	Medida F	Área ROC	Classe
	0,97	0	1	0,97	0,985	0,987	TORRENT
	1	0,03	0,929	1	0,963	0,987	WEB
Média Ponderada	0,979	0,008	0,98	0,979	0,979	0,987	

Onde,

- Taxa TP: é a taxa de Verdadeiros Positivos, ou seja, a porcentagem dos fluxos classificados corretamente em relação a todos os fluxos de determinada classe;

- Taxa FP: é a taxa de Falsos Positivos, ou seja, a porcentagem de fluxos classificados como determinada classe que não são realmente da referida classe.

- Precisão: Quantidade percentual das classes retornadas que são relevantes, de acordo com a equação 5.1 a seguir:

$$Precisão = \frac{VP}{VP + FP} \qquad \text{EQ (5.1)}$$

onde,

VP é a Taxa VP e

FP é a Taxa FP

- Cobertura: também chamada de revocação ou sensibilidade, definida de acordo com a equação 5.2 a seguir:

$$Cobertura = \frac{VP}{VP + FN} \qquad \text{EQ (5.2)}$$

onde,

VP é a Taxa VP (Verdadeiro Positivo) e

FN é a Taxa FN (Falso Negativo)

- Medida-F (*F-Measure*): é a medida de precisão do teste definida como uma média harmônica da precisão e da Cobertura (Sasaki, 2007), de acordo com a equação 5.3 a seguir :

$$Medida - F = 2 * \frac{Precisão * Cobertura}{Precisão + Cobertura} \qquad \text{EQ (5.3)}$$

- Área ROC (*Receiver Operating Characteristic* ou Característica de operação do receptor) é a disposição de valores de sensibilidade e especificidade em um gráfico, no qual se deve selecionar um ponto de corte que otimize os acertos (Bradley, 1997;

Prati, Batista, & Monard, 2008). Esta disposição gráfica que ilustra o desempenho de um sistema de classificador binário como seu limiar de discriminação é variada.

5.6.2. ÁRVORE DE DECISÃO – ALGORITMO J.48 com modo de teste por validação cruzada

Aqui, também utilizando o algoritmo J48, após a seleção do atributo, os dados de treino são divididos em sub-grupos, correspondendo aos diferentes valores dos atributos e o processo é repetido para cada sub-grupo até que uma grande parte dos atributos em cada sub-grupo pertençam a uma única classe. A indução por árvore de decisão é um algoritmo que habitualmente aprende um conjunto de regras com elevada acuidade (Bouckaert et al., 2008).

A seguir, nas tabelas 5.3 e 5.4, verificamos os dados do resultado da classificação através do algoritmo J.48, utilizando, para levantamento dos valores atributos o teste com a função de validação cruzada do WEKA.

Tabela 5.3: Resumo do teste ÁRVORE DE DECISÃO – ALGORITMO J.48 – teste com validação cruzada

Atributo	**Valor**
Instâncias corretamente classificadas	97,7544%
Instâncias incorretamente classificadas	2,2456%
Erro médio absoluto	0,0315

Tabela 5.4: Precisão detalhada por classe ÁRVORE DE DECISÃO – ALGORITMO J.48 – teste com validação cruzada

	Taxa VP	Taxa FP	Precisão	Cobertura	Medida F	Área ROC	Classe
	0,977	0,021	0,992	0,977	0,984	0,981	TORRENT
	0,979	0,023	0,942	0,979	0,96	0,981	WEB
Média Ponderada	0,978	0,021	0,978	0,978	0,978	0,981	

5.6.3. Algoritmo SVM com KERNEL RBF: exp(-gamma*|u-v|^2) - Modo de teste: arquivo externo

Algoritmos SVM (*Support Vector Machines*) são classificadores estatísticos. Eles utilizam dados de treinamento como vetores de suporte. No WEKA é possível selecionar a escolha do tipo de Kernel garantindo a otimização global [13]. RBF é a sigla inglês *radial basis function* (função de base radial)

A seguir, nas tabelas 5.5 e 5.6, verificamos os dados do resultado da classificação através do algoritmo SVM com KERNEL RBF: `exp(-gamma*|u-v|^2)`, utilizando, para levantamento dos valores atributos o teste um arquivo externo.

Tabela 5.5: Resumo do resultado do teste do algoritmo SVM com KERNEL RBF: `exp(-gamma*|u-v|^2)` - modo de teste: arquivo externo

Atributo	**Valor**
Instâncias corretamente classificadas	73,3906%
Instâncias incorretamente classificadas	26,6094%
Erro médio absoluto	0,2661

13 http://www.csie.ntu.edu.tw/~cjlin/libsvm/index.html

Tabela 5.6: Precisão detalhada por classe para algoritmo SVM com KERNEL RBF: `exp(-gamma*|u-v|^2)` - modo de teste: arquivo externo

	Taxa VP	Taxa FP	Precisão	Cobertura	Medida F	Área ROC	Classe
	1	0,954	0,73	1	0,884	0,523	TORRENT
	0,046	0	1	0,046	0,088	0,523	WEB
Média Ponderada	0,734	0,688	0,806	0,734	0,633	0,523	

5.6.4. Algoritmo SVM com KERNEL RBF: exp(-gamma*|u-v|^2) - Modo de teste: validação cruzada

A seguir, nas tabelas 5.7 e 5.8, verificamos os resultado da classificação através do algoritmo SVM com KERNEL: RBF: `exp(-gamma*|u-v|^2)`. Para obtenção dos valores foi utilizado o modo de teste de validação cruzada).

Tabela 5.7: Resumo dos resultados do teste do algoritmo SVM com KERNEL RBF: `exp(-gamma*|u-v|^2)` - modo de teste: validação cruzada

Atributo	**Valor**
Instâncias corretamente classificadas	74,7731%
Instâncias incorretamente classificadas	25,2269%
Erro médio absoluto	0,2523

Tabela 5.8: Precisão detalhada por classe para algoritmo SVM com KERNEL RBF: `exp(-gamma*|u-v|^2)` - Modo de teste: validação cruzada

	Taxa VP	Taxa FP	Precisão	Cobertura	Medida F	Área ROC	Classe
	1	0,906	0,741	1	0,851	0,547	TORRENT
	0,094	0	1	0,094	0,172	0,547	WEB
Média Ponderada	0,748	0,653	0,813	0,748	0,662	0,547	

5.6.5. Algoritmo SVM com KERNEL linear: `u'*v` - modo se teste: arquivo externo

A seguir, nas tabelas 5.9 e 5.10, verificamos os resultado da classificação através do algoritmo SVM com KERNEL linear: u'*v. Para obtenção dos valores foi utilizado o modo de teste de com arquivo externo fornecido pelo usuário.

Tabela 5.9: Resumo do teste do algoritmo SVM com KERNEL linear: `u'*v` - modo se teste: arquivo externo

Atributo	**Valor**
Instâncias corretamente classificadas	48,0687%
Instâncias incorretamente classificadas	51,9313%
Erro médio absoluto	0,5193

Tabela 5.10: Precisão detalhada por classe para algoritmo SVM com KERNEL linear: `u'*v` - modo se teste: arquivo externo

	Taxa VP	Taxa FP	Precisão	Cobertura	Medida F	Área ROC	Classe
	0,363	0,215	0,813	0,363	0,502	0,574	TORRENT
	0,785	0,637	0,323	0,785	0,457	0,574	WEB
Média Ponderada	0,481	0,333	0,676	0,481	0,49	0,574	

5.6.6. Algoritmo SVM com Kernel Linear: `u'*v` com modo de teste por validação cruzada

A seguir, nas tabelas 5.11 e 5.12, verificamos os resultado da classificação através do algoritmo SVM com KERNEL linear: u'*v. Para obtenção dos valores foi utilizado o modo de teste de validação cruzada.

Tabela 5.11: Resumo do teste do algoritmo SVM com KERNEL linear: `u'*v` – modo de teste: validação cruzada

Atributo	**Valor**
Instâncias corretamente classificadas	77,9264%
Instâncias incorretamente classificadas	22,0736%
Erro médio absoluto	0,2207

Tabela 5.12: Precisão detalhada por classe para algoritmo SVM com KERNEL linear: `u'*v` - modo de teste: validação cruzada

	Taxa VP	Taxa FP	Precisão	Cobertura	Medida F	Área ROC	Classe
	0,821	0,328	0,866	0,821	0,843	0,746	TORRENT
	0,672	0,179	0,591	0,672	0,629	0,746	WEB
Média Ponderada	0,779	0,286	0,79	0,779	0,783	0,746	

5.7. CONSOLIDAÇÃO DOS RESULTADOS

Para resumir os resultados é apresentada a seguir a tabela 5.13 com a consolidação das informações retornadas pelas execuções dos algoritmos sobre a base de dados, separada por grupos.

Tabela 5.13: Consolidação dos resultados

Algoritmo	**J.48**		**SVM**			
Kernel	**N/A**	**N/A**	**RBF**		**Linear**	
Teste	**Separação 90%/10%**	**Validação cruzada**	**Separação 90%/10%**	**Validação cruzada**	**Separação 90%/10%**	**Validação cruzada**
Instâncias corretamente classificadas	97,8541%	97,7544%	73,3906%	74,7731%	48,0687%	77,9264%
Instâncias incorreta-mente classificadas	2,1459%	2,2456%	26,6094%	25,2269%	51,9313%	22,0736%
Erro médio absoluto	0,0288	0,0315	0,2661	0,2523	0,5193	0,2207

No próximo capítulo apresentaremos a discussão final dos resultados e as propostas de trabalhos futuros sugeridas.

6. DISCUSSÃO DOS RESULTADOS E TRABALHOS FUTUROS

Foram avaliadas 2.093 instancias de um conjunto de dados no formato ARFF resultante da extração de informações de 6,1 GB de tráfego *BitTorrent* e 2,0 GB de outros tráfegos (excluindo *BitTorrent*) sendo principalmente de navegação WEB.

O algoritmo que demonstrou melhor desempenho geral na identificação foi o J.48 (árvore de decisão), sendo que o teste na separação manual dos dados em 90 % para treinamento e 10% para teste chegou a 97,85% de acurácia, sendo superior a (Le & But, 2009). Entre os testes executados com o algoritmo SVN, o feito com o Kernel Linear: u'*v apresentou resultado de 77,92 % de acurácia no teste com verificação com validação cruzada. Como, de acordo com (Bennett & Campbell, 2000), já foi mostrado que o SVM executa bem várias áreas de diversos problemas de classificação padrão, observamos neste caso que pode haver a necessidade de uma outra preparação mais elaborada dos dados de entrada. Como a árvore de decisão já apresentou resultados satisfatórios de forma mais fácil não entrou no escopo deste trabalho uma análise mais profunda do comportamento do SVM para este tipo de classificação, ficando como sugestão para trabalhos futuros.

A ferramenta WEKA mostrou-se viável para classificar esses dados, pois obteve uma boa acurácia nos resultados e pode ser usada no modo de linha de comando.

Outra sugestão de trabalhos futuros é a verificação se os algoritmos de classificação podem trabalhar de forma a diferenciar os diferentes tipos de conteúdos (mídia, programas, documentos etc) compartilhados através do BitRorrent.

REFERÊNCIAS BIBLIOGRÁFICAS

Acorn, J. (2008). *Forensics of BitTorrent*. *Test*. Roal Holloway, University of London Egham, Surrey TW20 0EX, England. Retrieved from http://www.ma.rhul.ac.uk/static/techrep/2008/RHUL-MA-2008-04.pdf

Bauer, K., McCoy, D., Grunwald, D., & Sicker, D. (2009). BitStalker: Accurately and efficiently monitoring bittorrent traffic. *2009 First IEEE International Workshop on Information Forensics and Security (WIFS)*, 181-185. Ieee. doi:10.1109/WIFS.2009.5386457

Bennett, K. P., & Campbell, C. (2000). Support vector machines: hype or hallelujah? *ACM SIGKDD Explorations Newsletter*, *2*(2), 13. Retrieved from http://dl.acm.org/citation.cfm?id=380995.380999

Bindal, R., Cao, P., Chan, W., Medved, J., Medval, J., Suwala, G., Bates, T., et al. (2006). Improving Traffic Locality in BitTorrent via Biased Neighbor Selection. , *2006. ICDCS 2006*. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1648853

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. (M. Jordan, J. Kleinberg, & B. Scholkopf, Eds.)*Pattern Recognition* (Vol. 4, p. 738). Springer. doi:10.1117/1.2819119

Bouckaert, R. R., Frank, E., Hall, M., Kirkby, R., Reutemann, P., Seewald, A., & Scuse, D. (2008). WEKA Manual for Version 3-6-0.

Bradley, A. P. (1997). The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, *30*(7), 1145-1159. doi:10.1016/S0031-3203(96)00142-2

Burges, C. J. C. (1998). A Tutorial on Support Vector Machines for Pattern Recognition, *167*, 121-167.

Chokkalingam, A., & Riyaz, F. (2002). Bittorrent Protocol Specification v1.0. Retrieved December 25, 2011, from http://wiki.theory.org/BitTorrentSpecification

Cohen, B. (2003). Incentives build robustness in BitTorrent. *Workshop on Economics of Peer-to-Peer systems*, *6*, 68–72. Retrieved from http://www.ittc.ku.edu/~niehaus/classes/750-s06/documents/BT-description.pdf

Cohen, B. (2008). The BitTorrent protocol specification. *Standard, January*. Retrieved from http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:The+BitTorrent+Protocol+Specification#0

Cuevas, R., Laoutaris, N., Yang, X., Siganos, G., & Rodriguez, P. (2009). Deep Diving into BitTorrent Locality. *IEEE INFOCOM'11*.

Dai, L., Yang, J., & Lin, L. (2010). A comprehensive system for P2P classification. *Network Infrastructure and Digital Content, 2010 2nd IEEE International Conference on* (pp. 561–563). IEEE.

Dalpian, G. M., & Benites, C. A. A. (2007). Ferramenta Para Monitoramento de Redes P2P-EspiaMule. *The Second International Conference of Forensic Computer Science* (pp. 70-72).

Diamantidis, N. a., Karlis, D., & Giakoumakis, E. a. (2000). Unsupervised stratification of cross-validation for accuracy estimation. *Artificial Intelligence*, *116*(1-2), 1-16. doi:10.1016/S0004-3702(99)00094-6

Dischinger, M., Marcon, M., Guha, S., Gummadi, K. P., Mahajan, R., & Saroiu, S. (2010). Glasnost: Enabling End Users to Detect Traffic Differentiation.

Doumen, J., Roozenburg, J., & Brinke, M. (2009). Next-Share Plataform M8-Specification Part, (4), 1-169.

Erman, D. (2005). *BitTorrent Traffic Measurements and Models*.

Erman, D., Ilie, D., & Popescu, A. (2006). BitTorrent Traffic Characteristics. *International Multi-Conference on Computing in the Global Information Technology - (ICCGI'06)*, 42-42. Ieee. doi:10.1109/ICCGI.2006.1690316

Erman, J., Mahanti, A., & Arlitt, M. (2006). Internet Traffic Identification using Machine Learning. *Global Telecommunications Conference, 2006. GLOBECOM '06. IEEE* (pp. 1-6). San Francisco, CA: IEEE. doi:10.1109/GLOCOM.2006.443

Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). From data mining to knowledge discovery in databases. *AI magazine*, *17*(3), 37-54. Retrieved from http://www.aaai.org/ojs/index.php/aimagazine/article/viewArticle/1230

Fernandes, D., & Fernandes, R. (2009). Bittorrent: um estudo avançado sobre um dos protocolos mais usados na internet. *connepi2009.ifpa.edu.br*, (1). Retrieved from http://connepi2009.ifpa.edu.br/connepi-anais/artigos/102_2317_209.pdf

Frawley, W. J., Piatetsky-Shapiro, G., & Matheus, C. J. (1992). Knowledge discovery in databases: An overview. *Ai Magazine*, *13*(3), 57-70. Retrieved from http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Knowledge+Discovery+in+Databases:+An+Overview#0

Geisser, S. (1993). *Predictive Inference: An Introduction*. *SIAM Review* (Vol. 36, p. 264). New York, New York, USA: Chapman & Hall.

HTCIA. (2010). 2010 Report on Cyber Crime Investigation, *95747*(916).

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA Data Mining Software: An Update. *SIGKDD Explorations*, *11*(1), 10-18.

Ipoque. (2009). *Internet Study 2008/2009*. *IPOQUE Report* (p. 14). Retrieved from http://www.ipoque.de/userfiles/file/ipoque-Internet-Study-08-09.pdf

Izal, M., Urvoy-Keller, G., Biersack, E., & Felber, P. (2004). Dissecting bittorrent: Five months in a torrent's lifetime. *Passive and Active*. Retrieved from http://www.springerlink.com/index/fg8hqw4136t0vtx9.pdf

Konrath, M. A., Barcellos, M. P., Silva, J. F., Gaspary, L. P., & Dreher, R. (2007). Atacando um Enxame com um Bando de Mentirosos: vulnerabilidades em BitTorrent, 883-896.

Lange, R. (2011). Identificação de tráfego do emule usando redes neurais artificiais rodrigo lange.

Le, T. M., & But, J. (2009). BitTorrent Traffic Classification. *caia.swin.edu.au*, (October). Retrieved from http://caia.swin.edu.au/reports/090227A/CAIA-TR-090227A.pdf

Luan, H., & Tsang, D. H. K. (2006). A simulation study of block management in BitTorrent. *Proceedings of the 1st international conference on Scalable information systems - InfoScale '06*, 58. New York, New York, USA: ACM Press. doi:10.1145/1146847.1146906

Maimon, O. (Tel A. U., & Rokach, L. (Ben-G. U. of the N. (Eds.). (2010). *Data Mining and Knowledge Discovery Handbook* (Second.). Springer.

Mansilha, R., Mezzomo, A., Facchini, G., & Gaspary, L. (2010). Observando o Universo BitTorrent Através de Telescópios. *sbrc2010.inf.ufrgs.br*, 233-246. Retrieved from http://sbrc2010.inf.ufrgs.br/anais/data/pdf/trilha/st05_03_artigo.pdf

Markey, J. (2011). Using Decision Tree Analysis for Intrusion Detection: A How-To Guide. *SANS Institute Reading Room*, (Security 503).

Melo, Marcelo Damasceno de () Instituto Federal de Educação, C. e T. do R. G. do N. M. (2010). Introdução a mineração de dados utilizando o weka. *V CONNEPI*, (1).

Paynter, G. (Machine L. P. at the D. of C. S. of T. U. of W. (2010). Attribute-Relation File Format (ARFF). Retrieved May 25, 2012, from http://weka.wikispaces.com/ARFF

Penna, B., & Carvalho, R. D. (1995). O estado da arte em métodos para reconhecimento de padrões: Support Vector Machine Bernardo Penna Resende de Carvalho 1.

Peron, A. (2012). *SIT E CLIT: Ferramentas e Metodologia para Aprimoramento de Investigações Criminais Utilizando Interceptações de Conexão à Internet.* Universidade de Brasília.

Pouwelse, J. A., Garbacki, P., Epema, D. H. J., & Sips, H. J. (2005). The Bittorrent P2P File-sharing System: Measurements and Analysis. *Department of Computer Science, Delft University of Technology, the Netherlands, Lecture No*(Oct), 1829–1841. Wiley Online Library. Retrieved from http://onlinelibrary.wiley.com/doi/10.1002/cbdv.200490137/abstract

Prati, R. C., Batista, G. E. A. P. A., & Monard, M. C. (2008). Curvas ROC para avaliação de classificadores. *Revista IEEE América*, 1-8. Retrieved from http://www.revistaieeela.pea.usp.br/ieee/issues/vol6issue2june2008/6TLA2_13Prati.pdf

Qiu, D., & Srikant, R. (2004). Modeling and performance analysis of BitTorrent-like peer-to-peer networks. *ACM SIGCOMM Computer Communication Review* (Vol. 34, pp. 367–378). ACM. Retrieved from http://dl.acm.org/citation.cfm?id=1030194.1015508

Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, 81-106. Retrieved from http://www.springerlink.com/index/ku63wm5513224245.pdf

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning.pdf*. *Morgan Kaufmann* (Vol. 5, p. 270). Morgan Kaufmann. doi:10.1016/S0019-9958(62)90649-6

Quinlan, J. R., & Kohavi, R. (1999). Decision Tree Discovery, *3*(Hunt 1962).

Roberto Santos, F., da Costa Cordeiro, W. L., Gaspary, L. P., & Barcellos, M. P. (2010). Choking polluters in BitTorrent file sharing communities. *2010 IEEE Network Operations and Management Symposium - NOMS 2010*, 559-566. Ieee. doi:10.1109/NOMS.2010.5488657

Sasaki, Y. (2007). The truth of the F-measure, 1-5.

Schaffer, C. (1993). Selecting a classification method by cross-validation. *Machine Learning*, *143*, 135-143. Retrieved from http://www.springerlink.com/index/K2873P131868H813.pdf

Sen, S., Spatscheck, O., & Wang, D. (2004). Accurate, scalable in-network identification of p2p traffic using application signatures. *Proceedings of the 13th conference on World*

Wide Web - WWW '04, 512. New York, New York, USA: ACM Press. doi:10.1145/988672.988742

Silva, M. P. S. (2004). Mineração de Dados - Conceitos, Aplicações e Experimentos com Weka. *Universidade do Estado do Rio Grande do Norte (UERN) BR 110, Km 48, 59610-090, Mossoró, RN, Brasil*.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. (Michael Jordan & S. L. Lauritzen, Eds.)*Springer* (Vol. 8, p. 188). Springer. doi:10.1109/TNN.1997.641482

Witten, I. H., Frank, E., & Hall, M. A. (2011). *Data Mining: Practical machine learning tools and techniques* (3ª ed., p. 665). Burlington, MA 01803, USA: Morgan Kaufmann.

Wu, D., Dhungel, P., Hei, X., Zhang, C., & Ross, K. W. (2010). Understanding Peer Exchange in BitTorrent Systems. *2010 IEEE Tenth International Conference on Peer-to-Peer Computing (P2P)*, 1-8. Ieee. doi:10.1109/P2P.2010.5569967

Yang, Z., Li, L., Ji, Q., & Zhu, Y. (2010). An Omnibus Identification of BitTorrent Traffic in a Stub Network. *2010 3rd International Symposium on Parallel Architectures, Algorithms and Programming*, (09), 346-353. Ieee. doi:10.1109/PAAP.2010.47

Yang, Z., Li, L., Ji, Q., & Zhu, Y. (2012). Cocktail method for BitTorrent traffic identification in real time. *Journal of Computers*, *7*(1), 85-95. doi:10.4304/jcp.7.1.85-95

Yu, L., & Chen, M. (2010). Geographical distribution of peers of BitTorrent in multi-granularity scale. *2010 International Conference on Computer and Communication Technology (ICCCT)*, 93-98. Ieee. doi:10.1109/ICCCT.2010.5640419