

**Metodologia para extração de conhecimento em rede de computadores**

**Trabalho de Conclusão de Curso**

**Engenharia da Computação**

**Diogo Lopes da Silva**

**Orientador: Edison de Queiroz Albuquerque**

**Co-orientadora: Zuleika Tenório**

****

**Universidade de Pernambuco**

**Escola Politécnica de Pernambuco**

**Graduação em Engenharia de Computação**

**Diogo Lopes da Silva**

**Metodologia para extração de conhecimento em rede de computadores**

Monografia apresentada como requisito parcial para obtenção do diploma de Bacharel em Engenharia de Computação pela Escola Politécnica de Pernambuco – Universidade de Pernambuco.

**Recife, Junho de 2011.**

**De acordo**

**Recife**

**\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_/\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Orientador da Monografia**

*Dedico à minha mãe Carmen.*

**Agradecimentos**

Primeiramente agradeço a Deus porque sei quem sem ele não teria forças para prosseguir diante de tantas adversidades da vida.

Também quero agradecer à minha mãe, Carmen, por todos os princípios e valores, e por sempre ter me apoiado para a conclusão da minha graduação. Em especial a minha amada avó materna, Isabel – in memória, por sempre ter acreditado em mim, por todos os conselhos e por ser um exemplo de vida a ser seguido. Ao meu irmão por sempre torcer por mim. Também agradeço à minha madrinha, Vânia, por me encorajar sempre a terminar minha graduação, e, por acreditar num sonho chamado educação e por saber como ninguém que a educação pode verdadeiramente transformar as vidas das pessoas.

Agradeço à minha esposa, Márcia, por estar ao meu lado a todo o momento e por sonhar junto comigo. À minha filha, Fernanda, ser a luz que me guia

.Agradeço aos professores do Curso de Engenharia da Computação, que certamente contribuíram para minha formação profissional e pessoal.

À minha chefa, Zuleika, e meu colega de equipe, Fernando Antônio, por me incentivarem a crescer profissionalmente e acreditarem no meu potencial como profissional.

Também agradeço aos meus colegas de turma em especial Cristiano, Emanuel, Hiroshi e Marcos pelos esforços realizados para execução dos infinitos projetos da POLI.

**Resumo**

Este trabalho tem como objetivo apresentar um método de identificação e classificação utilizando o algoritmo de Árvores de decisão para classificação de tráfego de rede gerado por aplicações P2P(*Peer to Peer*), por exemplo. Desta forma, utilizando o algoritmo de Árvores de Decisão J48, que é uma técnica de classificação muito usada, será possível apresentar a utilização de um algoritmo usado em aprendizagem de máquina e mineração de dados, na resolução de problemas de identificação de tráfego nas redes atualmente, para isto, neste trabalho será apresentada uma ferramenta de captura de tráfego que possibilitará a geração de bases de dados utilizadas pelo algoritmo de mineração de dados.

**Abstract**

This work aims to present a method of identification and classification algorithm using decision trees for classification of network traffic generated by P2P (Peer to Peer), for example. Thus, using the Decision Trees algorithm J48, which is a widely used classification technique, will be possible use of an algorithm used in machine learning and data mining, in solving identification problems in traffic networks currently For this, this work presents a tool to capture traffic that allow for the generation of databases used by the algorithm of data mining.

**Sumário**

Sumário

[1 Introdução 1](#_Toc296867800)

[1.1 Formulação do Problema 2](#_Toc296867801)

[1.2 Motivação 4](#_Toc296867804)

[1.3 Estrutura da monografia 8](#_Toc296867806)

[2 Revisão bibliográfica 9](#_Toc296867807)

[2.1 Técnicas para classificação de tráfego 9](#_Toc296867809)

[2.1.1 Classificação baseada em portas 9](#_Toc296867810)

[2.1.2 Classificação baseada na Inspeção de Payload 10](#_Toc296867811)

[2.2 Processo KDD 11](#_Toc296867815)

[2.2.1 Identificação de domínio da aplicação 12](#_Toc296867816)

[2.2.2 Seleção dos dados 12](#_Toc296867817)

[2.2.3 Pré-processamento dos dados 12](#_Toc296867818)

[2.2.4 Transformação dos dados 13](#_Toc296867819)

[2.2.5 Mineração de dados 14](#_Toc296867820)

[2.2.6 Interpretação e avaliação dos Padrões 14](#_Toc296867821)

[2.2.7 Utilização do conhecimento 15](#_Toc296867822)

[2.3 Mineração de dados 15](#_Toc296867823)

[2.4 Classificação 17](#_Toc296867824)

[2.4.1 Classificação com Árvores de Decisão 18](#_Toc296867825)

[2.4.2 Particionamento da Árvore de Decisão 20](#_Toc296867826)

[2.4.3 Algoritmo J48 22](#_Toc296867827)

[3 Ferramentas 25](#_Toc296867829)

[3.1 Coleta dos dados 25](#_Toc296867831)

[3.2 Descrição dos dados 29](#_Toc296867832)

[3.3 Ferramenta WEKA 31](#_Toc296867833)

[4 Resultados 34](#_Toc296867834)

[4.1 Metodologia 34](#_Toc296867835)

[4.2 Experimentos e Resultados 36](#_Toc296867836)

[5 Conclusão 43](#_Toc296867837)

[Anexo A Árvore de decisão sem poda. 47](#_Toc296867838)

[Anexo B Árvore de decisão com poda. 50](#_Toc296867839)

[Anexo C Código Fonte 53](#_Toc296867840)

**Índice de Figuras**

[**Figura 1.** Opção de exportação do *Wireshark.* 5](#_Toc296868691)

[**Figura 2.** Exemplo de saída de exportação *Wireshark.* 6](#_Toc296868692)

[**Figura 3.** Processo KDD Fonte: Fayyad, 1996. 11](#_Toc296868693)

[**Figura 4.** Uso do modelo de classificação, Fonte: TAN[29] 18](#_Toc296868694)

[**Figura 5.** Árvore de decisão para problema de classificação de mamíferos, Fonte: TAN [29] 19](#_Toc296868695)

[**Figura 6.** Exemplo de arquivo no formato CSV. 26](#_Toc296868696)

[**Figura 7.** Seleção das Interfaces de rede. 27](#_Toc296868697)

[**Figura 8.** Captura dos pacotes 27](#_Toc296868698)

[**Figura 9.** Criação da Base de dados. 28](#_Toc296868699)

[**Figura 10.** Tela Inicial do WEKA. 31](#_Toc296868700)

[**Figura 11.** Tela de pré-processamento do WEKA. 32](#_Toc296868701)

[**Figura 12.** Tela de classificação do WEKA. 33](#_Toc296868702)

[**Figura 13.** Arquitetura da metodologia 36](#_Toc296868703)

[**Figura 14.** Ramificação da árvore de decisão para ARES e BITTORRENT no primeiro experimento. 37](#_Toc296868704)

[**Figura 15.** Precisão da árvore gerada, para o treinamento sem poda. 38](#_Toc296868705)

[**Figura 16.** Matriz de confusão da árvore gerada, para o treinamento sem poda. 39](#_Toc296868706)

[**Figura 17.** Ramificação da árvore de decisão para ARES e BITTORRENT para o segundo experimento. 40](#_Toc296868707)

[**Figura 18.** Precisão da árvore gerada, para o treinamento com poda. 41](#_Toc296868708)

[**Figura 19.** Matriz de confusão da árvore gerada, para o treinamento com poda. 42](#_Toc296868709)

**Índice de Tabelas**

**Tabela 1.** Conjunto de dados dos vertebrados, Fonte: TAN [29] 17

**Tabela de Símbolos e Siglas**

API - Aplication Programming Interface

CSV - Comma Separated Values

DPI - *Deep Packet Inspection*

*FTP- File Transfer Protocol*

HTTP *- HiperText Transfer Protocol*

*HTTPS - Hiper Text Transfer Protocol Secure*

ID3 - Iterative Dichotomiser 3

KDD - *Knowledge Discovery in Databases*

MDL - Minimum Description Length

P2P - Peer to Peer

*SSH - Secure Shell*

WEKA - *Waikato Environment for Knowledge Analysis*

# 1 Introdução

Atualmente, existem muitas aplicações sendo desenvolvidas e comercializadas todos os dias, e conseguir identificar as características do fluxo de rede gerado por sua respectiva aplicação não é uma tarefa trivial. A identificação do fluxo de tráfego gerado entre aplicações é uma tarefa muito importante para o monitoramento e gerenciamento de uma rede.

Nesse contexto, criar uma ferramenta que coletasse dados referentes aos fluxos gerados por aplicações e protocolos dentro de uma rede é de fundamental importância, isso permitiria o uso do processo KDD (*Knowledge Discovery in Databases*) que pode ser utilizado para auxiliar a descoberta de conhecimento útil em grandes bases de dados [6].

Uma das principais etapas do processo KDD, a Mineração de Dados, consiste na aplicação de algoritmos com a finalidade de extrair padrões de comportamento em uma base de dados [7]. Uma das mais importantes tarefas de previsão é a classificação. Dada essa natureza, essa tarefa foi selecionada para ser utilizada nesse trabalho. Classificação é a tarefa de organizar objetos em uma entre diversas categorias pré-definidas. [29]

## Formulação do Problema

Existem dois métodos bastante utilizados na identificação/classificação de tráfego gerado entre aplicações. O primeiro é o método de classificação baseado em portas conhecidas, que realiza a análise do número da porta empregada para determinada aplicação. Essa técnica se mostra bastante ineficaz, já que grande parte das aplicações faz uso randômico de portas.

O segundo método é a classificação baseada na inspeção do *payload* ou *Deep Packet Inspection* (DPI) [17, 33]. Essa técnica, por sua vez, elimina o problema do uso aleatório do número da porta empregada para uma determinada aplicação. A técnica funciona a partir do classificador que extrai a carga útil da aplicação do pacote TCP/UDP e faz uma varredura em cada pacote em busca de assinaturas que possam identificar o tipo do fluxo.

Assinaturas geralmente incluem uma seqüência de *Bytes/strings* e *offset* que são únicos da aplicação e são característicos delas [2]. Porém, o DPI não é capaz de analisar a carga útil para um tráfego criptografado. A utilização de técnicas de mineração de dados [1] pode auxiliar na extração de informações importantes contidas nas bases de dados.

Este trabalho visa apresentar uma metodologia para identificação e classificação de tráfego de rede sem a realização de inspeção de *payload* e a utilização de portas conhecidas. Sendo assim, serão estudados conceitos e técnicas de mineração de dados e aplicar o conhecimento adquirido para descobrir informações não explicitas nas bases de dados referentes aos tráfegos de rede. Este trabalho também apresenta a utilização do WEKA (*Waikato Environment for Knowledge Analysis)* [9], que é um software livre com implementações de diversas técnicas de Mineração de Dados, para auxiliar na extração do conhecimento [4]. Especialmente, este trabalho se concentra em:

* Selecionar dados disponíveis nas bases de dados referentes a tráfegos de redes;
* Selecionar técnicas de Mineração de Dados para serem aplicadas aos dados obtidos;
* Projetar e desenvolver uma ferramenta para extrair os dados selecionados dos tráfegos de redes.
* Utilizar o WEKA para aplicar as técnicas selecionadas;
* Analisar e organizar as informações obtidas, de forma que o conhecimento gerado possa ser utilizado por gerentes e administradores de redes.

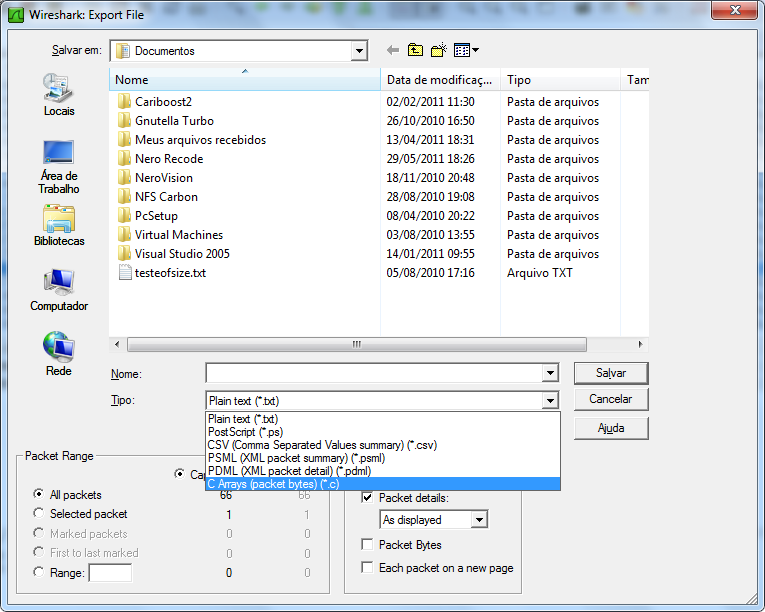


## Motivação

Para a realização deste trabalho de conclusão de curso de Extração de conhecimento em redes de computadores é preciso uma ferramenta que coletasse os dados que trafegam em uma rede de computadores e conseguisse exportar esses dados em base de dados para ser utilizada por uma ferramenta de mineração de dados, nesse trabalho a ferramenta usada para mineração de dados foi WEKA.

Porém, no inicio dos estudos para viabilização desse trabalho a ferramenta utilizada na coleta de dados foi a ferramenta *open source* *Wireshark[[1]](#footnote-2)*, porque é uma ferramenta de captura de pacotes com uma interface gráfica amigável e de uso bastante difundido entre administradores e gerentes de redes. Entretanto, o processo da exportação dos dados capturados para criação das bases de dados a serem utilizadas pelo WEKA era bastante complexo. E, além disso, nem todos os campos da pilha dos protocolos TCP/IP eram exportados e isso era de fundamental importância, pois, esses campos eram necessários para a criação das bases de dados consistentes para esse trabalho. Então, diante dessas dificuldades, existiam duas soluções possíveis, a primeira seria tratar a saída gerada pelo *Wireshark* e a segunda desenvolver uma ferramenta própria para coleta de dados.

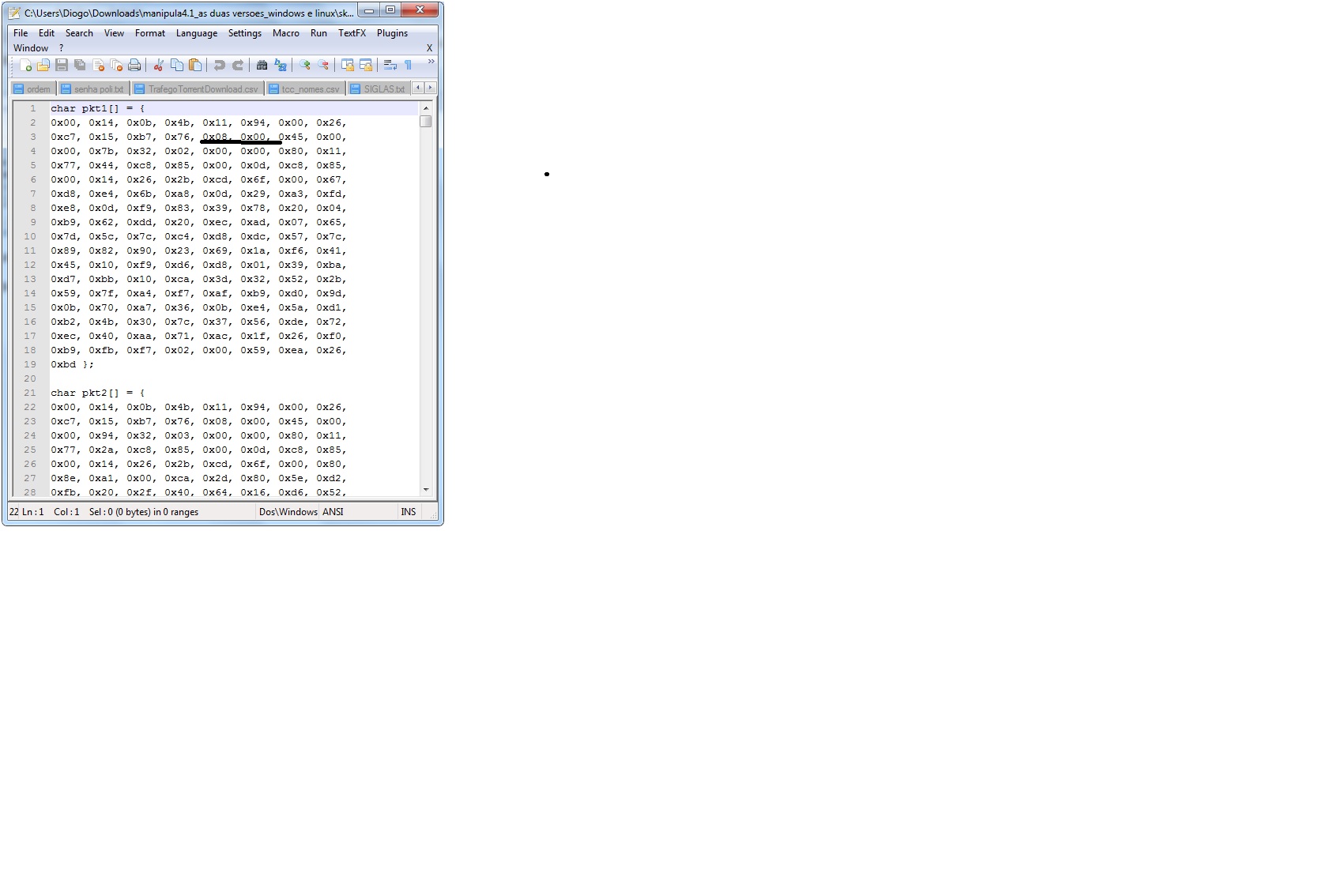
Na primeira, solução, era o desenvolvimento de uma aplicação que conseguisse ler a saída exportada pelo *Wireshark,* porém o único modo de exportar os dados contendo todos os campos que compõe cada pacote era exportar o arquivo no formato C arrays como mostrado na Figura 4.



1. Opção de exportação do *Wireshark.*

Onde o arquivo exportado contém *Arrays* que representam cada pacote contido no fluxo capturado, e cada *Array* écomposto pelo Hexadecimal correspondente aos campos do cabeçalho da camada de rede equivalente, como pode ser observado na Figura 5. Então, para conseguir tratar os dados exportados pelo *Wireshark*  era preciso saber previamente a posição do valor em Hexadecimal correspondente ao campo avaliado. Por exemplo, para avaliar o atributo TF, descrito com mais detalhes na seção 4.2, seria preciso tratar os valores que estão na posição sublinhada na Figura 5, neste caso o valor é, em Hexadecimal, 0800 que em decimal é 2048 como pode ser visto na Figura 8.

Então, devido à complexidade da primeira alternativa, para solução do problema da coleta de dados, foi escolhido implementar o desenvolvimento de uma ferramenta própria para coleta de dados.



1. Exemplo de saída de exportação *Wireshark.*

A ferramenta para coleta dos dados foi desenvolvida utilizando a linguagem Java[[2]](#footnote-3), pois é uma linguagem muito usada no meio acadêmico e multiplataforma. Também foi necessário o uso da API(*Aplication Programming Interface*) JPCAP[[3]](#footnote-4), que permite a captura de pacotes que trafegam em uma rede de computadores na linguagem Java. Maiores detalhes a respeito das funcionalidades desta ferramenta, desenvolvida neste trabalho de conclusão de curso, encontram-se no capítulo 4 nas seções 4.1 e 4.2. O código fonte da ferramenta encontra-se no Anexo C



## Estrutura da monografia

No capítulo 2, é apresentada uma revisão bibliográfica que abrange os principais conceitos necessários para a compreensão deste trabalho.

No capítulo 3, é apresentada a ferramenta desenvolvida para extração dos dados a partir do tráfego de rede gerado por aplicações e protocolos analisados neste trabalho.

No capítulo 4, são descritas as metodologias utilizadas durante o desenvolvimento e os resultados da aplicação dos algoritmos de mineração de dados.

O capítulo 5 contém as conclusões e dificuldades encontradas. Além disso, neste capítulo são sugeridos trabalhos futuros.

# 2 Revisão bibliográfica

Neste capítulo são apresentados conceitos e definições necessários à compreensão deste trabalho. Na seção 2.1, são apresentadas as etapas do processo KDD. A seguir, na seção 2.2, é apresentado o conceito de mineração de dados, bem como suas principais tarefas. Na seção 2.3 são apresentados os principais conceitos sobre classificação, funcionamento de uma árvore de decisão, métricas para selecionar a melhor divisão e o algoritmo J48.



## Técnicas para classificação de tráfego

Diferentes técnicas são propostas para classificação de rede. A classificação de tráfego consiste em dois passos. Primeiro, analisar o tráfego para extrair algumas características do tráfego e seus respectivos valores que caracterizam as diferentes classes de tráfego, ou seja, as assinaturas de cada classe de tráfego. Por ultimo, o tráfego desconhecido pode ser classificado avaliando suas características em relação às características descobertas durante a análise.

As características usadas durante o processo de classificação pode ser obtido dos cabeçalhos dos pacotes, carga útil dos pacotes ou fluxos de tráfego, onde fluxos são formados por uma filaunidirecional entre pacotes de um dado par de endereços IP.

### Classificação baseada em portas

A classificação baseada em portas leva em consideração os 16 bits dos campos *port numbers* da camada de transporte utilizadospor servidores para determinar o especifico processo que recebe certo fluxo de tráfego. O *port numbers para* específicos processos é controlado pela *Internet Assigned Numbers Authority (IANA)* [30].

Os primeiros estudos de classificação de tráfego confiavam no fato muitos serviços de rede executam nas “portas conhecidas”, ou seja, as portas padrões da camada de transporte registrada na lista da IANA [18]. Identificação por portas na camada de transporte foi empregada, especialmente para detecção de tráfego P2P (Peer to Peer) [25]. Alguns protocolos não são registrados na IANA, também é possível que IANA não defina portas para todas as aplicações, ou que algumas portas sejam definidas ou ambiguamente. Além disso, serviços padrões podem rodar em portas não padronizadas devido às restrições de controle de acesso dos sistemas operacionais [4]. E, algumas aplicações, como P2P, não usam intencionalmente um conjunto predefinido de portas para não ser detectadas [14]. Portanto, classificação baseada em portas pode produzir resultados falsos se uma aplicação usa outra porta conhecida ou falha se nenhuma combinação para a porta é encontrada.

### Classificação baseada na Inspeção de Payload

### Esta técnica de classificação de tráfego é baseada na investigação das múltiplas camadas de rede incluindo a camada de aplicação [15,19]. Cada pacote tem seu *payload* examinado bit por bit até ser localizado uma sequência especifica de bits que represente uma assinatura de um protocolo ou aplicação. Se tal sequência de bits é encontrada, o pacote pode ser corretamente rotulado.

Técnicas de inspeção de *payload* são comumente usadas na detecção de tráfego P2P [13, 26,12] e na detecção de intrusão [31]. Embora tenha uma alta precisão, a classificação baseada em inspeção de *payload* tem algumas desvantagens. Primeiro, é uma técnica complexa que requer um alto poder computacional e grandes capacidades para armazenamento. Por isso, para *links* de alta velocidade é preciso hardware específico, pois softwares que empregam inspeção de *payload* não são escaláveis. Segundo, estas técnicas podem falhar se na classificação se novas aplicações são encontradas com novas assinaturas, se o tráfego está em um túnel, e torna-se completamente ineficaz se o tráfego fim a fim está criptografado. E por ultimo, o problema da violação da privacidade dos usuários, pois essa técnica de classificação faz uma análise de todos os dados, de qualquer usuário, que estão trafegando na rede.



## Processo KDD

Historicamente, encontrar padrões em dados é conhecido por diferentes nomes em diferentes comunidades (por exemplo, extração de conhecimento, descoberta de informação, arqueologia de dados, processamento de padrões de dados e Mineração de Dados). O termo Mineração de Dados é muitas vezes utilizado em comunidades de estatísticos, pesquisadores de banco de dados, e mais recentemente por gerentes de sistemas de informação. Mas segundo Fayyad et. al [7], o termo KDD (*knowledge-discovery in databases* ) referencia o processo global de descobrir conhecimento útil em grandes bases de dados, sendo a Mineração de Dados um passo particular desse processo que consiste na execução de algoritmos de reconhecimento de padrões em base de dados.

KDD pode ser definido como [6]: O processo não trivial de identificação de padrões válidos, inovadores, potencialmente úteis e principalmente compreensíveis em bases de dados.

O processo KDD é interativo e iterativo, envolvendo numerosos passos com muitas decisões tomadas pelo analista. Nas próximas seções, são apresentados os passos básicos que compõem esse processo, conforme fluxograma ilustrado na Figura 3.



1. Processo KDD Fonte: Fayyad, 1996.

### Identificação de domínio da aplicação

Nesse passo é realizado um estudo do domínio da aplicação para definir os objetivos e metas a serem alcançados com o processo KDD. Esse passo é importante, pois para que seja extraída informação útil dos dados, as pessoas envolvidas no processo KDD devem possuir algum grau de conhecimento sobre a área da aplicação.

### Seleção dos dados

Os sistemas, normalmente, armazenam vários atributos de um objeto, mas nem sempre todos estes atributos são relevantes para a análise em questão. Assim, faz-se necessário uma criteriosa avaliação de quais atributos realmente agregam informações ao conjunto de dados para que esses venham a ser utilizados no processo KDD.

A qualidade dos dados armazenados é muito importante, pois ela determina a qualidade dos resultados obtidos, de forma diretamente proporcional. Porém, o processo KDD ainda é muito dependente da avaliação de analistas humanos e do seu conhecimento sobre a base de dados pesquisada, pois caso um atributo que contenha informações importantes seja desprezado nesta fase o resultado poderá não ser satisfatório.

### Pré-processamento dos dados

Nos problemas reais os dados tendem a ser incompletos, ruidosos e inconsistentes, o que torna necessária a aplicação de técnicas para corrigir essas falhas. A seguir são apresentadas algumas das atividades que podem ser realizadas na etapa de limpeza dos dados [10]:

* Valores ausentes - Algumas vezes, os valores de alguns atributos não estão presentes nas bases de dados, para contornar esse problema pode-se: ignorar os registros que possuem valores ausentes; preencher os valores ausentes manualmente; usar um valor constante, usar o valor médio do atributo ou usar valores estatísticos para preencher os valores ausentes.
* Valores fora do padrão (*outliers*) - Para resolver esse tipo de problema pode-se utilizar uma das seguintes técnicas: agrupamento, que consiste no agrupamento dos valores similares, facilitando a identificação e exclusão de valores fora do padrão; inspeção humana e computarizada, que consiste em uma inspeção feita por pessoas e computadores para identificar e excluir os valores fora do padrão; e regressão, na qual os dados podem ser ajustados por meio de funções de regressão.
* Dados inconsistentes - Esses erros ocorrem normalmente porque o usuário entra com um dado incorreto, mas pode ocorrer também redundância de dados, ou seja, dados que possuem o mesmo valor semântico, mas que foram inseridos com nomes diferentes.

Uma forma de resolver esse problema é por meio da análise de correlação, que consiste na medida de relacionamento entre dois atributos.

### Transformação dos dados

Muitas vezes é necessário realizar transformações nos dados para que os algoritmos de mineração possam ser executados, dentre as mais comuns estão:

* Normalização de atributos. Forma de harmonizar as escalas dos atributos em um pequeno intervalo especificado.
* Padronização de atributos. Atributos redundantes devem ser eliminados, utilizando a análise de correlação para mapear múltiplos atributos para uma simples entidade, por exemplo.
* Redução dos dados.
* Agregação. Agrega e sumariza os dados. Por exemplo, os dados de vendas diárias podem ser agregados de forma a calcular os montantes mensais e anuais.
* Compressão de dados. Mecanismos de codificação são usados para reduzir o tamanho do conjunto de dados.
* Redução da dimensionalidade. Eliminação de atributos irrelevantes à técnica de mineração aplicada.
* Redução Numérica. Diminuir o número de instâncias, por exemplo.

### Mineração de dados

Esse é o passo onde os padrões são descobertos por meio de algoritmos específicos, tais como K-means e Apriori, que são aplicados várias vezes até que sejam extraídas informações úteis dos dados. Por ser um dos passos mais importantes do processo KDD, este passo é descrito mais detalhadamente na seção 2.2.

### Interpretação e avaliação dos Padrões

Os resultados alcançados pelos algoritmos de mineração devem ser avaliados por analistas para que sejam julgados como úteis ou não. Os padrões que forem considerados úteis devem ser interpretados de forma que se tornem compreensíveis para os usuários finais do sistema.

O processo KDD é considerado iterativo, pois pode ser necessária sua execução por várias vezes até que, por meio da repetição de qualquer um dos passos anteriores, se obtenha resultados satisfatórios [6].

### Utilização do conhecimento

Nessa etapa, o conhecimento extraído é consolidado sendo incorporado a um sistema, utilizado diretamente pelo usuário final ou, simplesmente, documentado e relatado às pessoas interessadas. Os resultados dos passos anteriores devem ainda ser analisados para que possíveis conflitos entre o conhecimento existente e o conhecimento adquirido sejam solucionados [5].

## Mineração de dados

Um dos principais passos do processo KDD é a Mineração de Dados , que consiste na aplicação de algoritmos com a finalidade de extrair padrões de comportamento em uma base de dados [5].

Essa é uma área de pesquisa multidisciplinar, que inclui inteligência artificial, aprendizagem de máquina, reconhecimento de padrões, estatística e banco de dados. Um dos grandes desafios da Mineração de Dados é criar algoritmos que sejam capazes de lidar com escalabilidade e alta dimensionalidade dos dados, ou seja, trabalhar com grandes quantidades de dados.

Segundo TAN et al [29], o processo de mineração de Dados é dividido em duas categorias de tarefas principais: primeiro a predição, onde a idéia é prever o comportamento futuro de algumas variáveis da base de dados e, por ultimo, a descrição, onde a idéia é identificar padrões que representem a distribuição dos itens de tal forma que seja possível interpretar os padrões. Abaixo são apresentadas as principais tarefas da mineração de dados, de acordo o proposto em [5]:

* Classificação - uma tarefa preditiva, que consiste em determinar uma função que mapeie cada item de uma base dados a uma das classes previamente definidas. Um exemplo de classificação é identificação de objetos de interesse em grandes bases de dados.
* Regressão - é uma tarefa preditiva, cujo objetivo é estimar o valor de uma variável com base nos valores de outras variáveis. Alguns dos exemplos de regressão são: prever o PIB de um país; estimar a probabilidade de um paciente sobreviver, dado os resultados de um conjunto de exames; e prever séries temporais.
* Detecção de desvios/anomalias - é uma tarefa preditiva, cujo objetivo é detectar itens que possuam características significativamente diferentes do comportamento normal do restante dos dados, como por exemplo, detecção de fraudes em cartões de crédito.
* Sumarização - é uma tarefa descritiva, que consiste em definir um conjunto mínimo de características que seja capaz de identificar um subconjunto de objetos. As técnicas de sumarização são comumente aplicadas para análise exploratória de dados e geração de relatórios automatizados.
* Análise de Séries Temporais - modela características sequenciais, como dados que possuem dependências no tempo. O objetivo é modelar os estados do processo de geração da sequência, extraindo e relatando os desvios e tendências no tempo.

## Classificação

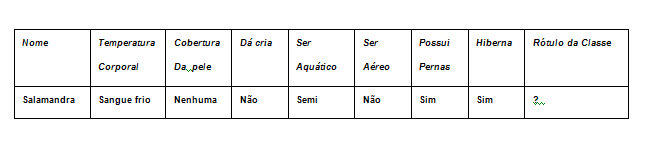
A previsão é uma das tarefas mais importantes de Mineração de Dados. O problema de classificação consiste em prever a classe à qual uma nova instância pertence. Um classificador é construído a partir de instâncias passadas (chamadas instâncias de treinamento) [5]

A classificação pode ser feita localizando as regras que particionem os dados indicados em grupos disjuntos [27]. Por exemplo, poderia ser utilizado um modelo de classificação usado na previsão de um rótulo de classe de registros desconhecidos. Tomemos como entrada para o modelo de previsão os dados contidos na Tabela 1.

Suponha que tenhamos recebido as seguintes características de uma criatura chamada Salamandra, apresentadas na Figura 4. Então, com a utilização de um modelo de classificação criado a partir dos dados contidos na Tabela 1, poderia ser determinada a qual classe a criatura pertence.

1. Conjunto de dados dos vertebrados, Fonte: TAN [29]

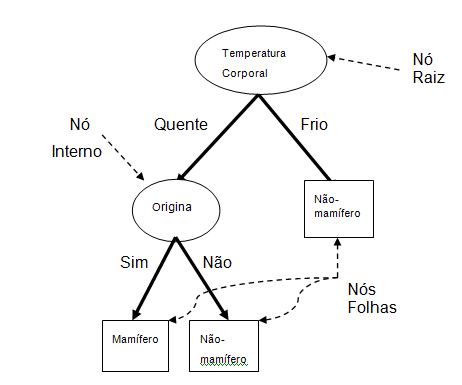
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***Nome*** | ***Temperatura***  ***Corporal*** | ***Cobertura***  ***Da pele*** | ***Dá cria*** | ***Ser***  ***Aquático*** | ***Ser***  ***Aéreo*** | ***Possui***  ***Pernas*** | ***Hiberna*** | ***Rótulo da Classe*** |
| **Humano** | **Sangue quente** | **Cabelo** | **Sim** | **Não** | **Não** | **Sim** | **Não** | **Mamífero** |
| **Píton** | **Sangue frio** | **Escamas** | **Não** | **Não** | **Não** | **Não** | **Sim** | **Réptil** |
| **Salmão** | **Sangue frio** | **Escamas** | **Não** | **Sim** | **Não** | **Não** | **Não** | **Peixe** |
| **Baleia** | **Sangue quente** | **Cabelo** | **Sim** | **Sim** | **Não** | **Não** | **Não** | **Mamífero** |
| **Sapo** | **Sangue frio** | **Nenhuma** | **Não** | **Sim** | **Não** | **Sim** | **Sim** | **Anfíbio** |
| **Dragão de Komodo** | **Sangue frio** | **Escamas** | **Não** | **Não** | **Não** | **Sim** | **Não** | **Réptil** |
| **Morcego** | **Sangue quente** | **Cabelo** | **Sim** | **Não** | **Sim** | **Sim** | **Sim** | **Mamífero** |
| **Pomba** | **Sangue quente** | **Penas** | **Não** | **Não** | **Sim** | **Sim** | **Não** | **Ave** |
| **Gato** | **Sangue quente** | **Pêlo** | **Sim** | **Não** | **Não** | **Sim** | **Não** | **Mamífero** |
| **Leopardo** | **Sangue frio** | **Pêlo** | **Sim** | **Sim** | **Não** | **Sim** | **Não** | **Mamífero** |
| **Tubarão** | **Sangue frio** | **Escamas** | **Não** | **Sim** | **Não** | **Não** | **Não** | **Peixe** |
| **Tartaruga** | **Sangue frio** | **Escamas** | **Não** | **Semi** | **Não** | **Sim** | **Não** | **Réptil** |
| **Pingüim** | **Sangue quente** | **Penas** | **Não** | **Semi** | **Não** | **Sim** | **Não** | **Ave** |
| **Porco-espinho** | **Sangue quente** | **Espinhos** | **Sim** | **Não** | **Não** | **Sim** | **Sim** | **Mamífero** |
| **Enguia** | **Sangue frio** | **Escamas** | **Não** | **Sim** | **Não** | **Não** | **Não** | **Peixe** |



1. Uso do modelo de classificação, Fonte: TAN[29]

### Classificação com Árvores de Decisão

A classificação com árvores de decisão é uma técnica amplamente utilizada em Mineração de Dados. Os classificadores de árvores de decisão constroem uma árvore, sendo que cada folha possui uma classe associada e cada nó interno possui um predicado (ou, geralmente, uma função) associado a ele [5]. A Figura 5 mostra um exemplo de árvore de decisão.



1. Árvore de decisão para problema de classificação de mamíferos, Fonte: TAN [29]

Para classificar uma nova instância, é preciso testar os valores dos atributos nos nós da árvore, começando na raiz e terminando em uma folha; a folha representa a classe prevista para a instância. Ao longo desse percurso, a avaliação do predicado (ou função) do nó é necessária para descobrir para qual filho prosseguir. Por exemplo, o nó raiz mostrado na Figura 5 usa o atributo temperatura corporal para separar vertebrados de sangue quente dos de sangue frio. Já que todos os vertebrados de sangue frio são não mamíferos o nó folha com rótulo Não Mamífero é criado como filho à direita do nó raiz. Caso o vertebrado possua sangue quente, um atributo subseqüente, Origina, é usado para distinguir mamíferos de outras criaturas de sangue quente, que são na sua maioria pássaros. Classificar um novo registro, entretanto, é apenas a etapa final de um classificador de árvore de decisão. O maior desafio é, a partir de um conjunto de instâncias de treino, montar uma árvore de classificação que possua um poder preditivo satisfatório.

A disposição dos atributos na árvore deve seguir algum critério. Esse critério deve maximizar a capacidade de predição da árvore. No exemplo da Figura 5, o atributo escolhido para estar no nó raiz da árvore é “Temperatura Corporal”; os valores possíveis desse atributos são: quente e frio. Se o valor para esse atributo é “quente” gera-se uma sub-árvore, cujo atributo que ocupa o novo nó, no exemplo é “Origina”, é escolhido pelo mesmo critério. Se o valor para o atributo “Temperatura Corporal” é “frio” é, então, gerado o nó folha. Assim, o atributo “Temperatura Corporal” ocupa a raiz da árvore por ser o mais representativo dos exemplos de treino; o segundo atributo mais representativo é “Origina”, para cada sub-árvore gerada a partir da raiz, e assim sucessivamente, até que se justifique a criação de um nó folha.

Intuitivamente, no processo de escolha dos atributos de particionamento, o algoritmo começa com o conjunto de todas as instâncias de treinamento, que é "impuro" no sentido de que contém instâncias de muitas classes e acaba com as folhas que são "puras" no sentido de que, em cada folha, todas as instâncias de treinamento pertencem a apenas uma classe.

### Particionamento da Árvore de Decisão

O critério de escolha dos melhores atributos de particionamento, na maioria dos algoritmos de classificação com árvores de decisão, utiliza os conceitos de Entropia e Ganho de Informação.

A entropia é uma medida que caracteriza a impureza de uma coleção arbitrária de instâncias. Mais precisamente, dada uma coleção S, a entropia de S é obtida da seguinte equação [8]:

onde n é a quantidade de classes presentes em S, e pi é a probabilidade de uma instância em S pertencer à classe i.

Para ilustrar o conceito de entropia, serão exibidos três exemplos de calculo de Entropia que podem ser encontrados em TAN [29]. Para o primeiro exemplo, suponha uma coleção S de 6 instâncias que possua 0 instâncias classificadas como "Sim" e 6 instâncias classificadas como "Não". Então, a entropia referente ao conjunto S = [0,5] é:

(S) = - - = 0

Para o segundo exemplo, suponha uma coleção S de 6 instâncias que possua 1 instância classificada como "Sim" e 5 instâncias classificadas como "Não". Então, a entropia referente ao conjunto S = [1,5] é:

(S) = - - = 0,650

Para o ultimo exemplo, suponha uma coleção S de 6 instâncias que possua 3 instâncias classificadas como "Sim" e 3 instâncias classificadas como "Não". Então, a entropia referente ao conjunto S = [3,3] é:

(S) = - - = 1

Então, conforme os exemplos mostrados é possível observar que o valor máximo para Entropia é atingido, no terceiro exemplo, quando existe uma distribuição uniforme das classes, e o valor mínimo é alcançado, no primeiro exemplo, quando todos os registros pertencem à mesma classe.

Vimos que utilizando o cálculo da entropia é possível calcular o grau de impureza de um atributo, mas é preciso determinar, também, se os atributos para serem avaliados durante os testes são realmente os melhores. Para isto é necessário comparar o grau de impureza do nó pai em relação aos nós filhos. O ganho é um critério que pode ser usado para determinar a qualidade de uma divisão [29].

O ganho de informação pode ser definido como a medida da efetividade de um atributo para classificar os dados de treinamento [8]. O ganho da informação, de um atributo A em relação a um conjunto de instâncias S é dado pela seguin te equação [23]:

Onde é a entropia de um determinado nó, é o numero total de instâncias do nó pai, é o número de valores dos atributos e é o número de instâncias do nó filho.

### Algoritmo J48

O algoritmo J48 é uma implementação do C4.5 *release* 8, por sua vez o algoritmo C4.5 é baseado no algoritmo ID3 (Iterative Dichotomiser 3), para compreensão do algoritmo J48 uma breve explicação sobre os algoritmos ID3, C4.5, C4.5 *release* 7 e C4.5 *release* 8 será realizada .

O algoritmo ID3 (Iterative Dichotomiser 3) foi desenvolvido por J. Ross Quinlan [20] na Universidade de Sidney e publicado em 1975. O ID3 constrói a árvore de decisão a partir da raiz, selecionando o melhor atributo classificador dentre todos os atributos do conjunto de dados. O melhor atributo classificador é selecionado com base numa avaliação estatística de todos os atributos. Após a escolha, os dados são separados de acordo com as classes do atributo escolhido, gerando uma subdivisão dos dados para cada descendente na árvore.

O algoritmo é aplicado recursivamente a cada nó descendente, até que algum critério de parada seja atingido. Isto gera uma árvore de decisão aceitável, na qual o algoritmo nunca retrocede para reconsiderar escolhas feitas anteriormente.

O ID3 foi criado para receber como entrada conjuntos de dados com atributos discretos (nominais), ou seja, atributos cujos valores são pré-definidos.

Em 1993, Quinlan [22] publicou o desenvolvimento de uma família de algoritmos de classificação com árvores para conjuntos contínuos na linguagem C, denominada C4.5. A principal diferença entre o C4.5 e ID3 é que o C4.5 permite o tratamento de dados cujo domínio é contínuo. Uma grande parcela das bases de dados mais interessantes é constituída por atributos contínuos (ou seja, os dados referentes pertencem a uma faixa infinita de valores).

Os critérios adotados para a escolha do melhor atributo classificador no C4.5 são similares aos do ID3, ou seja, utilizam os conceitos de entropia e ganho de informação apresentados na subseção 2.3.2. A principal característica do C4.5 *Release* 7 foi a inclusão de um importante critério para seleção desse limiar: a relação de ganho. Esse critério adota o cálculo da informação de corte para auxiliar a escolha do melhor limiar [23]:

onde é a quantidade de subconjuntos do atributo e representa a quantidade de exemplos de cada subconjunto de . Para atributos contínuos o valor de é sempre 2, pois Quinlan [23] propõe uma divisão binária para os intervalos de valores contínuos.

A relação de ganho é dada pela seguinte equação [16]:

Então, o critério da relação de ganho seleciona, entre os testes cujo ganho de informação é maior ou igual à média deste valor, aquele com melhor relação de ganho. Esse balanceamento no ganho de informação é crucial para a obtenção de árvores mais consistentes com a amostra (S).

A diferença básica entre as versões Release 7 e Release 8 do C4.5 é a utilização do princípio MDL (Minimum Description Length) [24]. A seguir, uma breve descrição, Segundo Quilan e Rivest [21], desse princípio:

Admita duas entidades: um transmissor e um receptor. Cada um possui uma lista ordenada de instâncias de uma base de dados. Suponha que o transmissor conhece as classes de suas instâncias e desenvolve uma teoria para classificá-las. Obviamente, a teoria de classificação pode ser imperfeita, ou seja, nem todos os casos serão corretamente classificados. Quanto melhor a teoria, maior o custo de desenvolvê-la e menor o número de classificações incorretas.

Quanto pior a teoria, menor o custo de desenvolvê-la e maior o número de classificações incorretas. O objetivo é encontrar um meio termo entre esses dois casos. Esta é a descrição geral do MDL [17].

Segundo [23], uma adaptação desse princípio no Release 8 considera que o custo para a escolha do melhor atributo classificador está em função do número de limiares distintos de cada atributo. Na prática, ele sugere que o ganho de informação de cada limiar para cada atributo seja decrementado pelo logaritmo na base 2 da razão entre o número de limiares distintos e o número de amostras:



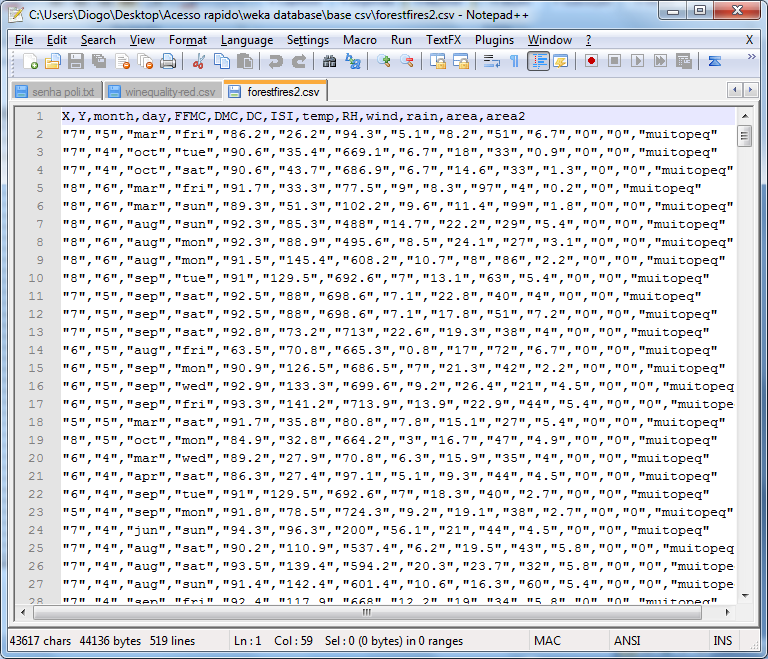
# 3 Ferramentas

Neste capítulo, são apresentadas as principais características da ferramenta desenvolvida, neste trabalho, para coleta dos dados, e da Ferramenta WEKA. Na seção 3.1 são apresentadas as principais funcionalidades da ferramenta para coleta dos dados. Na seção 3.2 são descritos os dados coletados pela ferramenta. Na seção 3.3 são apresentadas as principais funcionalidades da ferramenta WEKA.



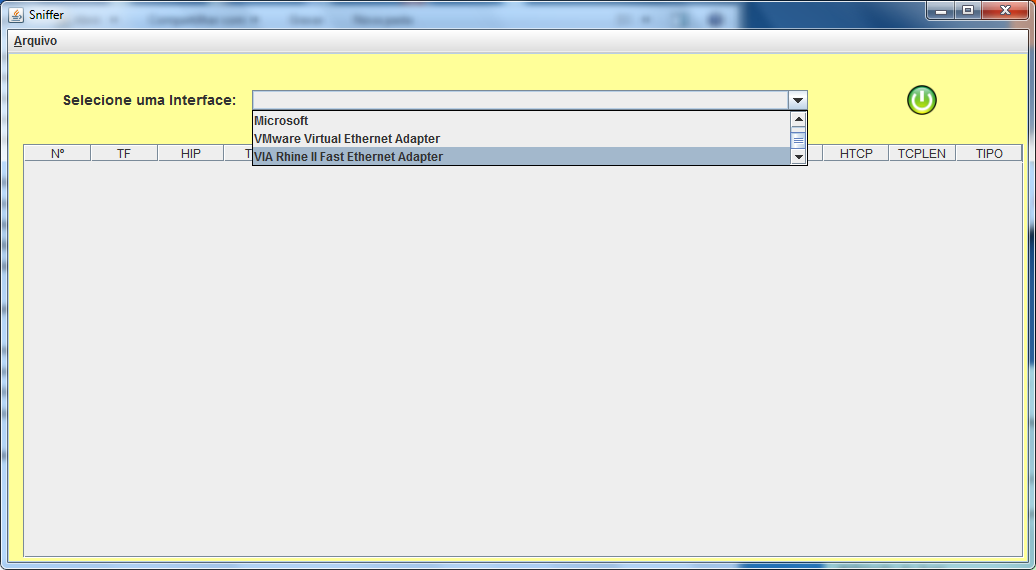
## Coleta dos dados

Uso dessa ferramenta permite que grandes quantidades de dados, obtidos através da captura de pacotes da rede, sejam armazenados em bases de dados no formato CSV (*Comma Separated Values*). Arquivos no formato CSV são bastante simples. Na Figura 6 são exibidos dados no formato CSV. Na primeira linha do arquivo, contida na Figura 6 temos os nomes de todos atributos presentes na base de dados. Perceba que esses atributos estão separados por vírgulas, e que os valores dos atributos estão distribuídos da segunda linha em diante e, também, estão separados por vírgulas. Estes tipos de bases de dados no formato CSV serão utilizadas pela ferramenta do Weka como dados de entrada para a realização do treinamento da Árvore de decisão J48.



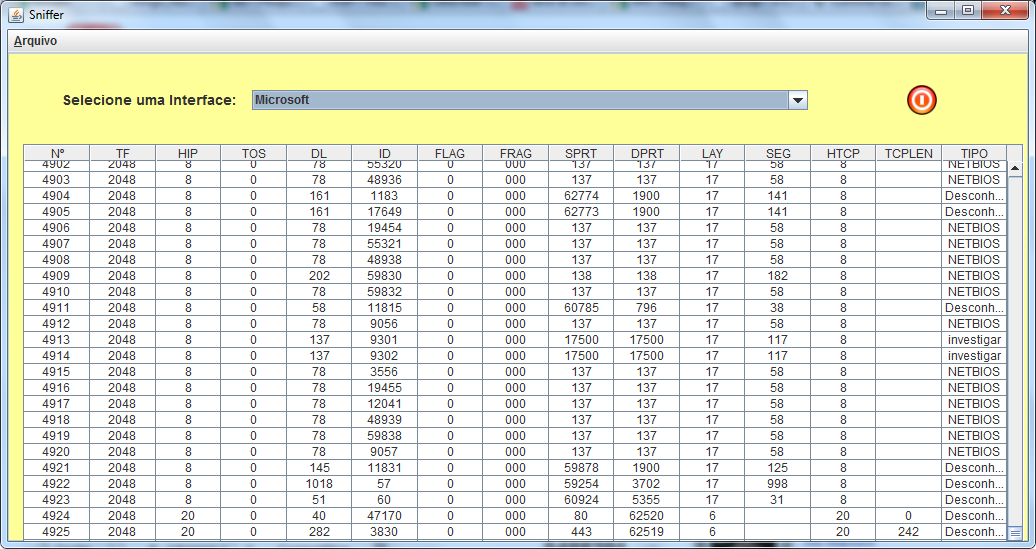
1. Exemplo de arquivo no formato CSV.

Para a realização da captura dos pacotes para posterior criação das bases de dados é necessário a seleção da interface de rede desejada, assim como nas ferramentas  *open source* TCPDUMP[[4]](#footnote-5)[11] e Wireshark. Na figura 7 é apresentada a seleção da interface desejada, da ferramenta desenvolvida neste trabalho de conclusão de curso, para o inicio das capturas dos pacotes e posterior criação da base de dados.



1. Seleção das Interfaces de rede.

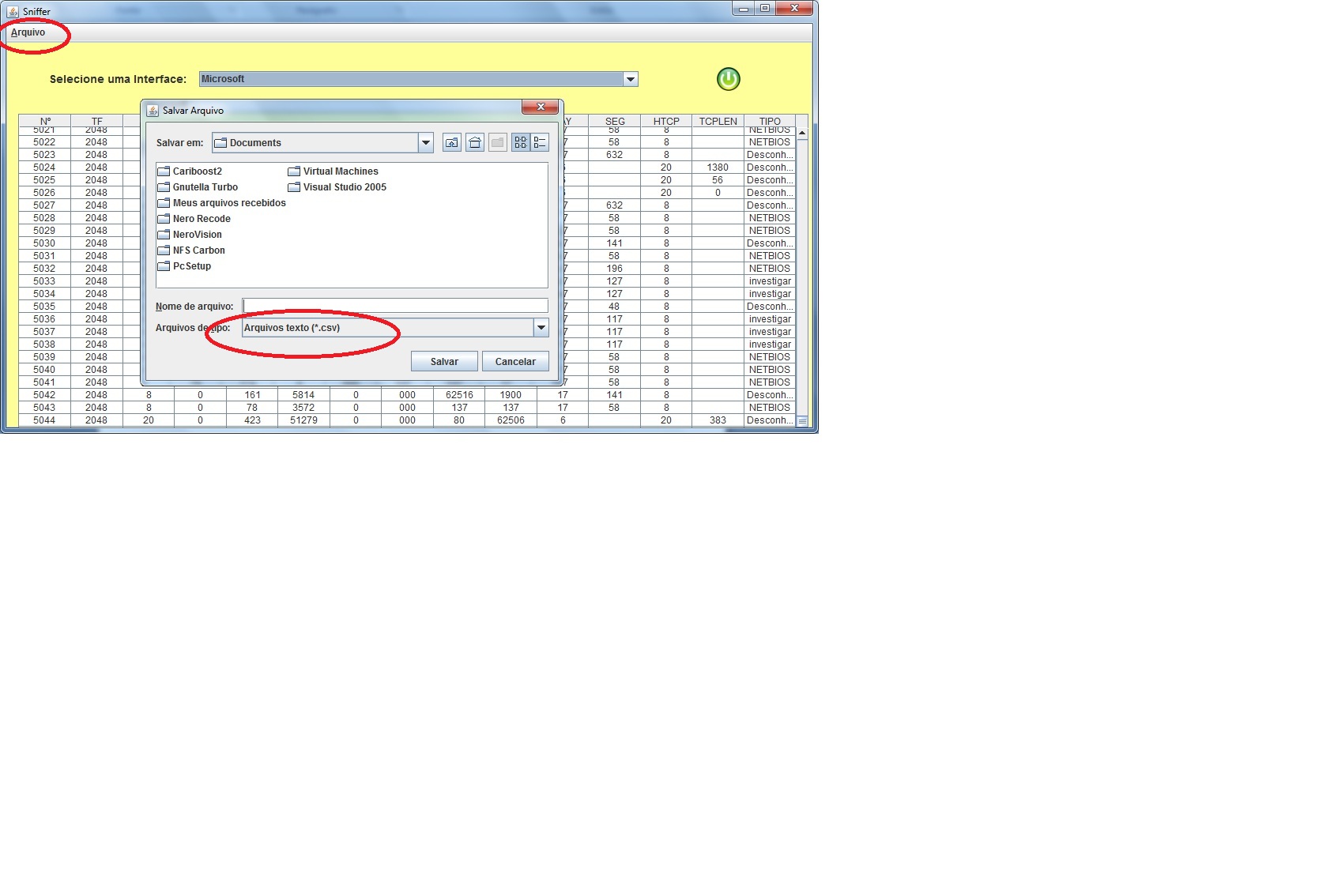
Depois que a interface de rede é selecionada é preciso acionar o botão verde para iniciar a captura dos pacotes, nessa etapa todos os pacotes que estão trafegando pela interface de rede selecionada estão sendo exibidas na ferramenta, este processo de captura dos pacotes poder ser observado conforme a figura 8.



1. Captura dos pacotes

É possível identificar na figura 8 que a cor do botão agora é vermelha sinalizando que todo o tráfego de rede que está passando pela interface em questão está sendo exibido pela ferramenta, os campos dos pacotes estão apresentados no formato de tabelas para facilitar a compreensão do usuário e também na criação das bases de dados.

Após a captura dos pacotes a etapa seguinte, mostrada na figura 9, é a criação da base de dados para a realização do processo em questão. É necessário selecionar no menu Arquivo da aplicação e, então, selecionar a opção “Salvar como”. Por ultimo, clicar no botão salvar. Perceba que no campo Arquivos de tipo já está pré-selecionado o formato CSV.



1. Criação da Base de dados.

## Descrição dos dados

Para cada pacote capturado pela ferramenta são extraídas informações consideradas importantes para realização deste trabalho, essas informações são referentes aos campos dos cabeçalhos da pilha dos protocolos TCP/IP, e estão contidas como valores dos atributos dentro das bases de dados, geradas pela ferramenta de coleta de dados. Esses campos estão detalhados em KUROSE [16]. A seguir, uma descrição dos atributos utilizados como dados de entrada para a realização da classificação:

* **TF:** Este atributo é referente ao ***Ethernet Frame Type*** da camada de enlace.
* **HIP:** Este atributo é referente ao **comprimento do cabeçalho**, camada de rede, o HIP exibe o comprimento, em Bytes, do cabeçalho do datagrama IP.
* **TOS:** Este atributo é referente ao campo ***Type of Service*** do IPV4 e Classe do Ipv6 .
* **DL:** Este atributo é referente ao campo **comprimento do datagrama** que especifica o comprimento total (cabeçalho mais dados) do datagrama em Bytes.
* **ID:** Este atributo é referente ao campo **Identificador,** pois quando um datagrama é fragmentado existe a necessdidade de um numero de identificação do datagrama original no momento da reconstrução do datagrama.
* **FLAG:** Este atributo é referente ao campo ***flags*** que informa se o datagrama está fragmentado e se ainda existem fragmentos do datagrama pra chegar.
* **FRAG:** Este atributo é referente ao campo deslocamento , que é utilizado para especificar a localização exata do fragmento no datagrama IP original [11].
* **LAY:** Este atributo é referente ao campo protocolo da camada superior que informa qual o protocolo da camada de transporte especifico que os dados contidos no datagrama deverá ser entregue.
* **SPRT:** Este atributo é referente ao campo ***porta fonte*** , na camada de transporte é possível identificar qual a porta fonte da aplicação.
* **DPRT:** Este atributo é referente ao campo  ***porta destino*** , na camada de transporte é possível identificar qual a porta destino da aplicação.
* **SEG:** Este atributo informa o tamanho, em Bytes, da carga útil quando o protocolo da camada de transporte é UDP.
* **TCPLEN:** Este atributo informa o tamanho, em Bytes, da carga útil quando o protocolo da camada de transporte é UDP.
* **HTCP:** Este atributo é referente ao campo **comprimento do cabeçalho TCP**, e informa comprimento total, em Bytes, do cabeçalho TCP.
* **TIPO:** Este atributo contém o rótulo da classe do pacote que está sendo analisado.

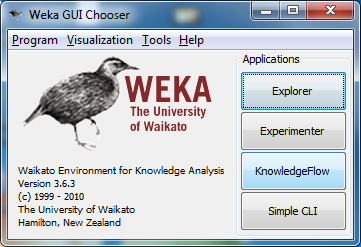
## Ferramenta WEKA

Para realização das tarefas de Mineração de Dados neste trabalho foi utilizada uma ferramenta de código aberto chamada WEKA (Waikato Environment for Knowledge Analysis ). Essa ferramenta foi desenvolvida na Universidade de Waikato na Nova Zelândia e possui implementações de vários algoritmos de Mineração de Dados [9].

A tela inicial do minerador Weka com três botões conforme figura 10. O primeiro botão

*Simple CLI* executa os algoritmos do Weka através de linha de comando. O segundo

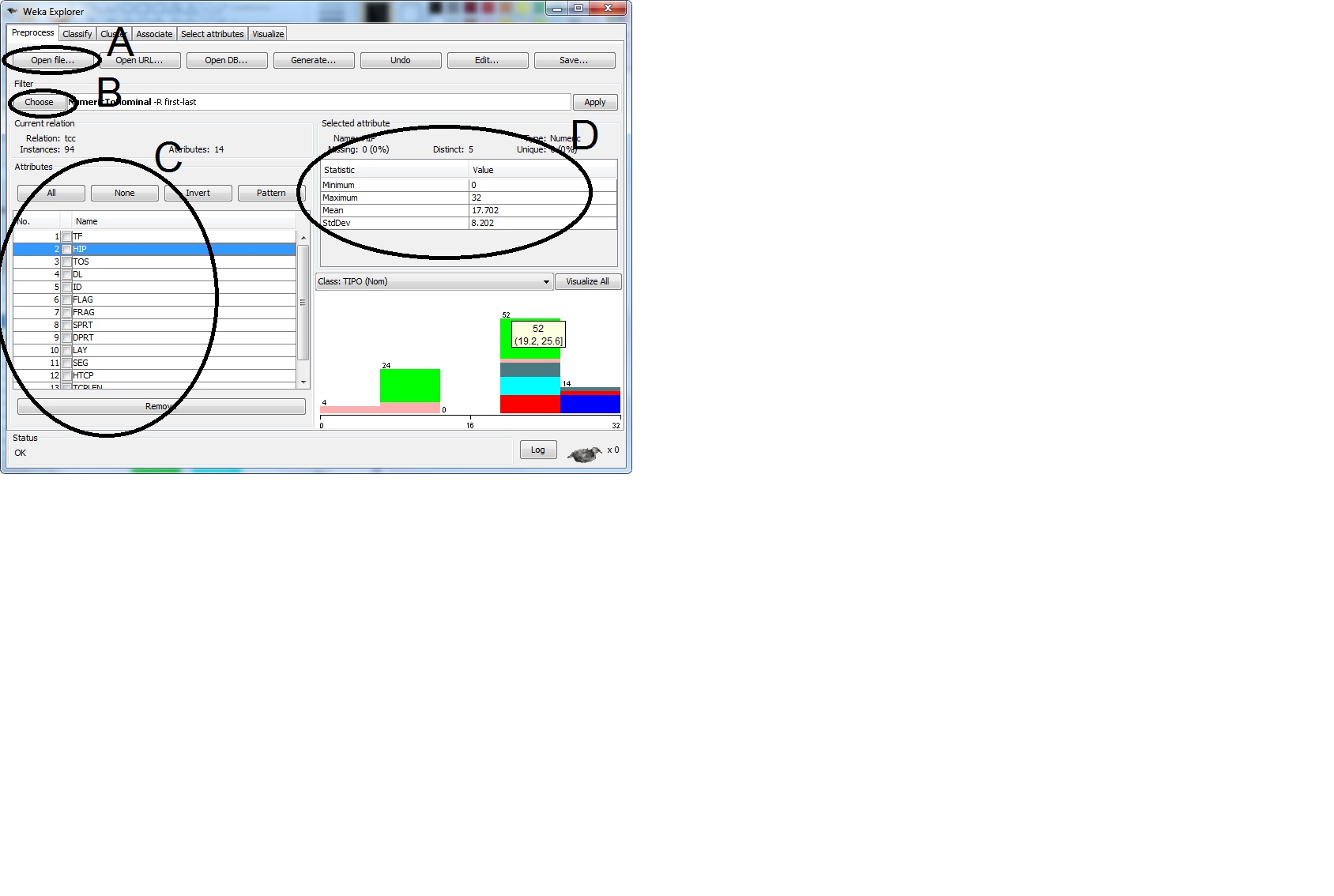
botão *Explorer* executa o módulo gráfico para execução dos algoritmos. O terceiro botão *Experimenter* executa o módulo para manipular base de dados.



1. Tela Inicial do WEKA.

A interface gráfica do WEKA é amigável, porém é importante destacar os principais elementos. A Figura 11 apresenta a tela de pré-processamento da ferramenta destacando os principais elementos:

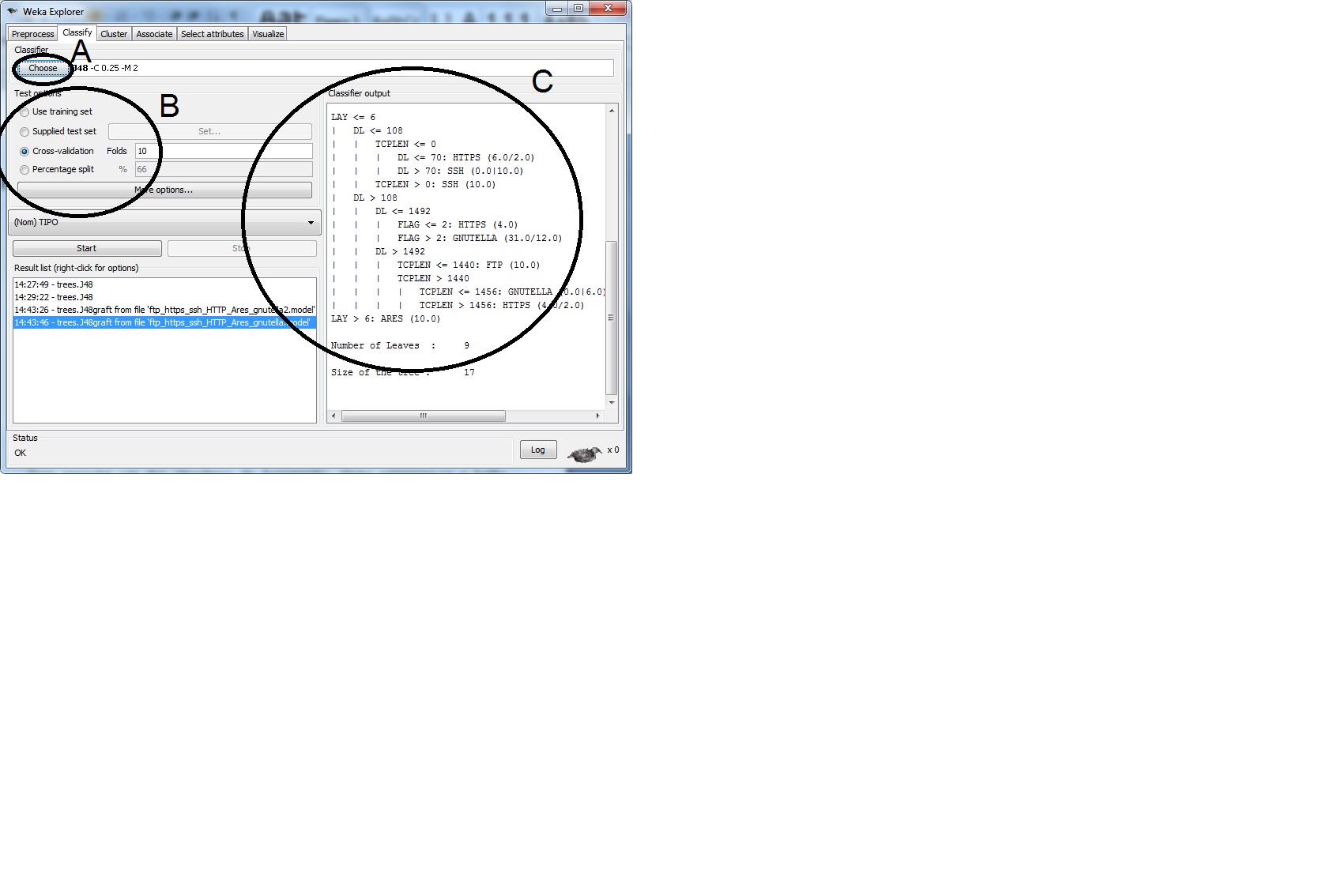
* (a) Esse botão permite a seleção de bases de dados no formato CSV (*Comma Separated Values*).
* (b) Nessa área podem ser selecioanados algoritmos para pré-processamento dos dados (Converter atributos numéricos em nominais, por exemplo.)
* (c) Nessa área são apresentados os atributos da base de dados.
* Apresenta informações quantitativas e estatísticas a respeito do atributo selecionado na área  *Attributes*.



1. Tela de pré-processamento do WEKA.

Neste trabalho será utilizada a tarefa de classificação para isto, é necessário selecionar a aba *Classify.* Na figura 12 é apresentada a tela de classificação , bem como as principais funcionalidades e elementos:

* + (a) O botão *choose* seleciona qual algoritmo será utilizado para classificação, por padrão o *ZeroR* é instanciado.
  + (b) Em *Test Options* define algumas opções como conjunto de treinamento (*use training set*) e validação cruzada (*cross-validation*).
  + (c) Em Classifier Output é possível visualizar os resultado obtidos com o treinamento da árvore de decisão.



1. Tela de classificação do WEKA.

# Resultados

Neste capítulo é apresentado como os experimentos foram realizados na tarefa de classificação de tráfego de rede, sem a inspeção de ***payload*** e análise de **portas conhecidas**, também é mostrado qual a metodologia utilizada para realização dessa tarefa. Na seção 4.1 é descrita a metodologia utilizada na tarefa de classificação, e na seção 4.2 são apresentados os experimentos realizados e os resultados.

## Metodologia

Este trabalho tem como objetivo apresentar um modo de classificar um fluxo de rede gerado por uma aplicação ou protocolo sem análise de *payload,* observando a composição dos campos nos cabeçalhos dos pacotes, buscando padrões que representam regras para identificar a aplicação ou protocolo. As aplicações e protocolos analisados neste artigo foram HTTP *(HiperText Transfer Protocol), HTTPS (HiperText Transfer Protocol Secure), FTP (File Transfer Protocol), SSH (Secure Shell), ARES, GNUTELLA, BITTORRENT.*

Para que fosse possível utilizar o algoritmo de árvore de decisão J48 [23] para classificar os protocolos e aplicações utilizando os campos dos pacotes IP, foi preciso criar uma base de dados para ser usada pelo algoritmo. A coleta dos dados foi dividida em duas etapas.

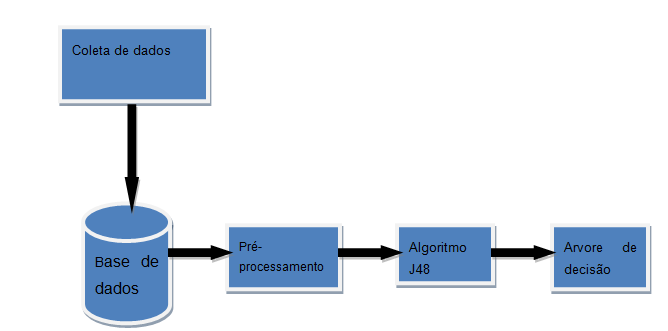
A primeira etapa foi a captura dos pacotes referentes à aplicação ou ao protocolo como *HTTP, FTP, SSH,* que possuem característicasconhecidas*,* isolando e avaliando a composição dos cabeçalhos desses, aplicando a ferramenta, desenvolvida neste trabalho, para analisar e referenciar o conjunto dos dados capturados. Em seguida, com base nas técnicas dos trabalhos estabelecidos para captura e análise dos pacotes, foi feito a captura isolada de fluxos do *BITTORRENT* e aplicações P2P passando a estudar o comportamento e a composição dos seus cabeçalhos na busca de extração de conhecimento.

.A segunda etapa consiste em criar a base de dados com base na captura dos pacotes referentes a aplicação ou protocolo em questão a partir dos fluxos gerados na etapa anterior. Para a realização dessa etapa foi desenvolvida a ferramenta, apresentada no capítulo 3, para manipular os dados contidos em cada fluxo gerado pela etapa anterior, criando uma base de dados, conforme mostrado na seção 3.2 , correspondente aos campos da pilha de protocolos da camada de enlace, rede e transporte de cada pacote pertencente ao fluxo de rede para aplicação ou protocolo em análise. Porém, como foi mostrado no capítulo 2, durante o processo de descoberta do conhecimento é preciso ter domínio da aplicação em análise, e muitas vezes antes de seguir para a etapa de mineração dos dados, é preciso realizar um pré-processamento dos dados. Neste trabalho, houve a necessidade da retirada de alguns atributos, reduzindo assim a dimensionalidade da base de dados. Os atributos removidos da base de dados foram SPRT, DPRT, ID, FRAG, TF e TOS.

Os dois primeiros atributos foram removidos, porque não é interesse deste trabalho levar em consideração as portas das aplicações e protocolos pelos motivos já citados anteriormente, no caso dos demais atributos sua remoção se justifica pelo fato de que eles não são significantes para a tarefa de classificação.

Após a geração desta base de dados e o pré-processamento realizado, o passo seguinte é utilizá-la no Weka [8,32] que representa uma ferramenta para mineração de dados bastante eficiente de amplo aporte a vários domínios de aplicação, como agrupamento, classificação, extração de regras, dentre outros. Para a tarefa de classificação foi utilizado o algoritmo J48 que pertence à biblioteca de algoritmos do Weka, onde demonstrou uma grande eficiência.

Na figura 13 está descrito graficamente o processo para classificação das aplicações e protocolos avaliados neste trabalho.



1. Arquitetura da metodologia

## Experimentos e Resultados

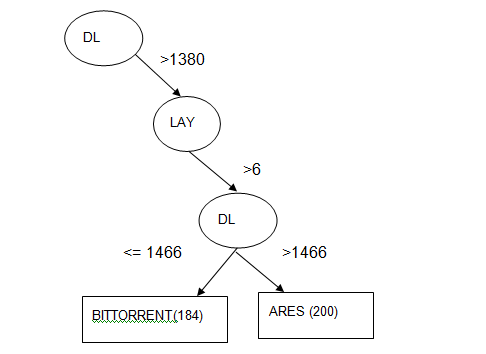
O primeiro experimento foi realizado utilizando uma base de dados com 1524 entradas para o treinamento, o algoritmo utilizado foi o J48 sem poda realizando a validação cruzada. A validação cruzada é um método de avaliação do desempenho de um classificador, onde o conjunto de dados é particionado em subconjuntos de treinamento e de teste.

No Anexo A é apresentado a árvore de decisão gerada para o primeiro experimento.

No Anexo A é possível identificar o nó raiz DL onde a condição de teste para entrar nos próximos nós da árvore está em avaliar se o DL é maior*,* menor ou igual a 1380 *Bytes* .

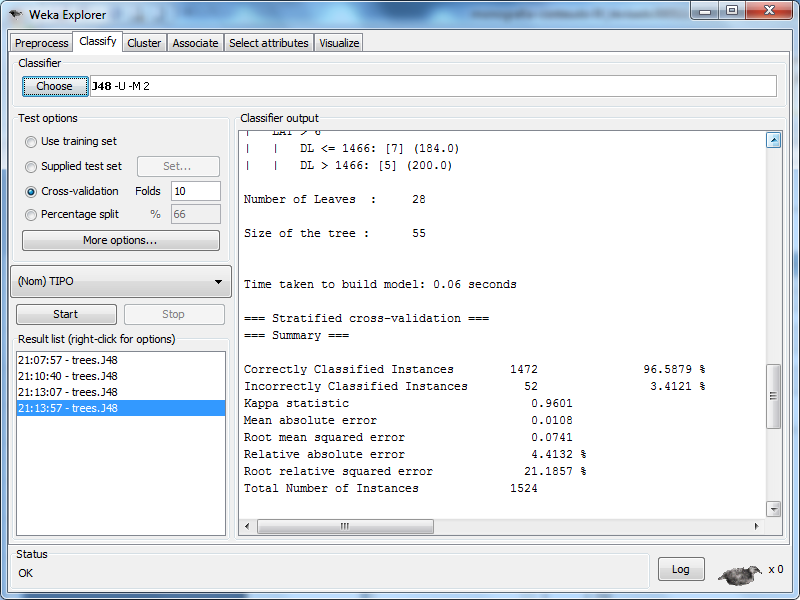
Para este primeiro experimento é possível identificar, nos ramos da árvore de decisão gerada, quais avaliações são necessárias para a identificação das aplicações ARES e BITTORRENT como mostrado na Figura 14.

É possível notar na Figura 14 que o primeiro teste realizado na árvore é verificar se o nó DL é maior que 1380 Bytes, se é maior caminha-se na árvore para o nó LAY. Se nó LAY for maior que 6, isto significa que o protocolo da camada de transporte é UDP, então caminha-se na árvore até o nó DL novamente. Então, se o nó DL é maior que 1466 Bytes a classe dessa aplicação é rotulada como ARES, se DL for menor ou igual a 1466 Bytes a classe dessa aplicação é rotulada como BITTORRENT.



1. Ramificação da árvore de decisão para ARES e BITTORRENT no primeiro experimento.

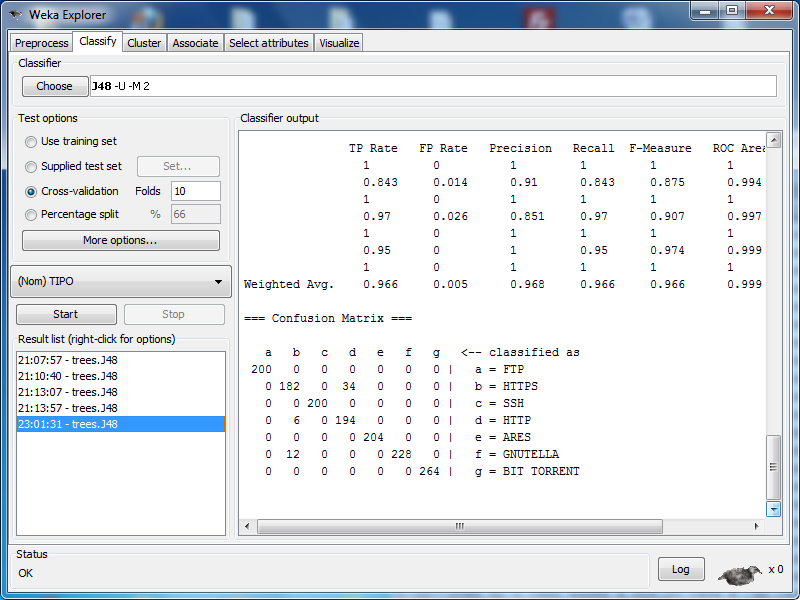
Na Figura 15 é mostrada a precisão da árvore de decisão gerada após o treinamento, sem poda e com a validação cruzada, note que a taxa de acerto da árvore é de 96.5879%. Esta taxa de acerto, 96.5879%, é um valor bastante significativo para classificação em tráfego de rede, como pode ser observado em Soysal [28].



1. Precisão da árvore gerada, para o treinamento sem poda.

Na Figura 16 é apresentada a matriz de confusão para a árvore de decisão gerada pelo treinamento, sem poda e com validação cruzada. O objetivo da matriz de confusão é mostrar o número de previsões corretas em relação às esperadas para cada regra [3]. É possível notar na Figura 15 que para a classe FTP todas a 200 aparições são classificadas, corretamente, como FTP. Já para o caso da classe HTTP das 200 aparições, 194 foram classificadas, corretamente, como HTTP e 6 foram classificadas como HTTPS.

Então, com a matriz de confusão é possível identificar quais são as classes que estão sendo classificadas de forma errada.



1. Matriz de confusão da árvore gerada, para o treinamento sem poda.

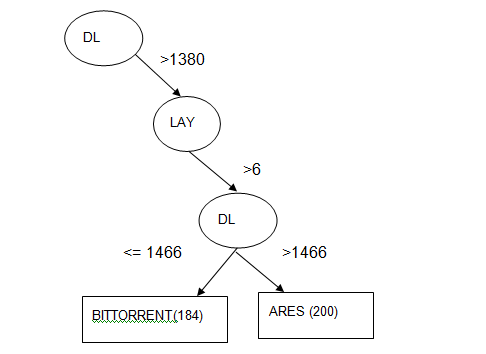
O segundo experimento foi realizado utilizando uma base de dados com 1524 entradas para o treinamento, o algoritmo utilizado foi o J48 com poda realizando a validação cruzada.

No Anexo B é apresentado a árvore de decisão gerada para o primeiro experimento.

No Anexo B é possível identificar o nó raiz DL onde a condição de teste para entrar nos próximos nós da árvore está em avaliar se o DL é maior*,* menor ou igual a 1380 *Bytes* .

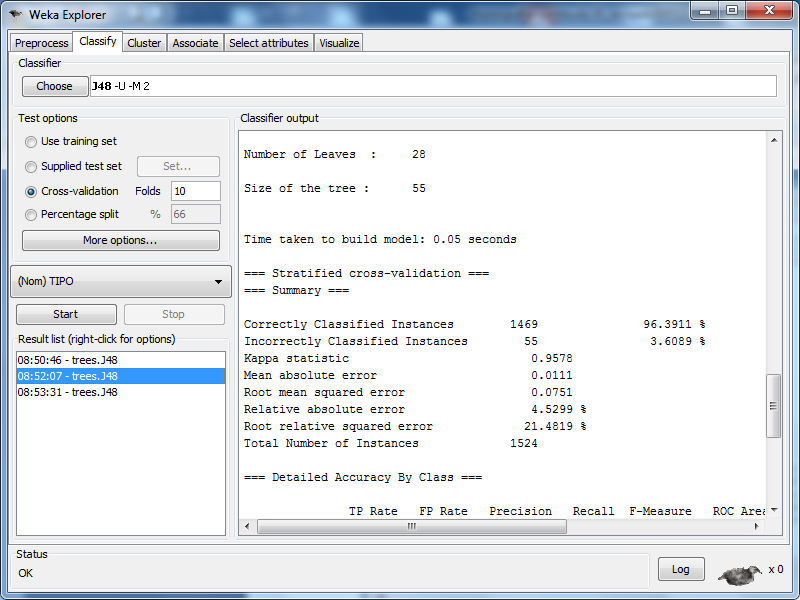
Para este segundo experimento é possível identificar, nos ramos da árvore de decisão gerada, quais avaliações são necessárias para a identificação das aplicações ARES e BITTORRENT como mostrado na Figura 17.

É possível notar na Figura 17 que o primeiro teste realizado na árvore é a verificar se o nó DL é maior que 1380 Bytes, se é maior caminha-se na árvore para o nó LAY. Se nó LAY for maior que 6, isto significa que o protocolo da camada de transporte é UDP, então caminha-se novamente na árvore até o nó DL. Então, se o nó DL é maior que 1466 Bytes a classe dessa aplicação é rotulada como ARES, se DL for menor ou igual a 1466 Bytes a classe dessa aplicação é rotulada como BITTORRENT.



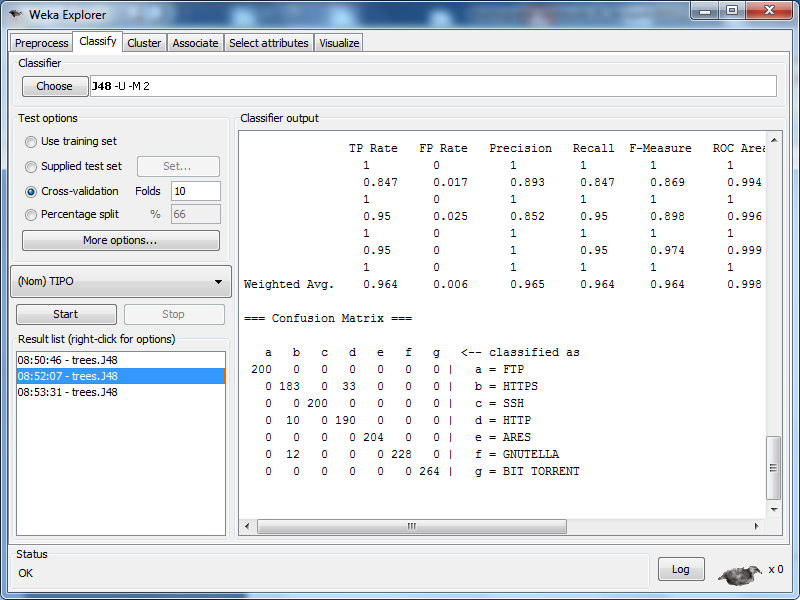
1. Ramificação da árvore de decisão para ARES e BITTORRENT para o segundo experimento.

É possível notar na figura 18 que a taxa de acerto é um pouco menor que a taxa de acerto quando o treinamento foi realizado sem poda e com validação cruzada. A única diferença entre o treinamento realizado no primeiro experimento e o segundo é o fato de que o primeiro foi feito sem poda, o segundo experimento com poda.



1. Precisão da árvore gerada, para o treinamento com poda.

Na Figura 19 é apresentada a matriz de confusão para a árvore de decisão gerada pelo treinamento, com poda e com validação cruzada. Também é possível notar que a variação na matriz de confusão é mínima em relação a primeira matriz de confusão , ou seja, os resultados obtidos tanto para o primeiro experimento quanto para o segundo são praticamente os mesmos. Os resultados também foram bastante animadores no que diz respeito à classificação já que os resultados para os dois experimentos foram praticamente iguais, isto mostra que os atributos escolhidos realmente ajudaram a classificar os tráfegos de rede e que o algoritmo de árvore de decisão também se mostrou bastante eficaz, frente ao problema de classificação para os protocolos e aplicações escolhidos para serem avaliados no presente trabalho.



1. Matriz de confusão da árvore gerada, para o treinamento com poda.

# Conclusão

Neste trabalho de conclusão de curso, foi apresentado o problema de classificação para tráfego de rede gerado por aplicações e protocolos. Na tentativa de solucionar esse problema foi proposta uma metodologia para a extração de conhecimento em tráfego. Sendo assim, foi necessário, para a implantação da metodologia, o desenvolvimento de uma ferramenta de coleta de dados em rede de computadores que criasse base de dados, juntamente com a aplicação da técnica de árvore de decisão com o algoritmo J48 por meio da ferramenta de mineração de dados WEKA.

Com os resultados alcançados neste trabalho foi possível comprovar que esta metodologia pode ser implementada em uma ferramenta que venha auxiliar os gerentes e administradores de redes no monitoramento de redes, pois essa ferramenta teria a capacidade de classificar o tráfego de rede utilizando as características das aplicações e protocolos sem utilizar as portas conhecidas e nem realizar inspeção de *payload*. Portanto, esses conceitos podem ser empregados no desenvolvimento de firewalls inteligentes que possam aprender de forma autônoma o tráfego de rede gerado por protocolos e aplicações.

Bibliografia

[1] AMO, S. D. Técnicas de Mineração de Dados. [S.1.], 2004.

[2] Bar-yanai, R.; Langberg, M.; Peleg, D.; and Roditty, L. “Realtime Classification for Encrypted Traffic,” 2010 LNCS, pp. 373-385.

[3] BARANAUSKAS, J. A; MONARD, M. C. Reviwing some machine learning

concepts and methods. Techical Report 102, ICMC-USP.

[4] Crotti M., Gringoli F., Pelosato P., Salgarelli L., A statistical approach to IP- level classification of network traffic, in: IEEE International Conference on Communications, ICC06, 2006, pp. 170–176.

[5] ELMASRI, R.; NAVATHE, S. B. Sistemas de Banco de Dados. [S.1.]: Pearson, 2005.

[6] FAYYAD, U.; PIATETSKY-SHAPIRO, G.; SMYTH, P. From data mining to knowledge discovery in databases. AI Magazine, v. 17, p. 37\_54, 1996.

[7] FAYYAD, U.; PIATETSKY-SHAPIRO, G.; SMYTH, P. The kdd process for extracting useful knowledge from volumes of data. Communications of the ACM, v. 39, p. 27\_34, 1996.

[8] GUARDA, A. Inteligência Artificial em Controle de Automação. [S.l.]: Editora Edgard Blücher, 2000.

[9] HALL, M. et al. The weka data mining software: An update. SIGKDD Explorations, v. 11,2009.

[10] HAN, J.; KAMBER, M. Data mining: concepts and techniques. [S.l.]: Morgan Kaufmann,2006.

[11] JACOBSEN, V.; LERES, C.; MCCANNE, S. ***tcpdump****.* LBNL, University of

California, June 1997.

[12] Karagiannis T., Broido A., Faloutsos M., Claffy K.C., Transport layer identification of P2P traffic, in: 4th ACM SIGCOMM Conference on Internet Measurement, 2004.

[13] Karagiannis T., Broido A., Brownlee N., Claffy K.C., Faloutsos M., Is P2P dying or just hiding, in: IEEE Global Telecommunications Conference,GLOBECOM 04, 2004.

[14] Karagiannis T., Broido A., Brownlee N., Claffy K.C., Faloutsos M., Is P2P dying or just hiding, in: IEEE Global Telecommunications Conference,GLOBECOM 04, 2004.

[15] Karagiannis T., Papagiannaki K., Faloutsos M., BLINC: Multilevel traffic classification in the dark, SIGCOMM Computer Communication Review 35 (4) (2005) 229–240.

[16] KUROSE, James F. Redes de computadores e a internet: uma abordagem top-down/James F. Kurose, Keith W. Ross ; Tradução Arlete Simille Marques ; revisão técnica Wagner Luiz Zucchi. – 3, ed. – São Paulo : Pearson Wesley, 2006.

[17] Mantia, G.L.; Rossi, D.; Finamore, A.; Mellia, M.; and Meo, M. “Stochastic Packet Inspection for TCP Traffic,” May 2010 IEEE International conference on Communications.

[18] Moore D., Keys K., Koga R., Lagache E., Claffy K.C., The CoralReef software suite as a tool for system and network administrators, in: 15th USENIX Conference on System Administration, LISA01, pp. 133–144.

[19] Moore A.W., Papagiannaki K., Toward the accurate identification of network applications, in: Lecture Notes in Computer Science, Springer, Berlin, 2005, pp. 41–54.

[20] QUINLAN, J. R. Machine Learning, n. 1, p. 81–106, 1986.

[21] Quinlan, J. R., & Rivest, R. L. (1989). Inferring decision trees using the minimum description length principle. Information and Computation, 80, 227-248.

[22] QUINLAN, J. R. C4.5: Programs For Machine Learning. [S.l.]: Morgan Kalfmann, 1993.

[23] QUINLAN, J. R. Improved use of continuous in c4.5. Journal of Artificial Inteligence Research, n. 6, p. 77–90, 1996.

[24] RISSANEN, J. A universal prior for integers and estimation by minimum description length. Annals of statistic, n. 11, p. 416–431, 1983.

[25] Sen S., Wang J., Analyzing peer-to-peer traffic across large networks, IEEE/ACM Transactions on Networking 12 (2004) 219–232.

[26] Sen S., Spatscheck C., Wang D., Accurate, scalable innetwork identification of P2P traffic using application signature, in: 13th International Conference on World Wide Web, 2004.

[27] SILBERSCHATZ, A.; SUDARSHAN, H. F. K. Sistema de Banco de Dados. [S.1.]: Editora Campus, 2006.

[28] Soysal M, Schmidt EG. Machine learning algorithms for accurate flow-based network traffic classification: Evaluation and comparison. *Performance Evaluation*. 2010;67(6):451-467.

[29] TAN, Pang-Ning; STEINBACH, Michael; KUMAR, Vipin; Introdução ao DATAMINING Mineração de Dados Rio de Janeiro: Editora Ciência Moderna Ltda., 2009.

[30] The Internet Assigned Numbers Authority, IANA, [Online], Available: http://www.iana.org

[31] Wang K., Stolfo S.J., Anomalous payload-based network intrusion detection, in: Lecture Notes in Computer Science, Springer, Berlin, 2004

[32] Witten, I. H.; and Frank, E. “ Data mining: Practical Machine Learning Tools and Techniques,” June 2005 Morgan Kaufmann Series in Data Management Systems Morgan Kaufmann, second edition.

[33] Ye, M.; Xu, K.; Wu, J.; and Po, H. “AutoSig-Automatically Generating Signatures for Applications,” Oct. 2009 Ninth IEEE International Conference on Computer and Information Technology, pp. 104-109.

# Anexo A Árvore de decisão sem poda.

J48 unpruned tree

------------------

DL <= 1380

| FLAG <= 2

| | HIP <= 8: GNUTELLA (108.0)

| | HIP > 8

| | | DL <= 137

| | | | DL <= 58: HTTPS (18.0)

| | | | DL > 58: GNUTELLA (6.0)

| | | DL > 137: HTTPS (72.0)

| FLAG > 2

| | TCPLEN <= 68

| | | TCPLEN <= 0

| | | | HIP <= 20: HTTPS (18.0)

| | | | HIP > 20: HTTP (76.0/36.0)

| | | TCPLEN > 0: SSH (200.0)

| | TCPLEN > 68

| | | DL <= 933

| | | | DL <= 470

| | | | | DL <= 228

| | | | | | DL <= 173: GNUTELLA (6.0)

| | | | | | DL > 173: HTTP (20.0)

| | | | | DL > 228: GNUTELLA (24.0)

| | | | DL > 470

| | | | | DL <= 511: HTTP (60.0)

| | | | | DL > 511

| | | | | | DL <= 862

| | | | | | | DL <= 580

| | | | | | | | DL <= 577: GNUTELLA (12.0)

| | | | | | | | DL > 577: HTTP (20.0)

| | | | | | | DL > 580: GNUTELLA (18.0)

| | | | | | DL > 862: HTTP (20.0)

| | | DL > 933

| | | | DL <= 1089: HTTPS (18.0)

| | | | DL > 1089

| | | | | DL <= 1204

| | | | | | DL <= 1093: HTTP (20.0)

| | | | | | DL > 1093

| | | | | | | DL <= 1107: GNUTELLA (6.0)

| | | | | | | DL > 1107: HTTP (20.0)

| | | | | DL > 1204: GNUTELLA (12.0)

DL > 1380

| LAY <= 6

| | HIP <= 20

| | | DL <= 1480

| | | | DL <= 1449: HTTPS (18.0)

| | | | DL > 1449: ARES (4.0)

| | | DL > 1480

| | | | DL <= 1492: GNUTELLA (36.0)

| | | | DL > 1492: HTTPS (48.0/12.0)

| | HIP > 20

| | | DL <= 1492: BIT TORRENT (80.0)

| | | DL > 1492: FTP (200.0)

| LAY > 6

| | DL <= 1466: BIT TORRENT (184.0)

| | DL > 1466: ARES (200.0)

# Anexo B Árvore de decisão com poda.

J48 pruned tree

------------------

DL <= 1380

| FLAG <= 2

| | HIP <= 8: GNUTELLA (108.0)

| | HIP > 8

| | | DL <= 137

| | | | DL <= 58: HTTPS (18.0)

| | | | DL > 58: GNUTELLA (6.0)

| | | DL > 137: HTTPS (72.0)

| FLAG > 2

| | TCPLEN <= 68

| | | TCPLEN <= 0

| | | | HIP <= 20: HTTPS (18.0)

| | | | HIP > 20: HTTP (76.0/36.0)

| | | TCPLEN > 0: SSH (200.0)

| | TCPLEN > 68

| | | DL <= 933

| | | | DL <= 470

| | | | | DL <= 228

| | | | | | DL <= 173: GNUTELLA (6.0)

| | | | | | DL > 173: HTTP (20.0)

| | | | | DL > 228: GNUTELLA (24.0)

| | | | DL > 470

| | | | | DL <= 511: HTTP (60.0)

| | | | | DL > 511

| | | | | | DL <= 862

| | | | | | | DL <= 580

| | | | | | | | DL <= 577: GNUTELLA (12.0)

| | | | | | | | DL > 577: HTTP (20.0)

| | | | | | | DL > 580: GNUTELLA (18.0)

| | | | | | DL > 862: HTTP (20.0)

| | | DL > 933

| | | | DL <= 1089: HTTPS (18.0)

| | | | DL > 1089

| | | | | DL <= 1204

| | | | | | DL <= 1093: HTTP (20.0)

| | | | | | DL > 1093

| | | | | | | DL <= 1107: GNUTELLA (6.0)

| | | | | | | DL > 1107: HTTP (20.0)

| | | | | DL > 1204: GNUTELLA (12.0)

DL > 1380

| LAY <= 6

| | HIP <= 20

| | | DL <= 1480

| | | | DL <= 1449: HTTPS (18.0)

| | | | DL > 1449: ARES (4.0)

| | | DL > 1480

| | | | DL <= 1492: GNUTELLA (36.0)

| | | | DL > 1492: HTTPS (48.0/12.0)

| | HIP > 20

| | | DL <= 1492: BIT TORRENT (80.0)

| | | DL > 1492: FTP (200.0)

| LAY > 6

| | DL <= 1466: BIT TORRENT (184.0)

| | DL > 1466: ARES (200.0)

# Anexo C Código Fonte

package sniffer.gui;

import java.awt.event.ActionListener;

import java.awt.event.ActionEvent;

import java.awt.Color;

import javax.swing.SwingUtilities;

import javax.swing.ButtonGroup;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JMenuItem;

import javax.swing.JMenuBar;

import javax.swing.JMenu;

import javax.swing.JFrame;

import javax.swing.JRadioButton;

import java.awt.Dimension;

import java.awt.Rectangle;

import java.io.IOException;

import javax.swing.JButton;

import jpcap.JpcapCaptor;

import jpcap.NetworkInterface;

import jpcap.packet.IPPacket;

import jpcap.packet.Packet;

import javax.swing.ImageIcon;

import javax.swing.table.DefaultTableModel;

import javax.swing.JTable;

import javax.swing.JComboBox;

import javax.swing.JTextField;

import javax.swing.JTextPane;

import javax.swing.BorderFactory;

import javax.swing.border.BevelBorder;

import javax.swing.border.TitledBorder;

import java.awt.Font;

import javax.swing.JScrollPane;

import java.awt.ComponentOrientation;

public class SnifferTampa {

private JFrame jFrame = null; // @jve:decl-index=0:visual-constraint="-6,45"

private JPanel jContentPane = null;

private JMenuBar jJMenuBar = null;

private JMenu fileMenu = null;

private JMenuItem exitMenuItem = null;

private JButton btLigar = null;

ButtonGroup groupRB = new ButtonGroup();

private JRadioButton[] rbInterface = null;

private NetworkInterface[] list;

private JpcapCaptor captor;

private JButton btDesligar = null;

private Thread thread;

private Color corAplicacao = new Color(255, 255, 153);

private String [][] dados;

private String [] colunas;

private JComboBox interfaces = null;

private JTextPane jTextPane = null;

private JScrollPane jScrollPane = null;

private JTable tabela = null;

/\*\*

\* This method initializes jFrame

\*

\* @return javax.swing.JFrame

\*/

private JFrame getJFrame() {

if (jFrame == null) {

jFrame = new JFrame();

jFrame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

jFrame.setSize(new Dimension(794, 414));

jFrame.setJMenuBar(getJJMenuBar());

jFrame.setLocationRelativeTo(null);

jFrame.setResizable(false);

jFrame.setTitle("Sniffer");

jFrame.setContentPane(getJContentPane());

jFrame.setVisible(true);

}

return jFrame;

}

/\*\*

\* This method initializes jContentPane

\*

\* @return javax.swing.JPanel

\*/

private JPanel getJContentPane() {

if (jContentPane == null) {

jContentPane = new JPanel();

jContentPane.setLayout(null);

jContentPane.setBorder(BorderFactory.createTitledBorder(null, "", TitledBorder.DEFAULT\_JUSTIFICATION, TitledBorder.DEFAULT\_POSITION, new Font("Dialog", Font.BOLD, 12), new Color(51, 51, 51)));

jContentPane.setBackground(corAplicacao);

jContentPane.add(getBtLigar(), null);

jContentPane.add(getBtDesligar(), null);

}

list = JpcapCaptor.getDeviceList();

rbInterface = new JRadioButton[list.length];

for (int i = 0; i < list.length; i++) {

rbInterface[i] = new JRadioButton();

rbInterface[i].setBounds(new Rectangle(15, 25 + 20\*i, 500, 20));

rbInterface[i].setBackground(corAplicacao);

rbInterface[i].setText(list[i].description);

groupRB.add(rbInterface[i]);

}

jContentPane.add(getInterfaces(), null);

jContentPane.add(getJTextPane(), null);

jContentPane.add(getJScrollPane(), null);

return jContentPane;

}

/\*\*

\* This method initializes jJMenuBar

\*

\* @return javax.swing.JMenuBar

\*/

private JMenuBar getJJMenuBar() {

if (jJMenuBar == null) {

jJMenuBar = new JMenuBar();

jJMenuBar.add(getFileMenu());

}

return jJMenuBar;

}

/\*\*

\* This method initializes jMenu

\*

\* @return javax.swing.JMenu

\*/

private JMenu getFileMenu() {

if (fileMenu == null) {

fileMenu = new JMenu();

fileMenu.setText("Arquivo");

fileMenu.add(getExitMenuItem());

}

return fileMenu;

}

/\*\*

\* This method initializes jMenuItem

\*

\* @return javax.swing.JMenuItem

\*/

private JMenuItem getExitMenuItem() {

if (exitMenuItem == null) {

exitMenuItem = new JMenuItem();

exitMenuItem.setText("Sair");

exitMenuItem.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

System.exit(0);

}

});

}

return exitMenuItem;

}

/\*\*

\* This method initializes btLigar

\*

\* @return javax.swing.JButton

\*/

private JButton getBtLigar() {

if (btLigar == null) {

btLigar = new JButton();

btLigar.setBounds(new Rectangle(698, 39, 34, 39));

btLigar.setBorderPainted(false);

btLigar.setContentAreaFilled(false);

btLigar.setIcon(new ImageIcon(getClass().getResource("/sniffer/gui/on.png")));

btLigar.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent e) {

if(thread == null){

thread = new Thread() {

public void run() {

try {

int i;

boolean isSelecionado = false;

for(i = 0; i < list.length; i++){

if(interfaces.getSelectedIndex()==i){

isSelecionado = true;

break;

}

}

if (isSelecionado){

captor = JpcapCaptor.openDevice(list[i], 65535, true, 20);

btLigar.setVisible(false);

interfaces.disable();

btDesligar.setVisible(true);

while (true) {

Packet pacote = captor.getPacket();

if (pacote instanceof IPPacket){

IPPacket ip = (IPPacket) pacote;

adicionaLinha(ip.src\_ip.getHostAddress().toString(),ip.dst\_ip.getHostAddress().toString(), ""+ip.length);

}

}

}else{

thread = null;

JOptionPane.showMessageDialog(null, "Selecione uma Interface.");

}

} catch (IOException ex) {

ex.printStackTrace();

}

}

};

thread.start();

}

}

});

}

return btLigar;

}

/\*\*

\* This method initializes btDesligar

\*

\* @return javax.swing.JButton

\*/

private JButton getBtDesligar() {

if (btDesligar == null) {

btDesligar = new JButton();

btDesligar.setFocusPainted(false);

btDesligar.setBorderPainted(false);

btDesligar.setContentAreaFilled(false);

btDesligar.setIcon(new ImageIcon(getClass().getResource("/sniffer/gui/off.png")));

btDesligar.setBounds(new Rectangle(697, 39, 37, 39));

btDesligar.setVisible(false);

btDesligar.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent e) {

if(thread != null){

thread.stop();

thread = null;

btDesligar.setVisible(false);

interfaces.enable();

btLigar.setVisible(true);

}

}

});

}

return btDesligar;

}

/\*\*

\* This method initializes interfaces

\*

\* @return javax.swing.JComboBox

\*/

private JComboBox getInterfaces() {

if (interfaces == null) {

String nomesInterfaces []= new String [list.length];

for (int i = 0; i < list.length; i++){

nomesInterfaces[i] = list[i].description.toString();}

interfaces = new JComboBox(nomesInterfaces);

interfaces.setBounds(new Rectangle(73, 47, 580, 23));

interfaces.setMaximumRowCount(3);

interfaces.setVisible(true);

}

return interfaces;

}

/\*\*

\* This method initializes jTextPane

\*

\* @return javax.swing.JTextPane

\*/

private JTextPane getJTextPane() {

if (jTextPane == null) {

jTextPane = new JTextPane();

jTextPane.setBounds(new Rectangle(76, 19, 176, 25));

jTextPane.setBackground(new Color(255, 255, 153));

jTextPane.setEditable(false);

jTextPane.setFont(new Font("Dialog", Font.BOLD, 14));

jTextPane.setText("Selecione uma interface");

}

return jTextPane;

}

/\*\*

\* This method initializes jScrollPane

\*

\* @return javax.swing.JScrollPane

\*/

private JScrollPane getJScrollPane() {

if (jScrollPane == null) {

jScrollPane = new JScrollPane();

jScrollPane.setBounds(new Rectangle(15, 90, 760, 261));

jScrollPane.setComponentOrientation(ComponentOrientation.UNKNOWN);

jScrollPane.setBorder(null);

jScrollPane.setViewportView(getTabela());

}

return jScrollPane;

}

/\*\*

\* This method initializes tabela

\*

\* @return javax.swing.JTable

\*/

private JTable getTabela() {

if (tabela == null) {

DefaultTableModel modelo = new DefaultTableModel(dados,getColunas());

tabela = new JTable(modelo);

tabela.disable();

}

return tabela;

}

/\*\*

\* This method initializes salvarComo

\*

\* @return javax.swing.JMenuItem

\*/

private JMenuItem getSalvarComo() {

if (salvarComo == null) {

salvarComo = new JMenuItem();

salvarComo.setText("Salvar Como...");

salvarComo.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent e) {

fc = new JFileChooser();

fc.setDialogType(JFileChooser.SAVE\_DIALOG);

fc.setDialogTitle("Salvar Arquivo");

fc.removeChoosableFileFilter(fc.getFileFilter());

fc.addChoosableFileFilter(new javax.swing.filechooser.FileFilter() {

public boolean accept(File f) {

String nome = f.getName().toLowerCase();

return nome.endsWith(".txt") || f.isDirectory();

}

public String getDescription() {

return "Arquivos texto (\*.txt)";

}

});

fc.addChoosableFileFilter(new javax.swing.filechooser.FileFilter(){

public boolean accept(File f) {

String nome = f.getName().toLowerCase();

return nome.endsWith(".csv") || f.isDirectory();

}

public String getDescription() {

return "Arquivos texto (\*.csv)";

}

});

fc.showDialog(getJFrame(), null);

try {

diretorio=null;

escreverArquivo(getTabela());

} catch (IOException e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

}

}

});

}

return salvarComo;

}

/\*\*

\* Launches this application

\*/

public static void main(String[] args) {

SwingUtilities.invokeLater(new Runnable() {

public void run() {

SnifferTampa application = new SnifferTampa();

application.getJFrame().setVisible(true);

}

});

}

public void adicionaLinha(String coluna1, String coluna2, String coluna3){

DefaultTableModel modelo=(DefaultTableModel)getTabela().getModel();

modelo.addRow(new String[]{coluna1,coluna2,coluna3});

}

public String[] getColunas(){

if(colunas==null)

colunas = new String []{"Nº","TF", "HIP", "TOS", "DL", "ID", "FLAG",

"FRAG", "SPRT", "DPRT", "LAY", "SEG", "HTCP","TCPLEN", "TIPO"};

return colunas;

}

}

1. http://www.wireshark.org/ [↑](#footnote-ref-2)
2. http://www.oracle.com/technetwork/java/javase/downloads/index.html [↑](#footnote-ref-3)
3. http://netresearch.ics.uci.edu/kfujii/Jpcap/doc/ [↑](#footnote-ref-4)
4. http://www.tcpdump.org/ [↑](#footnote-ref-5)