ẓes á-ti-co á-fi-cas

MINISTÉRIO DA DEFESA
EXÉRCITO BRASILEIRO
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA
CURSO DE MESTRADO EM SISTEMAS E COMPUTAÇÃO

RODRIGO MATHIAS PRAXEDES DA SILVA

ARQUITETURA PARA DETECÇÃO ONLINE DE BOTS P2P

Rio de Janeiro
2013

INSTITUTO MILITAR DE ENGENHARIA

RODRIGO MATHIAS PRAXEDES DA SILVA

ARQUITETURA PARA DETECÇÃO ONLINE DE BOTS P2P

Dissertação de Mestrado apresentada ao Curso de Mestrado em Sistemas e Computação do Instituto Militar de Engenharia, como requisito parcial para obtenção do título de Mestre em Sistemas e Computação.

Orientador: Prof. Ronaldo Moreira Salles - Ph.D.

Rio de Janeiro
2013

INSTITUTO MILITAR DE ENGENHARIA
Praça General Tibúrcio, 80-Praia Vermelha
Rio de Janeiro-RJ CEP 22290-270

004.65 Silva, Rodrigo Mathias Praxedes da
S586m Arquitetura para Detecção Online de Bots P2P/ Rodrigo Mathias Praxedes da Silva; Orientado por Ronaldo Moreira Salles. – Rio de Janeiro: Instituto Militar de Engenharia, 2013.

77 p.: il.

Dissertação (mestrado) – Instituto Militar de Engenharia – Rio de Janeiro, 2013.

1.Engenharia de computação - teses e dissertações. 2. Redes de Computadores 3. Tecnologia e Sistemas de Computação. I. Salles, Ronaldo Moreira II.Título III. Instituto Militar de Engenharia.

CDD 004.65

INSTITUTO MILITAR DE ENGENHARIA

RODRIGO MATHIAS PRAXEDES DA SILVA

ARQUITETURA PARA DETECÇÃO ONLINE DE BOTS P2P

Dissertação de Mestrado apresentada ao Curso de Mestrado em Sistemas e Computação do Instituto Militar de Engenharia, como requisito parcial para obtenção do título de Mestre em Sistemas e Computação.

Orientador: Prof. Ronaldo Moreira Salles - Ph.D.

Aprovada em 21 de janeiro de 2013 pela seguinte Banca Examinadora:

Prof. Ronaldo Moreira Salles - Ph.D. do IME - Presidente

Prof. Claudio Gomes de Mello - DSc, do IME

Prof. Sidney Cunha de Lucena - DSc, da UniRio

Rio de Janeiro
2013

Dedico esta dissertação
à Deus por tudo que ele representa em minha vida!
"O coração do homem pode fazer planos, mas a resposta certa dos lábios vem do Senhor"

Provérbios 16:1

AGRADECIMENTOS

A Deus nosso criador e sustentador, toda honra, louvor e glória por permitir a concretização deste trabalho.

A todas as pessoas que contribuíram para o desenvolvimento desta dissertação de mestrado, em especial ao meu orientador Prof. Ronaldo Moreira Salles, por ter me apresentado o assunto e fornecido toda a base necessária para o seu desenvolvimento.

Aos meus pais, Luiz Praxedes da Silva e Vera Lucia Mathias da Silva que não mediram esforços para que eu pudesse estudar e nortearam sempre a minha vida com exemplos e valores.

A minha avó Felisbella Mathias Carneiro da Silva (*in memoriam*) que cuidou de mim durante a infância com dedicação, apoio e afeto.

A minha namorada Lauana Figueiredo Silva pela paciência nos muitos momentos em que estive concentrado no trabalho.

Por fim, a todos os professores e funcionários do Departamento de Engenharia de Sistemas (SE/8) do Instituto Militar de Engenharia.

Rodrigo Mathias Praxedes da Silva

Cada um que passa em nossa vida, passa sozinho...
Porque cada pessoa é única para nós,
e nenhuma substitui a outra.
Cada um que passa em nossa vida, passa sozinho...
mas não vai só...
Levam um pouco de nós mesmos,
e nos deixam um pouco de si mesmos.
Há os que levam muito,
mas não há os que não levam nada.
Há os que deixam muito,
mas não há os que não deixam nada.
Esta é a mais bela realidade da vida...
A prova tremenda de que cada um é importante,
é que ninguém se aproxima do outro por acaso...

Saint Exupery

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

Bot	-	*Robot*
C&C	-	*Comando e Controle*
DDoS	-	*Distributed Denied of Service*
DHCP	-	*Dynamic Host Configuration Protocol*
DDNS	-	*Dynamic Domain Name Sstem*
DNS	-	*Domain Name System*
DHT	-	*Distributed Hash Table*
DPI	-	*Deep Packet Inspection*
FTP	-	*File Transfer Protocol*
HTTP	-	*Hypertext Transfer Protocol*
ID3	-	*Iterative Dichotomiser 3*
IDS	-	*Intrusion Detection System*
IM	-	*Instant Messages*
IP	-	*Internet Protocol*
IRC	-	*Internet Relay Chat*
ISP	-	*Internet Service Provider*
MP3	-	*MPEG 1/2 - Audio Layer 3*
P2P	-	*Peer-to-peer*
SVM	-	*Suport Vector Machine*
TCP	-	*Transmission Control Protocol*
TTL	-	*Time to Live*
UDP	-	*User Datagram Protocol*

RESUMO

Botnets são os principais veículos para atividades ilícitas na internet, o crescimento de suas atividades tem causado prejuízos de bilhões de dólares à economia mundial e colocado em risco a segurança de nações. Atualmente é observada uma migração das arquiteturas centralizadas (baseadas em protocolos HTTP e IRC) mais simples, para arquiteturas descentralizadas (baseadas em protocolos P2P) que são capazes de prover maior resiliência às suas redes devido à própria natureza do protocolo em que se fundamentam.

Com o objetivo de mitigar esta ameaça, muitas propostas de detecção têm sido elaboradas, porém, a maioria ainda com seu foco nas arquiteturas centralizadas através de métodos que executam análises *offline*. Estas análises buscam por padrões em longos históricos de rede, muitas vezes utilizando algoritmos de clusterização, que podem demorar dias para obter uma resposta satisfatória.

Neste trabalho é proposta uma arquitetura para a detecção de *bots* descentralizadas de forma *online*, ou seja, que não cause uma demora entre a produção dos dados a serem analisados e os resultados obtidos por estas análises. O tráfego de rede gerado pelos nós monitorados são transformados em fluxos de rede e filtrados por módulos de pronta execução que não necessitam de longos históricos, provendo desta forma resultados a tempo de serem utilizados com um maior proveito. A arquitetura proposta é dividida em duas fases: na primeira são buscados todos os nós que possuem aplicações P2P ativas para que posteriormente, na segunda fase, sejam diferenciadas as aplicações P2P legítimas das *bots* P2P.

Ao final deste trabalho é feita uma validação em dois cenários distintos que comprovam a eficiência da arquitetura para a detecção *online* de *bots* P2P.

ABSTRACT

Botnets are the main vehicles for illegal activities on the Internet cauing losses of billions of dollars to the world economy and putting at risk the security of nations. Currently it is observed a migration of centralized architectures (based on HTTP and IRC) simpler for decentralized architectures (based P2P protocols) that are capable of providing greater resilience to their networks due to the nature of the protocol on which they are based.

Aiming to mitigate this threat several botnet detection schemes have been proposed in the literature, but most of them were designed for centralized architectures and applied methods that perform offline analysis. Such approaches look for patterns in long network traces (past history) often using clustering algorithms, which can take days to get a satisfactory result.

This work proposes an online architecture to detect descentralized botnets, it does not cause a delay between the production of data to be analyzed and the results obtained by these analyzes. The network traffic generated by the nodes monitored are transformed into network flows and filtered by execution modules that do not require long records, thereby providing results in time to be usefull. The proposed architecture is divided into two phases. In the first phase, all nodes that have active P2P applications are sought so that later in the second phase, differentiation between legitimate P2P applications and P2P bots is achieved.

At the end of this work, experimental results are obtained for two distinct scenarios to show the efficiency of the proposed architecture.

1 INTRODUÇÃO

Segundo dados da empresa *Symantec* (GLOBO, 2012), no último ano, crimes cibernéticos geraram um prejuízo superior a US$ 110 bilhões no mundo. Dentre esses crimes, destacam-se fraudes bancárias, roubo de senhas, envio de *spams*, entre outros.

Inicialmente concebidos para executar autônoma e automaticamente funções nas mais diversos áreas (economia, computação distribuída,...), *botnets* são atualmente as principais ameaças à segurança na internet. O termo que designa uma coleção de agentes de softwares denominados *bots*, a partir do final da década de 90 começou a designar redes maliciosas voltadas para inúmeras atividades ilícitas. A capacidade de se controlar remotamente grandes quantidades de agentes autônomos, mostrou ser uma poderosa ferramenta para execução de atividades, como ataques de DDoS (*Distributed Denied of Service*), roubo de informações, dentre outras. Neste trabalho será tratado apenas de *bots* e *botnets* maliciosas, ou seja, voltado para a execução de atividades ilícitas.

Este tipo de *malware* dotado de um canal de comando e controle (C&C), por onde comandos são transmitidos, representam um dos principais responsáveis por atividades maliciosas. As redes constituídas por este *malware*, denominadas *botnets*, não possuem fronteiras geográficas para a sua atuação, podendo, desta forma, possuir membros nos mais diversos países. Isto torna a descoberta de seu controlador (*botmaster*) uma tarefa complicada e sem garantias de que, caso possível, ocorra sanções legais sobre ele, pois muitos países ainda não possuem uma legislação em relação a crimes cibernéticos.

Atualmente a principal atividade maliciosa exercida por *botnets* é o envio de *spams*. Muitas vezes, por serem constituídas por milhares de máquinas, mostram-se um veículo ideal para esta atividade. Ao invés de usar poucas máquinas para enviar milhares de e-mails para os mais distintos usuários, utiliza-se milhares de máquinas onde cada uma se torna responsável pelo envio de uma pequena quantidade de e-mails. Isso torna a detecção e inserção dos domínios utilizados para o envio dos e-mails em *blacklists* uma tarefa mais custosa. Segundo a empresa de segurança M86 (NOD, 2012), ao dia são enviados aproximadamente 230 bilhões de *spams*, quase a totalidade oriundo de *botnets*. A Tabela 1.1 mostra algumas *botnets* conhecidas com o seu número estimado de membros e suas capacidades de envio de *spams* por dia (FSECURE, 2009; DANCHEV, 2009; MILLER,

2008; GOODIN, 2010). Em uma pesquisa realizada pelo Instituto Sophos (CLULEY, 2012), o Brasil ocupou a sétima colocação entre os países que mais enviam *spams*, como mostrado na Figura 1.1.

BOTNET	QUANTIDADE DE MEMBROS	NÚMERO DE SPAMS/DIA
Rustock	150.000	30 bilhões
Mega-D	510.000	10 bilhões
Conficker	10.500.000	10 bilhões
Waledac	100.000	1,5 bilhão
Lethic	260.000	2 bilhões
Bobax	185.000	9 bilhões
Kraken	495.000	9 bilhões
Rustock	150.000	30 bilhões
Grum	560.000	40 bilhões
Storm	160.000	3 bilhões

TAB. 1.1: Quantidade de *spams* enviados por dia e número aproximado de membros por *botnet*.

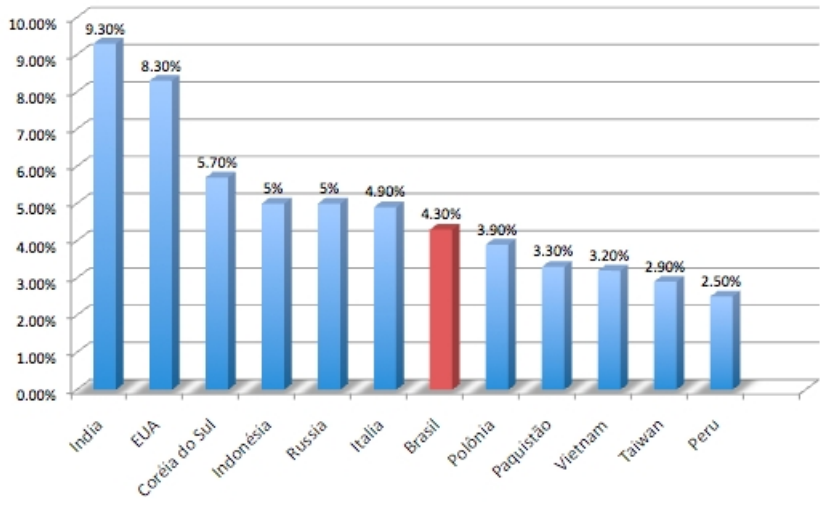

FIG. 1.1: Percentual de *spams* enviados pelos países.

Devido aos números alarmantes, muitos esforços com o objetivo de deter a ocorrência das *botnets* são realizados, diversas técnicas de detecção e mitigação do problema já foram propostas, como veremos no capítulo subsequente. Entretanto, as *botnets* vêm evoluindo com o intuito de se tornarem cada vez mais difíceis de serem detectadas e mais tolerantes às tentativas de desarticulação das suas redes.

O primeiro registro de uma *botnet* ocorreu em 1999, onde o *Trojan Sub7* e o *Worm Pretty Park* acessavam canais IRC (*Internet Relay Chat*) a partir da máquina infectada

com o intuito de receber comandos a serem executados. No ano 2000 a *bot Global Threat Bot* (GTBot) foi o primeiro *malware* classificado propriamente como uma *bot*, este também se utilizava da rede IRC para a recepção de comandos. Em 2002, a *bot Agobot* utilizou o protocolo HTTP em detrimento ao IRC para o canal de C&C, e no ano de 2003 a bot *Sinit* foi considerada a primeira a utilizar arquitetura descentralizada. Neste mesmo ano a bot *AgoBot* foi atualizada de forma a permitir seu funcionamento de modo descentralizado, baseando sua arquitetura nas redes P2P existentes na época. A Tabela 1.2 mostra a evolução das *botnets* ao longo dos quatorze anos de sua existência (RAYWOOD, 2010; SILVA, 2012b; HOLZ, 2008).

Atualmente *botnets* têm migrado de arquiteturas centralizadas (baseadas em protocolos HTTP e IRC), mais simples, para arquiteturas descentralizadas (baseadas em protocolos P2P), que são capazes de prover maior resiliência às suas redes devido à própria natureza do protocolo em que se fundamentam. Entretanto, muitas técnicas que são propostas, ainda nos dias de hoje, visam detectar *botnets* com arquiteturas centralizadas e em sua grande parte não contemplam a dinâmica de nós existente nas redes. Constantemente, nós são inseridos e removidos das redes, ou mesmo têm seus endereços IPs trocados, o que dificulta a descoberta de máquinas infectadas pelas metodologias desenvolvidas, caso estas demandem de grandes períodos de tempos para executar suas análises e retornar uma resposta. Nestes casos, as informações sobre nós infectados podem não chegar a tempo de serem úteis, os nós podem não mais existir nas redes ou mesmo podem estar utilizando outros endereços IPs.

Neste trabalho é proposta uma arquitetura capaz de detectar *bots* descentralizadas (P2P) de forma *online*, ou seja, que não causem uma demora entre a produção dos dados a serem analisados e os resultados obtidos por esta análise.

A arquitetura foi dividida em duas fases distintas: detecção de nós com aplicações P2P ativas (fase 1) e detecção de nós com *bots* P2P ativas (fase 2), visando assim maior otimização do processo, devido a uma redução do universo a ser analisado na segunda etapa. A divisão também é justificada pelo fato de que uma *bot* P2P, essencialmente, é uma aplicação P2P e muitas vezes utiliza a estrutura destas redes para seu funcionamento, como será visto posteriormente.

Este trabalho apresenta as seguintes contribuições:

- uma arquitetura de detecção online de máquinas com *bots* P2P ativas;

- módulo para diferenciação entre aplicações P2P legítimas e *bots* P2P;
- um laboratório para geração de *traces* visando auxiliar na execução de testes de validação da arquitetura.

A dissertação se encontra estruturada da seguinte forma: no Capítulo 2 são apresentados alguns trabalhos anteriores relacionados ao tema. São explicados no Capítulo 3 conceitos básicos relacionados ao protocolo P2P, *botnets* e a árvores de decisão, que são importantes para o melhor entendimento da proposta. No Capítulo 4 é detalhada a arquitetura em suas fases. Uma validação desta arquitetura é realizada no Capítulo 5, onde são mostrados os resultados obtidos. Finalmente, no Capítulo 6 é feita a conclusão desta dissertação bem como algumas propostas para trabalhos futuros.

BOTNET	REGISTRO	CARACTERÍSTICAS	CANAL DE C&C
Sub 7	1999	*Trojan* considerado um precursor das *botnets*	IRC
Pretty Park	1999	*Worm* considerado um precursor das *botnets*	IRC
GTBot	2000	Primeiro *malware* definido como *botnet* é uma atualização do *trojan Sub 7*	IRC
SDBot	2002	Primeiro *bot* a ter seu código fonte comercializado, deu origem a diversas *bots*	IRC
AgoBot	2002	Primeira *bot* a usar o conceito de fases de operação em sua atuação e a utilizar o protocolo HTTP	IRC e HTTP
SpyBot	2003	Evolução da *SDBot* foi a primeira *bot* a utilizar técnicas de *keylogging*	IRC
RBot	2003	Primeira *bot* a executar DDoS e utilizar algorítimos de encriptação e compressão com o objetivo de dificultar sua detecção	IRC
Sinit	2003	Primeira *bot* a utilizar arquitetura descentralizada, baseada em protocolo P2P	P2P
Polybot	2004	Evolução da *AgoBot*, foi a primeira *bot* a utilizar polimorfismo (alterações feitas em seu próprio código a fim de dificultar sua detecção)	IRC
Bagle	2004	Considerada uma das primeiras *bots* para envio de *spams*	HTTP
Bobax	2004	Considerada uma das primeiras *bots* para envio de *spams*	IRC
SpamThru	2006	Primeira *bot* a utilizar protocolo P2P customizado	P2P
Nugache	2006	*Bot* P2P que se conectava a *peers* prédefinidos	P2P
RuStock	2006	*Bot* para envio de *spams*	HTTP
ZeuS	2007	Principal *botnet* voltada para roubo de informações	HTTP
Storm	2007	*Botnet* utilizada para uma variedade crimes cibernéticos, permite seu desmembramento e venda para interessados em executar atividades criminosas	P2P
Peacomm	2007	*Bot* P2P baseada no protocolo *Kademlia* (DHT para redes P2P descentralizadas)	P2P
Mega-D	2008	Foi responsável por mais de 32% dos *spams* enviados no auge de sua infestação	HTTP
Conficker	2008	Considerada uma das mais poderosas *bots*, permite ao atacante controlar remotamente a máquina da vítima	HTTP, P2P
Waledac	2010	Primeira *bot* a utilizar em uma arquitetura descentralizada (P2P) o protocolo HTTP	P2P

TAB. 1.2: *Botnets* mais relevantes.

2 REVISÃO DA LITERATURA

Botnets são consideradas atualmente as principais ameaças à segurança da internet. Com isto, diversos pesquisadores têm direcionado seus estudos para a detecção e mitigação destas redes de *malware*.

Neste capítulo relacionamos alguns trabalhos que foram utilizados como base para a elaboração e execução desta dissertação. Com o intuito de facilitar a leitura, foram divididos em duas áreas: detecção de aplicações P2P e detecção de *botnets* descentralizadas.

O estudo de técnicas para detecção de aplicações P2P é justificado pelo fato de *bots* P2P serem essencialmente aplicações deste gênero e por isso serem suscetíveis às técnicas para detectá-las, guardando algumas peculiaridades, como será visto. Além disso, metodologias elaboradas para a detecção de *bots* descentralizadas são o alicerce para a arquitetura proposta.

2.1 DETECÇÃO DE APLICAÇÕES P2P

Spognardi et al. (SPOGNARDI, 2005) elaboraram um modelo de detecção de nós com aplicações P2P baseado na análise de palavras-chave contidas nos pacotes trafegados nas redes monitoradas. Para isto, um mecanismo semelhante a um IDS (*Intrusion Detection System*) foi utilizado e diversas assinaturas foram registradas para posterior detecção. Tal mecanismo, apesar de funcional, falha ao tentar descobrir novos protocolos, pois, devido a sua natureza, está limitado a encontrar protocolos com assinaturas conhecidas. Outro aspecto limitante na proposta é a facilidade com que aplicações podem evadir-se utilizando mecanismos de criptografia ou compressão de dados em seus pacotes.

Nos trabalhos (SAROIU, 2002; CHOI, 2004; KARAGIANNIS, 2004a; MOORE, 2005) foram estudadas as portas utilizadas pelas diversas aplicações com o objetivo de gerar um mapa que facilite a detecção das aplicações e protocolos P2P. Todavia, tal método se tornou ineficaz com a aleatoriedade na escolha das portas utilizadas pelas principais aplicações existentes na atualidade, visando a não detecção por tais técnicas.

Sen e Wang (SEN, 2004) e Constantinou e Mavrommatis (CONSTANTINOU, 2006) fizeram uma análise de um conjunto de características de redes, como volume de tráfego, tempo de conexão, número de máquinas conectadas por rede, entre outras, com o in-

tuito de encontrar um padrão adotado pelas aplicações P2P. Através desta caracterização foi possível encontrar protocolos P2P conhecidos e descobrir novos, entretanto, o tempo necessário para a execução das análises mostrou ser um fator crítico para ambas metodologias, dificultando a descoberta de máquinas em redes com grande dinâmica entre os nós.

Karagiannis et al. (KARAGIANNIS, 2004b) foram os primeiros a criticar os métodos baseados em assinaturas de *payload* ou que utilizem valores de portas para a identificação de aplicações. Neste trabalho foram elencadas uma série de características de rede inerentes ao protocolo P2P de modo a garantir que se uma aplicação não se adequar a nenhuma destas características, certamente não é uma aplicação P2P. Em testes executados em redes, buscando a validação da metodologia, os autores obtiveram resultados satisfatórios, conseguindo detectar nós com aplicações P2P através da análise de períodos inferiores a 20 minutos de tráfego.

Liu et al. (LIU, 2009a) utilizaram um conjunto de características de redes para a análise do comportamento de diversos nós e determinar quais possuem aplicações P2P ativas com o auxilio da técnica de *Support Vector Machine* (SVM). Dentre estas características podemos apontar o tamanho médio dos pacotes enviados, taxa de conexões bem sucedidas e a razão entre IPs conectados e portas utilizadas. Todas as características são de fácil obtenção e análise. Testes executados em um ambiente controlado obtiveram valores de falsos negativos inferiores a 13%.

Técnicas que exploram o fato da não existência de autoridades certificadoras nas redes P2P também são bastante utilizadas para a sua neutralização, entre estas, pode-se apontar o *Sybil Attack* e o envenenamento de índice.

O *Sybil Attack* (DOUCEUR, 2002) consiste na criação de um grande número de *peers* falsos mas com um identificador válido na rede P2P. Por não serem membros reais da rede, ao receberem requisições de *peers* verdadeiros, não as repassam tornando a rede ineficiente em buscar informações. Em (LEVINE, 2006), são apontadas soluções para este ataque, entretanto, tais soluções causam a perda de desempenho, já que a criação de autoridades certificadoras podem gerar gargalos.

O envenenamento de índice, como citado em (LIANG, 2006), consiste na inserção de falsos arquivos em tabelas, como a DHT (*Distributed Hash Table*), de modo que um *peer*, ao procurar por determinado dado, deve receber arquivos diferentes dos buscados ou comprometidos, minando a confiabilidade da rede em questão.

2.2 DETECÇÃO DE BOTNETS DESCENTRALIZADAS

Wang et al. (WANG, 2009) estudaram os modelos de propagação e comunicação utilizados pelas *bots* P2P e, fundamentados em técnicas como o *Sybil Attack*, utilizado em redes P2P, conseguiram comprometer o canal de C&C da *botnet* P2P. Apesar de ser eficiente, a abordagem não detecta os membros da rede e não garante que a comunicação não será bem sucedida para todos os nós membros.

Coskum et al. (COSKUN, 2010), utilizaram o número de conexões feitas por cada computador para encontrar *peers* membros de uma mesma rede de *bots*. Entretanto, como mencionado pelos próprios autores, redes muito populosas podem não ser detectadas pelo método pois seus *peers* são confundidos com servidores acessados por diversas máquinas clientes. Outra limitação deste método é a necessidade de se conhecer previamente pelo menos um membro da rede. Além disso, o conhecimento de um *bot*, não permite atestar que todos as máquinas das redes identificadas como livres de *bots* não estão infectados por algum outro *bot* de outra rede maliciosa.

Chang e Daniels (CHANG, 2009) propuseram uma técnica baseada nas características do tráfego de rede entre nós, como o valor da porta utilizada. Porém, com o intuito de evadir-se de técnicas como esta, diversas *botnets*, como a *Nugache* que conhecidamente empregava a porta 8 para sua comunicação, alteraram suas implementações para que as portas sejam escolhidas de forma aleatória.

Liu et al. (LIU, 2010) elaboraram uma metodologia em fases, semelhante à proposta neste trabalho, onde inicialmente é desejada a detecção de todos os nós com algum tipo de aplicação P2P ativa, para depois, na fase seguinte, verificar entre estes, quais são *bots* P2P. Entretanto, em sua proposta há o emprego de técnicas de clusterização, como *K-means*, que demanda longos períodos de processamento, em alguns casos podendo chegar a dias, como mostrado pelos próprios autores, o que a torna inviável para uma detecção em redes dinâmicas. De forma análoga, Zhang et al. (ZHANG, 2011) propuseram uma metodologia baseada em fases, porém utilizaram a persistência inerente a *bot* P2P para detectá-la, característica que demanda grande quantidade de tempo para ser verificada, não permitindo desta forma uma resposta em um período curto.

3 CONCEITOS BÁSICOS

Neste capítulo são abordados os três principais conceitos necessários para o entendimento da arquitetura proposta: redes *peer-to-peer* (P2P), *botnets* e árvores de decisão.

As redes P2P são de extrema importância devido à própria natureza da *botnet* P2P, que essencialmente é uma aplicação P2P, e, por consequência, possui características inerentes a este tipo de rede, podendo ser detectado por algumas metodologias desenvolvidas para a detecção destas redes.

A importância do entendimento de *botnets* está no fato de serem o objeto de estudo desta dissertação, sendo de fundamental relevância a razão do seu comportamento em detalhes.

As árvores de decisão são representações simples do conhecimento e têm sido aplicadas em sistemas de aprendizado. Elas são amplamente utilizadas em algoritmos de classificação, como um meio eficiente para construir classificadores que predizem classes baseadas nos valores de atributos. Assim, podem ser utilizadas em várias aplicações como diagnósticos médicos, análise de risco em créditos, entre outros exemplos. Devido ao rápido processamento para a sua geração e para realizar o seu percurso, ambos demandam intervalo de tempo curtos. Nesta dissertação esta estrutura será responsável por classificar os nós contendo aplicações P2P ativas dos que não as contém.

3.1 REDES PEER-TO-PEER

Redes *peer-to-peer* (P2P) são redes *overlays* que não utilizam o paradigma Cliente/Servidor comumente encontrado na *web* (SPOGNARDI, 2005). São constituídas por um conjunto de nós, podendo cada um deles possuir as mais distintas características (capacidade de processamento, sistema operacional, quantidade de memória, entre outras), denominados *peers*. Estes ora atuam como clientes, requisitando dados da rede, ora atuam como servidores, fornecendo dados para a rede.

Por não possuírem um nó central responsável pela distribuição de todo o conteúdo na rede, que geraria uma centralização, mas sim um conjunto de *peers*, que constituem a rede e contribuem cada um com uma parte das informações disponíveis, estas redes são denominadas redes descentralizadas.

Os *peers* se conectam diretamente uns aos outros provendo assim uma comunicação direta. As Figuras 3.1 e 3.2 mostram exemplos de uma topologia Cliente / Servidor e de uma topologia descentralizada, respectivamente.

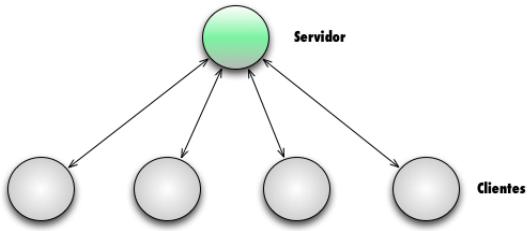

FIG. 3.1: Modelo de topologia Cliente / Servidor

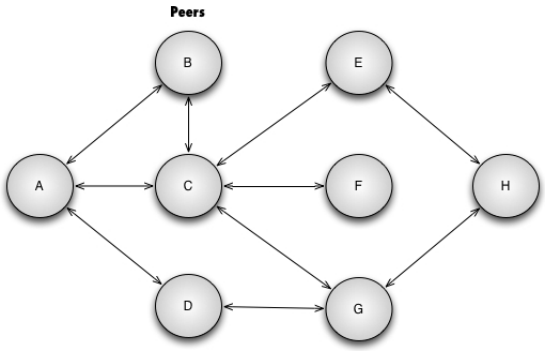

FIG. 3.2: Modelo de topologia descentralizada

Entre os benefícios introduzidos por essa nova topologia, aponta-se:

- *Auto-organização:* a não existência de uma figura controladora de toda a rede permite que um nó, ao se conectar à rede, escolha os *peers* que melhor o convier. Tal conveniência pode ser definida por proximidade geográfica, melhor largura de banda ou mesmo *peers* com grande número de conexões, o que contribuirá com a geração de redes complexas (DAGON, 2007).

- *Comunicação direta:* os *peers* trocam informações de modo direto, ou seja, não há um nó intermediário entre eles recebendo e repassando as informações a serem trocadas.

- *Resiliência:* a rede é capaz de se adaptar ao ambiente e de se manter operante mesmo com falhas em alguns nós. Na Figura 3.2, caso as ligações entre os *peers*

A-B e A-D sejam desfeitas, o nó A, ainda assim, será capaz de buscar e repassar informações para qualquer outro nó da rede através dos caminhos remanescentes.

- *Escalabilidade:* a rede cresce facilmente sem criar gargalos. Na Figura 3.2 é possível notar que entre dois nós quaisquer pode-se traçar múltiplos caminhos.

O termo *Peer-to-Peer* foi criado pela IBM em 1984 com o projeto *Advanced Peer-to-Peer Networking Architecture* (SILVA, 2012a) e posteriormente implementado nas redes *UseNet* (SIT, 2008) e *FidoNet* (BUSH, 1993), porém somente nos anos 90, com a popularização da internet e o aumento da largura de banda disponível, aliado ao advento de novas tecnologias, como o *mp3* (1995) e o *DivX* (1999), que permitiram maiores taxas de compressão de arquivos de áudio e vídeo, que estas redes se tornaram populares juntamente com as aplicações responsáveis pela distribuição de conteúdo: *Napster* (1999), *eDonkey* (2000) , *Kademlia* (2002), entre outras (HONG, 2011; LOCHER, 1995).

Aplicações que usam redes P2P, denominadas aplicações P2P, têm conquistado uma parcela cada vez maior da internet. Estatísticas atuais do site *Sourceforge* mostram que entre os cinco aplicativos mais procurados, três são aplicações P2P, totalizando quase um bilhão de *downloads*.

De acordo com (ERMAN, 2007), é estimado que aproximadamente 70% do tráfego na internet seja oriundo deste tipo de aplicação. Tal crescimento vem gerando uma preocupação por parte de provedores de acesso à internet (ISPs), pois apesar de tais redes serem a principal fonte de distribuição de informações, seu rápido crescimento degrada a qualidade dos serviços prestados por estas empresas.

Contudo, a sua arquitetura descentralizada aliada ao fato da falta de autoridades certificadoras nas redes para prover um gerenciamento e controle do conteúdo difundido, tem as transformado em um ambiente ideal para a difusão de arquivos piratas (áudio, vídeos, jogos, livros, etc) bem como para a proliferação de *malwares*. A Figura 3.3 nos mostra o percentual de utilização da internet pelos diversos protocolos e a Figura 3.4 os principais conteúdos buscados nas redes P2P (Fonte: IPoque).

Por este motivo, as redes P2P e suas possíveis arquiteturas são objetos de diversos estudos, tanto visando aprimorá-las de forma a obter um maior controle sobre as informações trafegadas, como objetivando sua detecção e posterior neutralização impedindo, assim sua utilização para a propagação de conteúdo pirata.

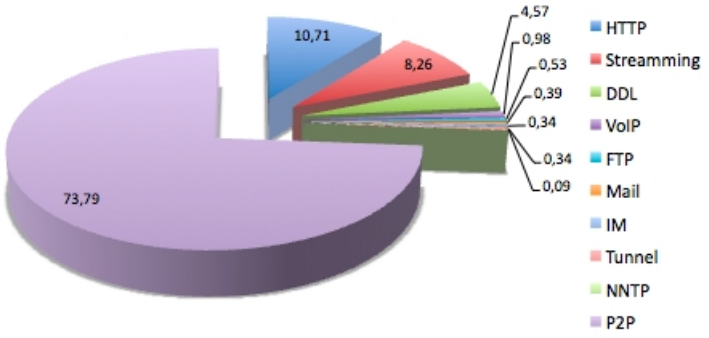

FIG. 3.3: Percentual de utilização da internet por protocolo

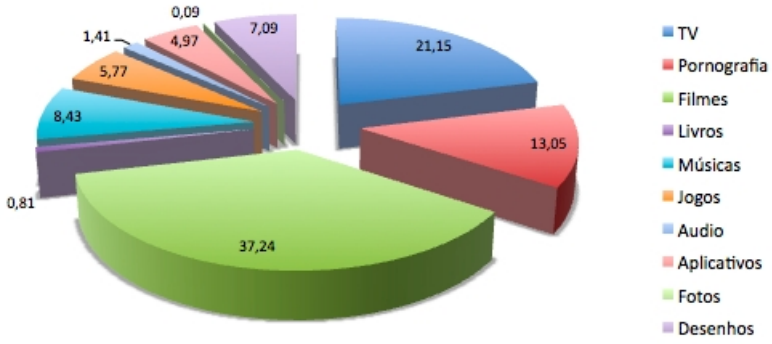

FIG. 3.4: Percentual de conteúdo buscado nas redes P2P

3.1.1 REDES P2P COM ARQUITETURA NÃO ESTRUTURADA

Redes P2P com arquitetura não estruturada utilizam algoritmos aleatórios para a sua construção (VISHNUMURTHY, 2007, 2006). Cada *peer* constituinte da rede possui uma lista, denominada *peer list*, com o endereço de seus vizinhos. A Figura 3.2 é um exemplo deste tipo de estrutura, onde as setas indicam as ligações entre os *peers*, em outras palavas indicam os nós contidos nas *peers lists* dos demais membros.

A busca por informações neste tipo de arquitetura é feita por inundação na rede, ou seja, todos os vizinhos do nó que efetua a busca recebem o pedido que por sua vez o propagam para seus vizinhos até que a informação seja encontrada, técnicas como *Random Walk* e *Random Walk 2* (BISNIK, 2005) estão entre as mais utilizadas. Entretanto, estas inundações podem causar congestionamentos na rede em que se propagam devido à quantidade elevada de pacotes gerados para localizar a informação de interesse. Visando evitar este problema, costuma-se utilizar mecanismos para a retirada destes pacotes da rede após certo tempo sem lograr exito (como por exemplo, TTL). Porém, tais mecanismos podem inviabilizar a localização da informação caso esta se encontre muito afastada do

peer que a busca. Utilizando a Figura 3.2 como exemplo, é possível perceber que com um TTL igual a 2 o *peer* A jamais conseguirá encontrar uma informação existente apenas no *peer* H.

3.1.2 REDES P2P COM ARQUITETURA ESTRUTURADA

Redes P2P com arquiteturas estruturadas possuem servidores, chamados *super peers*, que mantêm tabelas com *hashes* dos nomes ou metadados dos arquivos existentes na rede e os *peers* que os possuem. Tais tabelas são denominadas *Distributes Hash Table* (DHT) (VISHNUMURTHY, 2007).

As tabelas DHT apresentam diversas implementações (*Tapestry, Chord, CAN, Kademlia*), como mostrado em (TANNER, 2005). As mais comuns utilizam o par ordenado *(ID, Endereço IP)* para localizar a informação requerida, onde o ID é um *hash* de algum metadado referente ao arquivo e o endereço IP refere-se ao *peer* que o possui (Figura 3.5).

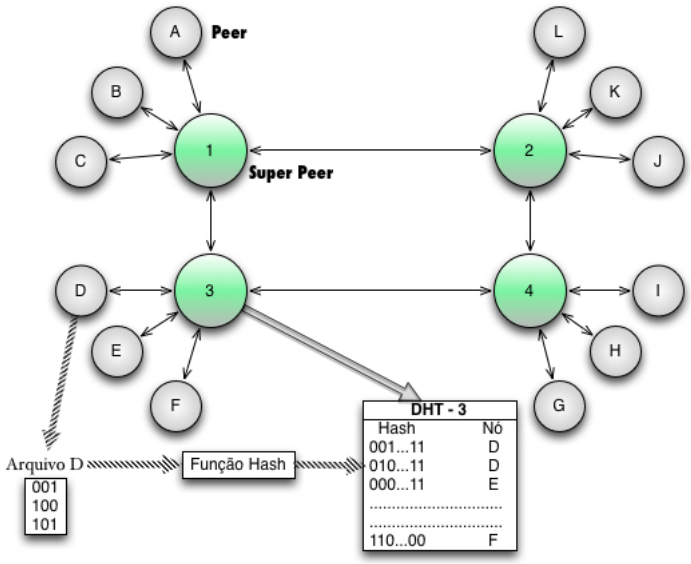

FIG. 3.5: Modelo de uma arquitetura P2P estruturada e de uma DHT.

Um *peer* ao se conectar a este tipo de rede P2P, o fará através de um destes servidores, que são ligados entre si através de um anel lógico por onde a consulta é propagada para os demais *super peers*.

Ao executar uma busca por determinada palavra chave, esta deverá ser transformada em uma sequência de bits resultantes da conversão em *hash* (na maioria das implementações tal sequência possui 128 ou 160 bits) (TANNER, 2005), e então ser enca-

minhada ao *super peer* a qual o nó esta diretamente conectado, que será responsável por realizar a busca em sua DHT e replicá-la para os demais *super peers*, a fim de encontrar o nó detentor da informação de interesse. Após a realização da busca, se for bem sucedida, o endereço IP do nó detentor da informação será repassado ao nó que gerou a busca e estes se conectarão de modo direto para que a informação seja enviada. Após a conclusão da transferência, o novo nó que agora também possui a informação, atualizará a tabela DHT do *super peer* a que esta conectado. Algumas implementações permitem que mais de um *peer* transfira a informação para o nó que originou uma busca, otimizando desta forma a transferência dos dados. A Figura 3.6 mostra um esquema de como é realizada a busca e transferência da informação.

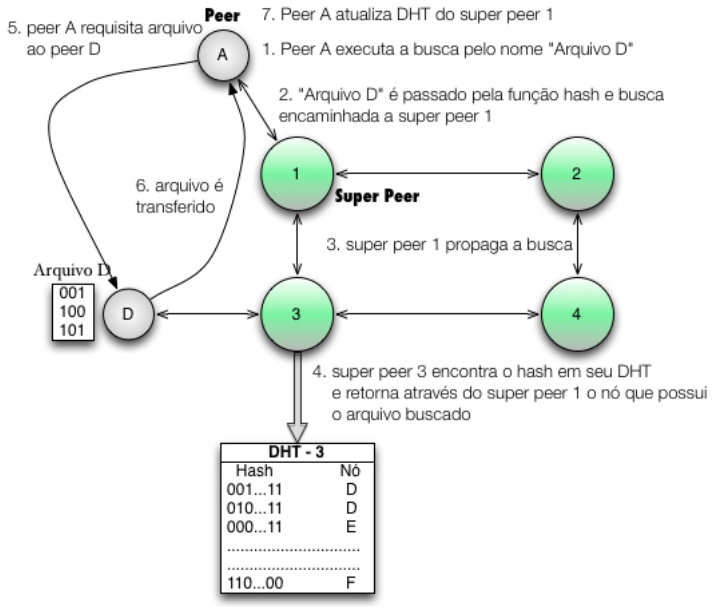

FIG. 3.6: Busca realizada pelo *peer* A ao Arquivo *D* pertencente ao *peer* D.

Neste tipo de arquitetura, o problema da degradação da rede devido ao aumento do tráfego gerado pelas buscas, como ocorre nas arquiteturas não estruturadas, é praticamente inexistente. Porém, falhas nos *super peers* podem comprometer toda a estrutura criada. Desta forma, técnicas de replicação das tabelas devem ser implementadas, evitando que parte das informações fiquem indisponíveis com eventuais falhas. Ademais, devido à utilização de *hashes* para o armazenamento e busca das informações, pequenas variações nos nomes podem gerar *hashes* completamente diferentes, prejudicando assim a localização do conteúdo desejado.

3.2 BOTNETS

Botnets são redes formadas por uma grande quantidade de *bots*. *Symantec* define *bot* como *malware*, semelhante a um *worm* ou *trojan*, dotado de um canal de C&C por onde informações são enviadas ou recebidas por um controlador desta rede.

Atualmente, representam a principal ameaça à segurança da internet e são responsáveis por diversas atividades maliciosas como roubo de informações sigilosas, fraudes bancárias, *Distributed Denial of Service* (DDoS), envio de *spams*, entre outras. Estima-se que quase a totalidade dos *spams* são enviados destas redes.

Segundo (ACCELERATORS, 2007), há diversos modos de um computador ser infectado por uma *bot*. Estes, constantemente, buscam na internet por máquinas com vulnerabilidades em *softwares* para que possam explorar, além de se enviarem através de e-mails, escondidos em programas ou em *sites* maliciosos. A Tabela 3.1 mostra os principais meios utilizados para a propagação de *bots*.

AMEAÇA	**DESCRIÇÃO**
E-mail	a bot é enviada por e-mail
Exploit	alguma vulnerabilidade do sistema é explorada
Compartilhamento de Arquivos	arquivos com o *malware* escondido são distribuídos pela rede
Mensagens Instantâneas (IM)	arquivos são enviados via IM
Ataques de Força Bruta	senhas são testadas em sistemas com o objetivo de se obter acesso
Downloads da internet	bots são obtidas diretamente de sites maliciosos
Dispositivos Móveis	bots são espalhadas através de dispositivos compartilhados entre diversos computadores
Varredura de rede	algumas bots realizam varreduras em redes com o objetivo de encontrar determinadas portas abertas ou buscando por endereços IPs aleatórios para serem atacados
Redes P2P	bots são instaladas através de arquivos buscados nestas redes de compartilhamento

TAB. 3.1: Principais meios para a propagação de bots (Fonte: Microsoft.com (ACCELERATORS, 2007)).

3.2.1 FASES DE OPERAÇÃO DA BOTNET

Botnets se distinguem dos demais *malwares* existentes por seu peculiar modo de operar. De acordo com (FEILY, 2009), *botnets* possuem cinco fases de operação distintas conforme mostrado na Figura 3.7. É através do conhecimento destas fases que se fundamenta o estudo para a sua mitigação.

- **Espalhamento Inicial ou Injeção:** consiste no espalhamento e auto-injeção nos sistemas vulneráveis a ela. Segundo (FEILY, 2009), *botnets* possuem técnicas de espalhamento semelhante à utilizada por outros *malwares*, dentre as quais, se podem destacar:

  - *Engenharia Social:* segundo o Instituto Sophos (sop, 2012), esta é a principal via de distribuição de *malwares*, e-mails com conteúdos atrativos são enviados como *spams* e, ao terem seus *links* acessados, a injeção do *malware* ocorre no sistema vulnerável.
  - *Exploração Remota de Vulnerabilidades:* uma máquina já infectada por uma *bot*, ou algum outro *malware*, de modo pró-ativo pode buscar por demais máquinas com a mesma vulnerabilidade para que possam ser infectadas, um exemplo bastante explorado é a aplicação LSASS (*Local Security Authority Subsystem Service*) do sistema operacional *Windows*.
  - *Mensagens Instantâneas:* alguns *malwares*, como *W32.pipeline*, se utilizam de *links* enviados através de mensagens instantâneas, ou mesmo imagens que, ao serem acessadas, o injetam no sistema vulnerável.
  - *Redes P2P:* diversas *bots*, em especial as que possuem arquitetura descentralizada, se utilizam deste método, devido a falta de uma autoridade certificadora na maioria das redes P2P para fazer o controle do que é compartilhado. Muitos arquivos compartilhados por estas redes estão corrompidos, trocados por conteúdos diversos ou mesmo infectados por algum *malware*, que ao serem executados são instalados no sistema vulnerável.
  - *Através de outras Botnets:* esta forma de espalhamento pode ser subdividida em duas outras, onde o espalhamento pode ser feito por uma *botnet* destinada a esta função apenas, ou uma máquina infectada por um *malware* mais antigo pode sofrer uma atualização de forma a se tornar uma *bot*.

- **Injeção Secundária:** nesta fase, segundo (FEILY, 2009), um *script* é executado localmente na máquina infectada com o intuito de obter, através de servidores HTTP, FTP ou P2P, na grande maioria das vezes, o arquivo binário da *bot*, para que esta seja instalada e assim habilitar a máquina a se tornar uma *bot* propriamente dita.

- **Conexão:** é nesta fase que é feito o acesso ao canal de C&C pela primeira vez, neste momento a *bot* se habilita a receber comandos do *botmaster* ou enviar informações para este. Segundo (XIAO-NAN, 2011) os métodos responsáveis pela troca de informações entre as *bots* e o *botmaster* são *push* e *pull*:

  - *Método Push:* comandos são enviados pelo *botmaster* para as *bots* através do canal de C&C, permitindo desta forma, que sua execução ocorra de forma simultânea em toda a rede. Entretanto, para que este método seja possível há a necessidade do conhecimento do endereço de acesso a cada uma das *bots*.
  - *Método Pull:* a *bot* é responsável por iniciar a comunicação buscando por novos comandos. Esta busca pode ser feita por agendamentos interativos, ou seja, a *bot* envia uma solicitação para o canal de C&C que é respondida, ou não-interativos, onde os comandos são disponibilizados em um repositório (FTP, *LinkedIn*, *Facebook*,...) e estes são buscados periodicamente pelos nós.

- **Canal de Comando e Controle:** nesta fase a *bot* já está operando na rede. O canal de C&C, por onde trafegam todas as informações das *botnets*, é considerado ponto de neutralização destas redes, sendo um dos principais objetos de estudo na área de detecção e mitigação. De acordo com (COOKE, 2005), as *bots* podem ser classificadas segundo seu canal de C&C em centralizadas e descentralizadas (P2P).

- **Atualizações:** é neste momento que ocorrem as atualizações no código da *bot*. Com o objetivo de dotar a *botnet* de novas funcionalidades ou mesmo de corrigir possíveis vulnerabilidades ou erros em seu código, *botmasters* estão sempre atualizando suas *botnets*. Tais atualizações comumente contemplam alterações na criptografia utilizada e mudanças de endereços de servidores de C&C.

3.2.2 BOTNETS CENTRALIZADAS

Botnets com esta arquitetura possuem um nó central responsável por encaminhar comandos e dados entre o *botmaster* e sua rede maliciosa. Possuem uma menor latência no envio

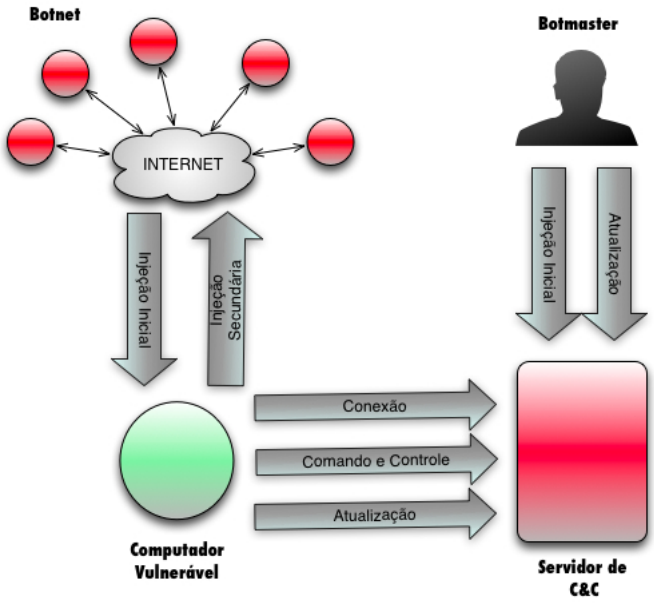

FIG. 3.7: Diagrama básico de operação de uma *botnet*.

de mensagens, pois todas as *bots* da rede estão conectadas diretamente a um conjunto de poucos servidores de C&C, como mostrado na Figura 3.8. Todavia, este pequeno conjunto de servidores é considerado a principal vulnerabilidade, já que sua desarticulação implica na total inutilização da rede. Os principais protocolos utilizados neste tipo de arquitetura são IRC e o HTTP, porém servidores que enviam e recebem informações através de SMS e IM já foram observados em *botnets* (SILVA, 2012b).

As primeiras *botnets* utilizavam o IRC como canal de C&C devido a facilidade de utilização e escalabilidade que este protocolo possui, é possível conectar em uma única sala IRC milhares de *bots* para que os comandos sejam repassados pelo *botmaster*, permitindo assim o gerenciamento da rede formada. Assim que uma máquina é infectada, ela automaticamente se junta a um determinado canal público ou privado IRC, que já é definido em seu código, para aguardar por comandos a serem enviados.

Apesar de ainda hoje ser utilizado por diversas *botnets*, este tipo de canal de C&C pode ser neutralizado através do bloqueio das portas usadas pelo IRC ou através de cancelamento de salas reconhecidamente frequentadas pelas *bots*. (MAZZARIELLO, 2008) mostrou que palavras chaves utilizadas nas salas podem ser empregadas como metodologia para encontrar as *bots* e as salas em que frequentam.

Com o intuito de evadir-se de técnicas de detecção para *botnets* baseadas no protocolo IRC, outro tipo de protocolo começou a ser utilizado por esta arquitetura, o HTTP, que

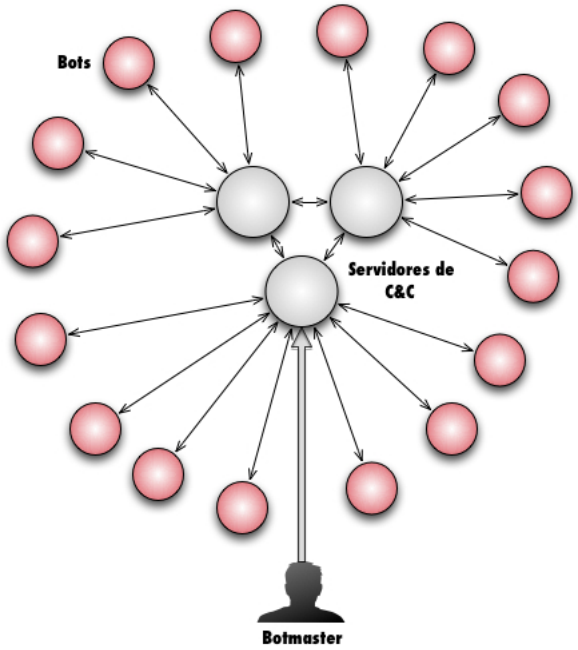

FIG. 3.8: Estrutura de uma *botnet* centralizada.

possibilita ao tráfego gerado por essas *bots* se misturar ao tráfego legítimo oriundo de acesso a sites e, desta forma, manter a rede maliciosa escondida das técnicas de detecção até então empregadas.

3.2.2.1 MECANISMO DE RALLY

Independente do protocolo que utiliza, arquiteturas centralizadas precisam necessariamente, em determinados períodos de tempo, fazer com que todas as *bots* ativas acessem o servidor de C&C a fim de receber novos comandos, atualizações ou repassar informações obtidas. Tal ato é denominado mecanismo de *rally*.

Para que a *bot* seja capaz de realizar este contato com o servidor é necessário que ela saiba como acessá-lo. Três métodos são utilizados para a sua execução: via endereço IP, através de *Dynamic DNS Domain Name* ou via *Distributed DNS Domain Service*.

O acesso feito diretamente pelo endereço IP do servidor de C&C foi adotado e logo substituído por outros meios, pois caso uma *bot* seja descoberta, uma análise de seu código revelaria o endereço, que poderia ser bloqueado inutilizando toda uma *botnet*. Tal implementação não permite a migração do servidor para outros endereços.

O *Dynamic DNS Domain Name* é a técnica mais utilizada para a definição de canais de C&C em arquiteturas centralizadas. Desta forma, o *botmaster* define apenas os nomes

de domínios que o servidor utilizará, estes nomes são registrados em um servidor DDNS e caso ocorra algum problema, basta migrá-los para outro endereço IP e fazer a atualização do servidor DDNS.

A principal diferença entre o *Dynamic DNS Domain Name* e o *Distributed DNS Service* é que neste, a *botnet* possui seu próprio servidor de nomes, não sendo necessário efetuar cadastros em DDNS alheio, possibilitando que parâmetros, como porta utilizada para consultas sejam alteradas a fim de dificultar a monitoração do tráfego entre *bots* e DNS.

Fundamentado no mecanismo de *rally* existente neste tipo de *botnet* diversas metodologias de detecção foram elaboradas.

Choi et al. (CHOI, 2007) definiram uma série de características do tráfego gerado por uma *botnet* para um servidor DNS que a distinguem de um acesso legítimo feito a este servidor. Segundo o autor existem cinco casos em que uma *bot* necessita consultar o DNS:

- **No procedimento de rally:** no momento em que o *malware* infecta o terminal, este precisa se conectar ao canal de C&C para que o *botmaster* possa utilizá-lo.
- **Durante a execução do comportamento malicioso:** muitas atividades maliciosas são acompanhadas de consultas a um DNS (DDoS, envio de *spam*).
- **Em caso de falhas no canal de C&C:** se a conexão com o servidor falhar, as *bots* devem fazer novas consultas ao DNS a fim de obter um novo acesso ao canal.
- **Durante a migração do servidor de C&C:** as *bots* efetuam consultas ao DNS a fim de obter o novo endereço do servidor de C&C.
- **Se o endereço IP do servidor de C&C for alterado:** as *bots* fazem consulta ao DNS a fim de obter o novo endereço IP.

A Tabela 3.2 mostra as principais diferenças entre requisições feitas pelas *bots* e feitas por usuários legítimos. Em primeiro, somente membros de uma *botnet* enviam requisições sobre servidores de C&C, motivo pelo qual se tem uma quantidade fixa de consultas, enquanto usuários acessam diferentes domínios com diferentes endereços IP. Em segundo, os membros de uma *botnet* agem juntos, executando requisições em situações específicas enquanto o tráfego legítimo é aleatório e contínuo. Além de que servidores de C&C normalmente utilizam DDNS enquanto servidores legítimos costumam utilizar DNS.

Fundamentados nestes apontamentos uma metodologia foi elaborada em (CHOI, 2007) para detecção de servidores de C&C para assim poder bloqueá-los e por consequência neutralizar a *botnet* associada a ele.

	IPs que acessam o domínio	**Padrões de Atividades**	**Tipo de DNS**
Botnet	Tamanho fixo (membros da botnet)	atividades em grupo intermitente (situações específicas)	normalmente DDNS
Legítimos	tamanho variável	acesso contínuo e aleatório	normalmente DNS

TAB. 3.2: Diferença entre domínios legítimos e relacionados a botnets.

Outra característica inerente aos registros DNS feitos para servidores de C&C é o baixo valor de TTL (*time to live*) associado a eles. O TTL registra o tempo que um determinado domínio deve ficar gravado em *cache* evitando assim consultas repetidas e, por consequência, diminuindo a carga nos servidores de nomes.

Valores de TTL muito baixos podem gerar sobrecargas nos servidores de nome, porém valores bastante elevados podem gerar inconsistência nos servidores, já que uma possível alteração em um endereço IP pode demorar para ser propagada, deixando o servidor detentor do domínio inacessível. Tal característica estimula *botmasters* a manter o TTL de seus domínios com valores baixos, entretanto, cautela é necessária ao se usar esta metodologia, devido a existência de domínios legítimos que também utilizam baixos TTL (CNN, YAHOO!, GOOGLE) devido a constantes alterações de endereços IP de seus servidores. Domínios legítimos registrados em DDNS costumam possuir, também, baixo valor de TTL.

3.2.3 BOTNETS DESCENTRALIZADAS

Botnets descentralizadas incorporam protocolos P2P em seu código. Apesar destes protocolos já serem relativamente antigos, seu uso em *botnets* é recente. Sua principal característica reside na inexistência de um servidor de C&C, que na arquitetura centralizada é considerado um ponto vulnerável. Por causa da descentralização de seu canal de C&C, este tipo de arquitetura se torna de difícil detecção e neutralização. Quando uma *bot* é desconectada do sistema P2P, não é causado grande impacto sobre as *bots* remanescentes.

Nesta arquitetura os comandos são repassados entre os *peers*, gerando uma maior latência para envio de mensagens quando comparado à arquitetura centralizada, todavia,

a descoberta e neutralização de alguns *peers* não comprometem o funcionamento da *botnet*, elevando assim sua resiliência, como explicado para redes P2P.

Em (WANG, 2009) as *botnets* descentralizadas são classificadas em três grupos distintos: parasitas, sangue-sugas e *botnets* customizadas.

- **Parasita:** as *botnets* P2P parasitas dependem do protocolo P2P para o seu funcionamento, seu canal de C&C é dependente da rede P2P que habitam, as *bots* são selecionadas de máquinas vulneráveis existentes nesta rede e, por consequência, as *botnets* deste tipo têm seu tamanho limitado pela quantidade de *peers* membros da rede legítima.

- **Sangue-sugas:** *botnets* P2P deste tipo, assim como as parasitas, dependem da rede P2P que habitam para o funcionamento do seu canal de C&C. Entretanto, são capazes de infectar não apenas máquinas vulneráveis pertencentes a rede P2P em que habitam mas também as que não pertencem, permitindo desta forma o seu crescimento de forma ilimitada. Versões iniciais da *botnet Storm* (HOLZ, 2008; DAVIS, 2008; YU, 2009) adotavam este comportamento.

- **Botnets customizadas:** uma *botnet* P2P customizada possui sua própria rede P2P, não havendo necessidade de habitar redes P2P legítimas. Máquinas vulneráveis são infectadas e levadas para estas redes, a versão atual da *Storm* e *Nugache* (STOVER, 2007; HOLZ, 2008; DAVIS, 2008; YU, 2009) são exemplos de *botnets* deste tipo.

3.2.3.1 CONSTRUÇÃO DAS BOTNETS P2P

Como visto, *botnets* se aproveitam de diversos mecanismos para espalhar seu código malicioso pela internet. De modo particular, podem se aproveitar da falta de autoridades certificadoras nas redes P2P para contaminar arquivos compartilhados, ou utilizar técnicas, como o envenenamento de DHT (LIANG, 2006), onde *bots* residentes nas redes P2P estruturadas propagam seu código malicioso como se fossem arquivos legítimos, os registrando nas tabelas DHT dos *super peers*. Tal mecanismo é suficiente para construir *botnets* parasitas, já que estas se aproveitam de toda a estrutura da rede P2P em que habitam, logo não há necessidade de compartilhar *peer lists* de *bots*, as próprias listas da rede legítima são suficiente para manter o contato entre as *bots*.

Entretanto, em *botnets* P2P com redes customizadas ou sangue-sugas, existe a infecção feita em máquinas fora das redes. Para estas máquinas comprometidas, um mecanismo de conexão à rede se faz necessário, dessa forma dois modelos são utilizados. No primeiro modelo, o código malicioso possui uma *peer list* com endereços de *bots* já definidos. A máquina, ao executar este código, tenta acessar cada um dos *peers* registrados para que estes repassem a ela uma lista mais atualizada. A outra forma de conexão é feita através de um servidor *web*, responsável por manter uma *peer list* atualizada. Ao se conectar pela primeira vez, a máquina infectada acessa este servidor que tem seu endereço no próprio código malicioso e atualiza a sua lista.

Esta conexão inicial feita pelas *bots* é denominada de procedimento de *bootstrap* e é considerado uma vulnerabilidade para essas redes maliciosas, uma vez que o servidor *web* ou os endereços de *bots* contidos na *peer list* inicial sejam desativados ou descobertos a *botnet* fica comprometida.

Visando sobrepujar tal vulnerabilidade, técnicas que evitam o procedimento de *bootstrap* foram elaboradas. Na *botnet* P2P híbrida elaborada em (WANG, 2007), quando uma *bot* A infecta uma máquina B, A passa toda a sua lista de *peers* para B e o adiciona em sua *peer list* também. Quando uma *bot* tentar infectar uma máquina já infectada, estas duas trocam suas listas, gerando assim novas listas para cada, tal metodologia foi abordada em (BARAKAT, 2010) para a construção das chamadas *super botnets*.

3.3 ÁRVORES DE DECISÃO

As árvores de decisão são ferramentas utilizadas para auxiliar um agente em sua capacidade de aprender e de tomar decisões fundamentadas no que foi aprendido anteriormente com a observação de um conjunto de informações.

O maior desafio relacionado a construção destas árvores é como encontrar um algoritmo capaz de formar a estrutura com menor quantidade de nós possível e, por consequência, menor custo.

Cada nó neste tipo de árvore indica o teste para um atributo e a escolha dos atributos a serem utilizados é feita com base no maior ganho de informação possível. Para se classificar uma amostra, basta percorrer a árvore a partir de sua raiz, respondendo aos atributos encontrados nos nós.

Um exemplo bastante citado de árvore de decisão está definido em (QUINLAN, 1986), que relaciona características relacionadas a um determinado dia com a possibilidade de

sair ou não de casa. Neste exemplo, os atributos e os valores que podem ser assumidos por eles são mostrados na Tabela 3.3.

ATRIBUTO	VALORES
Perspectiva	ensolarado, nublado, chuvoso
Temperatura	frio, média, quente
Umidade	alta, normal
Vento	sim, não

TAB. 3.3: Atributos e seus valores correspondentes.

A partir de um conjunto de treinamento, que possui objetos com todos os atributos preenchidos, bem como o resultado em que eles acarretam (em nosso caso, sair ou não de casa), algoritmos de árvores de decisão devem implementar uma árvore capaz de predizer os resultados de objetos subsequentes apenas conhecendo o valor de seus atributos. Desta forma, com um conjunto de treinamento definido na Tabela 3.4, aplicados a um determinado algoritmo de árvore de decisão irá gerar a árvore representada na Figura 3.9.

Pela Figura 3.9, observa-se que se o dia possuir uma perspectiva de ser nublado o resultado será sair de casa. Caso essa perspectiva seja diferente de nublado (ensolarado ou chuvoso) outros testes deverão ser feitos. No caso do dia ser ensolarado a umidade deverá ser verificada e no caso de ser chuvoso o vento será verificado.

Perspectiva	Temperatura	Umidade	Vento	Resultado
ensolarado	quente	alta	não	não sair
ensolarado	quente	alta	sim	não sair
nublado	quente	alta	não	sair
chuvoso	médio	alta	não	sair
chuvoso	frio	normal	não	sair
chuvoso	frio	normal	sim	não sair
nublado	frio	normal	sim	sair
ensolarado	média	alta	não	não sair
ensolarado	frio	normal	não	sair
chuvoso	média	normal	não	sair
ensolarado	média	normal	sim	sair
nublado	média	alta	sim	sair
nublado	quente	normal	não	sair
chuvoso	média	alta	sim	não sair

TAB. 3.4: Conjunto de treinamento para a geração da árvore de decisão.

Segundo (YAO, 2005), uma árvore de decisão para ser considerada eficiente deve atender aos seguintes requisitos:

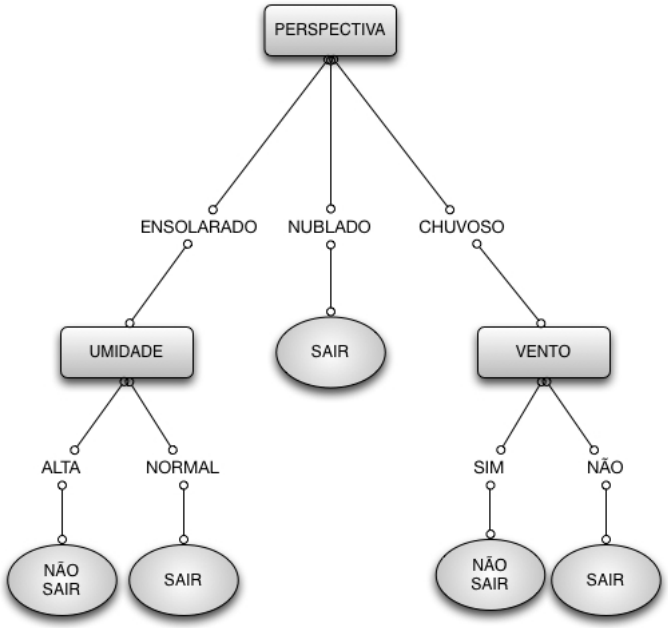

FIG. 3.9: Árvore gerada para a tabela 3.4

- **Acurácia:** deve ser capaz de gerar um modelo que prediga de maneira correta a classe em que dados ainda não visualizados deve pertencer.
- **Velocidade:** deve ter um custo computacional para geração do modelo que seja o menor possível.
- **Robustez:** deve gerar um modelo capaz de predizer corretamente dados com valores faltando.
- **Flexibilidade:** deve gerar um modelo eficiente baseado em grandes quantidades de dados.

(QUINLAN, 1986) elaborou o algoritmo ID3 (*Iterative Dichotomiser 3*), capaz de gerar árvores de decisões, tal algoritmo é o precursor do algorítimo C4.5 (QUINLAN, 1993), também elaborado pelo autor e utilizado neste trabalho. Ambos são fundamentados na entropia da informação (Equação 3.1), que pode ser definida como o grau de incerteza que uma variável aleatória pode assumir.

Os principais adventos trazidos pelo algoritmo C4.5 são a capacidade de ser utilizado com atributos que assumem valores contínuos, bem como em conjuntos onde os valores de alguns atributos estejam ausentes.

Nestes algoritmos quanto menor for a entropia, mais próxima da raiz da árvore estará localizado o atributo. Ou seja, para a árvore mostrada na Figura 3.9, o atributo perspectiva possui a menor entropia entre os atributos, isto por que há uma maior coincidência entre os resultados sair ou não sair de casa e os valores assumidos pelo atributo em questão, como observado na Tabela 3.4.

A equação $E(S) = -\sum_{j=1}^{n} f_s(j) log_2 f_s(j)$ define o valor da entropia para o atributo S; onde n é o número de valores do atributo e $f_s(j)$ é a frequência do valor j para o atributo de S.

4 ARQUITETURA DE DETECÇÃO DE BOTS P2P

A arquitetura proposta neste trabalho tem o objetivo de detectar *bots* P2P de maneira *online*. Por *online*, entende-se que não deve haver uma demora entre a produção dos dados a serem analisados e os resultados obtidos por esta análise. Como mencionado em capítulo anterior, a demora ocorrida entre a coleta de dados e posterior análise pode comprometer a usabilidade e veracidade dos resultados devido a dinâmica encontrada nas redes atuais. Nós são constantemente inseridos e retirados das redes e possuem seu endereços IPs, muitas vezes, alterados por servidores DHCP.

Esta arquitetura é possível devido às seguintes premissas adotadas:

- utilização de módulos para a análise do tráfego de rede de pronta execução, ou seja, que não necessitem de históricos de longa duração;
- não utilização de métodos baseados em *Deep Packet Inspection* (DPI) (CHEN, 2009b), evitando assim o retardo no envio de pacotes pela rede e a sobrecarga com o armazenamento de *payloads*;
- não utilização de métodos baseados em assinaturas, como valores de portas, mostrado em (AVIV, 2011), tais métodos limitam-se a encontrar *bots* já conhecidas e não a descobrir novas.

Para atingir este objetivo, a arquitetura foi dividida em duas fases distintas (como proposto em trabalhos anteriores (ZHANG, 2011; LIU, 2010)), detecção de nós com aplicações P2P ativas (fase 1) e detecção de nós com *bots* P2P ativas (fase 2), cada qual constituída por módulos. Desta forma é alcançada uma maior otimização do processo, devido a uma redução do universo a ser analisado na segunda etapa. Outro argumento que corrobora nessa divisão em fases, como mostrado no capítulo anterior, é o fato de *bots* P2P serem inerentemente aplicações P2P, sendo assim, suscetíveis a muitas técnicas de detecção destinadas a estas. A Figura 4.1 mostra o funcionamento da arquitetura em fases proposta.

FIG. 4.1: Fases da arquitetura proposta

4.1 ARQUITETURA MODULAR

O uso de uma arquitetura modular possibilita uma melhor adequação à rede em que for implantada. Dessa forma, evita-se o desenvolvimento de uma solução engessada a determinados tipos de redes com características pré-definidas.

Assim sendo, para um melhor desempenho da arquitetura proposta neste trabalho é necessário um estudo prévio das principais características da rede (percentuais de uso de pacotes TCP / UDP, tipos de aplicações em funcionamento na rede, presença de *firewall*, etc), para que sejam escolhidos os módulos mais adequados. Um conjunto de módulos que se adeque bem a uma rede não necessariamente se adequará da mesma forma a uma nova rede com características diferentes.

Botnets evoluem com o intuito de evadirem-se das mais diversas técnicas de detecção, inutilizando-as como mostrado em (HOLZ, 2008). Entretanto, é possível manter a arquitetura sempre atualizada através da implementação de módulos capazes de detectar as *bots* P2P desejadas, ou mesmo adaptá-la para novas *bots* que utilizem outros protocolos de comunicação. A Figura 4.4 mostra onde os bancos de módulos são utilizados na arquitetura proposta completa.

4.2 FASE 1: DETECÇÃO DE APLICAÇÕES P2P

Nesta fase, foram utilizados módulos capazes de detectar características de rede inerentes a aplicações P2P. Todos os módulos foram validados em trabalhos anteriores de diversos autores e nesta proposta foram adaptados para atuarem na análise dos fluxos de rede gerados pelos nós monitorados.

Entende-se por fluxo de rede uma tupla identificada pelo seguinte conjunto de dados: IP de origem, IP de destino, porta de origem, porta de destino e protocolo (equação 4.1).

O fluxo é considerado finalizado, quando um pacote TCP RST ou TCP FYN é enviado,

ou quando nenhum pacote é enviado por um período de tempo pré-determinado. A utilização dos fluxos visa a otimização no uso do espaço de armazenamento de dados.

$$Fluxo = (IP_{src}, IP_{dst}, Port_{src}, Port_{dst}, Proto) \tag{4.1}$$

4.2.1 MÓDULOS DE DETECÇÃO P2P

Para compor a arquitetura, foi feito um levantamento dos módulos em trabalhos anteriores, onde foram estudados e avaliados. Além disso, a fim de prover uma exemplificação do comportamento das redes P2P em relação a estes módulos, *traces* de duas redes com aplicações P2P em seus nós (*Bittorrent* e *Gnutella*) e de uma rede livre deste tipo de aplicação, denominada rede doméstica, tiveram a característica em estudo mensurada. O ambiente de teste é constituído por 3 computadores ligados através de uma rede *wireless* de 300 Mbps a uma conexão de internet de 10 Mbps. Somente um dos computadores foi monitorado e teve seu tráfego registrado para a análise das características em questão.

Abaixo são listados os filtros, bem como uma descrição sucinta e valores obtidos para cada um dos *traces* utilizados no teste. Dentre os filtros descritos, o percentual de conexões oriundos de uma mesma porta de origem foi elaborado neste trabalho.

- **Taxa de envio de pacotes TCP SYN / UDP:** aplicações P2P necessitam conectar-se a outros *peers* a fim de manter a sua rede funcional. Entretanto, sabe-se que os *peers*, em sua grande maioria, são computadores normais (e não servidores) que podem possuir endereços IPs dinâmicos ou mesmo podem estar atrás de *firewalls* ou NAT, impedindo assim que a conexão seja efetuada com sucesso. Almejando mitigar tal fato, os nós membros das redes P2P tentam efetuar conexões com diversas máquinas a fim de aumentar o número de sucessos nas conexões e desta forma o número de *peers* a que estão conectados, o que acarretará em uma maior eficiência na busca e propagação de dados (CHEN, 2009a). Este comportamento pode ser medido através da quantidade de fluxos TCP e UDP gerados em um determinado período de tempo.

- **Utilização do par de protocolos TCP/UDP:** segundo (CHEN, 2009a; KARAGIANNIS, 2004b), aplicações P2P tendem a utilizar ambos os protocolos (TCP e UDP). Normalmente, mensagens de controle utilizam, por hábito, o protocolo UDP

enquanto os dados transferidos utilizam o protocolo TCP. Dessa forma, a utilização de ambos os protocolos entre pares de *peers* para a troca de mensagens pode ser uma indicação da existência de aplicações P2P. O UDP é preferido para mensagens de controle devido ao seu menor *overhead* quando comparado ao protocolo TCP, entretanto, para o envio de dados há preferência pelo protocolo TCP devido ao maior controle propiciado.

- **Grau de entrada e saída de um nó:** a concepção das redes P2P almeja que todos os *peers* atuem ora como clientes (solicitando arquivos), ora como servidores (disponibilizando arquivos), resultando assim em máquinas que podem apresentar múltiplas conexões de entrada e de saída (*download* e *upload*). (YU, 2010) mostrou que tal característica é explorada pelas aplicações P2P e um valor limite pode ser utilizado para a separação de máquinas pertencentes e não pertencentes a redes P2P. Entretanto, tal característica é comumente burlada pelos denominados *freeriders* - usuários destas redes que possuem arquivos completos (*seeds*), ou em parte (*leechers*) e que não os disponibilizam a outros *peers*, desta forma, fazendo com que o grau de saída seja zero ou um valor próximo para eles. Este comportamento em redes P2P é repudiado e combatido de diversas maneiras dentre as quais a mais comum é a atribuição de limites de taxas de transferência de arquivos a usuários com baixos níveis de *uploads*.

- **Percentual de pacotes TCP:** da observação dos pacotes da camada de transporte, (CHEN, 2010) constatou que para usuários normais o percentual de utilização de pacotes UDP é muito baixo, pacotes TCP são mais utilizados pois possuem uma política de controle de erros não presente no UDP, aumentando assim a confiabilidade de muitas aplicações. Dessa forma, pacotes UDP são mais utilizados em aplicações em que a perda de pacotes não é tão crítica, ou é menos crítica de que o atraso em sua entrega devido ao reenvio de pacotes anteriores que não chegaram (como por exemplo, *streaming* de áudio e vídeo). Assim, o percentual de utilização de pacotes UDP pode se aproximar de zero, ou seja, o percentual de utilização de pacotes TCP se aproxima de 100%. Porém máquinas com aplicações P2P tendem a inundar as redes com pacotes UDP com o objetivo de trocar mensagens de controle ou de busca entre os *peers*, acarretando em um aumento percentual dos pacotes UDP e decréscimo dos pacotes TCP.

- **Valor médio do número das portas utilizadas:** segundo (ULLIAC, 2010), os protocolos P2P atuais, com o objetivo de evitar técnicas de detecção baseada em portas conhecidas, as utilizam com valores aleatórios para efetuar a comunicação entre os *peers*. Como existem 65.536 portas em uma máquina, tem-se uma probabilidade de aproximadamente 0,0015% de uma porta ser escolhida de forma aleatória, por consequência, temos aproximadamente 98.5% de chance de uma porta acima de 1024 (limite máximo das principais portas conhecidas) ser escolhida para ser utilizada, dando assim um valor médio elevado para esta característica em aplicações P2P.

- **Valor do desvio padrão do número das portas utilizadas:** devido a aleatoriedade na escolha da porta a ser utilizada pela aplicação o seu desvio padrão tende a ser elevado, segundo (ULLIAC, 2010). Se o valor das portas está distribuído entre todos os valores possíveis tal desvio deve atingir valores próximos a 30.000. Algumas aplicações P2P, utilizam portas escolhidas de forma sequencial para efetuar a conexão com outros *peers*, evitando assim este comportamento.

- **Uso do DNS:** em (ZHANG, 2011; SILVA, 2012a) requisições ao DNS foram utilizadas como um redutor do volume de tráfego a ser analisado. Isto é justificado pelo fato de que a comunicação P2P é feita diretamente através do endereço IP dos *peers* membros da rede, contidas nas *peers list*, o percentual de *peers* com domínios registrados em DNS é menor que 0,5% (normalmente apenas *super peers* ou servidores *web* que armazenam *peer list* utilizam registros em domínios), como mostrado em (SILVA, 2012a). Através da análise de *traces* de redes contendo aplicações P2P e aplicações não P2P, foi mostrado que é possível obter uma redução de volume superior a 25% do tráfego a ser analisado.

- **Percentual de conexões oriundas de uma mesma porta de origem:** em (SILVA, 2012a) foi mostrado que algumas aplicações P2P costumam utilizar uma única porta de origem para se conectar a outros *peers*. Uma máquina que executa uma aplicação P2P abre um grande número de conexões com outros membros da rede usando uma mesma porta de origem. Tal característica pode ser explorada para a detecção de portas usadas por aplicações P2P e pode ser utilizada para criar regras de bloqueio com o objetivo de impedir a comunicação entre nós com aplicações P2P.

- **Sequenciamento na escolha das portas:** segundo (LIN, 2009), muitas aplicações P2P tendem a escolher as portas para utilizar de modo sequencial, mesmo que inicialmente a porta selecionada tenha sido de forma aleatória. Uma vez escolhida a porta, as conexões seguintes serão feitas com portas de forma sequencial a partir desta primeira.

- **Tamanho médio dos pacotes:** em (LIN, 2009) é mostrado que diferentes aplicações possuem tamanhos de pacotes médios distintos, cada aplicação possui uma determinada frequência de tamanhos de pacotes que pode ser utilizada para sua classificação. Conforme mostrado anteriormente, aplicações P2P tendem a utilizar pacotes TCP para a transferência de arquivos e pacotes UDP apenas para a troca de mensagens de controle, o que indica a tendência da utilização de pacotes TCP de tamanho médio maior que pacotes UDP.

- **Percentual de conexões bem sucedidas:** *peers*, segundo (ZHOU, 2007; LIU, 2009b), apesar de possuírem as mesmas atribuições, muitas vezes possuem características distintas (largura de banda, sistema operacional, parâmetros da rede,...), sendo que qualquer alteração em sua configuração (alteração em regras do *firewall*, mudança de endereço IP, entre outras) pode alterar a conectividade deste com a rede P2P. Ademais, em sua maioria, não são servidores, permanecendo desligados ou sem conectividade em algum momento. Desta forma, para manter a rede P2P em funcionamento, um *peer* deve tentar se conectar com o máximo de nós possíveis a fim de obter um número mínimo de conexões bem sucedidas. Assim, nós P2P possuem uma taxa de conexões bem sucedidas inferior a de máquinas que só utilizam aplicações com a topologia Cliente / Servidor, que possuem servidores dedicados para estas aplicações.

- **Múltiplos fluxos entre nós:** operações que envolvam protocolos UDP / TCP possuem a escolha aleatória de pelo menos uma porta (de origem ou de destino), portanto, não é comum que fluxos distintos possuam um mesmo conjunto de atributos identificadores (IP_{src}, IP_{dst}, $Port_{src}$,$Port_{dst}$, Protocolo). Entretanto, isto acontece com aplicações P2P, já que ambos os *peers* dedicam uma porta para a troca de mensagens e transferência de dados. Esta é a base da heurística proposta em (MARCELL PERÉNYI, 2006): a existência de fluxos idênticos é um forte indício de comunicação entre aplicações P2P.

- **Tamanho do fluxo:** (MARCELL PERÉNYI, 2006) verificou que dados trafegados nas redes P2P costumam possuir tamanhos que variam de alguns *kilobytes* (no caso de arquivos de áudio ou pequenas aplicações, por exemplo) até milhares de *megabytes* (no caso de arquivo de vídeos, entre outros), o que pode demandar uma grande quantidade de pacotes para serem transferidos, gerando desta forma arquivos de fluxo com grandes quantidades de *bytes* trafegados.

- **Duração do fluxo:** segundo (MARCELL PERÉNYI, 2006) usuários de redes P2P são pacientes, transferências de arquivos nas redes P2P podem durar de alguns minutos até mesmo horas. Segundo esta heurística, os fluxos com duração superior a poucos minutos é um indicador do uso de aplicação P2P.

A Tabela 4.1, mostra os valores obtidos para as três redes que tiveram as características mensuradas, bem como indica quais módulos foram confirmados nesta medição.

	Valores			
Módulo	Rede Bittorrent	Rede Gnutella	Rede Doméstica	Característica Observada
Taxa de Envio TCP SYN / UDP	15 fluxos /seg	18 fluxos / seg	2 fluxos /seg	sim
Utilização do par de protocolos TCP /UDP	utiliza	não utiliza	não utiliza	não
Grau de entrada e saída de um nó	12 conexões de entrada e 4 de saída	4 conexões de entrada e 6 de saída	2 conexões de entrda e 0 e saída	sim
Percentual de pacotes TCP	22,4%	100%	93%	não
Valor médio do número das portas	21.027,66	6.411,90	753,43	sim
Desvio padrão do valor do número das portas	19.544,45	0	235,34	não
Percentual do uso do DNS	0,51%	0%	33,4%	sim
Percentual das conexões oriundas de uma mesma porta de origem	23%	2%	0,3%	não
Sequenciamento na escolha das portas	não possui	possui	não possui	não
Tamanho médio dos pacotes	UDP= 115,2 Bytes e TCP=600,15 Bytes	TCP=65 Bytes	UDP=430 Bytes e TCP=1.121 Bytes	não
Percentual de conexões bem sucedidas	74,3%	42,38%	89,4%	sim
Múltiplos fluxos entre nós	possui	possui	não possui	sim
Tamanho do fluxo	88.263,76 KB	2.431,11 KB	985,34 KB	sim
Duração do fluxo	304,72 segundos	11,83 segundos	6,4 segundos	não

TAB. 4.1: Valores obtidos para os módulos em estudo

Para a escolha dos módulos a serem adotados na fase, foram verificados, além da observância da característica nas redes mensuradas, os seguintes critérios:

- *A compatibilidade do módulo de detecção de aplicação P2P com as bots P2P:* como mencionado anteriormente, *bots* P2P são aplicações P2P, mas nem todas as características presentes nas redes P2P legítimas estão presentes em *botnets* P2P. Características como as relacionadas a transferência de grandes volumes de arquivos muitas vezes não são utilizadas por estas redes maliciosas, como exemplo, aponta-se o uso de protocolos UDP e TCP pela aplicação, *bot* P2P normalmente se restringem ao uso de apenas um destes, para a troca de mensagens.

- *A utilização de módulos que cubram os mais diversos protocolos P2P, bem como bots P2P:* diversas implementações das redes P2P são utilizadas, diferenças variam da política de escolha das portas para a comunicação à estrutura adotada pela rede (redes estruturadas ou não estruturadas). Desta forma, alguns módulos que detectam algumas implementações podem não detectar outras. Assim, é importante que estes sejam escolhidos de modo a encontrar os mais distintos protocolos possíveis, ou que se fundamentem em premissas básicas para o funcionamento de qualquer rede P2P.

- *Baixas taxas de falsos positivos:* apesar da arquitetura ser dividida em duas fases, o que possibilita que nós identificados erroneamente como detentores de aplicações P2P sejam descartados na segunda fase, baixos índices de falsos positivos são importantes para o bom desempenho da arquitetura.

- *Valores limites bem definidos:* os valores limites que distinguem aplicações P2P das não P2P devem ser bem definidos, valores muito próximos entre os dois grupos podem acarretar em falsos positivos e negativos.

Desta forma os seguintes módulos foram escolhidos para a implementação da primeira fase desta arquitetura:

- Envio de pacotes TCP SYN/ UDP;

- Percentual de conexões bem-sucedidas;

- Múltiplos fluxos entre nós;

- Valor médio da porta utilizada.

Opcionalmente, pode-se adotar na arquitetura um módulo redutor de volume, definido pela análise do uso do DNS, como explicado anteriormente. Para redes que trafegam grandes volumes de informação, este pode elevar a eficiência de toda a arquitetura eliminando previamente fluxos de redes não pertencentes a aplicações P2P.

Após a análise do fluxo de rede, todos os nós que possuírem características compatíveis com um ou mais módulos terão seus endereços IPs e características armazenadas em banco de dados, para que, por meio da árvore de decisão sejam escolhidos para formar a lista de nós possuidores de aplicações P2P e, posteriormente, serem analisadas na segunda fase da arquitetura.

O uso da árvore de decisão se fundamenta no fato de que nem todos os nós que possuem uma ou mais características buscadas necessariamente possuem aplicações P2P ativas. Para a execução da árvore foi utilizada a aplicação WEKA (OFWAIKATO, 2012) com a implementação J48 (implementação do algoritmo C4.5).

A Figura 4.2 mostra a arquitetura correspondente à fase 1.

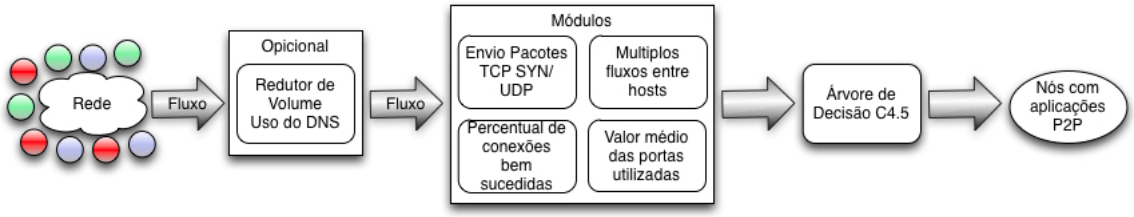

FIG. 4.2: Fase 1 da arquitetura proposta

4.3 FASE 2: DETECÇÃO DE BOTS P2P

Nesta fase é feita a distinção entre nós com aplicações P2P legítimas e nós com *bots* P2P, a separação fundamenta-se na principal diferença entre elas, a sua funcionalidade.

Aplicações P2P legítimas são utilizadas principalmente para disseminação de conteúdo entre os *peers* membros de suas redes. Por outro lado, *bots* P2P trafegam apenas informações simples como comandos enviados pelo *botmaster* para suas *bots*, ou *strings* contendo informações roubadas das máquinas onde estão instaladas (senhas, número de cartões de crédito, entre outras). Tais informações limitam-se a poucos *bytes* de tamanho e que muitas vezes conseguem ser transmitidas em poucos pacotes IP. Característica esta

que corrobora com o comportamento *stealthiness* inerente às *bots* que objetivam utilizar o mínimo de recursos das máquinas que habitam, evitando assim serem percebidas pelos usuários.

É nesta diferença que reside a principal forma para distinguir os dois tipos de rede. Devido ao maior tamanho dos dados trafegados nas redes P2P legítimas, estas possuem pacotes com maior tamanho médio, enquanto *bots* P2P tendem a utilizar pacotes com um tamanho médio muito inferior. (LIAO, 2010) mensurou em seu trabalho pacotes enviados por diversos protocolos, encontrando um valor para aplicações P2P superior a 1.300 *bytes*, entretanto *bots* P2P possuíam um tamanho inferior a 300 bytes.

Uma extensão desta característica é proposta neste trabalho para a diferenciação das redes: o número médio de pacotes por fluxo de dados. Aplicações P2P legítimas, por trafegarem grande volumes de dados, tendem a ter uma quantidade muito maior de pacotes por fluxo de dados que *bots* P2P.

Com o intuito de facilitar a visualização do tamanho médio dos pacotes e sua quantidade por fluxo, foram gerados gráficos (Figura 4.3) para estes dois atributos com sete conjuntos de *traces* de rede distintos, retirados dos repositórios CRAWDAD (CRA, 2012), OpenPacket (OPENPACKET, 2012), com a predominância das seguintes aplicações / protocolos:

- Aplicação P2P Bittorrent;
- aplicação P2P Gnutella;
- jogos online;
- protocolo HTTP;
- bot P2P Conficker;
- bot P2P Nugache;
- tráfego normal da internet (Internet's Traffic).

Como observado na Figura 4.3, as *bots* P2P (*Conficker* e *Nugache*) possuem um tamanho médio de pacote inferior ao de outras aplicações. Na medição feita, ambas apresentaram tamanho médio inferior a 100 *bytes*, enquanto aplicações P2P legítimas apresentaram um tamanho médio de pacote superior a 120 *bytes*. Porém, foi constatado que aplicações

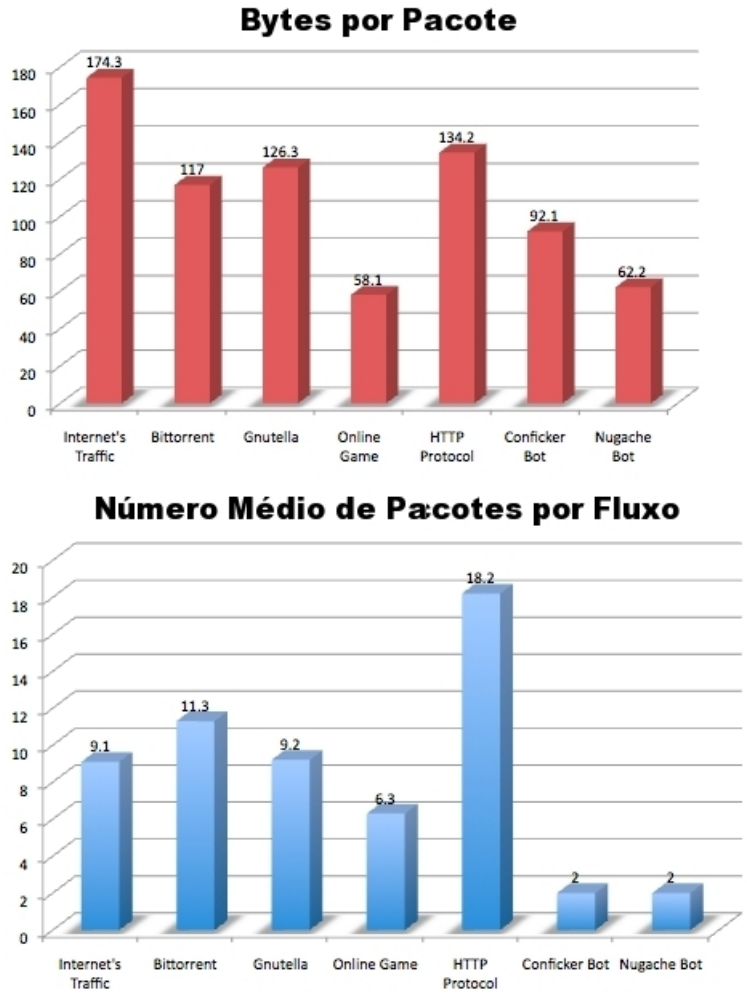

FIG. 4.3: Tamanho médio de pacotes (em vermelho) e número médio de pacotes por fluxo (em azul)

que envolvem jogos *online*, costumam apresentar tamanho médio de pacote inferior ao das *bots* P2P, como apresentado em (LIAO, 2010) e verificado neste artigo. Apesar do comportamento dos jogos *online* serem distintos do comportamento das aplicações P2P, o que os descartaria na primeira fase da arquitetura, uma eventual escolha de nós contendo este tipo de aplicação para serem analisados na fase 2 acarretaria em um aumento dos índices de falsos positivos, caso este parâmetro fosse adotado para a corrente fase.

Desta forma, com o objetivo de diminuir as chances da ocorrência deste tipo de falso positivo, para a fase 2 é proposta uma extensão da característica acima: o número médio de pacotes por fluxo. Como observado na Figura 4.3, o número médio de pacotes por fluxo das *bots* P2P é inferior ao de qualquer outra aplicação, inclusive as que envolvem jogos *online*, pois além das *bots* possuírem um tráfego de rede bastante inferior ao das outras

aplicações, estas almejam permanecer ao máximo sem serem detectadas nas máquinas hospedeiras, reduzindo o número de pacotes trocados entre os *peers*, característica denominada *stealthiness*, que os jogos *online* não possuem.

Outro fator que corrobora para a utilização do número médio de pacotes por fluxo em detrimento do tamanho médio dos pacotes é a facilidade para se estipular um valor limite entre aplicações P2P legítimas e as *bots*. Neste caso, a determinação de um valor limite para a distinção de *bots* P2P das demais aplicações se torna uma tarefa mais fácil, podendo ser determinado com uma boa margem de segurança a fim de evitar potenciais erros. Para esta proposta é adotado como fator de corte a distância média entre o maior tamanho encontrado para o pacote oriundo de uma *bot* e o menor tamanho encontrado para pacotes de demais aplicações, ou seja, para o gráfico da Figura 4.3 o valor será 4.

Sendo assim, na fase 2 a distinção é implementada pelo número médio de pacotes por fluxo, onde valores superiores a 4 são considerados de redes P2P legítimas.

Após a análise dos filtros adotados na fase 1 e na fase 2, é possível verificar que o tempo de execução da metodologia é limitado pelo tempo de captura do *trace* da rede a ser analisada, como será verificado durante a validação da arquitetura. Fato este que ratifica a classificação da arquitetura como *online*.

Na Figura 4.4 é exposto um esquema completo da arquitetura proposta para a detecção *online* de *bots* P2P.

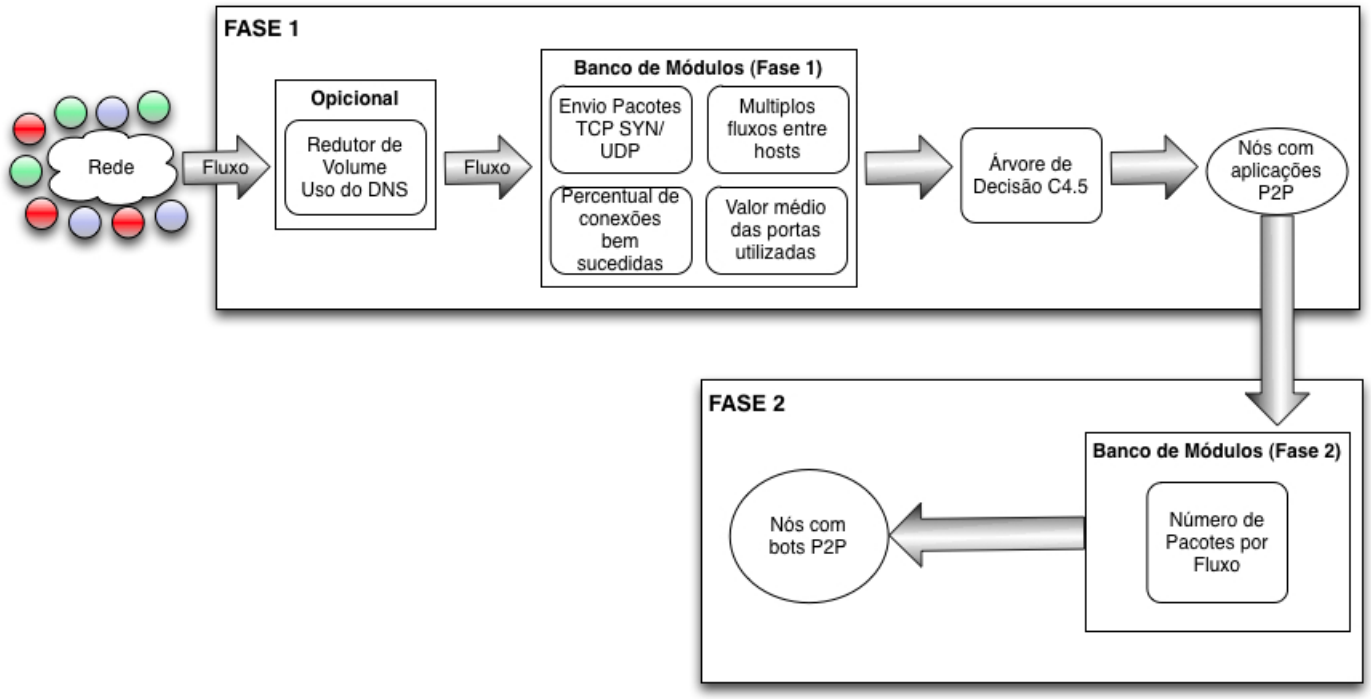

FIG. 4.4: Arquitetura proposta completa

5 VALIDAÇÃO DA ARQUITETURA

Para a realização de testes com a arquitetura proposta, dois cenários foram montados a partir de *traces* obtidos dos repositórios *CRAWDAD* (CRA, 2012), *OpenPacket* (OPENPACKET, 2012) e *DARPA* (DARPA, 1999), além de traces gerados artificialmente pela aplicação *Rubot* (LEE, 2009).

O *Rubot* (em Anexo), consiste de um *framework* para a emulação e análise de *botnets*. Desenvolvido na linguagem *Ruby*, permite a rápida construção de *botnets* integrando diferentes funcionalidades que se deseja estudar. Sua principal vantagem, diante das demais aplicações voltadas para o estudos destes *malwares*, é a possibilidade de se analisar todos os aspectos da *bot*, desde o seu *payload* (assinaturas, padrões de criptografia) até os *traces* de rede gerados por sua comunicação com outros membros, já que permite a emulação da *bot* em estudo.

Para este trabalho foram utilizados módulos prontos do *Rubot* que emulam a *botnet Nugache*. Adotando seu *framework* foi construído uma *botnet* contendo quinze membros. Apesar de sua complexidade, o *Rubot* não emula a fase de *bootstrap*, o que inviabiliza o uso de um dos módulos adotados na arquitetura proposta, que busca por esta característica. Como a ausência desta fase na *botnet Nugache* é irreal, foram inseridos artificialmente em seus *traces* pacotes que a simulam, retirados de *traces* reais da *bot Nugache*, permitindo assim uma maior veracidade entre o *trace* gerado pelo *Rubot* e o real.

5.1 LABORATÓRIO PARA TESTES

Com o objetivo de auxiliar e padronizar a criação dos cenários e na sua posterior validação, foi elaborado um laboratório para testes.

Para a geração de *traces* com o *Rubot* foi montada uma rede composta de 16 máquinas virtuais para simular uma *botnet Nugache* (1 *botmaster* e 15 *bots Nugache*). Para tal, foi escrito um *shell script* com o conjunto de comandos e parâmetros necessários para a ativação da rede maliciosa simulada. Após, uma sequência de comandos na linguagem *Ruby* foram enviados pelo *botmaster* para serem executados pelos *bots* e repassados aos vizinhos. A Figura 5.1 mostra a configuração gerada pelo *Rubot*.

Com o objetivo de obter os *traces* oriundos das máquinas virtuais utilizadas pelo *Rubot*,

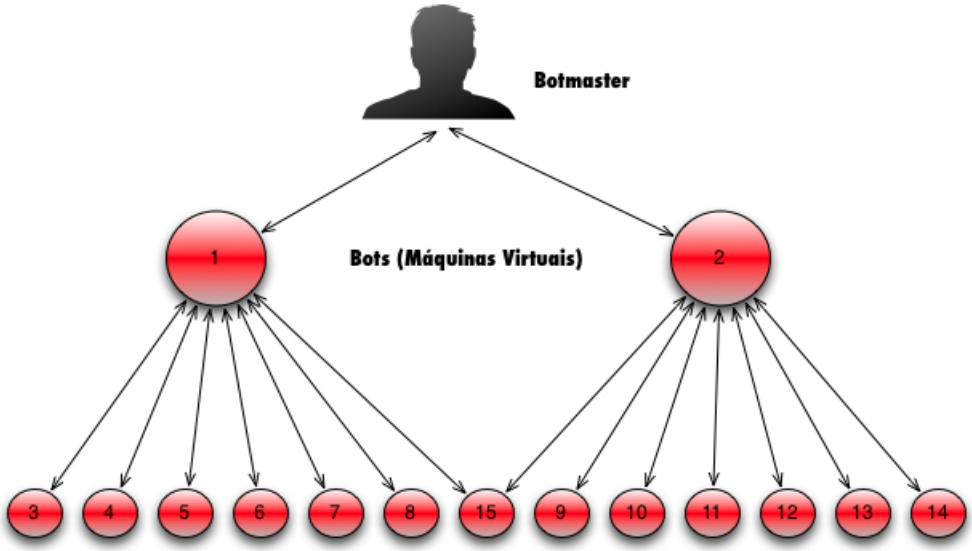

FIG. 5.1: Topologia configurada no *Rubot* para a *botnet Nugache*

instâncias da aplicação *TCPDUMP* (TCPDUMP, 2012) foram instaladas para monitorar cada uma das 16 máquinas virtuais e gerar os arquivos contendo os respectivos *traces*.

Os *traces* obtidos dos repositórios e do *Rubot* são reproduzidos por instâncias da aplicação TCPREPLAY (TCPREPLAY, 2012), em velocidade previamente definida, para uma porta que é monitorada pelo ARGUS (*Audit Record Generation and Utilization System*) (QOSIENT, 2012), em anexo.

A utilização do TCPREPLAY possibilitou que a velocidade do fluxo de dados reproduzido dos arquivos de *trace* pudesse ser acelerada ou reduzida conforme a necessidade do teste a ser executado. Além disso, permitiu a geração de uma nova temporização para o *trace* resultante ao ter diversas instâncias suas sendo executadas em paralelo com o fluxo sendo direcionado para uma determinada porta monitorada pelo ARGUS, como mostrado na Figura 5.2. Aplicações como TCPREWRITE (TCPREPLAY, 2012) e PCAPJOINER(PCAPJOINER, 2012), apesar de permitirem a junção de diversos arquivos de *traces*, não possibilitam uma nova temporização para o arquivo resultante, mantendo a dos arquivos originais.

O programa ARGUS foi utilizado por permitir a conversão de arquivos de *traces* em arquivos de fluxo, o que possibilita uma redução de até 50% em espaço para armazenamento, como foi observado neste trabalho. Ademais, o ARGUS permite que todo o fluxo de rede seja exportado para um banco de dados MYSQL (MYSQL, 2012) através da aplicação RA_SQLINSERT com os atributos que se deseja estudar. A Figura 5.2 mostra

a estrutura do laboratório utilizado.

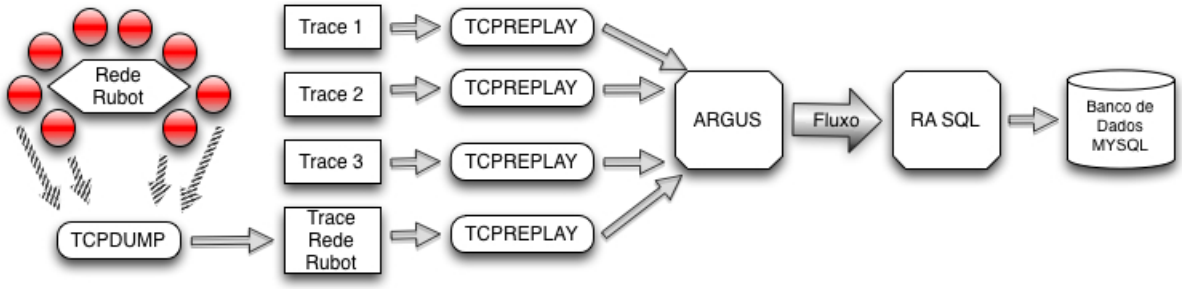

FIG. 5.2: Laboratório de testes

Os atributos importados para a base de dados em cada fluxo são: *timestamp*, duração do fluxo, endereços IP e portas de origem e destino, número de pacotes enviados pela origem e pelo destino, quantidade de bytes enviados pela origem e pelo destino e o estado da conexão.

Todos os módulos adotados na arquitetura proposta foram traduzidos para a linguagem SQL e testados no banco de dados a fim de obter valores relativos a sua acurácia, e tempo de processamento.

5.2 CENÁRIOS UTILIZADOS

Com a utilização do laboratório de testes, foram elaborados dois cenários distintos (C_1 e C_2) para a validação da arquitetura, como mostrado na Tabela 5.1.

O cenário C_1 contém a *bot* P2P *nugache* gerada pelo *Rubot*, bem como o *trace* de duas *bots* reais retiradas do repositório *Open Packet*. É também constituído de *traces* de uma rede com nós executando a aplicação *Bittorrent* e traces da rede *DARPA* que foram gerados sinteticamente e estão isentos de qualquer atividade relacionados a redes P2P.

O cenário C_2 não possui nenhuma *bot* P2P em seus *traces*, é constituído de traces de redes que executam a aplicação P2P *Gnutella* e *Bittorrent* e traces isentos de qualquer aplicação P2P oriundos do DARPA e de uma rede doméstica livre de aplicações P2P.

Ambos os cenários possuem uma duração de quarenta minutos.

CENÁRIO	TRACES
C_1	bot Nugache + Bittorrent + tráfego DARPA
C_2	Gnutella + Bittorrent + tráfego DARPA + rede doméstica

TAB. 5.1: Constituição dos cenários.

5.3 GERAÇÃO DA ÁRVORE DE DECISÃO

Antes da execução da validação da arquitetura nos cenários gerados, uma amostra contendo dez minutos de fluxo foi retirada do cenário C_1 para ser utilizada como treinamento com a finalidade de gerar a árvore de decisão da fase 1, para isto foi utilizada uma implementação do algoritmo C4.5 denominado J48, contido na aplicação WEKA. A Figura 5.3 mostra a árvore gerada e a Tabela 5.2 mostra o percentual de nós classificados de modo correto bem como as taxas de falsos positivos obtidas.

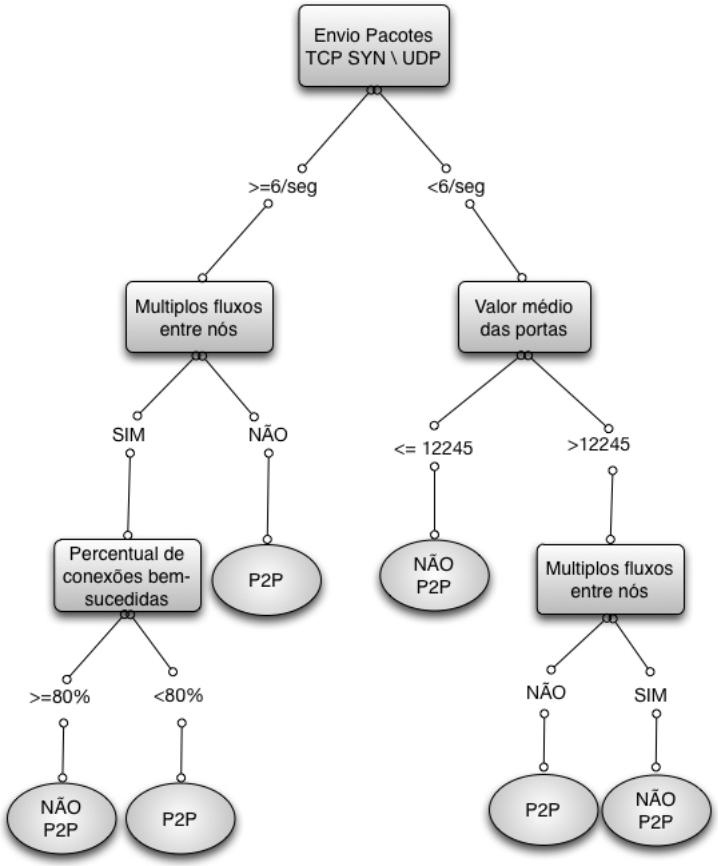

FIG. 5.3: Árvore de decisão gerada pelo WEKA

Percentual de nós classificados corretamente	95,58%
Percentual de nós não P2P classificados erroneamente	5,3%
Percentual de nós P2P classificados erroneamente	3,3%

TAB. 5.2: Nós classificados corretamente e percentuais de falsos positivos.

A Figura 5.4 mostra como a árvore executou a classificação de cada nó utilizado no conjunto de teste para os módulos adotados na fase 1. Nós representados pela cor azul são os possuidores da aplicações P2P, enquanto os representados pela cor vermelha não possuem aplicações P2P, o símbolo **X** indica os nós classificados corretamente.

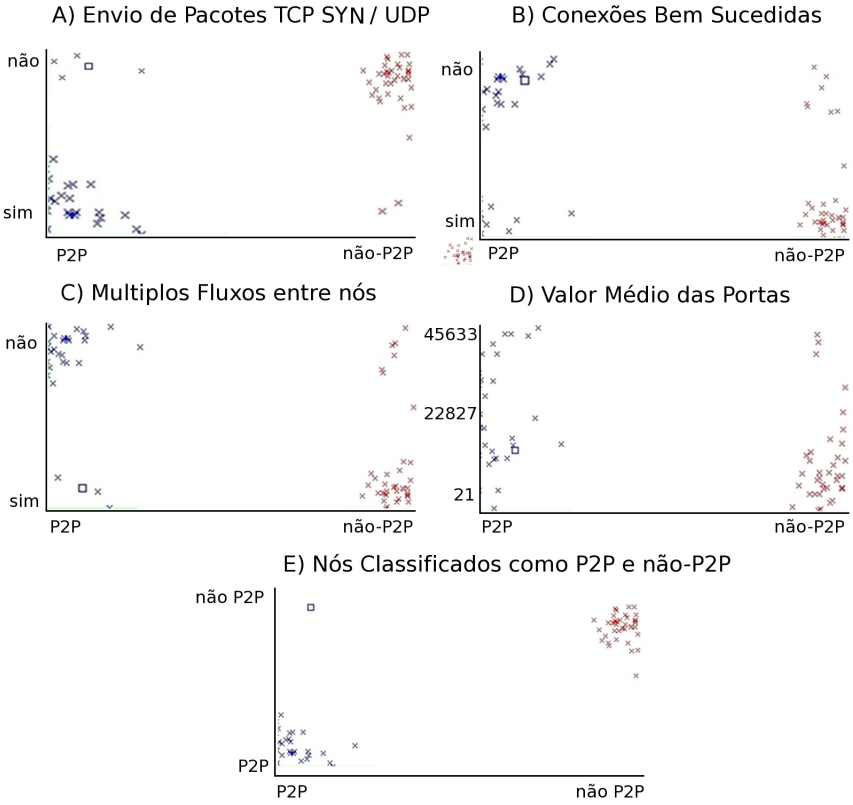

FIG. 5.4: Gráficos com a classificação dos nós da amostra de teste, para cada um dos módulos.

5.4 ANÁLISE DO CENÁRIO C_1

O cenário C_1, ao ser analisado na fase 1 da arquitetura, apresentou uma acurácia de 89,79% e percentual de falso positivo para os nós contendo aplicações P2P e não contendo tais aplicações de 10,3% e 10%, respectivamente (Tabela 5.3). Em números absolutos é possível observar que, entre os nós com aplicações P2P ativas, apenas 1 foi classificado

erroneamente e, entre os nós sem aplicações P2P ativas, 5 foram classificados de forma errada.

Através da análise de cada módulo empregado na fase 1 individualmente (Figura 5.5), é possível perceber que o nó classificado erroneamente apresentou o módulo que analisa a taxa de envio de pacotes TCP SYN / UDP classificado de forma adversa ao que as *bots* P2P possuem e valor de porta abaixo de 1024.

Através de uma observação mais detalhada dos *traces* adotados no cenário C_1 é possível observar que ambos os módulos falharam devido ao *trace* da *bot Nugache* real utilizado não possuir o procedimento de *bootstrap* e utilizar a porta 8 para sua comunicação, porta esta somente utilizada em implementações antigas desta *bot*.

Percentual de nós classificados corretamente	89,79%
Percentual de nós não P2P classificados erroneamente	10,3%
Percentual de nós P2P classificados erroneamente	10%

TAB. 5.3: Nós classificados corretamente e percentuais de falsos positivos.

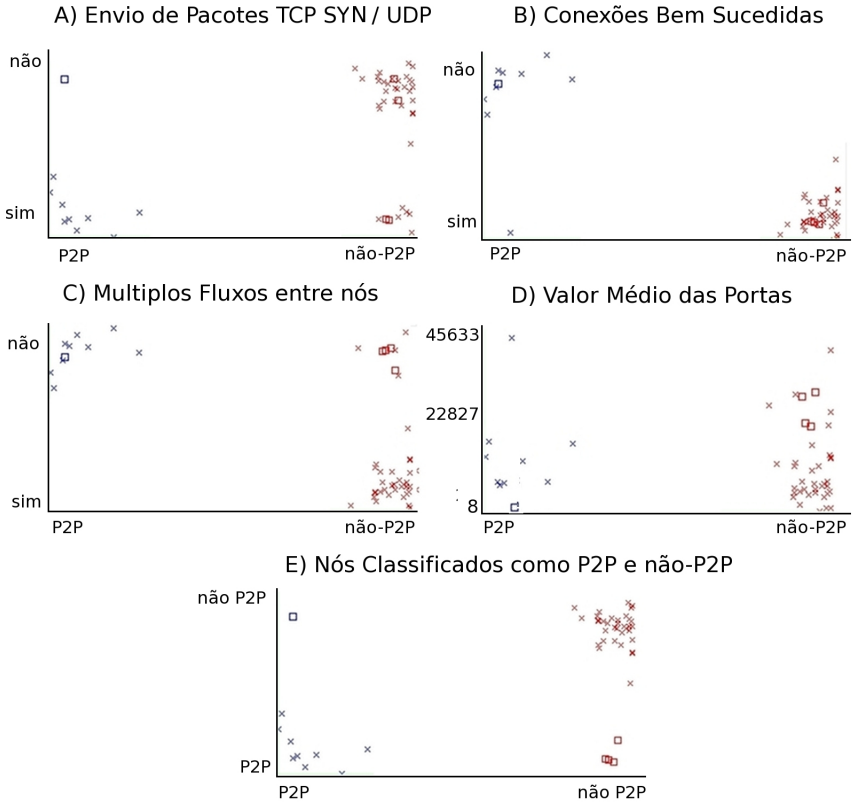

FIG. 5.5: Gráficos com a classificação dos nós do cenário C_1, para cada um dos módulos.

Após a análise executada pela fase 1, os nós identificados como aplicações P2P tiveram

o número médio de pacotes por fluxo extraídos para serem analisados na fase seguinte. A Tabela 5.4 indica os valores obtidos nesta fase.

Nós classificados como possuídores de aplicações P2P Ativas pela fase 1			
	Bots P2P	**Outras App P2P**	**App não P2P**
Número de nós	5	4	5
Número médio de pacotes por fluxo	2,1	67,17	14,13
Quantidade de nós classificados erroneamente	0	2	2
Número médio de pacotes por fluxo dos nós classificados erroneamente	-	1	2,3

TAB. 5.4: Valores obtidos após analise pela fase 2

Todos os nós possuidores de *bots* P2P analisados pela fase 2 foram classificados corretamente. Porém, entre os nós com aplicações P2P legítimas, 2 por possuírem número médio de pacotes por fluxo inferior a 4, foram considerados como *bots* P2P. Entre os nós não possuidores de aplicações P2P, situação semelhante foi observada; 4 nós foram erroneamente classificados como possuidores de *bots* P2P.

Desta forma, das 6 *bots* P2P existentes no cenário C_1, apenas 1 não foi detectada por ter sido erroneamente classificada durante a fase 1. E para os 43 nós não possuidores de *bots* P2P (nós com aplicações P2P legítimas e sem aplicações P2P), apenas 4 foram mal classificados (Tabela 5.5). Assim, encontramos para a arquitetura um percentual de falsos positivos de 9,3% e de falsos negativos de 16,7% (Tabela 5.5). Utilizando a métrica formulada por (POWERS, 2007), obtêm-se o valor de 0,56 para a precisão e 0,83 para *recall*, através da média harmônica destes dois parâmetros é obtido o F_1 *score* [1] no valor de 0,67. O tempo de processamento gasto para as fases 1 e 2 foi de 5,43 segundos.

RESULTADO PARA O CENÁRIO 1			
	Quantidade	**Classificadas Erroneamente**	**Percentual de Erros (%)**
Bots P2P	6	1	16,7
Outras Aplicações	43	4	9,3

TAB. 5.5: Valores obtidos para o Cenário C_1

[1] medida da acurácia dos testes

5.5 ANÁLISE DO CENÁRIO C_2

O cenário C_2 foi gerado sem que houvesse nenhum nó *bot* P2P presente, apenas aplicações P2P legítimas e nós sem nenhum tipo de aplicação deste gênero. Após a análise pela fase 1 foi obtido um percentual de acerto de 82,35% e uma taxa de falsos positivos para nós com aplicações P2P e nós sem este tipo de aplicação de 20% e 0%, respectivamente (Tabela 5.6).

A análise do desempenho de cada um dos módulos empregados na fase 1 deste cenário (Figura 5.6), mostra que todos os nós com aplicações P2P foram classificados corretamente, apesar de 1 nó referente a aplicação *Gnutella* apresentar um valor baixo para a porta de comunicação.

Entretanto, nós isentos de aplicações P2P tiveram um percentual elevado de erros em sua classificação. Analisando de forma detalhada os *traces* adotados neste cenário, observa-se a utilização, por parte de aplicações diversas não P2P, portas com valores elevados para a comunicação, que contribuíram para a classificação errônea.

Percentual de nós classificados corretamente	82,35%
Percentual de nós não P2P classificados erroneamente	20,3%
Percentual de nós P2P classificados erroneamente	0%

TAB. 5.6: Nós classificados corretamente e percentuais de falsos positivos.

Após a análise executada pela fase 1, os nós identificados como aplicações P2P tiveram o número médio de pacotes por fluxo extraídos para serem analisados na fase seguinte. A Tabela 5.7 indica os valores obtidos nesta fase.

Nós classificados como possuídores de aplicações P2P Ativas pela fase 1			
	Bots P2P	**Outras App P2P**	**App não P2P**
Número de nós	0	6	9
Número médio de pacotes por fluxo	-	19,82	21
Quantidade de nós classificados erroneamente	-	1	2
Número médio de pacotes por fluxo dos nós classificados erroneamente	-	3,1	3,1

TAB. 5.7: Valores obtidos após analise pela fase 2

No cenário C_2 não existem *bots* P2P somente nós com aplicações P2P legítimas (6 nós),

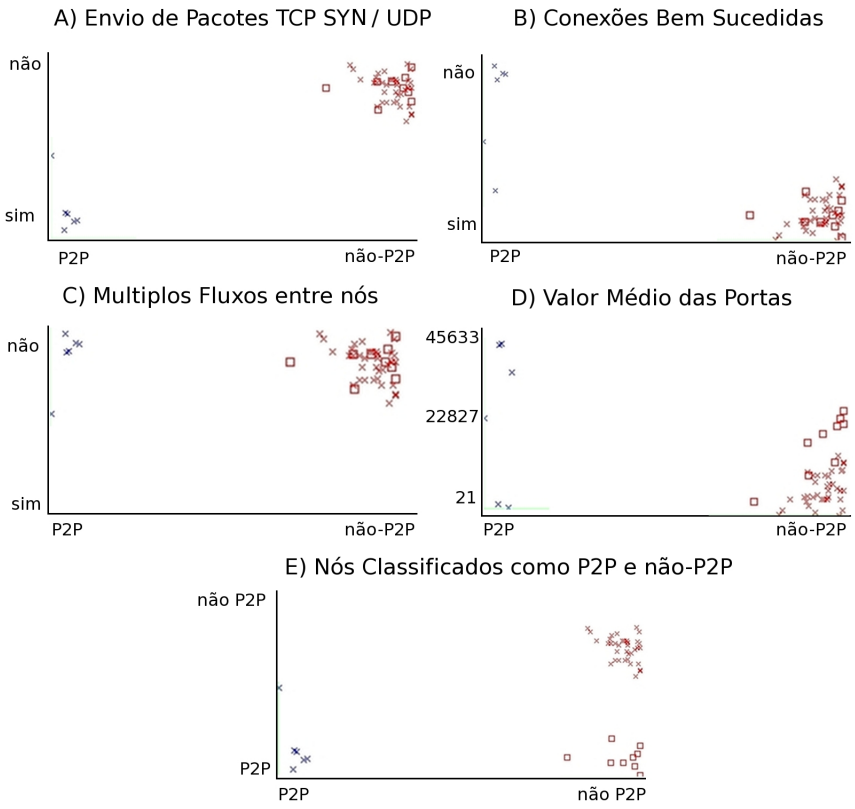

FIG. 5.6: Gráficos com a classificação dos nós do cenário C_2, para cada um dos módulos.

dentre os quais 1 foi classificado de forma errada, como *bot* P2P, e nós sem aplicações P2P (45 nós), dentre os quais 9 foram considerados possuidores de aplicações P2P na primeira fase e 2 foram classificados como *bots* P2P na segunda. Desta forma, para este cenário foi encontrado um percentual de falsos positivos de 5,9% (Tabela 5.8). O tempo de processamento gasto para a execução das fases 1 e 2 foi de 6,89 segundos.

RESULTADO PARA O CENÁRIO 2			
	Quantidade	**Classificadas Erroneamente**	**Percentual de Erros (%)**
Bots P2P	0	-	-
Outras Aplicações	51	3	5,9

TAB. 5.8: Valores obtidos para o Cenário C_2

6 CONSIDERAÇÕES FINAIS

Como plataforma para crimes cibernéticos, *botnets* são a principal ameaça à segurança na internet. Suas atividades ilícitas geram anualmente bilhões de dólares de prejuízo à economia mundial e despertam o interesse de pesquisadores que têm direcionado seus esforços para a elaboração de técnicas de detecção e neutralização destas redes.

Neste trabalho foi apresentado uma proposta de arquitetura para a detecção de *bots* P2P de modo *online*, ou seja, capaz de retornar com um elevado grau de acurácia máquinas infectadas com este tipo de *malware* sem que para isso seja necessário executar longas análises nos históricos de rede, que demandam tempo e geram resultados que muitas vezes não correspondem à situação atual da rede.

O modelo proposto é dividido em duas fases, buscando assim, uma otimização dos recursos. Na primeira fase, módulos de rápida execução, que não necessitam analisar longos históricos de redes, aliados a uma árvore de decisão são utilizados com o intuito de diferenciar aplicações P2P das demais, para que, posteriormente, em uma segunda fase, seja feita a distinção entre as aplicações P2P legítimas das *bots* P2P através da análise do número médio de pacotes por fluxo de rede.

Com o objetivo de justificar e implementar a primeira fase da arquitetura, foi feito um estudo do funcionamento dos protocolos P2P e como as *bots* P2P os utilizam em proveito de suas redes maliciosas, bem como um levantamento das mais distintas técnicas utilizadas para a sua detecção de modo a empregá-las para descobrir não apenas as redes P2P legítimas mas também as *bots*.

Para a implementação da segunda fase, foram observados os fins a que se destinam as redes P2P, legítimas ou não. Através de uma análise empírica foi mostrado que as redes P2P legítimas tendem a gerar maior número de pacotes por fluxo que *bots* P2P, pois essas tendem a trafegar grandes quantidades de informações enquanto *bots* P2P limitam-se a simples trocas de mensagens.

Visando uma validação a mais eficiente possível, um laboratório de testes foi elaborado e nele dois cenários foram gerados através da junção de diversos *traces* de rede, com características distintas. No primeiro cenário, *traces* da *bot* P2P *Nugache* foram implantados juntamente com traces de aplicações P2P legítimas. Para esse cenário, índices de

falsos positivos e negativos de 9,3% e 16,7%, respectivamente, foram obtidos e um valor de 0,67 para o F_1 *score*. No cenário 2, somente aplicações P2P legítimas foram implantadas, gerando um índice de falsos positivos de 5,9%.

Através de comparações com outros trabalhos, mencionados em capítulos anteriores, que utilizam técnicas *offline*, é possível notar que apesar da arquitetura proposta possuir taxas de acurácia e falsos positivos pouco inferiores a estes, possui um tempo de resposta muito superior, tendo como fator limitante para a duração do processamento, o intervalo de tempo de histórico de rede desejado para a análise (no caso deste trabalho 40 minutos).

Além disso, sua arquitetura modular é dividida em duas fases, o que permite uma melhor adequação em redes com características as mais distintas possíveis, bastando para isso um estudo prévio destas, para assim obter um conjunto de módulos ótimos. É possível, ainda, a inserção de módulos com o objetivo de detectar novas *bots* P2P ou mesmo *bots* que se utilizem de outros protocolos para sua comunicação. Além disso, com a remoção da fase 2 da arquitetura, há a possibilidade de se detectar qualquer tipo de aplicação P2P ativa na rede.

Outro diferencial proposto na arquitetura, em relação aos demais trabalhos com objetivos semelhantes, é o seu tempo de execução bem inferior. Porém, com o objetivo de diminuir, ainda mais, este tempo e otimizar os valores de acurácia e falsos positivos encontrados, é necessário um estudo da variação da eficiência pelo tempo de histórico de rede em análise. Estudos semelhantes já foram feitos por (KARAGIANNIS, 2004b) para a detecção de aplicações P2P, mas para *bots* P2P permanecem inédito.

Ademais, uma análise da arquitetura em redes maiores e com outros tipos de *bots* P2P se faz necessário visando verificar sua escalabilidade e obter valores de falsos positivos e negativos mais apurados.

7 REFERÊNCIAS BIBLIOGRÁFICAS

A community resource for archiving wireless data at dartmouth. http://crawdad.cs.dartmouth.edu/, 2012. URL `http://crawdad.cs.dartmouth.edu`.

ACCELERATORS, M. S. **Malware removal starter kit: How to combat malware using windows**. Technical report, 2007.

AVIV, A. J. e HAEBERLEN, A. **Challenges in experimenting with botnet detection systems**. Em ***Proceedings of the 4th USENIX Workshop on Cyber Security Experimentation and Test (CSET'11)***, 2011.

BARAKAT, A. e KHATTAB, S. **A comparative study of traditional botnets versus super-botnets**. Em ***Informatics and Systems (INFOS), 2010 The 7th International Conference on***, págs. 1 –5, march 2010.

BISNIK, N. e ABOUZEID, A. **Modeling and analysis of random walk search algorithms in p2p networks**. Em ***Hot Topics in Peer-to-Peer Systems, 2005. HOT-P2P 2005. Second International Workshop on***, págs. 95 – 103, july 2005.

BUSH, R. **Fidonet: technology, tools, and history**. *Commun. ACM*, 36:31–35, August 1993. ISSN 0001-0782. URL `http://doi.acm.org/10.1145/163381.163383`.

CHANG, S. e DANIELS, T. E. **P2P botnet detection using behavior clustering & statistical tests**. Em ***Proceedings of the 2nd ACM workshop on Security and artificial intelligence***, AISec '09, págs. 23–30, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-781-3. URL `http://doi.acm.org/10.1145/1654988.1654996`.

CHEN, F., WANG, M., FU, Y. e ZENG, J. **New detection of peer-to-peer controlled bots on the host**. Em ***Wireless Communications, Networking and Mobile Computing, 2009. WiCom '09. 5th International Conference on***, págs. 1 –4, sept. 2009a.

CHEN, H., HU, Z., YE, Z. e LIU, W. **A new model for p2p traffic identification based on dpi and dfi**. Em ***Information Engineering and Computer Science, 2009. ICIECS 2009. International Conference on***, págs. 1 –3, dec. 2009b.

CHEN, J.-B. **A method of identifying the p2p file sharing**. Em ***International Journal of Computer Science and Network Security,***, nov 2010.

CHOI, H., LEE, H., LEE, H. e KIM, H. **Botnet detection by monitoring group activities in dns traffic**. Em ***Computer and Information Technology, 2007. CIT 2007. 7th IEEE International Conference on***, págs. 715 –720, oct. 2007.

CHOI, T., KIM, C., YOON, S., PARK, J., LEE, B., KIM, H., CHUNG, H. e JEONG, T. **Content-aware internet application traffic measurement and analysis**. Em ***Network Operations and Management Symposium, 2004. NOMS 2004. IEEE/IFIP***, volume 1, págs. 511 –524 Vol.1, april 2004.

CLULEY, G. **India becomes the king of the spammers, stealing america's crown**, 2012. URL `http://nakedsecurity.sophos.com/2012/04/23/india-becomes-the-king-of-the-spammers-stealing-americas-crown`.

CONSTANTINOU, F. e MAVROMMATIS, P. **Identifying known and unknown peer-to-peer traffic**. Em ***Network Computing and Applications, 2006. NCA 2006. Fifth IEEE International Symposium on***, págs. 93 –102, july 2006.

COOKE, E., JAHANIAN, F. e MCPHERSON, D. **The zombie roundup: Understanding, detecting, and disrupting botnets**. págs. 39–44, 2005.

COSKUN, B., DIETRICH, S. e MEMON, N. **Friends of an enemy: identifying local members of peer-to-peer botnets using mutual contacts**. Em ***Proceedings of the 26th Annual Computer Security Applications Conference***, ACSAC 10, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0133-6. URL `http://doi.acm.org/10.1145/1920261.1920283`.

DAGON, D., GU, G., LEE, C. P. e LEE, W. **A taxonomy of botnet structures**. Em ***In Proc. of the 23 Annual Computer Security Applications Conference (ACSAC'07***, 2007.

DANCHEV, D. **Research: Small diy botnets prevalent in enterprise networks**, 2009. URL `http://www.zdnet.com/blog/security/research-small-diy-botnets-prevalent-in-enterprise-networks/4485`.

DARPA. **Intrusion detection evaluation**, 1999. URL `http://www.ll.mit.edu`.

DAVIS, C., FERNANDEZ, J., NEVILLE, S. e MCHUGH, J. **Sybil attacks as a mitigation strategy against the storm botnet**. Em ***Malicious and Unwanted Software, 2008. MALWARE 2008. 3rd International Conference on***, págs. 32 –40, outubro 2008.

DOUCEUR, J. R. **The sybil attack**. Em ***Revised Papers from the First International Workshop on Peer-to-Peer Systems***, IPTPS '01, págs. 251–260, London, UK, UK, 2002. Springer-Verlag. ISBN 3-540-44179-4. URL `http://dl.acm.org/citation.cfm?id=646334.687813`.

ERMAN, J., MAHANTI, A., ARLITT, M. e WILLIAMSON, C. **Identifying and discriminating between web and peer-to-peer traffic in the network core**. Em ***Proceedings of the 16th international conference on World Wide Web***, WWW '07, págs. 883–892, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-654-7. URL `http://doi.acm.org/10.1145/1242572.1242692`.

FEILY, M., SHAHRESTANI, A. e RAMADASS, S. **A survey of botnet and botnet detection**. Em ***Proceedings of the 2009 Third International Conference on Emerging Security Information, Systems and Technologies***, SECURWARE '09, págs. 268–273, Washington, DC, USA, 2009. IEEE Computer Society. ISBN 978-0-7695-3668-2. URL http://dx.doi.org/10.1109/SECURWARE.2009.48.

FSECURE. **Calculating the size of the downadup outbreak**, 2009. URL http://www.f-secure.com/weblog/archives/00001584.html.

GLOBO, J. **Crime cibernético gera prejuízos de quase r$ 16 bilhões no brasil**, 2012. URL http://g1.globo.com/tecnologia/noticia/2012/10/crime-cibernetico-gera-prejuizos-de-quase-r-16-bilhoes-no-brasil.html.

GOODIN, D. **Waledac botnet decimated by ms takedown**, 2010. URL http://www.theregister.co.uk/2010/03/16/waledac_takedown_success/.

HOLZ, T., STEINER, M., DAHL, F., BIERSACK, E. e FREILING, F. **Measurements and mitigation of peer-to-peer-based botnets: a case study on storm worm**. Em ***Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats***, Berkeley, CA, USA, 2008. USENIX Association. URL http://portal.acm.org/citation.cfm?id=1387709.1387718.

HONG, S.-H. **Measuring the effect of napster on recorded music sales: Difference-in-differences estimates under compositional changes**. *Journal of Applied Econometrics*, págs. n/a–n/a, 2011. ISSN 1099-1255. URL http://dx.doi.org/10.1002/jae.1269.

KARAGIANNIS, T., BROIDO, A., BROWNLEE, N., CLAFFY, K. e FALOUTSOS, M. **Is p2p dying or just hiding? [p2p traffic measurement]**. Em ***Global Telecommunications Conference, 2004. GLOBECOM '04. IEEE***, volume 3, págs. 1532 – 1538 Vol.3, nov.-3 dec. 2004a.

KARAGIANNIS, T., BROIDO, A., FALOUTSOS, M. e CLAFFY, K. **Transport layer identification of p2p traffic**. Em ***Proceedings of the 4th ACM SIGCOMM conference on Internet measurement***, IMC '04, págs. 121–134, New York, NY, USA, 2004b. ACM. ISBN 1-58113-821-0. URL http://doi.acm.org/10.1145/1028788.1028804.

LEE, C. P. ***Framework for botnet emulation and analysis***. Tese de Doutorado, Atlanta, GA, USA, 2009. AAI3364236.

LEVINE, B. N., SHIELDS, C. e MARGOLIN, B. **A survey of solutions to the sybil attack**. Technical report, 2006. URL http://prisms.cs.umass.edu/brian/pubs/levine.sybil.tr.2006.pdf.

LIANG, J., NAOUMOV, N. e ROSS, K. W. **The index poisoning attack in p2p file sharing systems**. Em ***INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings***, págs. 1 –12, april 2006.

LIAO, W.-H. e CHANG, C.-C. **Peer to peer botnet detection using data mining scheme**. Em ***Internet Technology and Applications, 2010 International Conference on***, págs. 1 –4, aug. 2010.

LIN, Y.-D., LU, C.-N., LAI, Y.-C., PENG, W.-H. e LIN, P.-C. **Application classification using packet size distribution and port association**. *Journal of Network and Computer Applications*, 32(5):1023 – 1030, 2009. ISSN 1084-8045. URL http://www.sciencedirect.com/science/article/pii/S1084804509000484.

LIU, D., LI, Y., HU, Y. e LIANG, Z. **A p2p-botnet detection model and algorithms based on network streams analysis**. Em ***Future Information Technology and Management Engineering (FITME), 2010 International Conference on***, volume 1, págs. 55 –58, oct. 2010.

LIU, F., LI, Z. e NIE, Q. **A new method of p2p traffic identification based on support vector machine at the host level**. Em ***Proceedings of the 2009 International Conference on Information Technology and Computer Science - Volume 02***, ITCS '09, págs. 579–582, Washington, DC, USA, 2009a. IEEE Computer Society. ISBN 978-0-7695-3688-0-02. URL http://dx.doi.org/10.1109/ITCS.2009.257.

LIU, F., LI, Z. e NIE, Q. **A new method of p2p traffic identification based on support vector machine at the host level**. Em ***Proceedings of the 2009 International Conference on Information Technology and Computer Science - Volume 02***, ITCS '09, págs. 579–582, Washington, DC, USA, 2009b. IEEE Computer Society. ISBN 978-0-7695-3688-0-02. URL http://dx.doi.org/10.1109/ITCS.2009.257.

LOCHER, T., MYSICKA, D., SCHMID, S. e WATTENHOFER, R. **A peer activity study in edonkey kad**, 1995.

MARCELL PERÉNYI, TRANG DINH DANG, A. G. e MOLNÁR, S. **Identification and analysis of peer-to-peer traffic**. Em ***Journal of Communications***, nov 2006.

MAZZARIELLO, C. **Irc traffic analysis for botnet detection**. Em ***Information Assurance and Security, 2008. ISIAS '08. Fourth International Conference on***, págs. 318 –323, sept. 2008.

MILLER, C. **The rustock botnet spams again**, 2008. URL http://www.scmagazine.com/the-rustock-botnet-spams-again/article/112940/.

MOORE, A. W. e PAPAGIANNAKI, K. **Toward the accurate identification of network applications**. Em ***In PAM***, págs. 41–54, 2005.

MYSQL. ***MySql Wiki***, 2012.

NOD, M. **Os dez maiores botnets - os spams mais frequentes na internet**, 2012. URL http://www.mknod.com.br/?q=spam-botnets.

OF WAIKATO, U. ***WEKA***, 2012.

OPENPACKET. **Open packet- capture repository**, 2012. URL https://www.openpacket.org/.

PCAPJOINER. ***PCAPJOINER***, 2012.

POWERS, D. M. W. **Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation**. Technical Report SIE-07-001, School of Informatics and Engineering, Flinders University, Adelaide, Australia, 2007.

QOSIENT. ***ARGUS Wiki***, 2012.

QUINLAN, J. R. **Induction of decision trees**. *Mach. Learn.*, 1(1):81–106, março 1986. ISSN 0885-6125. URL http://dx.doi.org/10.1023/A:1022643204877.

QUINLAN, J. R. ***C4.5: programs for machine learning***. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993. ISBN 1-55860-238-0.

RAYWOOD, D. **A condensed history of the botnet**, 2010. URL http://www.scmagazineuk.com/a-condensed-history-of-the-botnet/article/191636/.

SAROIU, S., GUMMADI, K. P., DUNN, R. J., GRIBBLE, S. D. e LEVY, H. M. **An analysis of internet content delivery systems**. *SIGOPS Oper. Syst. Rev.*, 36(SI):315–327, dezembro 2002. ISSN 0163-5980. URL http://doi.acm.org/10.1145/844128.844158.

SEN, S. e WANG, J. **Analyzing peer-to-peer traffic across large networks**. *Networking, IEEE/ACM Transactions on*, 12(2):219 – 232, april 2004. ISSN 1063-6692.

SILVA, R. M. P. e SALLES, R. M. **Methodology for detection and restraint of P2P applications in the network**. volume 7336 of *Lecture Notes in Computer Science*, chapter 24, págs. 326–339. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2012a. ISBN 978-3-642-31127-7. URL http://dx.doi.org/10.1007/978-3-642-31128-4_24.

SILVA, S. S., SILVA, R. M., PINTO, R. C. e SALLES, R. M. **Botnets: a survey**. *Computer Networks*, 2012b. ISSN 1389-1286. URL http://www.sciencedirect.com/science/article/pii/S1389128612003568?v=s5.

SIT, E., MORRIS, R. e KAASHOEK, M. F. **Usenetdht: a low-overhead design for usenet**. **Em *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation***, NSDI'08, págs. 133–146, Berkeley, CA, USA, 2008. USENIX Association. ISBN 111-999-5555-22-1. URL http://dl.acm.org/citation.cfm?id=1387589.1387599.

SPOGNARDI, A., LUCARELLI, A. e DI PIETRO, R. **A methodology for p2p file-sharing traffic detection**. **Em *Hot Topics in Peer-to-Peer Systems, 2005. HOT-P2P 2005. Second International Workshop on***, págs. 52 – 61, july 2005.

STOVER, S., DITTRICH, D., HERNANDEZ, J. e DIETRICH, S. **Analysis of the storm and nugache: P2p is here**. Em ***Proceedings of the 4th USENIX Workshop on Cyber Security Experimentation and Test (CSET'11)***. USENIX Association, Dec 2007.

TANNER, T. **Distributed hash tables in p2p systems - a literary survey, seminar on internetworking**, 2005.

TCPDUMP. ***TCPDump Manual***, 2012.

TCPREPLAY. ***TCPREPLAY Pcap edit and replay tools***, 2012.

ULLIAC, A. e GHITA, B. V. **Non-intrusive identification of peer-to-peer traffic**. Em ***Proceedings of the 2010 Third International Conference on Communication Theory, Reliability, and Quality of Service***, CTRQ '10, págs. 116–121, Washington, DC, USA, 2010. IEEE Computer Society. ISBN 978-0-7695-4070-2. URL `http://dx.doi.org/10.1109/CTRQ.2010.27`.

VISHNUMURTHY, V. e FRANCIS, P. **On heterogeneous overlay construction and random node selection in unstructured p2p networks**. Em ***INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings***, págs. 1 –12, april 2006.

VISHNUMURTHY, V. e FRANCIS, P. **A comparison of structured and unstructured p2p approaches to heterogeneous random peer selection**. Em ***2007 USENIX Annual Technical Conference on Proceedings of the USENIX Annual Technical Conference***, ATC'07, págs. 24:1–24:14, Berkeley, CA, USA, 2007. USENIX Association. ISBN 999-8888-77-6. URL `http://dl.acm.org/citation.cfm?id=1364385.1364409`.

WANG, P., SPARKS, S. e ZOU, C. C. **An advanced hybrid peer-to-peer botnet**. Em ***Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets***, HotBots'07, págs. 2–2, Berkeley, CA, USA, 2007. USENIX Association. URL `http://dl.acm.org/citation.cfm?id=1323128.1323130`.

WANG, P., WU, L., ASLAM, B. e ZOU, C. **A systematic study on Peer-to-Peer botnets**. Em ***Computer Communications and Networks, 2009. ICCCN 2009. Proceedings of 18th Internatonal Conference on***, págs. 1 –8, agosto 2009.

XIAO-NAN, L., YANG, L. e HUA, Z. **Peer-to-peer botnets: Analysis and defense**. Em ***Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on***, págs. 140 –143, may 2011.

YAO, Z., LIU, P., LEI, L. e YIN, J. **R-c4.5 decision tree model and its applications to health care dataset**. Em ***Services Systems and Services Management, 2005. Proceedings of ICSSSM '05. 2005 International Conference on***, págs. 1099 – 1103 Vol. 2, june 2005.

YU, F., SUO, Y. e SONG, D. **Peer-to-peer traffic detection based on periodic sampling**. **Em *Database Technology and Applications (DBTA), 2010 2nd International Workshop on***, págs. 1 –4, nov. 2010.

YU, J., LI, Z., HU, J., LIU, F. e ZHOU, L. **Using simulation to characterize topology of peer to peer botnets**. *Computer Modeling and Simulation, International Conference on*, 0:78–83, 2009.

ZHANG, J., PERDISCI, R., LEE, W., SARFRAZ, U. e LUO, X. **Detecting stealthy p2p botnets using statistical traffic fingerprints**. **Em *Dependable Systems Networks (DSN), 2011 IEEE/IFIP 41st International Conference on***, págs. 121 –132, june 2011.

ZHOU, L.-J., LI, Z.-T. e HAO, T. **Proposition and provement of a tcp feature of p2p traffic- an example of bittorrent and emule**. **Em *Communications and Networking in China, 2007. CHINACOM '07. Second International Conference on***, págs. 61 –65, aug. 2007.

8 ANEXOS

8.1 RUBOT

O *Rubot*, consiste de um *framework* para a emulação e análise de *botnets*. Sua instalação foi executada em 16 sistemas operacionais *Linux Ubuntu 12.04*.

Os pacotes necessários para seu correto funcionamento foram: *screen, ruby1.8-full, ruby1.8-dev, vim-ruby, rubybook, rubygems1.8, subversion, sqlite3, libsqlite3-dev, build-essential, hping3*. Todos foram instalados através da aplicação *Aptitude*, nativa do próprio sistema.

Após sua correta instalação, uma *botnet Nugache* contendo 15 membros foi configurada, cada uma das máquinas representou um membro da rede maliciosa. Com o objetivo de acelerar a inicialização da *botnet* foi montado um *shell script* contendo todos os comandos necessários para tal (Figura 8.1).

```
#!/usr/bin/env bash

if [ ! "x$1" == "x-nosvn" ]
then
    for host in serv01 serv02 serv03 serv04 serv05 serv06 serv07 serv08 serv09 serv10 serv11 serv12 serv13
serv14 serv15
    do
        ssh -t $host "cd rubot;svn up"
    done
fi

for host in serv01 serv02
do
    xterm -T $host -e ssh -t $host "cd rubot;./experiments/nugache.rb 2008" &
done
sleep 4
for host in serv03 serv04 serv05 serv06 serv07 serv08
do
    xterm -T $host -e ssh -t $host "cd rubot;./experiments/nugache.rb 2008 serv01:2008" &
done

for host in serv09 serv10 serv11 serv12 serv13 serv14
do
    xterm -T $host -e ssh -t $host "cd rubot;./experiments/nugache.rb 2008 serv02:2008" &
done

xterm -T   serv15   -e ssh -t   serv15    "cd rubot;./experiments/nugache.rb 2008 serv01:2008 serv02:2008" &

./experiments/nugache_controller.rb serv01:2008
sleep 10000
```

FIG. 8.1: *Shell script* utilizado pelo *Rubot* para a geração da *botnet Nugache* com 15 nós.

No *script*, o arquivo *nugache.rb* é responsável pela inicialização da *bot* na máquina em que se encontra instalado. Como parâmetros podem ser passados a porta que a *bot* utilizará para receber e enviar comandos, no caso 2008, e o endereço das demais *bots* a que está conectada.

A aplicação *nugache_controller.rb* é responsável pela representação do *botmaster* na rede. É através dela que os comandos serão repassados as demais *bots*. Como parâmetros são utilizados os endereços das *bots* a que estão conectadas. Todos os comandos enviados e interpretados pela *botnet* deverão ser escritos na linguagem *Ruby*.

8.2 ARGUS

O ARGUS (*Audit Record Generation and Utilization System*) é aplicação responsável pela conversão de *traces* de redes em arquivos de fluxo de rede (Figura 8.2). Além disso, a aplicação possui um módulo que permite a exportação dos fluxos de rede gerados para uma base de dados *MySQL*, o RASQL_INSERT (Figura 8.3).

```
sh-3.2# argus -S 3600 -mAJZRU 1024 -r arquivo_pcap -w arquivo_fluxo.arg3
```

FIG. 8.2: Comando utilizado pelo ARGUS para converter arquivos de *trace* (extensão *pcap*) para arquivos de fluxo a serem utilizados na arquitetura (extensão *arg3*).

```
sh-3.2# rasqlinsert  -r ARQUIVO_FLUXO.arg3  -w mysql://localhost/flow/BASE_DADOS
-s "stime dtime dur  proto saddr sport dir daddr dport spkts dpkts sbytes dbytes
 state sloss dloss sttl dttl" -m none
```

FIG. 8.3: Comando utilizado para exportar arquivo de fluxo para o banco de dados.

Os parâmetros indicados no comando (Figura 8.3) e que foram exportados para a base de dados (Figura 8.4) para posterior análise são:

- *stime* : *timestamp* do fluxo;
- *dur* : duração do fluxo em segundos;
- *proto*: protocolo utilizado;
- *saddr*: endereço da origem;
- *sport*: porta da origem;
- *dir*: direção do fluxo;
- *daddr*: endereço do destino;
- *dport*: porta do destino;
- *spkts*: número de pacotes enviados pela origem;
- *dpkts*: número de pacotes enviados pelo destino;
- *sbytes*: quantidade de *bytes* enviados pela origem;

- *dbytes*: quantidade de *bytes* enviados pelo destino;
- *state*: estado da conexão no término do fluxo;
- *sloss*: número de pacotes perdidos pela origem;
- *dloss*: número de pacotes perdidos pelo destino;
- *sttl*: TTL utilizado pela origem;
- *dttl*: TTL utilizado pelo destino.

stime	dur	proto	saddr	sport	dir	daddr	dport	spkts	dpkts	sbytes	dbytes	state	sloss	dloss	sttl
1349654336.169595	3.096584	tcp	10.0.1.3	50963	?>	50.136.71.176	54070	1	0	54	0	RST	0	0	64
1349653931.582444	1071.270508	tcp	94.173.1...	4977	->	10.0.1.3	51413	5	5	406	342	FIN	0	0	49
1349655776.296239	915.198547	tcp	75.189.2...	51767	->	10.0.1.3	51413	6	6	907	615	FIN	0	0	48
1349653869.537107	593.382568	udp	10.0.1.3	55717	->	255.255.255.255	2008	2	0	88	0	INT	0	0	64
1349653869.610508	0.000000	udp	10.0.1.3	61772	->	255.255.255.255	2223	1	0	114	0	INT	0	0	64
1349653869.992393	0.000000	udp	10.0.1.3	51413	->	82.12.196.125	48266	1	0	100	0	INT	0	0	64
1349653870.037311	0.000000	udp	10.0.1.3	52538	->	255.255.255.255	2008	1	0	44	0	INT	0	0	64
1349653870.537777	0.000000	udp	10.0.1.3	53788	->	255.255.255.255	2008	1	0	44	0	INT	0	0	64
1349653871.038040	3133.304199	udp	10.0.1.3	56849	->	255.255.255.255	2008	2	0	88	0	INT	0	0	64
1349653871.538250	0.000000	udp	10.0.1.3	60849	->	255.255.255.255	2008	1	0	44	0	INT	0	0	64
1349653872.038480	0.000000	udp	10.0.1.3	57368	->	255.255.255.255	2008	1	0	44	0	INT	0	0	64
1349653872.538704	0.000000	udp	10.0.1.3	53777	->	255.255.255.255	2008	1	0	44	0	INT	0	0	64
1349653873.038910	0.000000	udp	10.0.1.3	54115	->	255.255.255.255	2008	1	0	44	0	INT	0	0	64
1349653873.539115	0.000000	udp	10.0.1.3	52637	->	255.255.255.255	2008	1	0	44	0	INT	0	0	64
1349653874.054264	0.000000	udp	10.0.1.3	57775	->	255.255.255.255	2008	1	0	44	0	INT	0	0	64
1349653874.061386	2885.433838	udp	10.0.1.3	51413	<->	31.42.177.117	34768	7	7	844	1646	CON	0	0	64
1349653874.365014	159.251572	udp	10.0.1.3	51413	<->	110.246.130.191	16001	5	4	536	356	CON	0	0	64
1349653874.365088	16.226015	udp	10.0.1.3	51413	->	173.81.69.235	23606	2	0	200	0	INT	0	0	64
1349653874.365130	36.024799	udp	10.0.1.3	51413	->	180.241.111.24	30641	3	0	300	0	INT	0	0	64
1349653874.365170	2739.746338	udp	10.0.1.3	51413	<->	67.215.242.138	6881	4	4	400	400	CON	0	0	64
1349653874.365210	2737.697998	udp	10.0.1.3	51413	<->	99.43.230.30	1562	4	4	400	400	CON	0	0	64
1349653874.555127	0.000000	udp	10.0.1.3	50006	->	255.255.255.255	2008	1	0	44	0	INT	0	0	64
1349653874.919568	13.936160	udp	10.0.1.3	51413	<->	89.233.206.224	51153	2	2	200	220	CON	0	0	64
1349653875.055392	0.000000	udp	10.0.1.3	49698	->	255.255.255.255	2008	1	0	44	0	INT	0	0	64
1349653875.191665	480.826111	tcp	10.0.1.3	63013	?>	65.55.71.199	1863	9	0	622	0	RST	0	0	64
1349653875.555676	0.000000	udp	10.0.1.3	61572	->	255.255.255.255	2008	1	0	44	0	INT	0	0	64
1349653876.055971	2822.194824	udp	10.0.1.3	62780	->	255.255.255.255	2008	3	0	132	0	INT	0	0	64

FIG. 8.4: Tabela gerada pelo RASQL_INSERT referente ao fluxo de uma aplicação P2P.

8.3 COMANDOS SQL UTILIZADOS PARA A IMPLEMENTAÇÃO DOS MÓDULOS

Os códigos SQL utilizados na obtenção das informações necessárias a execução dos 4 módulos adotados na fase 1 da arquitetura e para a diferenciação entre *bots* P2P e aplicações P2P legítimas na fase 2 são:

- Taxa de envio de pacotes TCP SYN/ UDP:

  SELECT stime , saddr, sport **FROM** BASE_DADOS
  WHERE proto **like** "TCP"**or** proto **like** "UDP"

- Percentual de conexões bem-sucedidas:

  SELECT saddr, state, **COUNT**(*) **FROM** BASE_DADOS
  WHERE proto **like** "TCP"**or** proto **like** "UDP"
  GROUP BY saddr, state

- Múltiplos fluxos entre hosts:

  SELECT saddr, daddr, sport, dport , **COUNT**(*) **FROM** BASE_DADOS
  WHERE proto **like** "TCP"**or** proto **like** "UDP"
  GROUP BY saddr, daddr

- Valor médio da porta utilizada:

  SELECT saddr, daddr, **AVG**(dport) **FROM** BASE_DADOS
  WHERE proto **like** "TCP"**or** proto **like** "UDP"
  GROUP BY saddr, daddr

- Número médio de pacotes por fluxo:

  SELECT saddr, daddr , **AVG**(spkts), **AVG**(dpkts), **count**(*) **FROM** BASE_DADOS
  WHERE proto **like** "TCP"**or** proto **like** "UDP"
  GROUP BY saddr, daddr