UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

DETECÇÃO DE ANOMALIAS EM REDES WAN USANDO ESTIMATIVA DE HOLT-WINTERS APLICADA A MEDIDAS DE ENTROPIA

Alex Soares de Moura

Orientador
Prof. DSc. Sidney Cunha de Lucena

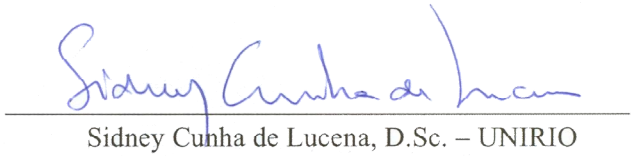
RIO DE JANEIRO, RJ – BRASIL
SETEMBRO DE 2009

DETECÇÃO DE ANOMALIAS EM REDES WAN USANDO ESTIMATIVA DE HOLT-WINTERS APLICADA A MEDIDAS DE ENTROPIA

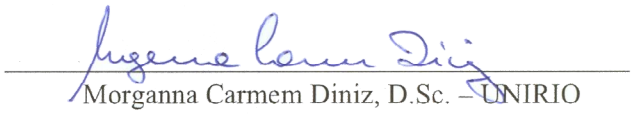
Alex Soares de Moura

DISSERTAÇÃO APRESENTADA COMO REQUISITO PARCIAL PARA OBTENÇÃO DO TÍTULO DE MESTRE PELO PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA DA UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO (UNIRIO). APROVADA PELA COMISSÃO EXAMINADORA ABAIXO ASSINADA.

Aprovada por:



Sidney Cunha de Lucena, D.Sc. – UNIRIO



Morganna Carmem Diniz, D.Sc. – UNIRIO

Ronaldo Moreira Salles, Ph.D. – IME

Magnos Martinello, D.Sc. – UFES

RIO DE JANEIRO, RJ – BRASIL

SETEMBRO DE 2009

Moura, Alex Soares de.
M929 Detecção de anomalias em redes WAN usando estimativa de Holt-Winters aplicada a medidas de entropia / Alex Soares de Moura, 2009.
xv, 96f.

Orientador: Sidney Cunha de Lucena.
Dissertação (Mestrado em Informática) – Universidade Federal do Estado do Rio de Janeiro, Rio de Janeiro, 2009.

1. Sistemas operacionais distribuídos (Computadores). 2. Redes de computadores. 3. Informação – Medidas de segurança. 4. Gerência de redes. 5. Anomalias de tráfego em redes. 6. Redes WAN. I. Lucena, Sidney Cunha de. II. Universidade Federal do Estado do Rio de Janeiro (2003-). Centro de Ciências Exatas e Tecnologia. Curso de Mestrado em Informática. III. Título.

CDD – 004.36

A meus pais

A dúvida é o principio da sabedoria.

Aristóteles

Agradecimentos

Primeiramente, agradeço a meu orientador, professor Sidney Lucena e à professora Morganna Diniz pela acolhida no PPGI da UNIRIO, pelos ensinamentos, apoio e estímulos que me proporcionaram ao longo destes dois anos e meio, e também pela agradável convivência que tivemos durante este período. Ao professor Sidney agradeço também por sua competência, rigor, persistência, paciência e atenção aos detalhes, que foram fundamentais para a conclusão deste trabalho.

Agradecimentos especiais também ao Centro de Engenharia e Operações da Rede Nacional de Ensino e Pesquisa – RNP, sob a direção de Alexandre Grojsgold e Ari Frazão, pelo apoio fundamental, e também a João Luiz de Brito Macaíba, analista de operações e Aluizio Hazin analista de engenharia e operações, ambos da RNP, cujas contribuições foram cruciais para conclusão deste trabalho.

Ao professor Angelo Ciarlini, cujas observações e comentários foram extremamente relevantes e contribuíram durante o processo de avaliação e evolução deste trabalho.

Um agradecimento muito especial a Andréia, pela paciência, apoio e companhia. Agradecimentos também a Mônica e Valentina, pelo apoio.

Por fim, agradeço a minha mãe e a meu pai pela torcida e incentivo e, sem os quais este trabalho não existiria, e à minha irmã, de quem me orgulho pelas recente conquista de ser aprovada como médica cardiologista.

MOURA, Alex Soares. **Detecção Anomalias em Redes WAN Usando Estimativa de Holt-Winters Aplicada a Medidas de Entropia** . UNIRIO, 2009. 107 páginas. Dissertação de Mestrado. Departamento de Informática Aplicada, UNIRIO.

RESUMO

Detectar anomalias de tráfego em redes WAN é uma tarefa de relativa complexidade. Propostas mais promissoras se baseiam em séries temporais contendo medidas de entropia para descrever padrões de tráfego. Este trabalho avalia o uso de estimadores tradicionais e de simples implementação, aplicado às medidas de entropia, para sinalizar a ocorrência de eventos que possam comprometer o bom funcionamento da rede e que não sejam facilmente detectados pelas ferramentas de gerência comumente usadas. Os resultados experimentais, extraídos para amostras de tráfego da Rede Ipê, avaliam o uso da estimativa de Holt-Winters na sinalização de anomalias artificialmente injetadas nas amostras de tráfego. Conclui-se que o método utilizado é capaz de identificar a ocorrência de tráfegos anômalos em enlaces de uma rede WAN, ocorrências estas que dificilmente conseguiriam ser detectadas pelas formas tradicionais monitoramento. Além disso, o sistema proposto consegue ser facilmente adaptado às ferramentas mais comuns usadas no gerenciamento de redes.

Palavras-chave: Sistemas distribuídos; redes de computadores; segurança da informação; gerência de redes; detecção de anomalias.

ABSTRACT

To detect anomalies in wide-area network traffic is a relatively complex task. Most promising propositions are based in time series of entropy measurements to describe traffic patterns. This present work evaluates the use of traditional predictors of simple implementation, applied to entropy measurements, to signalize events that may compromise the well behavior of the network and that cannot be easily detected by commonly used network management tools. Experimental results, obtained for traffic samples of Rede Ipê, show the use of Holt-Winters estimator on signalizing artificially injected anomalies in the traffic samples. It can be concluded that the proposed method can identify the presence of anomalous traffic in WAN links, what would be hard to do using traditional monitoring proceeds. Besides, the proposed system can be easily coupled with most used network management tools.

Keywords: Distributed systems; computer networks; information security; network management; anomaly detection.

Índice

Índice de Figuras

Índice de Tabelas

Lista de Abreviaturas Utilizadas

AS *Autonomous System*

bps *bits por segundo*

BSD *Berkeley Software Distribution*

CAIDA *Cooperative Association for Internet Data Analysis*

CAIS *Centro de Atendimento a Incidentes de Segurança*

CDEF *Consolidation Function Definition*

CEO *Centro de Engenharia e Operações*

CIDR *Classless Inter-Domain Routing*

CLI *Command Line Interface*

DoS *Denial of Service*

DDoS *Distributed Denial of Service*

EWMA *Exponential Weighted Moving Average*

FCAPS *Fault, Configuration, Accounting, Performance, Security*

IDS *Intrusion Detection System*

IP *Internet Protocol*

IPFIX *IP Flow Information eXport (IPFIX) protocol*

IPS *Intrusion Prevention System*

ISO *International Organization for Standardization*

ISP *Internet Service Provider*

GB *Gigabyte*

Gbps *Gigabits por segundo*

HW *Holt-Winters*

LAN *Local Area Network*

MB *Megabyte*

Mbps *Megabits por segundo*

MTTR *Mean Time to Repair*

NOC *Network Operation Center*

NREN *National Research and Education Network*

PDP *Primary Data Point*

PoP *Point of Presence*

pps *pacotes por segundo*

PTT *Ponto de Troca de Tráfego*

RRA *Round Robin Archive*

RNP *Rede Nacional de Ensino e Pesquisa*

RPM *Rotações Por Minuto*

RTT *Round Trip Time*

SCTP *Stream Control Transmission Protocol*

SLA *Service Level Agreement*

SNMP *Simple Network Management Protocol*

SPAN *Switched Port Analyzer*

TCP *Transmission Control Protocol*

ToS *Type of Service*

TTS *Trouble Ticket System*

UDP *User Datagram Protocol*

WAN *Wide Area Network*

1 Introdução

As bases da sociedade da informação são o acesso democrático à informação, à disseminação do conhecimento e aos serviços *online*. Para que tudo isso funcione, as pessoas e comunidades dependem da infra-estrutura das redes de computadores, particularmente nas áreas de educação, pesquisa e comércio. Com a dependência cada vez maior do bom funcionamento das redes, a compreensão da natureza das anomalias de tráfego em redes de computadores - principalmente as redes avançadas e de alta capacidade - tornou-se um problema importante e de solução complexa.

Anomalia de tráfego é o que foge a um padrão conhecido, considerado como normal no tráfego da rede. Independente das anomalias serem intencionais ou não, é importante que nas atividades preventivas de operação das redes seja possível analisá-las, devido aos prejuízos que podem decorrer dos variados tipos de incidentes, cujo ápice seria a completa falha nas comunicações de alguma parte de uma rede, que pode ser local, regional, nacional ou intercontinental, com variada amplitude de comunidades afetadas. Além da infra-estrutura da rede, as anomalias presentes no tráfego da rede podem impactar no uso de serviços por inúmeras comunidades de usuários.

"Diagnosticar anomalias envolve a detecção, identificação e quantificação de eventos no tempo que tenham causado algum problema no tráfego da rede." [Monsores 2006]. Ao longo do tempo, com a crescente quantidade de incidentes de segurança e uma maior necessidade por serviços com alta disponibilidade, a investigação de padrões que possam estar associados a anomalias de tráfego em redes de computadores torna-se cada vez mais importante para a oferta de serviços de qualidade.

As anomalias podem ser classificadas em diversos tipos. Aquelas com causa intencional, e que causam transtornos aos usuários de serviços são chamadas de **ataques**. Alguns exemplos são os ataques de negação de serviço (DoS), negação de serviço com origem distribuída (DDoS), infestações automatizadas de código malicioso

(*worms*), exércitos de máquinas controladas sem autorização (*botnets*), varreduras de portas maliciosas (*portscans*), envio massivo de correio eletrônico não solicitado (*spam*) e crescimento de tráfego repentino – às vezes com motivos legítimos – e/ou temporários (*flash crowds*), para citar alguns.

A detecção de anomalias vem sendo tema cada vez mais pesquisado nos últimos anos, cujos resultados já proporcionaram diversas metodologias para identificação e classificação de anomalias presentes no tráfego de rede. Alguns métodos são baseados em assinatura, que é a identificação de elementos no tráfego explicitamente associados a uma anomalia. Este tipo de análise costuma ser computacionalmente onerosa por exigir a inspeção do conteúdo de cada pacote IP trafegando nos fluxos das conexões de rede. Outra forma de detecção de anomalias é baseada em assinatura estatística do tráfego de rede associado a determinado tipo de anomalia [Estevez-Tapiador 2004]. Este método analisa o comportamento dos fluxos de pacotes para verificar se há presença de alguma anomalia correlata, sem a necessidade de inspecionar cada pacote IP. Esta seria uma abordagem mais viável para uso em redes de *backbone*, que geralmente possuem grandes volumes de pacotes atravessando seus roteadores.

[Lakhina 2005] apresenta uma metodologia que usa análise de entropia para detectar anomalias, baseada em assinatura estatística de tráfego. Os autores adotam esta análise tanto para a detecção quanto também para a classificação das anomalias, incluindo ameaças do tipo *day-zero*, ou seja, anomalias maliciosas ainda desconhecidas (como, por exemplo, um novo *worm*). A metodologia consegue inferir um padrão de tráfego normal através do uso de técnicas de estruturação e agrupamento de dados para mineração, aprendizado e classificação do tráfego anômalo.

A eficiência da metodologia de [Lakhina 2005] foi aferida com sucesso após testes com amostras de tráfego da rede acadêmica dos EUA, a Abilene, e da rede acadêmica européia GÉANT, que congrega diversas redes acadêmicas nacionais da Europa.

Nas redes acadêmicas de pesquisa e educação, geralmente financiadas por instâncias governamentais de cada país, seja a nível metropolitano, estadual ou nacional, é comum os grupos responsáveis por sua operação - como os NOCs (*Network Operations Centers*) - adotarem ferramentas gratuitas de *software* livre [FSF 2008] para controlar, monitorar e sinalizar alarmes. Algumas ferramentas de *software* livre vêm

sendo usadas na descoberta de alguns tipos de anomalias de tráfego, porém ainda são extremamente limitadas quando comparadas a ferramentas comerciais ([Arbor 2008]). Estas, por sua vez, ainda são muito especializadas e possuem custos de aquisição, licenciamento e manutenção (atualizações periódicas) muitas vezes proibitivos a organizações sem fins lucrativos, tendo seu uso limitado a empresas de grande porte, como as grandes operadoras de telecomunicações provedoras de redes para o mercado.

Outro forte apelo para o uso de ferramentas de código livre, além do custo, é obviamente o acesso ao seu código-fonte, o que permite modificações pelos próprios usuários para adequá-las e incrementá-las sem custos. Estas modificações muitas vezes retornam à comunidade de usuários do produto em forma de contribuições para os desenvolvedores das ferramentas, como correções (*patches*) ou *plugins* (*add-ons*). Estas contribuições geralmente são rapidamente redistribuídas aos usuários das ferramentas que, além de usufruir dos novos recursos, ajudam os desenvolvedores a testá-las e a depurar eventuais *bugs*.

As soluções de sistemas de detecção de anomalias atualmente disponíveis no mercado se destinam à detecção de variados tipos de anomalias em redes, sendo a maioria das soluções mais voltada para o uso em ambientes de redes locais corporativas e em *datacenters* de grande porte como, por exemplo, os IDS (*Intrusion Detection Systems*) e os IPS (Intrusion Prevention Systems). Já outras soluções são voltadas para redes maiores, como as redes de campus e de provedores ISP (*Internet Service Providers*). Em geral, estas soluções possuem um custo muito elevado quando comparado ao investimento de implantação e operação destas redes.

Tanto em [Lakhina 2005] quanto em [Zhang 2005], o arcabouço necessário para todo um processo de detecção, classificação e identificação dos fluxos anômalos é bastante complexo. Nestes trabalhos, a abordagem é do tipo *"network-wide"*, ou seja, faz-se necessário extrair informações de toda a rede para que seja possível detectar uma anomalia qualquer. Isto significa que é necessário ter visibilidade da rede como um todo. Entretanto, em [Silveira 2008], é mostrado que, surpreendentemente, a grande maioria das anomalias encontradas por métodos do tipo *network-wide* são também encontradas por métodos mais simples, do tipo *"single-link"*, ou seja, baseados em informações de apenas uma única interface. No caso dos métodos *single-link*, somente as anomalias que afetam a interface monitorada podem ser detectadas. Se, por exemplo,

a interface monitorada é o único enlace de conexão da rede com a Internet, então é por este enlace que passam todos os ataques oriundos de redes externas.

O presente trabalho se propõe a abordar o problema de detecção e sinalização de anomalias para um cenário de gerenciamento de redes típico de várias instituições brasileiras, principalmente aquelas ligadas a governo, como as redes acadêmicas das IFES (Instituições Federais de Ensino Superior) e outras similares na América Latina e no mundo. Deste cenário, podemos destacar as seguintes características: possuem soluções de gerenciamento de redes baseadas em *software* livre e arquitetura SNMP; monitoramento restrito às interfaces de rede dos equipamentos sob responsabilidade dos centros de gerência, na maioria destes centros não há visibilidade do sistema autônomo como um todo; histórico de estatísticas de tráfego, coletadas via SNMP, armazenado em bases RRD (*Round-Robin Database*), geralmente usando medidas extraídas a cada intervalo de cinco minutos; alarmes de ataques baseados em limiares pré-estabelecidos para a taxa de bits/s e pacotes/s nas interfaces monitoradas.

Sendo assim, este trabalho propõe o uso de séries temporais contendo medidas de entropia extraídas para fluxos de pacotes IP que passam por uma dada interface, combinado ao uso de um estimador de comportamento para estas séries temporais, no caso a estimativa de Holt-Winters [Brutlag 2000], a fim de melhorar a eficiência das técnicas atualmente usadas pelos administradores de redes para sinalizar anomalias, técnicas estas que se valem de ferramentas de *software* livre com abordagens mais simples e de baixo custo computacional. As medidas de entropia são extraídas para endereços IP e portas dos pacotes que trafegam numa dada interface de rede e a estimativa de Holt-Winters é usada para capturar o padrão normal de variação destas séries temporais, servindo também para sinalizar as possíveis anomalias em curso, onde os ataques à segurança da rede são o foco principal. Até onde foi possível verificar, a combinação da estimativa como Holt-Winters com medidas de entropia é uma abordagem nova para a detecção de anomalias em tráfegos IP de redes WAN.

A proposta deste trabalho foi validada a partir de amostras de tráfego coletadas da rede Ipê, que é o *backbone* da Rede Nacional de Ensino e Pesquisa (RNP). Os resultados mostraram a validade deste método para a detecção de tipos mais comuns de ataques passando por uma rede WAN, ataques estes que não necessariamente seriam verificados através da simples monitoração do tráfego de pacotes na interface de rede. Também foi possível verificar que esta metodologia consegue ser agregada com muito

pouco custo às ferramentas de *software* livre comumente usadas para o gerenciamento de redes.

Podem ser destacadas, como contribuições do presente trabalho:

- a proposta de uma metodologia simples para detecção de anomalias usando medidas de entropia e abordagem *single-link*, compatível com cenários mais comuns de gerenciamento de redes;
- uma análise comparativa entre os estimadores EWMA e HW quanto à capacidade de sinalizar anomalias nas séries temporais de medidas de entropia;
- um refinamento da proposta a partir do uso de filtros que capturam as características conhecidas de determinados tipos de ataques;
- a criação de um *framework* para a implementação da metodologia proposta utilizando ferramentas de *software* livre amplamente adotadas no gerenciamento de redes;
- e a obtenção de resultados e validação da proposta usando dados reais da rede Ipê.

Com relação à estrutura deste documento: o Capítulo 2 descreve a fundamentação teórica aplicada a detecção de anomalias; o Capítulo 3 mostra as técnicas para obtenção de medidas de tráfego e gerenciamento em redes WAN; o Capítulo 4 descreve o método proposto; o Capítulo 5 apresenta os resultados obtidos; e o Capítulo 6 apresenta as conclusões e trabalhos futuros. Ao final deste documento, após as referências, há um apêndice contendo os *scripts* desenvolvidos durante o trabalho.

2 Fundamentos para Identificação de Anomalias

2.1 Definição de anomalia

Entende-se por anomalia de tráfego tudo aquilo que foge a um padrão previamente reconhecido como normal no tráfego de um determinado ponto de medição de uma rede. Os pontos de medição de tráfego em redes são tipicamente interfaces de equipamentos ativos de rede, como roteadores e *switches.*

Particularmente em redes WAN, anomalias no tráfego são eventos transitórios com potencial para ocasionar perturbações em larga escala, afetando total ou parcialmente o acesso de grandes quantidades de usuários a serviços internos e/ou externos, e também degradando ou mesmo impedindo a comunicação entre eles.

Na medida em que serviços de comunicação de telefonia e vídeo se tornam cada vez mais comuns sobre as redes IP, inclusive como alternativas econômicas para encontros presenciais, torna-se evidente a importância da detecção rápida de eventos anômalos que possam comprometer o funcionamento e o desempenho das redes e seus serviços, como foi observado em eventos de *worms* de rápido alastramento por toda a internet ([Moore 2003]).

2.2 Análise do comportamento do tráfego

Para se analisar o comportamento do tráfego, o primeiro passo é definir os parâmetros que se deseja observar, parâmetros estes que, de alguma forma, possuam as informações necessárias para o objetivo da análise. No caso da detecção de anomalias em redes WAN, faz-se necessária uma distinção entre os diversos tipos de tráfegos que passam pela rede, uma vez que cada um tem suas próprias características.

O tráfego de dados em redes pode ser entendido como um conjunto dinâmico de diversos fluxos de pacotes, sendo que as informações contidas em cada pacote podem servir para caracterizar estes fluxos. Cada fluxo possui um conjunto de pacotes que compartilham as mesmas características de parâmetros, como IP e porta de origem e IP

e porta de destino, isto em um único sentido e em determinado período de tempo. Sendo assim, cada fluxo pode ser considerado como o tráfego decorrente de uma determinada aplicação em rede.

Segundo [Estevez-Tapiador 2004], é fato conhecido que certos parâmetros relacionados ao montante de recursos consumidos pelo tráfego de rede – como a quantidade de pacotes transmitidos por segundo – possuem um formato de onda regular, quando observados como uma série temporal. Os comportamentos observados são característicos de cada rede, mesmo dependendo de grande quantidade de fatores, como quantidade de usuários, quantidade de equipamentos de rede, horários de observação etc. Enquanto os fatores permanecem inalterados, as características dos padrões de tráfego na rede permanecem os mesmos.

2.3 Métodos para identificação de anomalias

2.3.1 Assinatura de tráfego

Um dos métodos para a identificação de anomalias, principalmente aquelas ligadas a tráfego malicioso, é o que faz uso de assinaturas de tráfego ([Estevez-Tapiador 2004]). Entende-se por assinatura de tráfego um conjunto de características particulares de um determinado tipo de tráfego. Muitas vezes esta assinatura ocorre apenas no dado da aplicação, onde é possível identificar alguma seqüência de caracteres que caracterize a presença de algum código malicioso, por exemplo.

Este método possui a desvantagem de requerer que a anomalia, para ser detectada, tenha uma assinatura previamente conhecida. Além disso, este método possui alto custo computacional para captura e inspeção do conteúdo de todos os pacotes que atravessam as conexões de rede monitoradas, em busca de assinaturas conhecidas. No caso de uma rede WAN, que possui um grande volume de pacotes trafegando, o custo computacional desta inspeção pode ser proibitivo.

2.3.2 Assinatura estatística de tráfego

Outro método de detecção de anomalias de tráfego requer a observação do comportamento estatístico do tráfego de pacotes, sem a necessidade de investigar o conteúdo dos pacotes trafegados pelas conexões de rede monitoradas ([Estevez-Tapiador 2004]). Este método se baseia no conceito de que uma anomalia de tráfego altera o comportamento estatístico do mesmo. Por exemplo, no caso de um ataque massivo de negação de serviço passando por uma dada interface de rede, é comum

verificar um aumento substancial na taxa de pacotes por segundo dos dados que trafegam por esta interface. Este é um método mais indicado para monitorar *backbones* WAN, por conta do grande volume de pacotes que atravessam as interfaces da rede.

2.3.3 Detecção por limiar

A detecção por limiar (*threshold*) é outra forma para detecção de anomalias no tráfego de rede ([Estevez-Tapiador 2004]). Após um procedimento para avaliar os níveis normais de variação de uma dada medida de tráfego (*baselining*), como a taxa de pacotes, são determinados – de forma empírica ou através de algum método estatístico – valores de limites superior e/ou inferior que não devem ser ultrapassados por esta medida. Os sistemas de monitoramento podem ser configurados para gerar alertas com base nos desvios que ultrapassarem os limiares estabelecidos, reportando anomalias.

Os alertas baseados em *limiar* são úteis para certas medidas como, por exemplo, consumo de CPU ou de memória em equipamentos, mas também são limitados quanto à sua capacidade de evitar falsos positivos em métricas sujeitas à sazonalidade como, por exemplo, alterações no volume de tráfego de bits por segundo ou pacotes por segundo em fora dos horários comerciais, em fins de semana, feriados e em ocorrências de *flash crowds*.

As limitações deste método de detecção são a dificuldade de se determinar corretamente os valores dos limiares, para evitar falsos positivos, e também a falta de suporte para ajustes a parâmetros sujeitos a variações sazonais.

2.4 Forecasting

A detecção de anomalias de tráfego por estimativas – conhecida como *forecasting* – possui características do método de detecção por limiar combinado a algoritmos de estimativas, automatizando os ajustes nos valores dos limites superior e inferior através de parâmetros que contemplem as tendências de variação da medida, de acordo com [Brutlag 2000]. Estes algoritmos de estimativa costumam ser aplicados a séries temporais e se valem do histórico de medidas anteriores, algumas vezes dentro de uma dada janela de tempo, para estimar uma medida futura. Este é um dos métodos empregados neste trabalho.

2.5 Detecção utilizando entropia

A entropia de Shannon ([Shannon 1948]) é definida como:

$$E_s = -\sum_{i=0}^{N} p_i \log_2(p_i) \qquad (1)$$

onde N é o número de diferentes ocorrências no espaço amostral e p_i a probabilidade associada a cada ocorrência i. Em [Lakhina 2005], N corresponde ao número de diferentes valores do parâmetro em análise (IP de origem, IP de destino, porta de origem ou porta de destino), que ocorreram durante um intervalo de cinco minutos, e p_i é a probabilidade de cada um desses diferentes valores no intervalo medido. O resultado varia entre *zero* e log_2N, onde *zero* indica concentração máxima na distribuição medida – ou seja, um único valor i ocorreu durante todo o intervalo de observação – e log_2N indica máxima dispersão na distribuição medida – ou seja, uma probabilidade igual a $1/N$ para todas as ocorrências dentro do intervalo de observação. Isto significa que, quanto menor o valor da entropia, mais concentrada é a distribuição e, quanto maior seu valor, mais dispersa é a distribuição.

A técnica de detecção de anomalias de tráfego usando entropia é recente, conforme apresentado por [Lakhina 2005] e [MacKey 2003]. Em [Lakhina 2005], os autores empregaram a entropia de Shannon para avaliar o grau de concentração das distribuições de probabilidade dos endereços IP de origem, dos endereços IP de destino, das portas de origem e das portas de destino. Estes parâmetros de tráfego são obtidos para os fluxos IP amostrados durante um intervalo fixo de tempo, no caso, cinco minutos (a definição de fluxo IP pode ser encontrada na seção 3.4.1). Uma variação inesperada na concentração ou dispersão de uma ou mais dessas quatro distribuições serve como indicativo da presença de algum tipo de anomalia.

2.6 Estimadores para séries temporais (*forecasting*)

Para se detectar um comportamento que fuja a um dado padrão histórico, faz-se necessária a construção de um *baseline* para a série analisada. *Baseline* pode ser definida como uma "linha de referência" que serve de base para medições, construída a partir de dados usados como base - ponto inicial - para cálculos ou comparações. Um exemplo simples seria usar a taxa média do tráfego ao longo de uma janela de tempo e seu desvio padrão para arbitrar um limiar mínimo e máximo de variação desta taxa. Qualquer novo valor fora destes limites seria considerado uma anormalidade. A seguir, são mostrados dois estimadores bastante conhecidos: o *Exponential Weighted Movin Average* (EWMA) e o *Holt-Winters* (HW).

2.6.1 EWMA

Esta técnica é também chamada de aproximação exponencial, ou *exponential smoothing* ([Brutlag 2000]). Trata-se de um estimador que faz uma soma ponderada entre o valor atual e um valor representando o acúmulo destas ponderações ao longo do tempo. A expressão do EWMA é:

$$x_{t+1} = \alpha X_t + (1-\alpha)x_t \qquad (2)$$

onde x_t é a média histórica, X_t o valor corrente e $0 < \alpha < 1$.

A constante α indica o peso que uma amostra recente tem sobre a previsão da próxima amostra. Usando de recursividade, é fácil verificar que o peso das amostras passadas, em relação a uma nova previsão, decai exponencialmente à medida que estas amostras tornam-se mais antigas. Valores típicos de α costumam ser inferiores a $0,1$.

Embora muito aplicado em diversos cenários da computação (por exemplo, na estimativa de *timeouts* a partir de medidas de RTT no protocolo TCP), o EWMA não foi concebido para casos onde exista algum tipo de periodicidade ou tendência de crescimento. Para estes casos, um estimador mais apropriado é o Holt-Winters (HW) ([Brutlag 2000] e Lucena [2009]).

2.6.2 Holt-Winters

O algoritmo Holt-Winters divide a série temporal em três partes superpostas: um termo que denota a periodicidade da série,

$$c_t = \gamma(X_t - a_t) + (1-\gamma)c_{t-m}$$

um segundo termo que indica a tendência de crescimento da série

$$b_t = \beta(a_t - a_{t-1}) + (1-\beta)b_{t-1}$$

e, por fim, um termo que expressa uma parte residual.

$$a_t = \alpha(X_t - c_{t-m}) + (1-\alpha)(\alpha_{t-1} + b_{t-1})$$

Cada um desses três termos é tratado de forma separada através de um EWMA, ou *exponential smoothing*. Isto significa que três coeficientes, α, β e γ, devem ser atribuídos, um para cada EWMA. Entretanto, a maneira como estes termos são combinados pode refletir dois tipos de sazonalidade: a aditiva e a multiplicativa ([Koehler 1999]).

A sazonalidade aditiva é aquela onde o termo que representa a variação periódica da série temporal possui comportamento estatístico que independente da taxa de crescimento (positiva ou negativa) da série. A sazonalidade multiplicativa é aquela

onde o termo que representa a variação periódica da série temporal possui comportamento estatístico proporcional à taxa de crescimento da série. No trabalho aqui descrito, utilizou-se apenas a forma aditiva do Holt-Winters, uma vez que ela apresentou melhores resultados durante os experimentos. Abaixo estão as expressões para o Holt-Winters usando modelo aditivo e multiplicativo, onde a_t corresponde à componente residual, b_t à componente de tendência, c_t à componente periódica e m é o tamanho do período:

$$x_{t+1} = a_t + b_t + c_{t+1-m} \quad (3)$$

$$a_t = \alpha(X_t - c_{t-m}) + (1-\alpha)(\alpha_{t-1} + b_{t-1}) \quad (4)$$

$$b_t = \beta(a_t - a_{t-1}) + (1-\beta)b_{t-1} \quad (5)$$

$$c_t = \gamma(X_t - a_t) + (1-\gamma)c_{t-m} \quad (6)$$

$$x_{t+1} = (a_t + b_t)c_{t+1-m} \quad (7)$$

$$a_t = \alpha(X_t \div c_{t-m}) + (1-\alpha)(a_{t-1} + b_{t-1}) \quad (8)$$

$$b_t = \beta(a_t - a_{t-1}) + (1-\beta)b_{t-1} \quad (9)$$

$$c_t = \gamma(X_t \div a_t) + (1-\gamma)c_{t-m} \quad (10)$$

Em [Brutlag 2000], o modelo aditivo da estimativa de Holt-Winters é usado para detecção de comportamentos anômalos numa série temporal contendo a taxa de bits por segundo do tráfego de saída na interface de um roteador.

Ainda de acordo com [Brutlag 2000], a medida dos desvios das estimativas de Holt-Winters pode ser indicada através de margens ou "bandas" de confiança, margens estas proporcionais ao desvio entre medida real e a estimativa computada para cada ponto no ciclo sazonal, modelando a variabilidade sazonal. A medida do desvio é uma media ponderada do desvio absoluto, atualizado através de *exponential smoothing*:

$$desvio_t = \gamma \left| X_t - x_t \right| + (1-\gamma)\ desvio_{t-m} \quad (11)$$

onde d_t é o desvio estimado no ponto temporal t.

A expressão de atualização de d é parecida com a de c_t e, inclusive, compartilham o mesmo parâmetro de adaptação γ.

As margens de confiança são a coleção de intervalos:

$$(x_t - \delta.\ desvio_{t-m}, x_t + \delta.\ desvio_{t-m}) \quad (12)$$

para cada ponto da série temporal y_t.

Os valores de escala para a largura das margens são δ_+ e δ_- e, como geralmente são desejadas margens simétricas, então $\delta_+ = \delta_-$. Ainda de acordo com [Brutlag 2000], com base em algumas suposições e na teoria de distribuição estatística, valores razoáveis para δ estão entre 2 e 3, com base em [Ward 1998].

3 Medições de Tráfego em *Backbones*

3.1 Arquitetura de medição típica de um *backbone*

O monitoramento de um *backbone* WAN é tipicamente centralizado, com algum nível de redundância para evitar que falhas nas conexões do centro de gerenciamento, ou NOC *(Network Operations Center*), deixem o *backbone* sem supervisão ou capacidade de ser controlado. A maioria das redes adota o modelo FCAPS da ISO para gerenciamento de redes, empregando servidores rodando sistema de monitoramento com suporte ao protocolo SNMP, para executar consultas periódicas aos principais equipamentos da rede controlados pelo NOC. As consultas SNMP, feitas tipicamente a cada cinco minutos, coletam dados relevantes para a operação da rede, auxiliando principalmente na detecção de falhas, no controle de configurações, no monitoramento do desempenho e na bilhetagem, quando for o caso.

As principais métricas de desempenho são monitoradas através de consultas SNMP em cada uma das principais interfaces de rede dos equipamentos roteadores e de *switches* do *backbone*, coletando as taxas de bits e pacotes por segundo, tanto de entrada quanto de saída. Tipicamente também são coletadas as taxas de erros de cada interface para monitorar a qualidade de funcionamento dos circuitos. Os dados coletados para cada uma das métricas são armazenados em bases – séries temporais – permitindo consultas aos valores armazenados e a representações gráficas da série. A coleção das últimas métricas coletadas fornece um retrato instantâneo do tráfego na rede, que podem ser apresentadas visualmente por ferramentas como os *network weather maps* ([Moura 2005]).

3.2 Limitações do monitoramento SNMP

O protocolo cliente-servidor SNMP, padrão para gerenciamento de redes, permite consultas específicas a informações permanentes e transitórias em dispositivos de redes – roteadores, *switches* e servidores – que o suportam.

Apesar das ferramentas de gerenciamento SNMP serem capazes de detectar falhas e de extrair informações de desempenho dos dispositivos de rede, elas não possuem recursos para correlacionar eventos ou para determinar os impactos de certos incidentes.

Tipicamente, para se executar certas atividades de correções de falhas em redes, geralmente se requer dos operadores análises adicionais em registros de *syslogs* e, em ações pró-ativas, são necessárias investigações para estabelecimento de *baselines* de certas métricas úteis para a operação normal da rede. A determinação dos *baselines* ajuda numa configuração de monitoramento que pode gerar alertas simples baseados em limiares (*thresholds*).

3.3 Medições passivas versus medições ativas

Algumas métricas de desempenho em redes podem ser mensuradas de forma ativa e outras métricas de forma passiva. Cada uma das formas de medição permite monitorar determinadas métricas, de acordo com o detalhamento e precisão desejados e com a infra-estrutura de *hardware* e *software* disponíveis.

Nas medições ativas são introduzidos pacotes na rede, que alteram o perfil do tráfego e, geralmente, permitem o monitoramento de métricas entre dois pontos da rede como: tempos de atraso e perdas de pacotes em um sentido, variação no atraso (*jitter*), tempo de resposta ida e volta (*Round Trip Time – RTT*), largura máxima de banda disponível, duplicação de pacotes e pacotes fora de ordem.

As medições passivas são feitas de forma a não modificar o perfil do tráfego na rede, efetuando a captura de pacotes por exportação de dados – *NetFlow* ou *sFlow* – ou por técnicas de "escuta" (*wiretapping*) ou espelhamento (técnica conhecida como *port mirroring ou port SPAN*), onde todos, ou parte, dos pacotes que entram ou saem por uma determinada interface são copiados integralmente ou parcialmente, possibilitando uma posterior análise mais profunda do tráfego na rede, a partir dos dados armazenados em um coletor. É importante observar que a técnica de espelhamento tem sérias limitações, por não estar amplamente disponível em todos os tipos de equipamentos de rede e não ser escalável para quaisquer larguras de banda, além da tecnologia geralmente suportar somente a escuta de uma única porta de cada vez.

Para analisar anomalias nos fluxos de tráfego é preciso adotar uma técnica de medição passiva que permita a análise de parâmetros dos pacotes trafegados, como a distribuição de IPs e portas de origem e de destino, e o protocolo de transporte, sendo as

tecnologias mais indicadas os protocolos *NetFlow*, *sFlow* e IPFIX.

3.4 Medição de fluxos

3.4.1 Definição de fluxo IP

Um fluxo IP é definido como uma seqüência unidirecional de pacotes onde cada pacote contém os mesmos valores para IP de origem, IP de destino, porta de origem, porta de destino e campo *protocol*. O intervalo de tempo entre pacotes de um mesmo fluxo não deve ultrapassar um valor máximo, que por *default* é igual a 15 segundos na maioria das implementações dos fabricantes de roteadores. Caso o intervalo de tempo ultrapasse este limite, o fluxo expira e um novo se inicia, conforme descrito em [Cisco 2008]. Outros critérios adotados para decretar o término de um fluxo IP são os pacotes RST e FIN, para conexões TCP, e o tempo de vida máximo do fluxo, geralmente configurado como sendo 30 minutos.

3.4.2 Formatos *NetFlow* e *SFlow*

A maneira mais otimizada de se capturar os fluxos IP que passam por um roteador é usar a capacidade destes roteadores de exportar esta informação. Trata-se de um recurso muito comum em roteadores de gerações mais recentes, porém nem sempre encontrada em roteadores de pequeno porte.

Dentre os padrões usados para estruturar as informações dos fluxos, podemos citar o *SFlow* ([Phaal 2001]) e o *NetFlow* ([Cisco 2008]). Destes, as versões 5 e 9 do padrão *NetFlow* ([Claise 2004]), desenvolvido e patenteado pela Cisco Systems em 1996, são as mais usadas. A versão 9 serviu como ponto de partida para um grupo de trabalho do IETF desenvolver um novo padrão, chamado IPFIX ([Leinen 2004]). Embora seja um formato proprietário da Cisco Systems, Inc., o *NetFlow* é um formato aberto e amplamente usado em equipamentos de diversos fabricantes.

O registro *NetFlow* de cada fluxo é composto pelos seguintes campos: versão do NetFlow, número seqüencial, interfaces de entrada e saída no roteador, carimbos de tempo de início e fim do fluxo, número de bytes e pacotes observados no fluxo, endereços IP de origem e destino, portas de origem e destino, campo *protocol*, valor do campo *Type of Service* (ToS) e, nos fluxos TCP, a união de todas as *flags* TCP observadas durante o tempo de vida do fluxo. A Tabela 3.1 apresenta dois exemplos de registro *NetFlow*. A captura destas informações é realizada para cada interface lógica do roteador mediante configuração. Os registros dos fluxos *NetFlow* são exportados

pelos equipamentos de rede usando pacotes UDP ou SCTP (*Stream Control Transmission Protocol*) que são coletados e armazenados em uma ou mais estações, chamadas de “coletores NetFlow”.

Hora início	Interface	IP orig.	Porta orig.	IP Dest.	Porta Dest.	Proto	Pacotes	Bytes	Flags TCP
1:20.12.221	26	172.16.10.4	1024	10.1.5.7	80	TCP	7	1025	SYN, ACK
1:20.12.871	35	192.168.1.5	1035	172.16.5.6	1434	UDP	19	29714	SYN

Tabela 3.1 - Exemplos de registro NetFlow

3.4.3 Arquitetura de medição

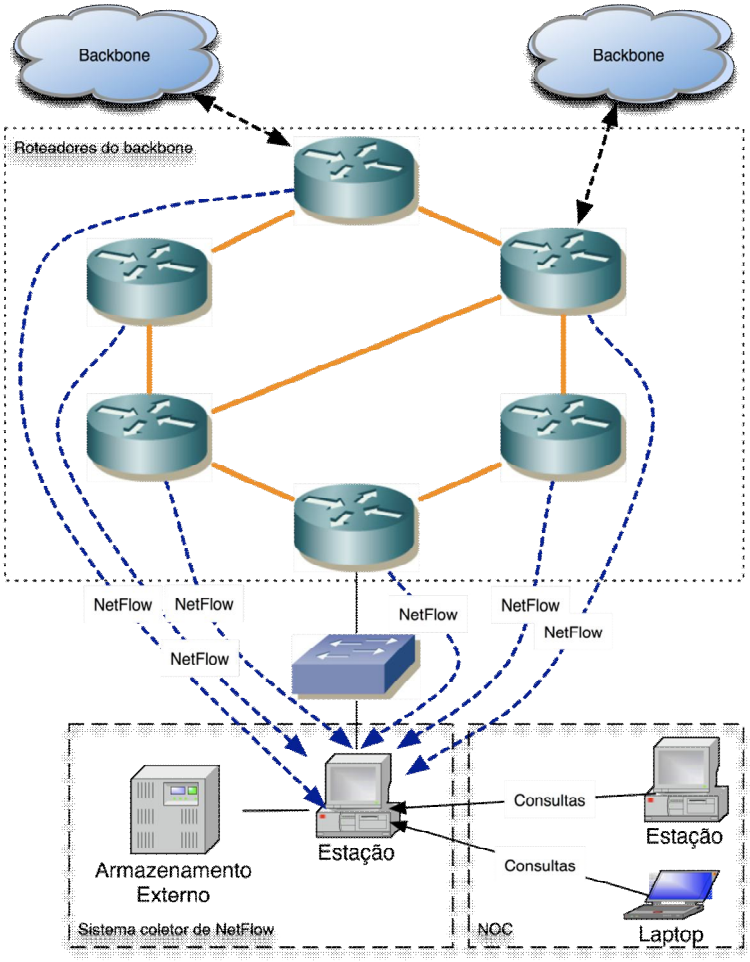


Figura 3.1 - Arquitetura de coleta de fluxos usando NetFlow

A coleta de fluxos *NetFlow* é executada através de uma infra-estrutura para

coleta, incluindo um ou mais servidores, com espaço de armazenamento interno – ou externo – suficiente para a guarda de dados durante um período de tempo considerado adequado pelo NOC para suas atividades ou, mais comumente, restrito pelo orçamento disponível para investimento na infra-estrutura de operações. A Figura 3.1 mostra uma arquitetura genérica para coleta de fluxos *NetFlow*.

Os equipamentos com suporte a *NetFlow* são todos configurados de acordo com a estratégia definida pelo NOC para a coleta. Há questões que devem ser previamente definidas como, por exemplo, em quais interfaces deve ser ativada a coleta de *NetFlow*, se esta coleta deve ser feita somente na entrada ou na saída das interfaces, ou em ambas, e, em seguida, os equipamentos devem ser configurados para exportar os fluxos *NetFlow* para os endereços IP do(s) servidor(es) coletor(es) para armazenamento.

O formato dos fluxos de *NetFlow* exportados é padronizado, mas o mesmo não acontece para armazenamento que, em termos de *software* livre, possui alguns padrões, como o *argus*, o *cflowd*, o *flow-tools*, o *lfapd* e o *nfdump*, sendo os mais populares o *cflowd*,o *flow-tools* e o *nfdump*. O *cflowd* é o formato de armazenamento originalmente implementado pela suíte *cflowd*, desenvolvido pela CAIDA ([CAIDA 1998]), posteriormente suportada pela ferramenta de análise *FlowScan*, desenvolvida e mantida por David Plonka ([Plonka 2000]), também autor do formato *flow-tools*, usado na suíte de ferramentas homônima. O formato *nfdump* é o usado pela suíte *Nfdump* desenvolvida por Peter Haag ([Haag 2005]). Cada formato só é lido pelas respectivas suítes de ferramentas e as principais diferenças entre elas estão nas características de compressão, otimização e no desempenho do armazenamento dos dados. Existem ferramentas para conversão dos formatos, particularmente do *flow-tools*, um padrão mais antigo, para o *nfdump*.

3.4.4 Top-N

Nos fluxos que atravessam a rede, existem os endereços de origem e/ou destino mais ativos (*top*), cuja listagem pode ser extraída dos dados de *NetFlow* através de geração de relatórios (*reports*) chamados de *Top-N*, onde "N" é o número de fluxos mais freqüentes segundo um determinado critério. Por exemplo, os dez endereços IP de origem mais freqüentes, ou os dez fluxos com a maior quantidade de bytes trafegados (*top talkers*). Relatórios contendo os *Top-N* endereços mais freqüentes de um determinado critério ajudam na investigação de anomalias, exibindo quais são os endereços responsáveis pela maior quantidade de fluxos, bytes ou pacotes transmitidos em determinado momento ou período.

Os relatórios *Top-N* não são indicados como método para gerar alertas de qualquer tipo de anomalia. Eles são gerados por ferramentas através das quais são configurados os critérios para seleção dos *N* mais freqüentes e esta configuração não se dá de forma automática, o que traz problemas de escalabilidade.

O método proposto neste trabalho sinaliza anomalias indicando os parâmetros que devem ser investigados mais apuradamente. Os relatórios *Top-N* gerados a partir dos parâmetros sinalizados podem exibir rapidamente as informações detalhadas sobre os fluxos anômalos – conforme exemplo da Tabela 3.1 – possibilitando a sua identificação e apoiando uma tomada de decisão, caso realmente se trate de um evento que está causando algum tipo de prejuízo, exigindo intervenção por parte do operador.

3.5 Abordagem network-wide

De acordo com [Lakhina 2004], a abordagem *network-wide* para detecção de anomalias se baseia numa visão completa da rede, e já foram investigadas algumas técnicas - [Soule 2005] e [Huang 2007] - com esta abordagem. Em [Lakhina 2005], utilizou-se o termo “par origem-destino” (OD *pair*) para indicar todos os fluxos que possuem um mesmo ponto de entrada (origem) e um mesmo ponto de saída (destino) na rede. Cada fluxo faz parte de um determinado par OD e as medidas de entropia são realizadas para todos os possíveis pares OD. Isto significa que, para cada par OD, as entropias de IP de origem, IP de destino, porta de origem e porta de destino são calculadas a cada cinco minutos de acordo com os fluxos que passam neste par. O resultado desta operação são quatro matrizes, uma para cada parâmetro, com os pares OD num eixo, a seqüência de intervalos de 5 minutos no outro eixo e as respectivas entropias em cada posição da matriz. Toda a metodologia para identificação e classificação das anomalias, utilizada pelos autores, baseia-se nestas matrizes.

No caso do uso de pares OD, a simples informação contida nos fluxos que passam por uma determinada interface nem sempre é suficiente. Faz-se necessário saber o ponto de entrada e de saída deste fluxo no âmbito do sistema autônomo (AS) e, caso esta interface não esteja na borda do AS, será necessário um ferramental mais complexo que considere topologia e tabelas de roteamento para ser possível saber que pontos são esses. Além disso, estas tabelas mudam dinamicamente por conta de variações, comuns a qualquer rede WAN, que são reportadas pelos protocolos de roteamento (quedas de enlace, manutenções, ampliações da rede, etc). Esta investigação torna-se mais difícil ainda quando se deseja saber, por exemplo, o ponto de entrada de um fluxo em um AS

multihomed, ou seja, que possua múltiplas conexões com um mesmo AS vizinho. Há também outros casos muito comuns onde, por exemplo, um AS possui múltiplos pontos de saída para outro AS, vizinho ou não, e várias políticas de balanceamento de rotas. Isto dificulta a estratégia para descobrir o ponto de saída de um dado fluxo IP.

Portanto, a construção das matrizes contendo a série temporal de entropias para cada par OD e, mais especificamente, a identificação do par OD para o qual um determinado fluxo pertence, adiciona um grau de complexidade razoável na implementação da solução. Torna-se necessário conhecer de antemão o ponto de entrada e de saída da rede a partir dos IPs de origem e de destino do fluxo, respectivamente, o que nem sempre é trivial. Outra conseqüência desta abordagem é a necessidade de um sistema com larga capacidade de armazenamento, de maneira que todos os fluxos que passam por todas as interfaces de entrada do AS sejam coletados para posterior análise.

3.6 Abordagem single-link

A abordagem *single-link* se restringe a analisar informações de fluxos IP colhidos para uma única interface da rede, que obviamente tenha relevância no que diz respeito ao tráfego que por ela passa. Por exemplo, a interface de rede que atende a um determinado cliente ou localidade, ou ainda a interface de rede para outro ISP, possivelmente um *upstream provider*.

Esta abordagem simplifica significativamente a implementação de uma solução capaz de sinalizar anomalias de um modo geral, principalmente em termos de sistemas de armazenamento e processamento. Além disso, em [Silveira 2008], é mostrado que a grande maioria das anomalias encontradas na abordagem *network-wide* é também encontrada na abordagem *single-link*. Uma restrição desta abordagem é que somente as anomalias que afetam a interface monitorada podem ser detectadas. Todavia, em [Lakhina 2005], os autores alegam que a metodologia usando pares OD amplia bastante as possibilidades de análise, como no caso da detecção de novas classes de anomalias (por exemplo, ataques do tipo *zero-day*).

4 Detecção de Anomalias usando Entropia e Estimativa de Holt-Winters

Conforme mencionado na seção 2, a técnica de detecção de anomalias em tráfego de redes usando entropia é relativamente recente, podendo ser vista em [Lakhina 2005], [MacKey 2003] e Lucena [2008]. Em [Lakhina 2005], a entropia de Shannon é usada para medir o grau de concentração de determinadas distribuições de probabilidade. No caso, o enfoque está nas distribuições dos números de portas e nas distribuições dos endereços IP, tanto de origem quanto de destino nos dois casos. Estas distribuições são obtidas a partir dos fluxos de pacotes agrupados e armazenados em arquivos, cada um contendo cinco minutos do total das amostras de tráfego, a partir dos quais serão construídas séries temporais. A concentração ou dispersão de uma ou mais das quatro distribuições serve como indicativo da presença de algum tipo de anomalia. Séries temporais construídas a partir das entropias medidas para cada um destes parâmetros de tráfego são usadas para identificar a ocorrência de anomalias, assim como para identificar os fluxos anômalos ([Lucena 2008]).

Um exemplo claro da aplicabilidade desta medida é o caso dos DDoS, ataques distribuídos de negação de serviço. Neste tipo de ataque, várias máquinas infectadas disparam pacotes para uma vítima específica. Na rede que conduz os atacantes à vítima, durante o ataque é possível notar a presença de vários fluxos com valores diversificados para o IP de origem e um mesmo valor para o IP e/ou porta de destino. Isto implica numa distribuição dispersa para os endereços IP de origem e concentrada para os endereços IP e/ou portas de destino destes fluxos ([Lakhina 2005]). Outro exemplo interessante é o caso dos ataques do tipo *port scan*. Neste tipo de ataque, uma determinada máquina executa uma varredura de diversas portas TCP e/ou UDP de uma ou mais máquinas de uma ou mais redes quaisquer buscando algum tipo de vulnerabilidade que seja um possível vetor de ataque. Desta forma, os fluxos durante o ataque apresentam distribuição concentrada para os valores de endereços IP, tanto de

origem como de destino, e distribuição dispersa para os valores de porta de destino ([Lakhina 2005]).

4.1 Motivação para o uso de medidas de entropia e estimativa de Holt-Winters

A simples adoção de medidas de entropia, por si só, não fornece um processo eficiente de detecção de anomalias. Faz-se necessário a uso de alguma técnica que seja capaz de verificar automaticamente, e em tempo real, se a medida de entropia fugiu de seu padrão de normalidade. Não fosse desta maneira, esta detecção teria que se dar visualmente, através de um operador de rede inspecionando as curvas de entropia.

Há formas bem simples de se obter um acionamento automático de alarme quando determinado evento provoca alteração no padrão de normalidade de uma dada medida. O melhor exemplo é o uso de limiares fixos: uma vez que a medida ultrapasse (para mais ou para menos, dependendo do caso) este limiar, tem-se a sinalização de um evento anômalo. Entretanto, é fácil perceber que tais limiares, na prática, não podem ser fixados. Isto porque eles variam de acordo com a dinâmica natural do tráfego, o que acaba se refletindo nas medidas extraídas. Este também é o caso das medidas de entropia.

Assim sendo, uma alternativa bastante empregada para se verificar mudanças de padrão numa série temporal, que por natureza tenham algum padrão intrínseco de variação, é usar métodos de *forecasting*, conforme mostrado na seção 2.4 e investigado em [Kiss 2008], [Lucena 2008] e [Lucena 2009]. Dentre os estimadores relacionados, a estimativa de Holt-Winters foi escolhida por apresentar duas características importantes dentro do escopo deste trabalho: capacidade para capturar sazonalidade em séries temporais e facilidade de implementação. Conforme dito anteriormente, este trabalho investiga o uso de estimadores tradicionais e de simples implementação, aplicados às medidas de entropia, para sinalizar a ocorrência de eventos que possam comprometer o bom funcionamento da rede e que não sejam facilmente detectados pelas ferramentas de gerência comumente usadas.

4.1.1 Foco em cenários de redes WAN

Detectar determinados tipos de anomalias,– principalmente em redes WAN,– é tarefa desafiadora devido à grande quantidade de dados que deve ser analisada em busca de padrões. Por esta razão, detecções baseadas em assinaturas estatísticas de tráfego são mais adequadas para grandes redes, do tipo *backbone* ([Estevez-Tapiador 2004]).

Através do comportamento dos fluxos de pacotes é possível verificar se há ou não a presença de alguma anomalia correlata, sem que para isso seja necessário inspecionar cada pacote IP trafegado. Este é o caso do método proposto em [Lakhina 2005].

Outra particularidade desafiadora das redes WAN é o escopo administrativo, onde, diferente das redes do tipo LAN, os administradores geralmente possuem acesso administrativo a apenas uma parte do conjunto total de *hosts* e ativos de rede pertencentes ao mesmo determinado sistema autônomo (AS). Este, por sua vez, pode ter uma grande quantidade de organizações clientes, como no caso de redes acadêmicas como a RNP. No caso da RNP, atualmente são mais de 600 instituições clientes e em torno de um milhão de usuários, o que torna a atividade de monitoramento, com cobertura a todas as sub-redes clientes do AS, algo extremamente difícil de ser feito por razões de escalabilidade. Mesmo considerando que o NOC (*Network Operation Center*) do AS pudesse ter acesso administrativo aos ativos de rede apenas para consultas *read-only* de informações, ainda assim haveria um dispêndio considerável de recursos para se executar estes serviços de monitoramento ativo com a qualidade mínima desejada. Por isso, técnicas como as investigadas neste trabalho são importantes, uma vez que oferecem um refinamento nos procedimentos de monitoramento e detecção de anomalias por analisarem somente uma fração do enorme – e sempre crescente – volume de tráfego que atravessa a rede.

4.1.2 Simplicidade da Análise

O uso de medidas de entropia para os fluxos coletados se constitui em importante ferramenta de alerta para administradores de rede. As propostas mais promissoras se baseiam em séries temporais contendo medidas de entropia para descrever padrões de tráfego. Diferente do que é mostrado em [Lakhina 2005], a abordagem aqui proposta é mais simples, se prestando apenas a sinalizar que determinado tipo de anomalia, dentre algumas de características mais conhecidas, pode estar em curso. Esta abordagem, em termos computacionais, é mais econômica e simples de implementar, face o grande volume de dados que precisam ser processados em um típico *backbone* WAN de alta velocidade, como as redes acadêmicas de pesquisa e educação atuais. Posteriormente, um analista de redes poderá usar outras ferramentas para uma investigação mais direcionada ao tipo de anomalia sinalizada. Pistas para a identificação do tipo de anomalia em curso podem ser conseguidas verificando-se o conjunto dos valores de entropia calculados para os quatro parâmetros (IPs e portas,

origem e destino) dentro do mesmo intervalo de tempo.

Para melhorar a eficiência do ferramental adotado pela maioria dos administradores das redes mencionadas, o método aqui proposto se valerá da ferramenta de software livre RRDTool ([Bogaerdt 2008]), amplamente usada como componente em diversas ferramentas orientadas à atividade de gerência de redes, tipicamente para armazenamento e exibição de dados históricos.

4.1.3 Adequação aos cenários típicos de gerenciamento SNMP

Do cenário típico de gerenciamento de redes WAN em várias instituições brasileiras, principalmente aquelas ligadas a governos, como as redes acadêmicas das IFES (Instituições Federais de Ensino Superior) e outras similares na América Latina e no mundo, podemos destacar as seguintes características:

- possuem soluções de gerenciamento de redes baseadas em *software* livre e arquitetura SNMP;
- monitoramento restrito às interfaces de rede dos equipamentos sob responsabilidade dos centros de gerência, sendo que na maioria destes centros não há visibilidade do sistema autônomo como um todo;
- histórico de estatísticas de tráfego, coletadas via SNMP, armazenado em bases RRD (Round-Robin Database), geralmente usando medidas extraídas a cada intervalo de 5 minutos;
- alarmes de ataques baseados em limiares pré-estabelecidos para a taxa de bits/s e pacotes/s nas interfaces monitoradas.

Conforme mostrado a seguir, a arquitetura proposta para a coleta de fluxos e medição da entropia é totalmente compatível com os cenários típicos de gerenciamento e faz uso de ferramentas de *software* livre que têm sido utilizadas com sucesso em várias redes acadêmicas no mundo. No caso, as ferramentas adotadas são o RRDtool ([RRDtool 2008]) e o pacote NfSen, NFdump e NFcapd ([Nfdump 2007]).

4.2 Arquitetura proposta para detecção de anomalias

A arquitetura proposta emprega recursos de ferramentas de gerenciamento de redes amplamente usadas pelos centros de operações de redes WAN. Dentre as tecnologias mais utilizadas para análise de tráfego, o *NetFlow* ([Cisco 2008]) é uma das mais empregadas e que oferece boa relação de quantidade de informações versus requisitos de processamento. Ao invés de se capturar todos os pacotes de tráfego de um

backbone, com *NetFlow* é possível trabalhar somente com parte das informações do cabeçalho IP e isto somente para uma porcentagem do total de pacotes trafegados. Isto permite uma otimização no processamento, permitindo uma análise em tempo real de um grande volume de tráfego, como o das redes WAN.

A partir de uma coleta de dados de *NetFlow*, que em geral podem ser obtidos de praticamente todos os roteadores de portes médio e grande dos diversos fabricantes, estes são armazenados em arquivos e pós-processados por ferramentas comerciais ou de *software* livre, como o *Nfdump* ([Nfdump 2007]). A partir destes arquivos, geralmente contendo cinco minutos de amostras de fluxos, já é possível efetuar o processamento de entropia e aplicação dos algoritmos estimadores, cujo resultado final são séries temporais armazenadas e representadas graficamente por ferramentas amplamente utilizadas para visualização de séries históricas, como o *RRDtool*. A Figura 4.1 ilustra a arquitetura proposta.

um sistema de coleta responsável por organizá-los em arquivos. Em geral, os sistemas coletores fazem a armazenagem dos dados dos fluxos em arquivos com cinco minutos de duração cada um. Caso um determinado fluxo fique ativo por mais do que cinco minutos, seus dados ficarão registrados em mais de um arquivo.

Em geral, os roteadores são configurados para usar amostragem de pacotes como forma minimizar a carga de processamento na geração dos registros dos fluxos. Isto é muito comum em redes que concentram um volume considerável de tráfego, como nos *backbones* WAN. Esta amostragem se dá capturando-se apenas um pacote a cada "tanto" que passa pela interface monitorada. Valores típicos são um para 100 ou mesmo um para 1000, dependendo do volume de tráfego. Outro motivo para se fazer amostragem é diminuir a quantidade de registros gerados e, conseqüentemente, usar menos espaço de armazenamento da estação coletora.

4.2.2 Extração da entropia

Conforme mencionado anteriormente, o cenário em que este trabalho se baseia considera sistemas de gerenciamento de redes onde o histórico das estatísticas de tráfego é armazenado em bases RRD (*Round-Robin Database*), geralmente usando medidas extraídas a cada intervalo de cinco minutos. Assim sendo, propõe-se que os registros *NetFlow* capturados tenham os valores de entropia calculados para cada intervalo de cinco minutos, e que os resultados sejam armazenados em bases do tipo RRD ([Bogaerdt 2008]), uma para cada parâmetro (IP de origem, IP de destino, porta de origem e porta de destino). O armazenamento em bases RRD permite que os administradores de rede tratem estas medidas a partir de ferramentas tradicionais de visualização, como o Cacti ([Cacti 2007]), ou mesmo de manipulação de bases RRD, como o *RRDtool* (RRDtool 2008), possibilitando a geração de alarmes conforme os recursos destas ferramentas. Vale notar que, em [Lakhina 2005], as medidas de entropia são também calculadas para intervalos de cinco minutos.

A entropia de cada parâmetro, a cada intervalo de cinco minutos, é calculada contabilizando-se todos os pacotes de todos os fluxos registrados neste intervalo, montando-se os histogramas de cada parâmetro e aplicando-se a expressão descrita em (1). De maneira a uniformizar o grau de concentração/dispersão informado pela medida de entropia, os valores calculados são normalizados por $log_2 N$, onde N corresponde ao número de ocorrências do respectivo parâmetro para cada intervalo de cinco minutos. No caso de não se fazer a normalização, a cada intervalo de cinco minutos poderá haver

um novo valor de $log_2 N$ e, portanto, um limite superior diferente dos demais para a respectiva entropia. Todavia, na prática, é esperado que estes diversos limites superiores guardem pouca diferença entre si devido ao ajuste pelo logaritmo.

Diferente do trabalho em [Lakhina 2005], deseja-se analisar um sistema cujo objetivo se resume a sinalizar que determinado tipo de anomalia, dentre algumas mais conhecidas, pode estar em curso. Posteriormente, o administrador de rede poderá usar outras ferramentas para uma investigação mais direcionada ao tipo de anomalia sinalizada. A identificação do tipo de anomalia em curso pode ser conseguida verificando-se o conjunto dos valores de entropia, calculados para os quatro parâmetros dentro do mesmo intervalo de tempo. Segue abaixo o exemplo de uma possível classificação a partir dos quatro valores de entropia:

- DoS: entropia baixa para IP de origem (origem específica), entropia baixa para IP de destino (alvo específico);
- DDoS: entropia alta para IP de origem (origem dispersa), entropia baixa para IP de destino (alvo específico), entropia alta para porta de origem (portas aleatórias);
- Port Scan: entropia baixa para IP de destino (mesmo IP com portas sendo varridas), entropia alta para porta de destino (muitas portas sendo varridas);
- Proliferação de Worms: entropia alta para IP de origem (possível *Botnet*), entropia alta para IP de destino (procura por possíveis vítimas), entropia alta para porta de origem (várias conexões para múltiplos destinos), entropia baixa para porta de destino (explora a vulnerabilidade de alguns serviços).

O processo para obtenção das entropias para os quatro parâmetros em cada uma das interfaces listadas começa pela utilização do *Nfdump*. Como a captura de amostras de fluxos é armazenada em vários arquivos e cada arquivo gerado pelo *Nfcapd* (ferramenta que faz captura dos fluxos *NetFlow* exportados pelos roteadores), por padrão, possui todos os fluxos ativos no roteador durante um período de cinco minutos, usando o *Nfdump* pode-se filtrar os fluxos capturados na entrada de uma interface específica e, ao mesmo tempo, obter um dos parâmetros desejados (IP de origem, IP de destino, porta de origem e porta de destino). O comando é repetido para cada um dos quatro parâmetros, gerando arquivos separados contendo todos os valores ocorridos em cada fluxo durante um intervalo específico. A partir daí, usa-se um programa escrito

para calcular as entropias de cada parâmetro para cada intervalo de cinco minutos.

4.2.3 Aplicação de estimadores e verificação das anomalias

A estimativa de Holt-Winters (HW) foi calculada utilizando uma função especial do *RRDtool* que realiza esta operação, conforme descrito em [Brutlag 2000]. A partir da série de predições do HW, o *RRDtool* calcula um limiar superior e inferior para o que pode ser considerado como comportamento normal. Ou seja, se o valor de uma nova amostra de entropia está fora deste intervalo, é sinal de uma anomalia. O cálculo deste intervalo nada mais é do que um *exponential smoothing*, ou EWMA, para o valor de desvio, que é a diferença absoluta entre valor estimado e valor real. No caso, este *exponential smoothing* considera o ciclo sazonal da série temporal e usa o mesmo γ como coeficiente de amortização:

$$desvio_t = \gamma \left| X_t - x_t \right| + (1-\gamma)\ desvio_{t-m} \qquad (11)$$

onde X_t é o valor real e x_t o valor estimado. Assim, limiares superior e inferior limitam o intervalo

$$(x_t - \delta.\ desvio_{t-m}, x_t + \delta.\ desvio_{t-m}) \quad (12)$$

onde δ é um fator multiplicador, geralmente com valor entre 2 e 3 (ver [Brutlag 2000] e [Ward 1998]).

Os resultados gerados para os desvios são armazenados numa fila circular cujo tamanho também é um parâmetro do *RRDtool* e seu valor deve ser maior que *m*.

4.2.3.1 Verificação de anomalias a partir de limiares proporcionais

Outra forma, mais simples, de estabelecer limares superior e inferior para o padrão de comportamento normal da medida de entropia é a adoção de uma margem percentual em torno da estimativa computada. Assim, caso a medida real esteja, por exemplo, mais que 10% acima ou abaixo da estimativa, considera-se então que há uma anomalia.

4.3 Ferramentas de apoio

4.3.1 RRDtool

O *RRDtool* é uma ferramenta padrão na indústria de *software* livre que faz uso de

uma base de dados de alto desempenho, chamada *Round-Robin Database*, ou RRD, e fornece um sistema gerador de gráficos para dados de séries temporais. Além disso, versões mais recentes desta ferramenta já incorporam a estimativa de Holt-Winters para as séries temporais, incluindo a verificação de desvio usando EWMA, conforme descrito na seção 4.2.3. O *RRDtool* é amplamente empregado como componente de ferramentas tanto comerciais como de *software* livre, portanto gratuitas.

4.3.2 NfSen e Nfdump

Das ferramentas de *software* livre que processam informações de fluxo, uma das mais populares é o *NfSen* ([Haag 2005]). O *NfSen* é uma ferramenta gráfica, de acesso via *browser*, que proporciona uma interface de alto nível para o uso das funcionalidades da ferramenta *Nfdump* e visualização dos resultados gerados. É desenvolvido por Peter Haag na SWITCH, rede acadêmica de pesquisa e educação da Suíça, e distribuído gratuitamente com licença de uso BSD. É comum usar o *NfSen* para verificar se há uma porta desconhecida dentre as *n* mais usadas, o que pode ser um indicativo de algum *worm* se alastrando. Ou ainda para olhar os fluxos IP de maior volume em busca de algum possível ataque do tipo DoS.

O *Nfdump* é um conjunto de ferramentas para coleta e processamento de dados *NetFlow* versões 5, 7 e 9, sendo que sua interface é por linha de comando (*Command Line Interface*, ou CLI). O *Nfdump* é capaz de filtrar e extrair diversas informações de fluxos *NetFlow* armazenados pelo programa *nfcapd* ([Nfdump 2007]). O *Nfdump* é parte do projeto *NfSen*, que é composto pelas ferramentas *nfcapd*, *nfdump*, *nfsen*, *nfprofile*, *nfreplay*, *nfclean* e *ft2nfdump*. Destas, o *Nfcapd* (*daemon* de captura *NetFlow*) e o *Nfdump* (*NetFlow dump*) são as ferramentas usadas neste trabalho.

4.3.3 Scripts para medição e análise

Para este trabalho foram desenvolvidos *shell scripts* e programas em linguagem Python para processamento dos dados *NetFlow* obtidos através da ferramenta *Nfdump* e para a geração das séries temporais de entropia. Os *scripts* executaram as ações de:

- filtragem e armazenamento em arquivos dos dados provenientes dos fluxos em formato *NetFlow*;
- injeção artificial de dados obtidos para amostras de tráfego anômalo nos dados de tráfegos potencialmente livres de anomalias, para fins de validação do método;

- cálculo das entropias para cada uma das séries amostrais geradas;
- automação de processamentos para geração dos gráficos.

4.3.4 Escolha de parâmetros

A escolha dos parâmetros influencia diretamente o quão próxima ou afastada será a estimativa, conseqüentemente a verificação de anomalias. Neste trabalho, esta escolha se baseou nos valores usados em [Brutlag 2000], com alguns ajustes empíricos. No caso do HW, alguns pontos relacionados à parametrização podem ser destacados:

- de maneira a diminuir o número de parâmetros, pode-se assumir que γ é igual a α;
- é fácil constatar que a taxa de crescimento da série temporal, associada ao parâmetro β, tem pouca influência em cenários práticos;
- é razoável assumir que a sazonalidade das séries de entropia tenham período igual a 24 horas, uma vez que estas guardam algum tipo de relação com a periodicidade de uso das redes.

Seguindo estas considerações, pode-se concluir que o parâmetro fundamental para a estimativa de HW passa a ser o α, e esta abordagem diminui bastante a complexidade na parametrização deste estimador.

4.3.5 Alarmes de eventos suspeitos

Alarmes de eventos suspeitos podem ser gerados após o processamento das entropias e do estimador em uso. Os arquivos das bases RRD do *RRDtool*, com o recurso de Holt-Winters habilitado, possuem cinco estruturas de dados associadas aos registros arquivados: HWPREDICT, SEASONAL, DEVPREDICT, DEVSEASONAL e FAILURES:

- HWPREDICT: armazena um *array* de estimativas computadas pelo algoritmo de Holt-Winters, um para cada ponto;
- SEASONAL: trata-se de um *array* de coeficientes sazonais com comprimento igual ao do período sazonal onde, para cada ponto, o coeficiente sazonal que for equivalente ao índice no ciclo sazonal será atualizado;
- DEVPREDICT: trata-se de um *array* de desvios das estimativas para onde, essencialmente, são copiados valores do *array* DEVSEASONAL para preservar o histórico, sem executar nenhum processamento;
- DEVSEASONAL: trata-se de um *array* de desvios sazonais onde, para cada

PDP, o desvio sazonal que for equivalente aos índice no ciclo sazonal é atualizado;

- FAILURES: trata-se de um *array* de indicadores booleanos, sendo o valor 1 indicativo de falha, armazenados num buffer circular de maneira que cada atualização remove o valor mais antigo deste *buffer* e insere uma nova observação.

Sendo assim, uma forma de verificar indicações de anomalias em um determinado momento é através da consulta dos valores no FAILURES. Para tal, o *RRDtool* possui uma maneira de fazê-lo através de linha de comando. Isto significa que é possível desenvolver uma aplicação, ou *script*, que execute esta leitura periodicamente.

4.3.6 Tempos de resposta em cenários reais

Foi observado que o método pode ser considerado praticamente de tempo real, a exemplo das tradicionais medições feitas a cada cinco minutos por sistemas de gerenciamento de rede aderentes à arquitetura FCAPS, que tipicamente se valem de consultas via protocolo SNMP aos equipamentos ativos da rede.

O tempo de indicação de uma anomalia por este método pode ser considerado rápido, com pequenas variações no tempo de processamento conforme a quantidade de dados *NetFlow* contidas em um determinado arquivo representando cinco minutos de captura. Este conteúdo pode variar de dezenas a centenas de MB, dependendo da intensidade do tráfego que estiver passando pelas interfaces do roteador no momento da exportação dos fluxos.

Nos tempos medidos durante o processamento do método, implementado por código escrito na linguagem Python, o tempo de processamento das entropias para arquivos contendo cinco minutos de fluxo variou de acordo com o tamanho dos arquivos. Foi observada uma média de 0,21 segundos de processamento para cada arquivo de uma seqüência amostral totalizando 10 dias de dados *NetFlow*. Ao todo, foram 12.673 arquivos totalizando aproximadamente 7GB (7.363.380 bytes), com tamanhos variando entre 1MB e 11MB. A duração total de processamento do método foi de aproximadamente 45 minutos em um notebook Apple Macbook, CPU Intel Core 2 Duo de 2.4 GHz, 4GB RAM e HD SATA externo Samsung S2, conectado por USB 2.0, com 500GB de capacidade, 5.400 RPM, 8MB cache e barramento Serial ATA/300. Segue abaixo a saída dos *scripts* de processamento contendo o tempo de

execução:

```
==== Calculando entropias e gerando séries temporais ====
Inicio do processamento: Qua 16 Set 2009 20:03:34 BRT
Final do processamento: Qua 16 Set 2009 20:48:45 BRT
Duração: 00:45:11
 ==== Gerando graficos das séries temporais ====
Inicio do processamento: Qua 16 Set 2009 20:48:45 BRT
Final do processamento: Qua 16 Set 2009 20:48:47 BRT
Duração: 00:00:02
```

Tabela 4.1 - Tempos de Processamento

5 Resultados obtidos e Análise Comparativa

Este capítulo aborda todos os aspectos relacionados com a validação experimental do método proposto.

5.1 Metodologia experimental

A metodologia para a validação experimental do método proposto se consiste em obter dados reais de anomalias confirmadas, ou ainda gerados artificialmente, para que estes sejam inseridos numa seqüência de dados oriunda de um tráfego supostamente livre de anomalias (aqui chamado de "tráfego de fundo"). Desta forma, é possível verificar se tais anomalias conseguem ser corretamente detectadas pelo método.

5.1.1 Dificuldades no uso de amostras reais de tráfego de *backbone*

Uma das dificuldades encontradas para executar os experimentos desta pesquisa foi a grande quantidade de dados amostrais que necessitavam ser processados. Em uma primeira captura de dez dias de amostras *NetFlow* coletadas para seis interfaces de 1 Gbps localizadas num mesmo roteador e usadas para *peerings* da RNP com outras redes WAN, e ainda com a taxa amostragem de 1:100 usada na rede Ipê, os arquivos gerados, incluindo arquivos de captura, arquivos com IPs e portas, arquivos contendo as entropias e os arquivos RRD, totalizaram aproximadamente 68GB de dados.

Outra dificuldade inerente ao uso de dados reais é a falta de confirmação da presença ou não de anomalias no tráfego amostrado, o que gera complicações na validação da eficácia do método. Eventos confirmados de anomalias na rede Ipê são possíveis apenas quando uma vítima notifica o centro de operação da rede (*Network Operation Center*, ou NOC), o que não é tão freqüente.

Para se obter amostras de tráfego real com anomalias confirmadas foi necessário solicitar ao Centro de Engenharia e Operações (CEO) da RNP, que mantém seus atendimentos registrados em um sistema de *trouble-ticket*, uma busca por ocorrências recentes de algum ataque que tenha sido reportado por algumas das instituições clientes

da RNP. A partir destas informações, foi verificado se as amostras de fluxo de uma das interfaces pela qual o ataque passou – amostras estas relacionadas ao período do ataque – se encontravam na base de dados do CEO e, em caso positivo, fez-se uma solicitação formal para acesso destes dados. A partir de informações sobre as características do ataque, como origem, porta de destino e rede vitimada, filtrou-se os fluxos do ataque uma vez que, originalmente, estes se encontravam misturados ao tráfego lícito no arquivo de captura.

Já para a obtenção de amostras de tráfego livres de anomalias, utilizou-se de expediente similar, mas desta vez verificando qual interface apresentava menos indícios de ataques dentre aquelas que possuíam informações de fluxo armazenadas na base de dados do CEO. Feita a escolha, mais uma vez foi solicitado formalmente o acesso a estes dados.

5.1.2 Ataques artificialmente gerados

Face à dificuldade de se obter amostras reais de ataques, foi usada também uma ferramenta *web* que se vale de códigos maliciosos conhecidos para gerar ataques segundo parâmetros especificados pelo usuário. Esta ferramenta pode ser encontrada em *www.pcapr.net* ([pcapr 2009]) e, a partir dela, é gerado um arquivo descrevendo o ataque desejado. Este arquivo deve ser baixado e, a partir dele, gera-se um ataque real com auxílio da ferramenta *Mu Dos* (mudos, [Mu Dynamics 2009]). O *mudos* lê o arquivo com a descrição do ataque, gera os respectivos pacotes e os envia para a rede de destino. Através de um aplicativo de *sniffing*, como *tcpdump* ([Jacobson 1987]) e *tshark* ([Combs 1998]), faz-se a captura do ataque. Sendo assim, é necessário tomar cuidado para não ter nenhum outro aplicativo gerando tráfego para a interface de rede monitorada, de maneira que somente o ataque esteja presente no arquivo *pcap* ([Jacobson 1987]) gerado pelo *tcpdump* ou pelo *tshark*. O passo seguinte para se obter os arquivos de fluxo é dividir o arquivo *pcap* gerado em arquivos menores, cada qual contendo cinco minutos de tráfego. Esta operação pode ser realizada através de um *script* perl chamado *pcap-util* ([Boddington 2006]). Após a geração dos vários arquivos *pcap* contendo cinco minutos de ataque, é possível aplicar amostragem de pacotes, segundo uma taxa específica, utilizando o comando *pcapdump*, que pertence ao pacote de ferramentas *pcaputils* ([Edmonds 2007]). Após esta etapa, são executados mais dois programas para gerar os arquivos no formato *NetFlow* versão 5: o *softflowd* ([Miller 2002a]) e o *flowd* ([Miller 2002b]).

O *softflowd* é um programa capaz de ler um arquivo *pcap* e, a partir dele, exportar conteúdo *NetFlow* versão 5 para um determinado IP e porta. O *flowd* é um *daemon* que escuta os pacotes *NetFlow* em uma determinada porta e os armazena em um arquivo. Após a gravação de cada arquivo, foi executado o comando *flowd-reader* para ler e filtrar os parâmetros necessários ao método para o cálculo das entropias - IPs e portas de origem e destino - gravando cada métrica em arquivos separados, mantendo os *timestamps* dos arquivos originais.

5.1.3 Considerações sobre a injeção artificial de ataques

É importante ressaltar que a injeção artificial de dados referentes a tráfego anômalo em dados oriundos de tráfegos supostamente livres de anomalia não elimina o problema de como fazer a identificação de falsos positivos. Em termos práticos, isto significa que qualquer sinalização de anomalia fora do período de injeção pode ser um falso positivo ou uma anomalia que, de fato, estava presente no tráfego de fundo e não se tinha conhecimento.

Para ser possível a injeção das amostras de ataques reais, foi necessário um tratamento preliminar das amostras destes tráfegos. Primeiramente, todos os arquivos contendo os fluxos IP do ataque continham também outros fluxos IPs referentes ao tráfego normal da interface. Portanto, foi necessário identificar os fluxos do ataque para extraí-los dos respectivos arquivos. Em seguida, o número de pacotes computado para cada fluxo atacante foi multiplicado por um fator corretivo de maneira que o volume do ataque não fosse, em média, superior a 25% do volume médio do tráfego de fundo. Este valor foi empiricamente escolhido de maneira a evitar que o ataque injetado tenha um volume de pacotes grande demais, o que o tornaria facilmente detectável por simples inspeção da taxa de pacotes do tráfego resultante. Por fim, as amostras de ataque, já processadas, são adicionadas ao tráfego de fundo de forma bastante simples.

Analisar o método proposto sem o artifício da inserção artificial de uma anomalia previamente conhecida, num determinado ponto do tráfego amostrado, traria uma incerteza sobre qualquer anomalia indicada pelo método, uma vez que não seria possível afirmar ser um falso positivo ou não. Não há garantias de que todo e qualquer ataque presente num *backbone* como rede Ipê tenha sido percebido e registrado pelos operadores de rede. Portanto, a estratégia usada foi inserir uma anomalia previamente confirmada num tráfego supostamente livre de anomalias para, então, verificar se o método consegue perceber a anomalia introduzida.

Vale ressaltar que não há necessidade de que o tráfego de fundo e o tráfego de ataque tenham sido extraídos para uma mesma interface de rede. O mais relevante é garantir que o tráfego de fundo seja livre de anomalias, ou aproximadamente isto, e que a anomalia inserida tenha as características desejadas, tanto no tipo quanto no volume.

5.2 Cenário de teste

5.2.1 Estrutura da rede Ipê

A rede Ipê é o *backbone* acadêmico brasileiro operado pela RNP e projetado para atender demandas das comunidades de pesquisa, educação ciência e tecnologia do Brasil, oferecendo a largura de banda necessária de tráfego para acesso à Internet comercial com os serviços tradicionais de navegação *web*, correio eletrônico e transferência de arquivos, bem como a serviços avançados como telefonia, videoconferências e transmissões de vídeo em alta definição, computação distribuída e colaboração entre grades computacionais localizadas dentro do país e no exterior.

A rede Ipê possui 27 pontos de presença (PoPs) nas principais capitais e mais o Distrito Federal, interligando aproximadamente 600 instituições de ensino e pesquisa e redes de alcance regional e metropolitano. Possui circuitos com larguras de banda variando entre 4Mbps e 10Gbps, possui conectividade internacional própria, conexão à rede avançada latino-americana CLARA, conexões em pontos de troca de tráfego (PTTs) metropolitanos e acordos de troca de tráfego (*peering*) com outros *backbones* comerciais nacionais. As Figuras 5.1 e 5.2 apresentam, respectivamente, o mapa geográfico da rede Ipê e um retrato de seu panorama de tráfego, onde fica evidente a topologia “lógica” da rede e a carga de tráfego dos enlaces.

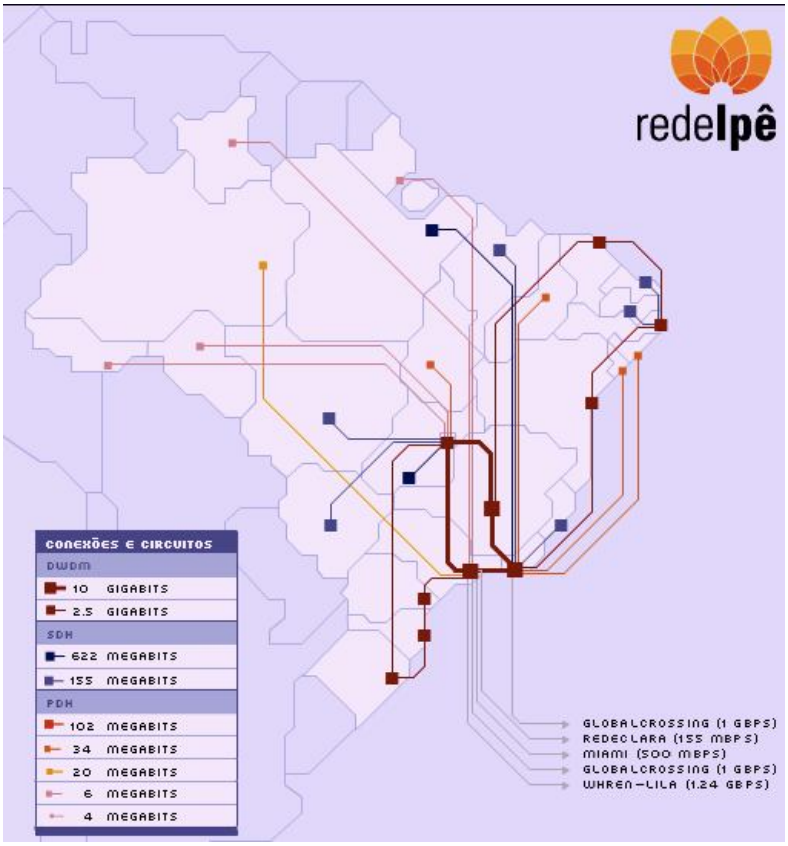


Figura 5.1 – Mapa da rede Ipê

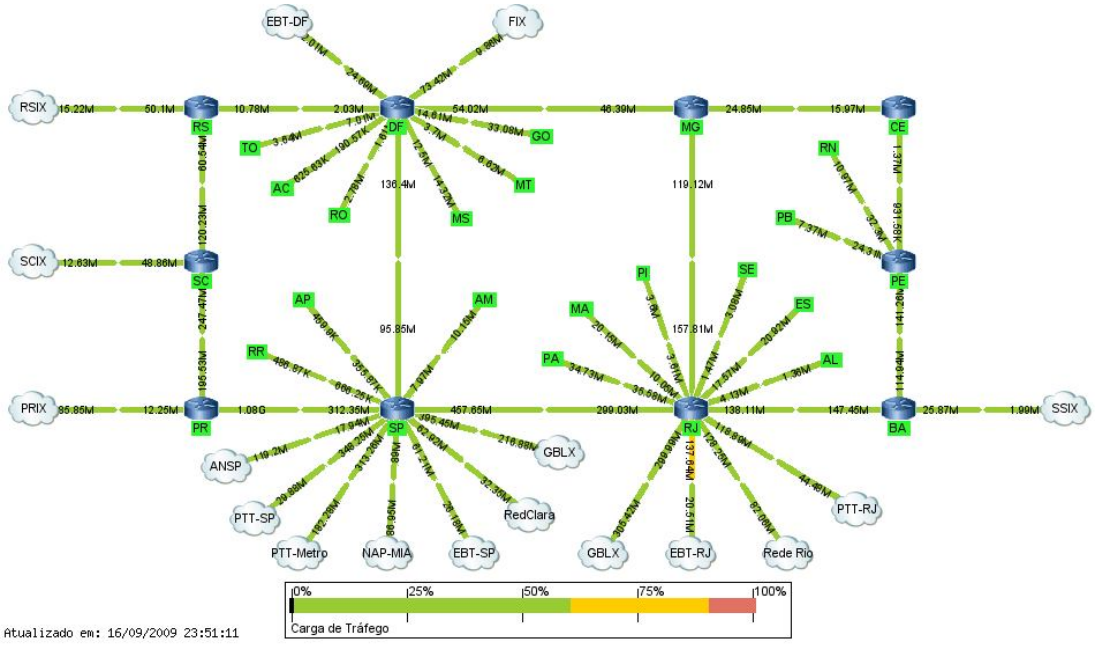


Figura 5.2 - Panorama do tráfego na rede Ipê

5.2.2 Gerenciamento da rede Ipê

O gerenciamento da rede Ipê é executado pelo CEO da RNP, que atua como o NOC e cujas atividades são apoiadas por diversas ferramentas baseadas em *software* livre. As ferramentas empregadas pelo CEO aderem à arquitetura FCAPS da ISO, onde é extensivamente utilizado o protocolo SNMP para coleta de informações de estado dos elementos ativos da rede, como roteadores e comutadores (*switches*). Esta coleta, por padrão na grande maioria das ferramentas, é feita a cada cinco minutos de forma contínua, ao longo de todos os dias do ano.

As principais métricas monitoradas pelo NOC através das ferramentas de monitoramento são as taxas de *bits* por segundo (bps), pacotes por segundo (pps) e falhas de funcionamento em todas as interfaces e em componentes dos chassis dos equipamentos, incluindo as respectivas durações (*outages*). Também são feitas medidas de tempos de resposta – RTT – entre pontos distintos da rede. A partir destas métricas, são inferidas outras métricas de avaliação da qualidade e desempenho da rede, como o percentual de disponibilidade (e de indisponibilidade) e tempo de entrega de pacotes entre quaisquer dois pontos da rede. O conjunto destas medições compõe o relatório a ser apresentado para o MCT, para apreciação da conformidade com o acordo de nível de serviço (SLA) contratado.

5.2.3 Coleta de fluxos na rede Ipê

No caso da RNP, o sistema usado para coleta dos registros *NetFlow* é o *Nfcapd*. O *Nfcapd* armazena em arquivo os registros referentes a todos os fluxos IP que estavam ativos num certo intervalo de cinco minutos [Nfdump 2007]. Ou seja, a cada período de cinco minutos, um novo arquivo contendo os registros dos fluxos IPs ativos é gerado. Este processo de armazenamento não realiza nenhum tipo de média que implique na perda de detalhamento das informações registradas. O único elemento que ocasiona perda na informação coletada é a taxa de amostragem dos pacotes, que captura apenas o primeiro pacote de uma seqüência contendo um número definido de pacotes que passam pela interface num dado sentido (entrada ou saída, dependendo da configuração). A taxa de amostragem *NetFlow* empregada nas amostras recebidas da RNP é de 1:100, ou seja, do volume total de pacotes que atravessaram as interfaces de um cada roteador da rede, foram exportados os dados *NetFlow* de um pacote de cada 100 pacotes.

5.2.4 Extração das entropias

O processo para obtenção das entropias começa pela utilização do *Nfdump*. Cada arquivo gerado pelo *Nfcapd* possui todos os registros referentes a todos os fluxos capturados para todas as interfaces do roteador que tenham sido configuradas para tal, isso para cada intervalo de cinco minutos. Através do *Nfdump* é possível filtrar os fluxos capturados na entrada de uma interface específica e, ao mesmo tempo, obter um dos parâmetros desejados (IP de origem, IP de destino, porta de origem e porta de destino). O comando é repetido para cada um dos quatro parâmetros, gerando arquivos separados contendo todos os valores ocorridos em cada fluxo durante um intervalo específico. A partir daí, usa-se um programa escrito para calcular as entropias de cada parâmetro para cada intervalo de cinco minutos. A Figura 5.3 ilustra o processo de extração de entropia.

A forma de armazenamento do *Nfcapd* não obriga que as entropias sejam computadas para cada período de cinco minutos. Optou-se por manter este intervalo para que a granularidade dos gráficos gerados seja igual a dos gráficos utilizados pela grande maioria dos sistemas de gerenciamento de redes. Além disso, esta é a mesma granularidade usada em [Lakhina 2005].

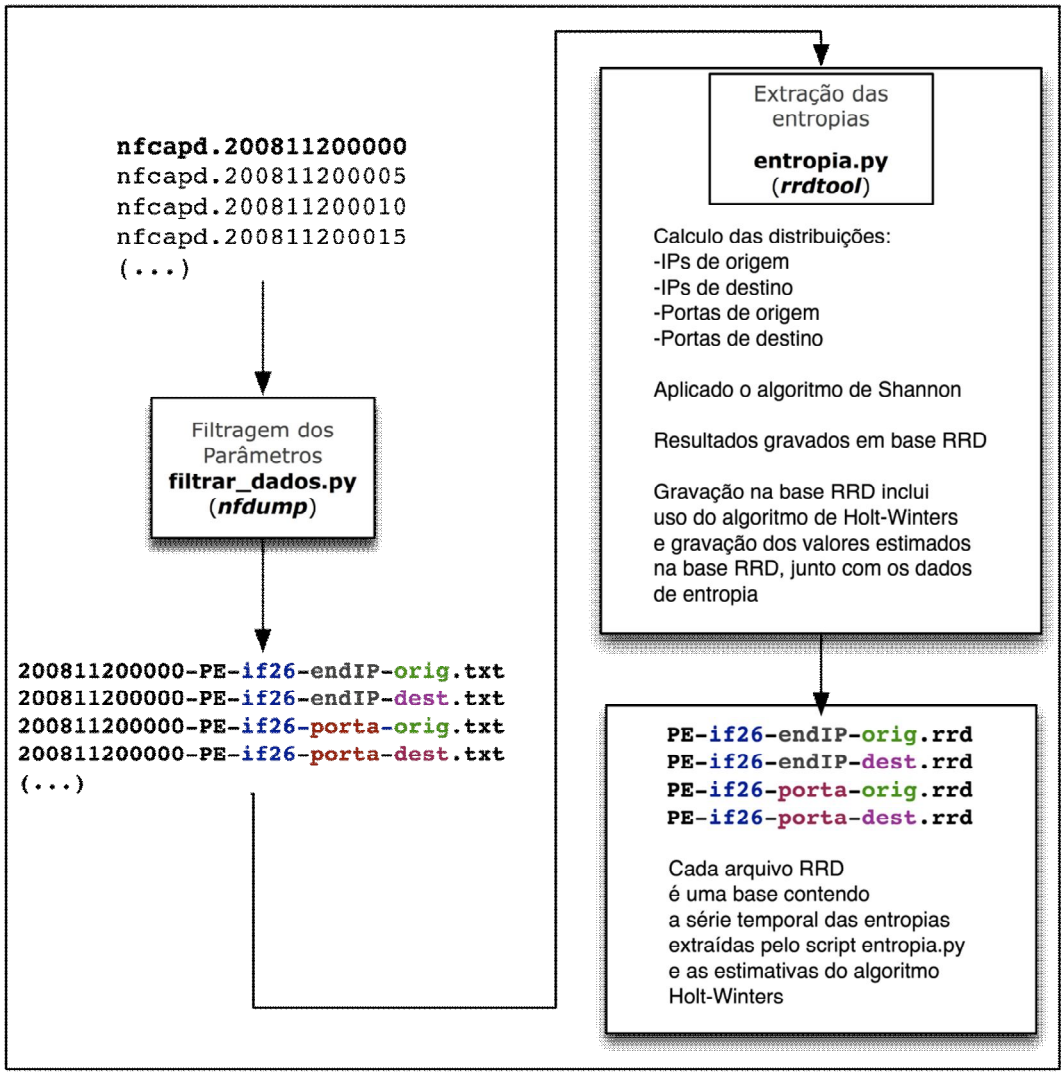


Figura 5.3 - Processo de extração das entropias

5.3 Amostras de tráfego

5.3.1 Tráfego de fundo

As amostras de tráfego da rede Ipê, representando um período de normalidade (não houve indícios de ataques massivos no período amostrado), foram obtidas do enlace de 2,5 Gbps entre BA e PE, sentido PE, das 00h do dia 20/11/2008 às 00h do dia 01/12/2008, totalizando dez dias de amostragem. A captura destas seqüências usou registros *NetFlow* versão 5 obtidos da respectiva interface do roteador modelo Juniper M40 localizado no ponto de presença do Estado de Pernambuco (PoP-PE). A taxa média de pacotes por segundo medida para estas amostras é de 300 pps, o que significa

que, devido à amostragem de 1:100, o tráfego de fundo possui taxa média real de 30 Kpps.

5.3.2 Amostras reais de ataque

O ataque amostrado foi um DoS que ocorreu das 16:50h do dia 26/05/2008 às 12:40h do dia 27/05/2008, totalizando 19h50 de duração, no enlace de 2,5 Gbps entre SC e RS, sentido RS. Este ataque foi informado ao Centro de Engenharia e Operações (CEO) da RNP, a pedido dos administradores da rede vitimada, para que este fosse bloqueado. Assim como para o tráfego de fundo, a captura destas seqüências usou registros *NetFlow* versão 5 da respectiva interface do roteador modelo Juniper M40 localizado no ponto de presença do Estado do Rio Grande do Sul (PoP-RS). Na injeção deste ataque, o volume de pacotes foi alterado para que a taxa média em pacotes por segundo seja igual a 25% da taxa média do tráfego de fundo, ou seja, 75 pps, o que corresponderia a 7,5 Kpps de tráfego real.

É importante mencionar que todas as informações da rede Ipê referentes a tráfegos reais, tanto as de ataque como as usadas para tráfego de fundo, foram gentilmente cedidas pelo Centro de Engenharia e Operações da RNP mediante solicitação formal.

5.3.3 Amostras de ataques artificialmente gerados

Conforme descrito na seção 5.1.2, os ataques artificialmente gerados usando as ferramentas *pcapr* e *mudos* foram o *TCP SYN flood* e o *worm Slammer*.

O *TCP SYN flood* é um ataque de negação de serviço (DoS), podendo ter origem distribuída (portanto um DDoS), que procura exaurir os recursos computacionais da vítima através da criação de múltiplas seções TCP. Os parâmetros usados para gerar o ataque foram os seguintes: IPs de origem randômicos a partir da rede 192.0.2.245/22, portas de origem randômicas, IP de destino 192.0.2.2/32 e porta de destino 80/TCP, com duração de 1,5 h (5.400 segundos) e taxa de pacotes por segundo variando entre 4.000 e 6.000 pps. Estas taxas correspondem a uma variação entre 13,3% e 20% da taxa média real do tráfego de fundo. De maneira a manter a coerência com as amostras de fluxo para o tráfego de fundo, durante a geração dos fluxos deste ataque foi usada uma taxa de amostragem de 1:100.

O *worm Slammer* é um ataque que explora uma vulnerabilidade do SQL. Os parâmetros para a geração deste ataque foram os seguintes: IP de origem 192.0.2.223/32, IPs de destino randomizados, portas de origem aleatória e porta de

destino 1434/UDP, com duração de 2 h (7.200 segundos) e taxa de pacotes por segundo fixa em 500 pps. Para este caso, foi usada uma taxa de amostragem de 1:10 durante a geração dos fluxos deste ataque, o que faz com que a taxa média considerada para este ataque corresponda a 16,6% da taxa média real do tráfego de fundo.

5.3.4 Uso de filtros para aumentar a eficácia na identificação das anomalias

Como a medida de entropia depende das distribuições de IPs e portas dos fluxos amostrados, um ataque de grandes proporções, cujo volume de pacotes seja muito maior que o volume médio do tráfego normal, causará uma mudança por demais abrupta nestas distribuições que será evidenciada da mesma forma na medida de entropia. Por outro lado, usando da mesma analogia, ataques de baixa intensidade podem causar variações muito sutis na medida de entropia, fazendo com que o ataque não consiga ser percebido pelos métodos automáticos de detecção.

Um recurso que pode ser usado para evitar este problema, no caso dos ataques de baixa intensidade, é o de filtrar o tráfego monitorado de maneira a somente considerar fluxos que possuam características comuns aos ataques que se deseja identificar. Por exemplo, no caso do *worm Slammer*, pode-se aplicar o método apenas a fluxos que usem o protocolo UDP. Ou no caso de ataques direcionados a servidores *web*, pode-se extrair a entropia apenas de fluxos direcionados à porta 80. Este recurso foi testado para as três amostras de ataque injetadas e os resultados encontram-se na seção 5.4.

5.4 Resultados

Em todos os testes realizados, foram usados os mesmos valores para os parâmetros da estimativa de Holt-Winters no *RRDtool*. Estes valores foram atribuídos de forma empírica, baseados em sugestões encontradas em [Brutlag 2000]. São eles: $\alpha = 0{,}01$; $\beta = 0{,}0035$; $\gamma = 0{,}01$; $\delta = 2$ e $m = 288$ (equivalente ao número de conjuntos de 5 minutos contidos em um dia). De mesma forma, todos os resultados gerados utilizaram um tamanho de fila circular igual a 1440, equivalente a cinco dias, para armazenamento do EWMA aplicado ao desvio entre a estimativa de Holt-Winters e a medida real.

O motivo para se usar este mesmo grupamento de parâmetros é que todos os testes usaram o mesmo tráfego de fundo. Ou seja, os parâmetros são escolhidos de maneira que a estimativa de Holt-Winters seja a mais aproximada possível da medida

real de entropia do tráfego de fundo. Entretanto, quando do caso da aplicação de algum tipo de filtro no tráfego de fundo, estes parâmetros poderiam ser novamente ajustados – principalmente α, que exerce maior influência. Todavia, por razões de simplicidade, todos os parâmetros foram mantidos para os exemplos envolvendo filtragem.

Tendo como base as datas e horas referentes ao tráfego de fundo, todos os ataques foram injetados às 00h do dia 27/11/2008. Apesar dos dez dias de coleta, para uma melhor visualização as figuras mostram apenas uma janela que vai do dia anterior à injeção do ataque até três dias após o final do mesmo. Entretanto, o cálculo das estimativas considerou toda a série amostral. Os limiares do critério de normalidade estão identificados pelas linhas escuras dos gráficos. A linha mais fina, de cor vermelha, indica a entropia calculada em cada gráfico. A linha mais cheia dos gráficos indica a respectiva estimativa.

Exemplos de resultados para estimativas com estimativas de EWMA e Holt-Winters

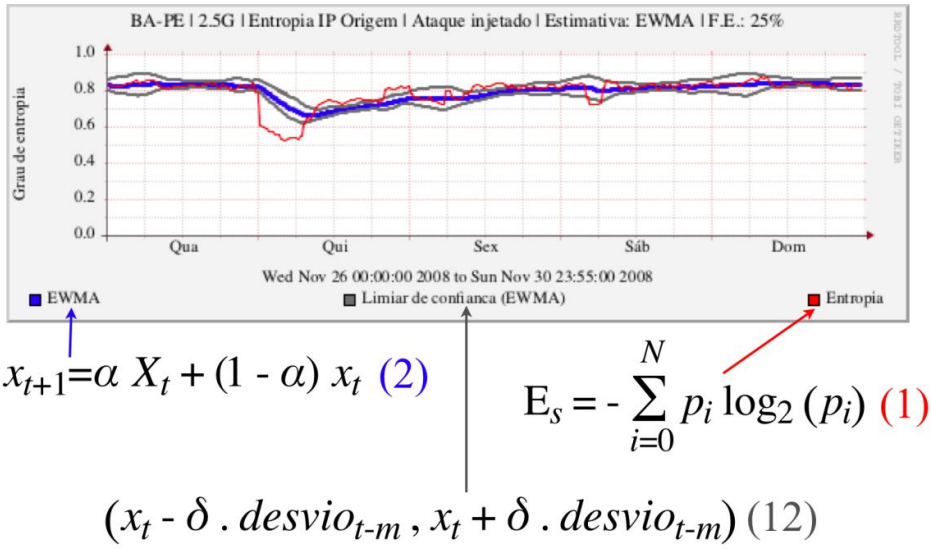
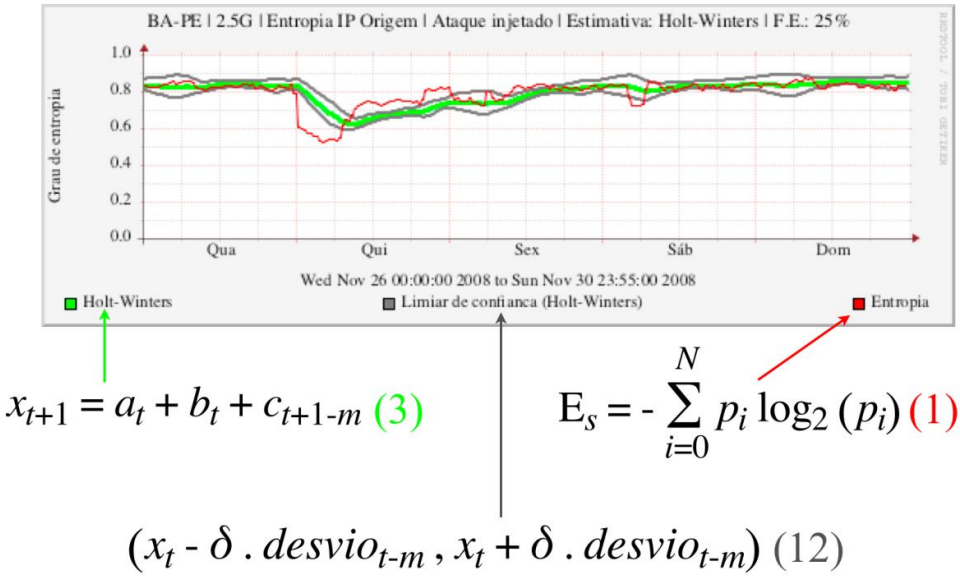


Figura 5.4 - Exemplo de resultados gerados com estimativas de EWMA



$$x_{t+1} = a_t + b_t + c_{t+1-m} \quad (3)$$

$$E_s = -\sum_{i=0}^{N} p_i \log_2 (p_i) \quad (1)$$

$$(x_t - \delta \,.\, desvio_{t-m} \,,\, x_t + \delta \,.\, desvio_{t-m}) \quad (12)$$

Figura 5.5 - Exemplo de resultados com estimativas de Holt-Winters

5.4.1 Ataque real injetado no tráfego de fundo

A Figura 5.6 mostra a aplicação do método para o teste da injeção de ataque real – no caso, um DOS.

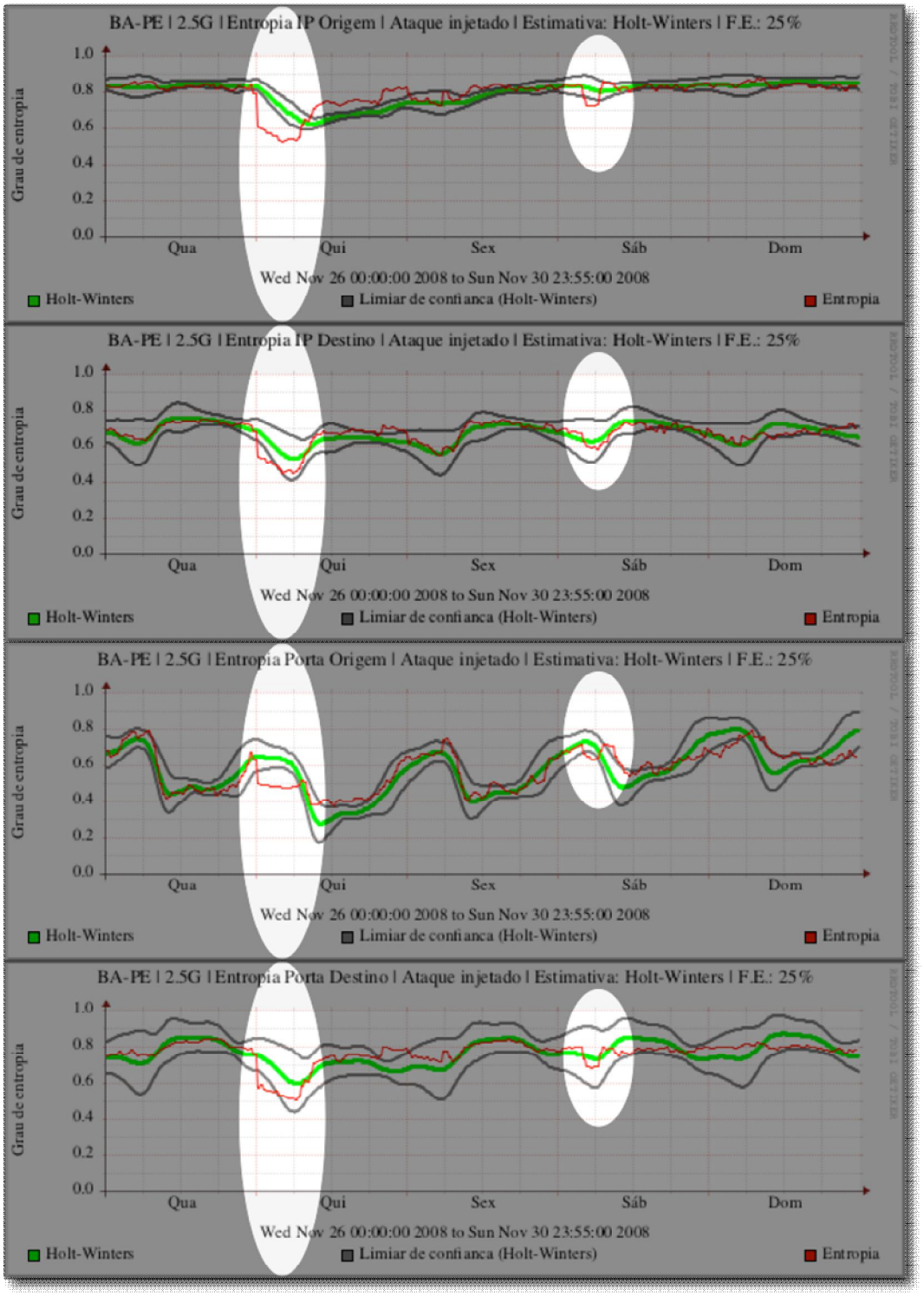


Figura 5.6 - Resultados de ataque DOS injetado em amostra de fundo BA/PE

Em todos os gráficos da Figura 5.6 é possível verificar o início do ataque através de uma queda abrupta nos valores de entropia no mesmo momento, em todas as séries

temporais. Isto é coerente com os dados do ataque, já que o resultado é uma grande concentração de pacotes com mesmo IP de origem, IP de destino, porta de origem e porta de destino. O retorno à "normalidade" do tráfego de fundo se dá cerca de 20 h depois, porém de forma gradual. Interessante verificar que a variação da entropia durante este período indica uma respectiva variação na intensidade do ataque, o que é bastante comum quando este tem duração longa. Outro detalhe interessante é que, por volta das 06 h de sábado, pode-se notar em todos os gráficos uma anomalia não registrada pela operação da Rede Ipê.

Ataque real injetado no tráfego de fundo, teste variando γ

A **Figura** 5.7 mostra como fica a estimativa de Holt-Winters com um valor de γ diferente de α. No caso, o valor de γ aumentou para *0,1*, ficando dez vezes maior, mas a diferença entre esta predição e a da Figura 5.6 é quase imperceptível, o que mostra a pouca influência de γ.

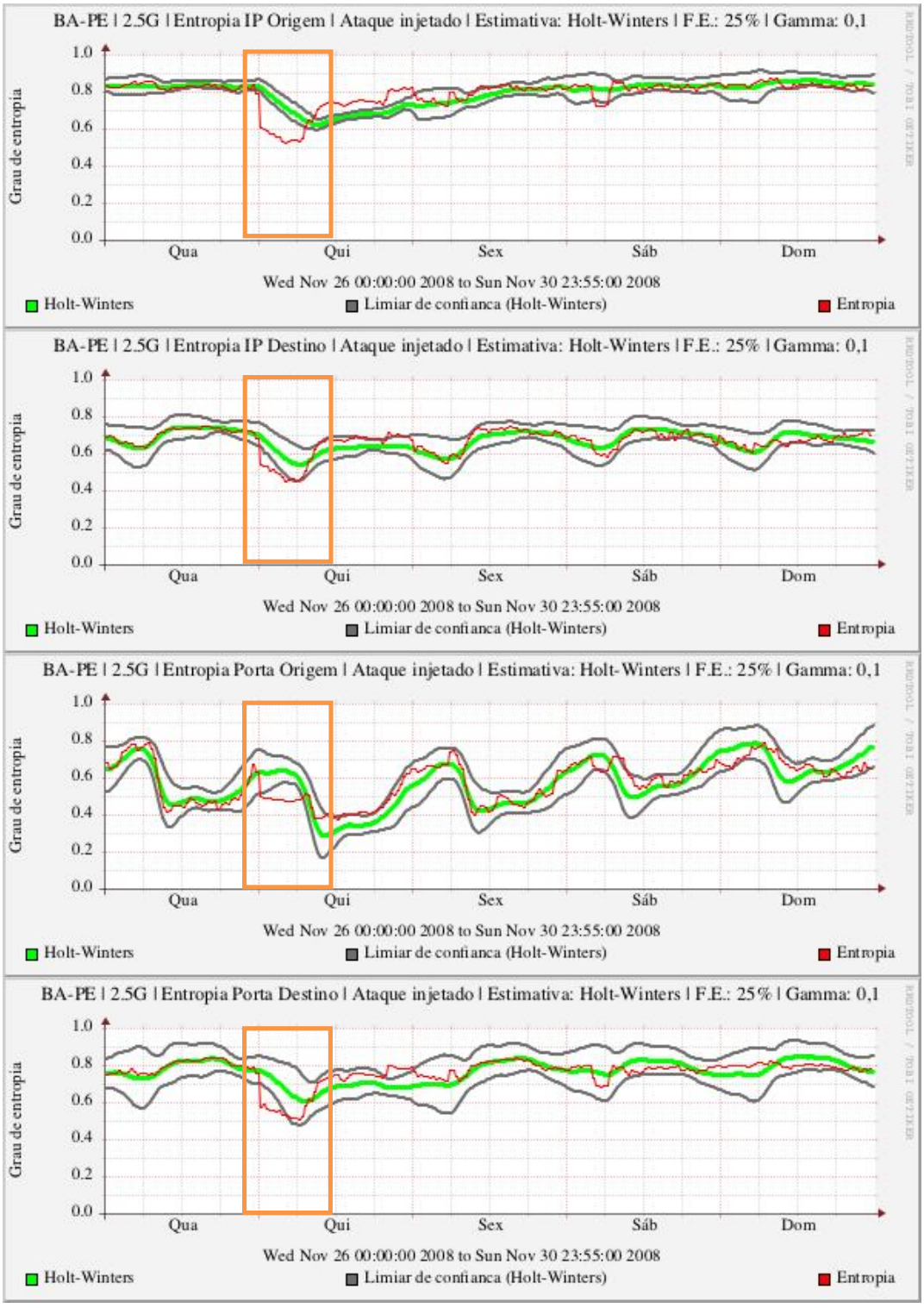


Figura 5.7 - Ataque DoS injetado no tráfego de fundo com variação em γ

Ataque real injetado no tráfego de fundo, uso do EWMA como estimador

A **Figura** 5.8 mostra a aplicação do EWMA como uma forma mais simples de estimador. Para tal, foi escrito um programa para ler os valores das bases RRD, contendo as medidas de entropia, e gerar a seqüência correspondente às predições do EWMA. Estes valores são também armazenados numa base RRD e, a partir daí, o *RRDtool* é usado para calcular os limiares superior e inferior que definem o critério de normalidade, tal qual realizado para o HW. O valor usado para α é o mesmo que nos testes usando HW. O valor de γ, usado no EWMA para cálculo do desvio, passa a ser igual a α.

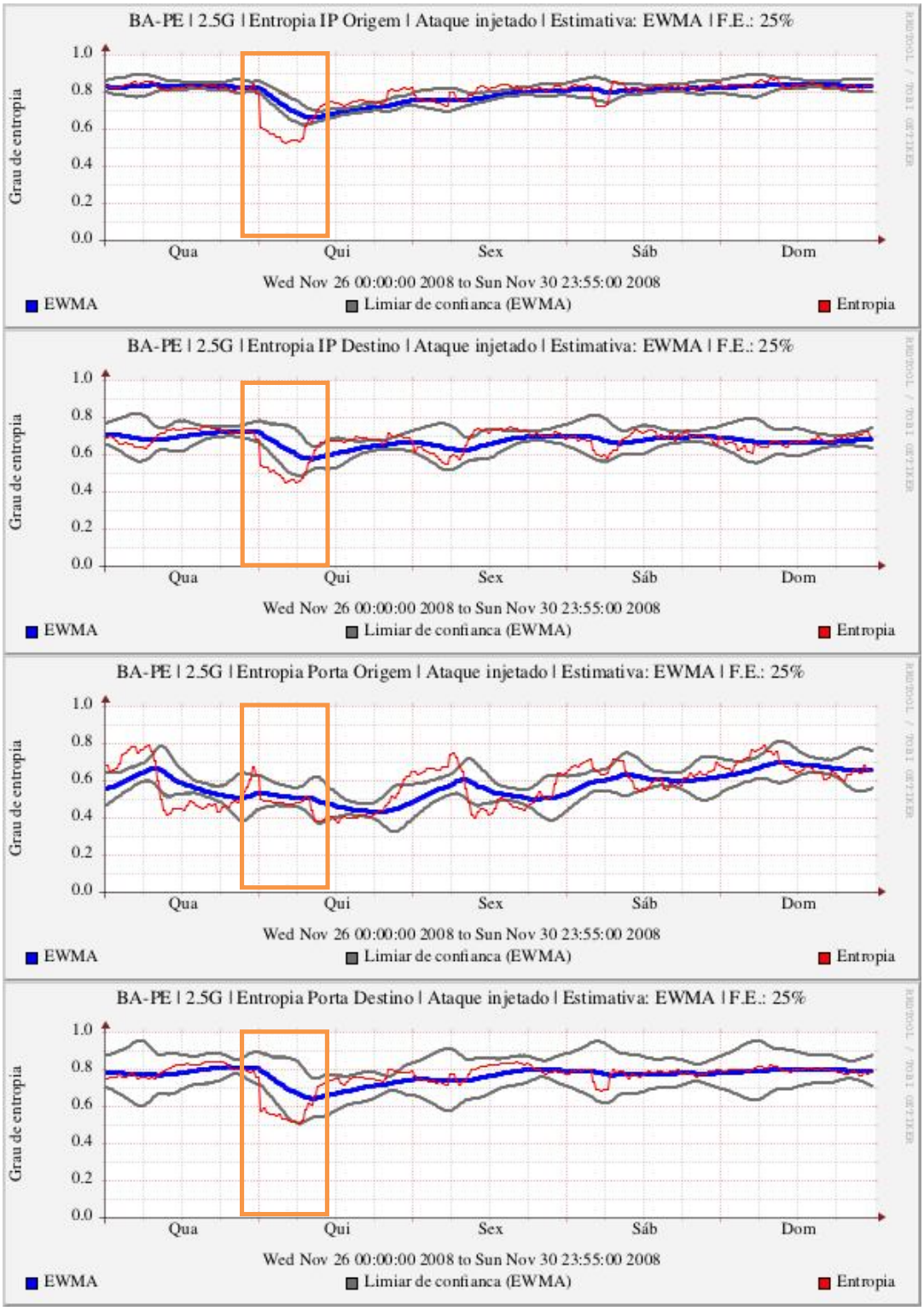


Figura 5.8 - Ataque DoS injetado em tráfego de fundo com estimativas de EWMA

Como era de se esperar, a estimativa usando HW conseguiu aproximar melhor a

sazonalidade diária das entropias medidas. Isto fica mais evidente na medida referente à porta de origem, cuja componente sazonal é mais intensa. A explicação está no comportamento dos usuários que, durante o horário comercial, acessam com mais freqüência diversos serviços em portas conhecidas. Estes serviços, por sua vez, respondem com vários pacotes contendo estes mesmos valores conhecidos na porta de origem, o que favorece a concentração. Para os demais parâmetros (porta de destino, IP de origem e IP de destino), esta sazonalidade não é tão presente, o que possibilitaria o uso do EWMA como estimador. Todavia, o HW "copia" de forma muito mais eficaz o padrão de entropia e é a única opção que se adéqua à entropia para a porta de origem.

5.4.2 TCP SYN flood injetado no tráfego de fundo

A **Figura** 5.9 ilustra o uso do método para a identificação do ataque *TCP SYN flood*.

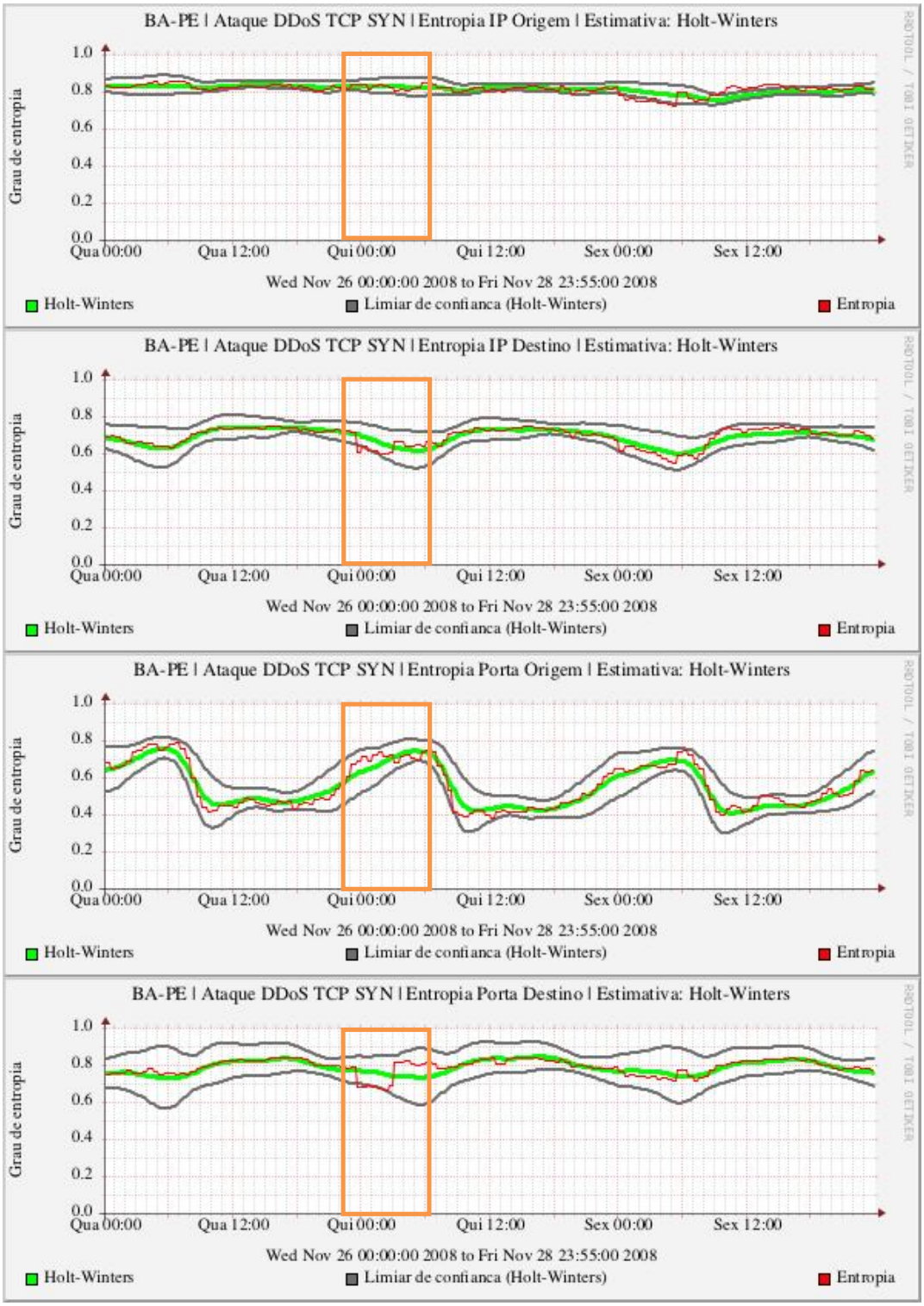


Figura 5.9 - Ataque DDoS TCP SYN *Flood* injetado no tráfego de fundo

Conforme pode ser observado nas medidas de entropia para IP de destino e porta de destino, o ataque fica evidenciado pela forma como a entropia reduz abruptamente seu valor, indicando uma concentração destes parâmetros. A estimativa de HW acompanha a tendência normal da entropia, podendo ser usada para indicar o momento do ataque. No entanto, a indicação baseada em EWMA do desvio, com os parâmetros adotados, não foi eficiente. Para este caso, uma indicação baseada em limiares proporcionais em torno da estimativa é mais adequada.

Apesar deste ataque ser do tipo DDoS, ou seja, com IP de origem distribuído, a randomização deste parâmetro ficou limitado a uma rede /22, conforme descrito na seção 5.3.3.

TCP SYN flood injetado no tráfego de fundo, filtro pelo protocolo TCP

A **Figura** 5.10 mostra como ficam os resultados do método quando aplicado somente aos fluxos de tráfego que usam protocolo TCP. Conforme conhecimento comum, boa parte dos fluxos na Internet usam o protocolo TCP. Por conta disto, este filtro não implica numa seleção de alguma característica mais particularizada para este tipo de ataque e, por conseguinte, os gráficos praticamente se mantiveram como antes.

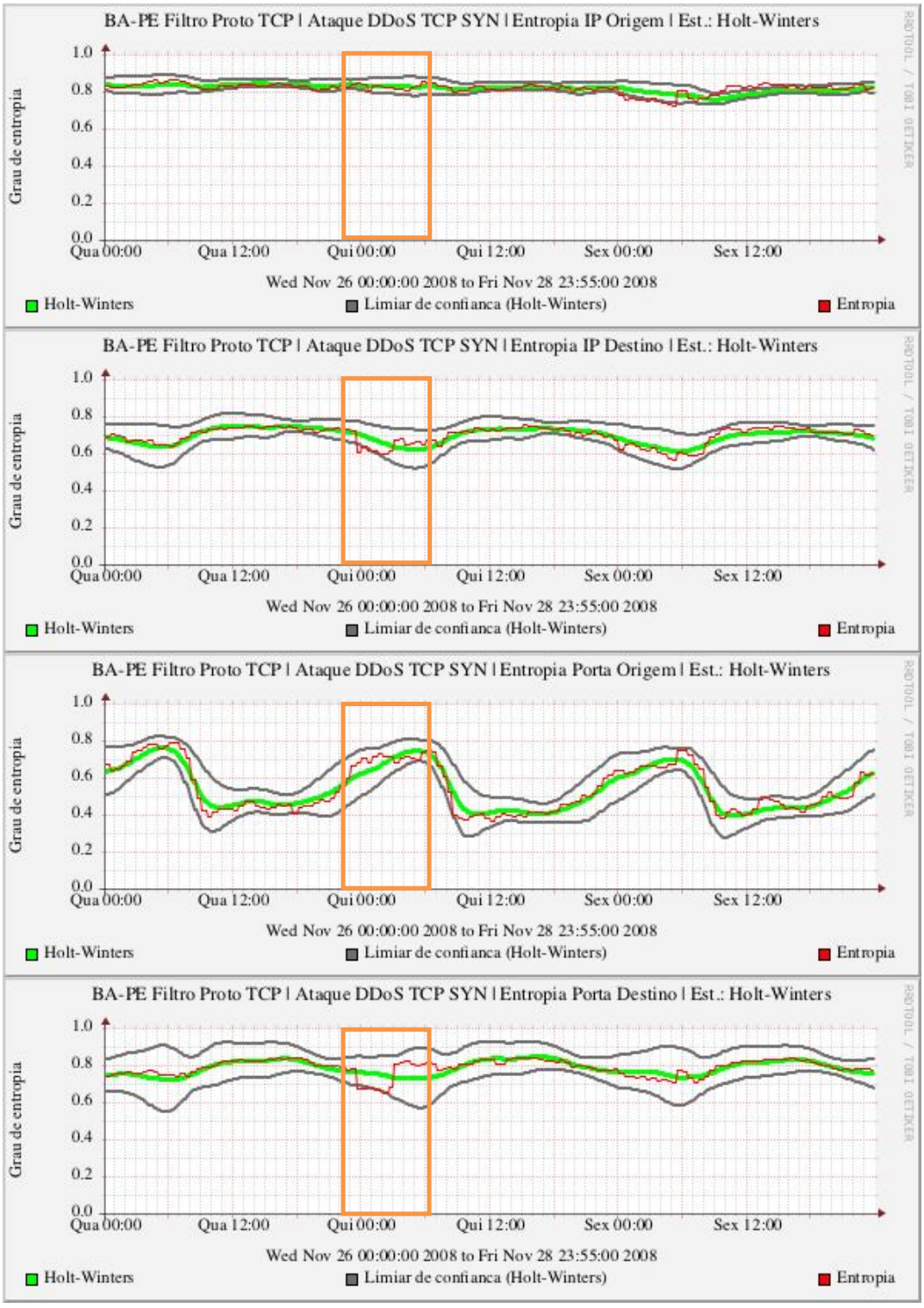


Figura 5.10 - TCP SYN flood injetado em tráfego filtrado por protocolo TCP

TCP SYN flood injetado no tráfego de fundo, filtro pela porta destino 80

A Figura 5.11 mostra como ficam os resultados do método quando aplicado somente aos fluxos de tráfego com porta de destino 80. Neste caso, é possível observar que a entropia da porta de destino será sempre zero, uma vez que há concentração máxima da entropia durante todo o período da amostra.

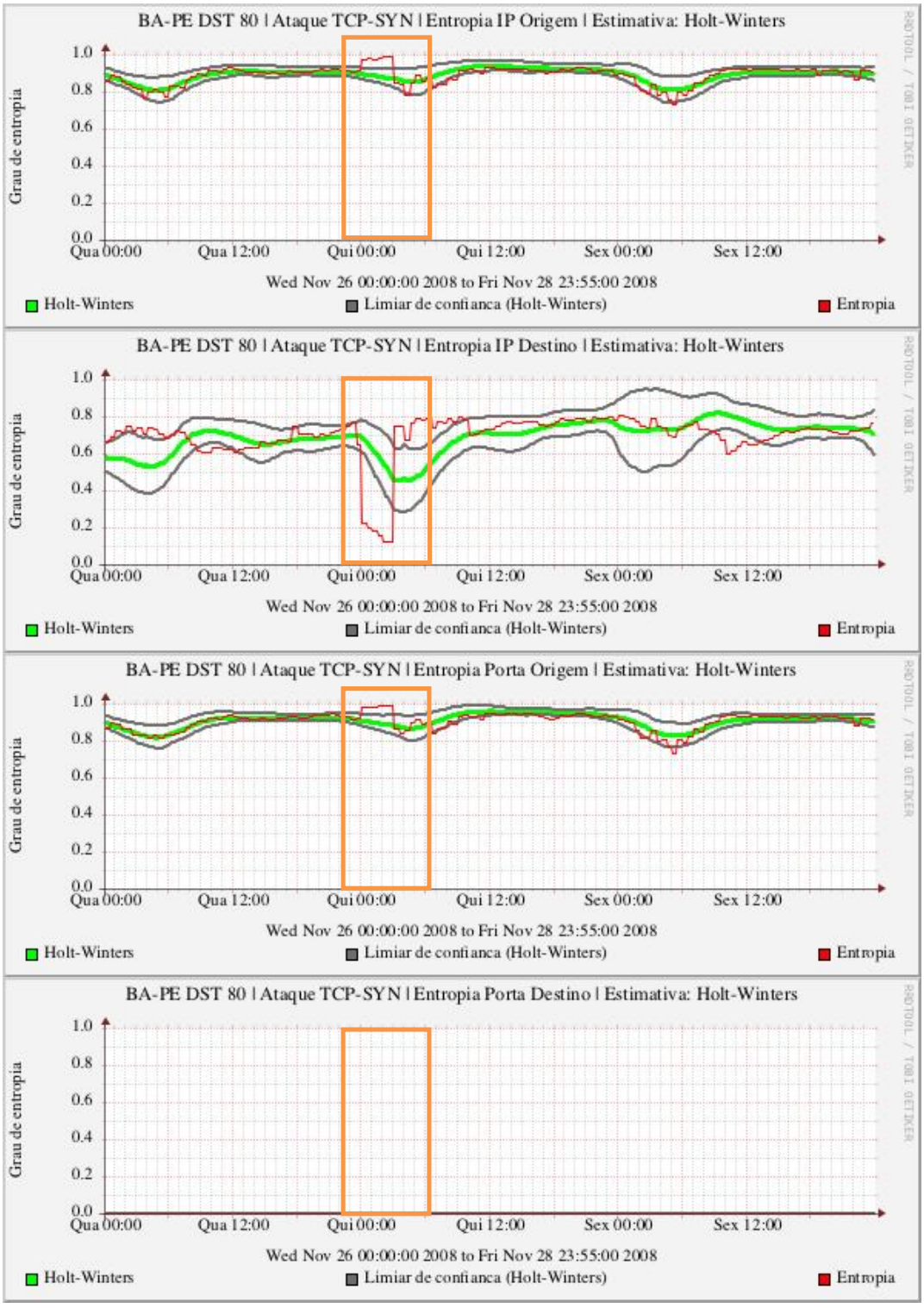


Figura 5.11 - DDoS TCP SYN injetado em tráfego filtrado na porta destino 80

Como houve uma filtragem específica e não houve ajustes dos parâmetros do HW, a estimativa passa a ser um pouco mais falha. Entretanto, na hora do ataque, pode-

se observar um grande pico de concentração no IP de destino e uma dispersão maior no IP e porta de origem, tornando a ocorrência do ataque muito mais evidente.

5.4.3 Worm Slammer injetado no tráfego de fundo

A Figura 5.12 ilustra o uso do método para a identificação do ataque do tipo *worm Slammer*.

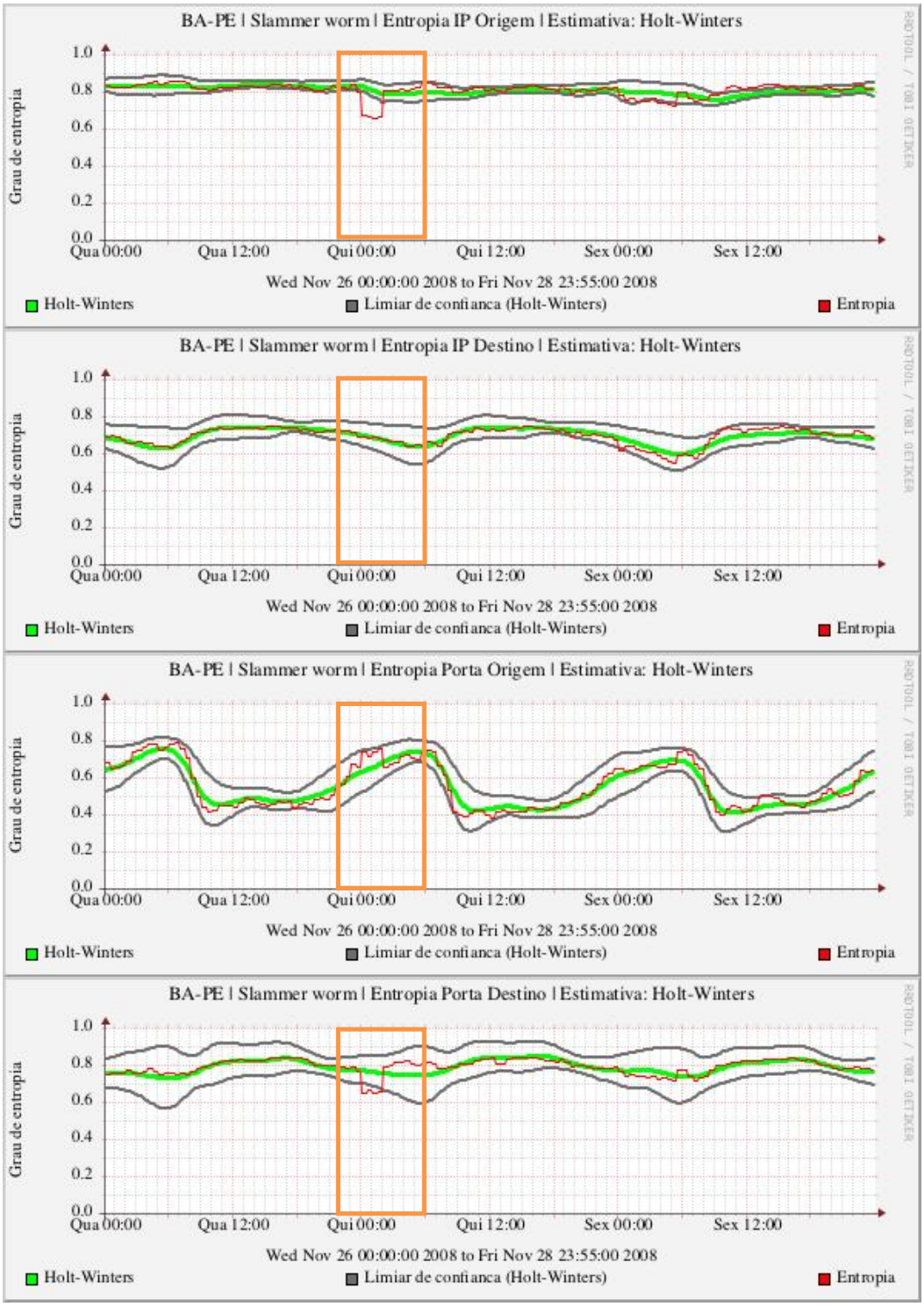


Figura 5.12 - Worm Slammer injetado em tráfego de fundo

Conforme pode ser observado, a concentração nas medidas de entropia para IP de origem e porta de destino evidenciam o ataque, o mesmo com relação à dispersão para porta de origem. A estimativa de HW acompanha a tendência normal da entropia,

podendo ser usada para indicar o momento do ataque. No entanto, a indicação baseada em EWMA do desvio, com os parâmetros adotados, não é tão eficiente quanto poderia ser. Para este caso, uma indicação baseada em limiares proporcionais em torno da estimativa pode ser mais adequada.

Worm Slammer injetado no tráfego de fundo, filtro pelo protocolo UDP

A Figura 5.13 mostra como ficam os resultados do método quando aplicado somente aos fluxos de tráfego que usam protocolo UDP. Conforme se verifica, as concentrações das entropias para IP de origem e porta de destino ficaram muito mais evidenciadas, assim como as dispersões para IP de destino e porta de origem.

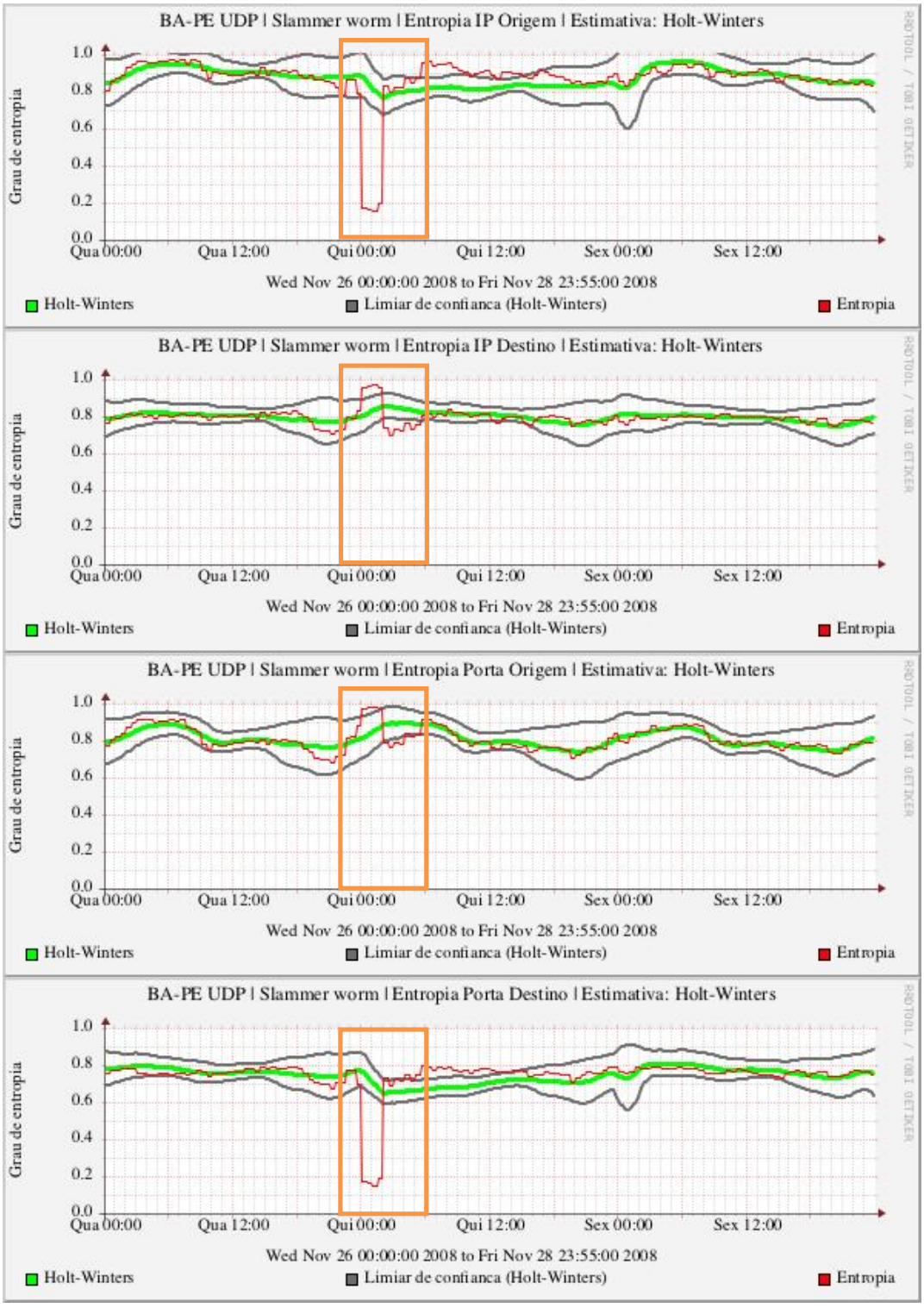


Figura 5.13 - Worm Slammer injetado em tráfego de fundo filtrado por protocolo UDP

5.5 Análise geral dos resultados

Os resultados apresentados confirmam a eficácia do método. Entretanto, fica claro que o uso de EWMA no desvio entre medidas de entropia e a estimativa de Holt-Winters precisa de um estudo mais aprofundado. Em alguns casos, usar limiares proporcionais à estimativa corrente pode ser mais adequado.

Os critérios de verificação das anomalias do método podem ser melhorados através de ajustes nos parâmetros de adaptação α, β, e γ do algoritmo Holt-Winters. Valores maiores significam que o algoritmo se adapta mais rapidamente e as estimativas recaem sobre os valores observados mais recentemente; enquanto valores menores fazem o algoritmo se adaptar mais lentamente, colocando maior peso sobre os valores mais antigos da série temporal.

A percepção de anomalias no tráfego através do método proposto varia em função da proporção entre volume do tráfego de fundo e volume do tráfego anômalo. As medições de entropia das distribuições de IPs e portas, de origem e de destino, podem ficar bem evidentes ou passar despercebidas, dependendo das características dos tráfegos envolvidos, ainda que somente para algumas das métricas, como foi possível observar. Ainda assim, o uso de entropias traz vantagens em relação ao uso de gráficos contendo apenas a taxa de pacotes por segundo, por exemplo. Basta ver o caso de um evento de mudança de rota devido à falha de um enlace. Todo o tráfego que passava pelo enlace que caiu será desviado para outro enlace, o que certamente causará um súbito aumento na taxa de pacotes por segundo, mas não necessariamente haverá uma forte modificação nas distribuições de IPs e portas. Certamente que isto se trata de um evento anômalo, embora não malicioso, e que costuma ser registrado pelo monitoramento SNMP. Fica, portanto, claro que uma análise baseada em medidas de entropia é mais adequada para a detecção de atividade maliciosa.

Certamente que o tráfego decorrente de um usuário remoto conectado a um servidor baixando arquivos a uma taxa elevada tem características similares a um DoS. Diferenciar tais situações de forma automática não é trivial. Para os objetivos deste trabalho, o importante é que o método usado sinalize rapidamente esta anomalia, seja ela um ataque ou não, cabendo aos operadores da rede investigar e tomar as devidas providências.

6 Conclusão e trabalhos futuros.

O presente trabalho apresentou uma metodologia para detecção de anomalias em redes WAN baseada em medidas de entropia associadas a um estimador tradicional e de simples implementação, o Holt-Winters. O método proposto usa a abordagem *single-link* por esta ser mais adequada ao cenário de gerenciamento de redes tipicamente encontrado nas instituições brasileiras, principalmente aquelas ligadas ao governo, e nos centros de operação de redes acadêmicas de pesquisa e educação. Esta abordagem simplifica bastante sua implementação, uma vez que ela pode se valer de ferramentas de *software* livre amplamente utilizadas no gerenciamento de redes, como o *RRDtool*. No caso, o objetivo é simplesmente a sinalização de anomalias que dificilmente seriam notadas pelos métodos de gerência tradicionalmente praticados. Uma vez dado o alerta, cabe aos administradores da rede usar os instrumentos adequados para mitigar o problema.

Os resultados mostrados no capítulo 5 confirmam a eficácia do método. Os experimentos realizados consideraram amostras de tráfego real da Rede Ipê. Amostras comprovadas de um ataque do tipo DoS, com duração aproximada de 20 horas, foram injetadas num tráfego de fundo, também extraído da rede Ipê, para o qual não houve registro de ataque durante o período observado, que foi de dez dias. Além disso, foram também injetados neste tráfego de fundo, em experimentos separados, dois tipos de ataques artificialmente gerados por códigos maliciosos já mapeados e de conhecimento público: o *TCP SYN flood* e o *worm Slammer*. O método proposto corretamente identificou a ocorrência destes três tipos de ataques no tráfego amostrado e, para melhorar sua eficiência, foi mostrado que o uso de filtros para capturar fluxos com características semelhantes aos ataques torna a detecção mais robusta, ou seja, os ataques ficam mais evidentes, evitando erros na sinalização dos mesmos. Vale salientar que o presente trabalho não se preocupou em estabelecer critérios visando um ajuste ótimo dos parâmetros usados na estimativa de Holt-Winters, mas sim uma avaliação

qualitativa da eficácia do método. Dentre as contribuições desta pesquisa, o autor destaca os artigos [Lucena 2008] e [Lucena 2009] publicados em workshops de conferencias nacionais.

Como trabalhos futuros, deseja-se estudar uma forma para melhorar o ajuste dos parâmetros da estimativa de Holt-Winters e, em especial, do EWMA aplicado aos desvios entre estimativa e medida real de entropia, usado para estabelecer os limites superiores e inferiores para o critério de normalidade. Além disso, deseja-se também implementar a metodologia proposta em sistemas de gerenciamento que usam código aberto e testá-la em ambiente de produção.

7 Referências

Bogaerdt, A. V. D. (2008) "*RRD Tutorial*", http://oss.oetiker.ch/rrdtool/tut/rrdtutorial.en.html. Acessado em 08/04/2009.

Boddington, Mark (2006) "*pcap-util Version 1.0*", http://www.badpenguin.co.uk/main/content/view/46/1/. Acessado em 01/09/2009.

Brutlag, J. D. (2000) "*Aberrant Behavior Detection in Time Series for Network Monitoring*". *Proceedings of the 14th Systems Administration Conference (LISA 2000)*.

CAIDA - Cooperative Association for Internet Data Analysis (2009), "*cflowd: Traffic Flow Analysis Tool*", http://www.caida.org/tools/measurement/cflowd/. Acessado em 01/09/2009.

CAIDA – Cooperative Association for Internet Data Analysis (2009), "*FlowScan - Network Traffic Flow Visualization and Reporting Tool*", http://www.caida.org/tools/utilities/flowscan/. Acessado em 01/09/2009.

Cacti (2007) "*The Complete RRDtool-based Graphing Solution*", http://cacti.net/features.php, acessado em 08/04/2009.

CEO-RNP, Rede Nacional de Ensino e Pesquisa (2008) "*Operação do Backone RNP*", http://www.rnp.br/ceo/. Acessado em 08/04/2009.

Cisco Systems, Inc. (2008) "*NetFlow Services Solution Guide*", http://www.cisco.com/en/US/docs/ios/solutions_docs/netflow/nfwhite.pdf. Acessado em 08/04/2009.

Claise, B. Ed. (2004) "*RFC 3954 - Cisco Systems NetFlow Services Export Version 9*", http://www.faqs.org/rfcs/rfc3954.html. Acessado em 08/04/2009.

Combs, Gerald (1998). "*tshark - Dump and analyze network traffic"*,

http://www.wireshark.org/docs/man-pages/tshark.html. Acessado em 01/09/2009.

Edmonds, Robert S. (2007) "*pcaputils*". https://launchpad.net/ubuntu/+source/pcaputils. Acessado em 01/09/2009.

Estevez-Tapiador, J. M., Garcia-Teodoro, P., Diaz-Verdejo, J. E. (2004) "*Anomaly Detection Methods in Wired Networks: a Survey and Taxonomy*", *Computer Comunications*, 27, 1569-1584.

Free Software Foundation, Inc. (2009), "*The Free Software Definition*", http://www.gnu.org/philosophy/free-sw.html. Acessado em 15/09/2009.

Haag, P. (2005) "*Watch your Flows with NfSen and Nfdump*", *50th RIPE Meeting, Stockholm*. http://www.ripe.net/ripe/meetings/ripe-50/presentations/ripe50-plenary-tue-NfSen-nfdump.pdf. Acessado em 08/04/2009.

Huang, L., Nguyen, X. L., Garofalakis, M. et al. (2007). "*Communication-efficient online detection of network-wide anomalies.*", *IEEE Conference on Computer Communications.* INFOCOM, 2007.

Jacobson, V., Leres, C., McCanne, S. (1987) "*TCPDUMP/LIBPCAP public repository*", http://www.tcpdump.org. Acessado em 01/09/2009.

Koehler, A. B., Snyder, R. D., and Ord, J. K. (1999) "*Forecasting Models and Prediction Intervals for the Multiplicative Holt-Winters Method*", http://www.buseco.monash.edu.au/depts/ebs/pubs/wpapers/1999/wp1-99.pdf. Acessado em 08/04/2009.

Lakhina, A., M. Crovella, C. Diot. (2004). "*Diagnosing network-wide traffic anomalies.*", *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications.*, p. 219–230.

Lakhina, A., Crovella, M., Diot, C. (2005) "*Mining anomalies using traffic feature distributions*", *Proceedings of the ACM SIGCOMM'2005*, Philadelphia, PA, USA.

Leinen, S. (2004) "*RFC 3955 - Evaluation of Candidate Protocols for IP Flow Information Export (IPFIX)*", http://www.faqs.org/rfcs/rfc3955.html. Acessado em 08/04/2009.

Lucena, S. C.; Moura, A. S. (2008). "*Detecção de Anomalias Baseada em Análise de Entropia no Tráfego da RNP*". In: *XIII Workshop de Gerência e Operação de Redes e Serviços (WGRS)*, 2008, Rio de Janeiro. Anais do XIII Workshop de Gerência e Operação de Redes e Serviços, 2008. p. 163-176.

Lucena, S. C.; Moura, A. S. (2009). "*Análise dos Estimadores EWMA e Holt-Winters para Detecção de Anomalias em Tráfego IP a partir de Medidas de Entropia*". In: *VIII Workshop em Desempenho de Sistemas Computacionais e de Comunicação (WPerformance)*, 2009, Bento Gonçalves. Anais do VIII WPerformance, 2009.

MacKey, D. J. C. (2003) "*Information Theory, Inference, and Learning Algorithms*", Cambridge University Press, Cambridge, UK.

Miller, Damien (2002) "*flowd - fast, secure and flexible NetFlow collector*", http://www.mindrot.org/projects/flowd/. Acessado em 01/09/2009.

Miller, Damien (2002) "*softflowd – fast software NetFlow probe*", http://www.mindrot.org/projects/softflowd/. Acessado em 01/09/2009.

Monsores, M. L., Ziviani, A., Rodrigues, P. S. S. (2006) "*Detecção de Anomalias de Tráfego usando Entropia Não-Extensiva*", Anais do XXIV Simpósio Brasileiro de Redes de Computadores – SBRC'2006.

Moore, D. et al. (2003) "*Inside the Slammer worm*". *Security & Privacy, IEEE* 1(4): 33-39.

Moura, Alex Soares de (2005), "*Visualização de Tráfego em Redes com Network Weathermaps*", ftp://ftp.registro.br/pub/gter/gter19/11-weathermap.pdf. Acessado em 19/09/2009.

Mu Dynamics, "*Mu Dos*", http://www.pcapr.net/faq#dos. Acessado em 01/09/2009.

Nfdump (2007) "*NFDUMP*", http://nfdump.sourceforge.net/, acessado em 08/04/2009.

Pcapr.net, "*Web 2.0 for packets | pcapr*", Mu Dynamics Research Labs, http://www.pcapr.net. Acessado em 01/09/2009.

Phaal, P., Panchen, S., McKee, N. (2001) "*RFC 3176 - InMon Corporation's sFlow: A*

Method for Monitoring Traffic in Switched and Routed Networks", http://www.ietf.org/rfc/rfc3176.txt. Acessado em 08/04/2009.

Plonka, David (2000) "*Flowscan: A Network Traffic Flow Reporting and Visualization Tool*". *Proceedings of the USENIX Fourteenth System Administration Conference LISA XIV*, Dezembro 2000.

Rede Nacional de Ensino e Pesquisa (2009), "*Mapa do backbone*", http://www.rnp.br/backbone/index.php. Acessado em 17/09/2009.

Rede Nacional de Ensino e Pesquisa (2009), "*Panorama do Tráfego*", http://www.rnp.br/ceo/trafego/panorama.php. Acessado em 17/09/2009.

RRDtool (2008) "*Aberrant Behavior Detection with Holt-Winters Forecasting*", http://oss.oetiker.ch/rrdtool/doc/rrdcreate.en.html. Acessado em 08/04/2009.

Shannon, C. E. (1948) "*A mathematical theory of communication*", *Bell System Technical Journal*, 27:379-423 and 623-656.

Silveira, F., Diot, C., Taft, N., Govindan, R. (2008) "*Empirical Evaluation of Network-wide Anomaly Detection", Thomsom Technical Report*", http://www.thlab.net/~fernando/papers/CR-PRL-2008-09-0004.pdf. Acessado em 08/04/2009.

Soule, A., K. Salamatian, N. Taft. (2005). "*Combining filtering and statistical methods for anomaly detection.*" *ACM Internet Measurement Conference*, p. 1–14.

Ward, A., Glynn, P., Richardson, K. (1998) "*Internet Service Performance Failure Detection*", *ACM SIGMETRICS Performance Evaluation Review*, Volume 26, number 3.

Zhang, Y., Ge, Z., Greenberg, A., Roughan, M. (2005) "*Network Anomography*", *Proceedings of the IMC'05*, Berkeley, CA, USA.

8 Apêndices

8.1 Script filtrar-dados.py

```
#!/usr/bin/env python

# $1 = Roteador | # $2 = Interface | # $3 = Workdir (ouput dir)
# $4 = Arquivo netflow | # $4 = Diretorio com arquivos netflow

import os, sys, string

prefix = '/usr/local/bin/programas/entropia'
prefix = os.getcwd()

roteador   = raw_input('Nome do roteador.............: ')
interface  = raw_input('Interface...................: ')
add_filtro = raw_input('Mais filtros p/ nfdump.......: ')
workdir    = raw_input('Diretorio de saida..........: ')
dir_nfcapd = raw_input('Diretorio c/ arqs. nfcapd....: ')

dir_nfcapd = os.path.join(prefix, dir_nfcapd)

try:
    os.chdir(dir_nfcapd)
finally:
    arqs_nfcapd = os.listdir(os.getcwd())  # Lendo nomes dos arqs da
amostra normal para a lista arqs_nfcapd
    arqs_nfcapd = sorted(arqs_nfcapd)

print "\nVariaveis usadas:\n"
print "Workdir...........: " + workdir

i = 0
```

```
j = 0
nfdump_fmt = [ 'fmt:%sa,%pkt', 'fmt:%da,%pkt', 'fmt:%sp,%pkt',
'fmt:%dp,%pkt' ]
nfdump_cmd = "nfdump -q -o " + nfdump_fmt[j] + " -r " + arqs_nfcapd[i]
+ " \'in if" + interface + " " + add_filtro + "\'"

while i <= len(arqs_nfcapd) - 1:
   j = 0
   while j <= 3:
#     nfdump_cmd = "nfdump -q -o " + nfdump_fmt[j] + " -r " +
arqs_nfcapd[i] + " \'in if" + interface + " " + add_filtro + "\'"
      timestamp = string.split(arqs_nfcapd[i], ".")[1]
      filt = string.split(string.split(nfdump_fmt[j], ":%")[1],
",")[0]
      filename = timestamp+"-"+roteador+"-"+filt+"-
if"+interface+".txt"
      arq_destino = os.path.join(prefix, workdir, filename)
      filtro = nfdump_fmt[j]
      nfcapd_file = arqs_nfcapd[i]

      print "Processando arquivo: " + arqs_nfcapd[i]
      print "Roteador...........: " + roteador
      print "Interface..........: " + interface
      print "Filtro.............: " + nfdump_fmt[j]
      print "Arquivo nfcapd.....: " + arqs_nfcapd[i]
      print "Comando............: " + "nfdump -q -o %s -r %s 'in if %s
%s' > %s" % (filtro, nfcapd_file, interface, add_filtro, arq_destino)
      print "Gravando arquivo...: " + arq_destino
      print " "
      os.system("nfdump -q -o %s -r %s 'in if %s %s' > %s" % (filtro,
nfcapd_file, interface, add_filtro, arq_destino))
      j = j + 1
   i = i + 1
```

8.2 Script entropia.py

```
#!/usr/bin/env python
# coding=utf-8

import os
import sys
import time

sys.path.append('/Users/alex/Mestrado/anomaly/bin/new/lib')

from flows.utils      import Configuration
from flows.utils      import progressbar
from flows.entropy    import Entropy
from flows.algorithms import createrrd
import flows.algorithms

# configuration loading
if len(sys.argv) == 2:
  configuration_file =  sys.argv[1]
else:
  configuration_file = 'flows.ini'

configuration = Configuration(configuration_file)

# creating EWMA RRD files #EWMA
#ewma_files = [ "sa-ewma.rrd", "sp-ewma.rrd","dp-ewma.rrd", "da-
ewma.rrd" ] # EWMA
#for rrd_file_ewma in ewma_files: #EWMA
#    if not os.path.isfile(rrd_file_ewma): #EWMA
#        createrrd(rrd_file_ewma, '200801010000') #EWMA

# samples directory
directory = configuration.data['entropy']['samples']

# mapping 'type' to its RRDTools file
types = {}

types_list = configuration.data['entropy']['types']
```

```
for type in types_list.split(','):
      type_key, type_value = type.split(':')
      types[type_key.strip()] = type_value.strip()

# algorithm
algorithm_name = configuration.data['entropy']['algorithm'].lower()
algorithm = eval('flows.algorithms.%s' % (algorithm_name))

# evaluating Entropy from the information in the configuration file:
flows.ini by default
for type, file in types.iteritems():
      e = Entropy(directory, '*%s*' % type)
      e.evaluate(algorithm, file)
```

8.3 Script entropy.py

```
# coding=UTF-8

# Descrição:
# Esta classe sabe tratar arquivos de "fluxos", gerar um histograma
dessas informações, aplicar um algoritmo para cálculo de entropia a
este e, por fim, anexar o resultado num arquivo RRD.

import glob
import os
import time

from flowssample import FlowsSample
from algorithms import shannon
from algorithms import ewma
from algorithms import
createrrd,createrrdhw,createrrdmhw,rrdgraph,rrdgraphhw,tunerrdgamma
from utils import progressbar

class Entropy:

    """
    Calcula entropia de arquivos de fluxos existentes num determinado
diretório.
    """

    def __init__(self, input_dir, filename_wildcard):

        """
        - input_dir: diretório que contém os dados filtrados
        - filename_wildcard: expressão regular utilizada para filtrar
os arquivos desejados
        """
        self.dir = input_dir
        self.wildcard = filename_wildcard

    def evaluate(self, algorithm, rrd_file):
```

```
        """
        Processa as informações contidas nos arquivos selecionados em
"self.input_dir" e 'appenda' as informações no "rrd_file".
        """
        vetor_ewma = [] # EWMA
        janela = 1      # EWMA
        files = glob.glob(self.dir + '/' + self.wildcard)
        first_timestamp = files[0].split('/')[-1].split('-')[0] # EWMA
        first_timestamp_epoch = time.mktime( time.strptime(
first_timestamp, "%Y%m%d%H%M" ) ) # EWMA

        idx_arq = 0
        #pb = progressbar(288, "*")
        pb = progressbar(len(files), "*")

        for file in files:
            idx_arq = idx_arq + 1
            histogram = FlowsSample(file).histogram()

            result = algorithm(histogram, to_be_normalized = True)   #
Calcula entropia de shannon

            timestamp = file.split('/')[-1].split('-')[0]
            timestamp_epoch = time.mktime( time.strptime( timestamp,
"%Y%m%d%H%M" ) )

            vetor_ewma.append(result) # EWMA - Guarda as entropias
calculadas no vetor

            rrd_line = "%s:%f" % (timestamp_epoch, result) # rrd_line
= parametros para gravar entropia no RRD

            if not os.path.isfile(rrd_file):   # Verifica se existe o
arquivo RRD
                createrrdhw(rrd_file, '200801010000')   # Cria o
arquivo da base RRD
                tunerrdgamma(rrd_file, '0,1')           # Modifica o
parametro gamma do Holt-Winters no RRD
```

```
            os.system('rrdtool update %s %s' % (rrd_file, rrd_line)) #
Grava o valor da entropia na base RRD
            pb.progress(idx_arq)  # Exibe barra de progresso do
processamento de SHANNON
            #print "[%s] [%d] %s" % (rrd_file, idx_arq, rrd_line)

        # Grafico Entropia com Holt-Winters
        #graphtitle = rrd_file + " Entropia" # Titulo do grafico
SHANNON c/ Holt-Winters
        #startdate = '200811270000'          # Data inicial grafico
SHANNON c/ H.W.
        #enddate = '200811302355'            # Data final   grafico
SHANNON c/ H.W.
        #rrdgraphhw(rrd_file, graphtitle, startdate, enddate) # Gera
grafico SHANNON c/ H.W.

        resultado_ewma = []                         # EWMA - Inicializa
vetor p/ EWMA
        ewma_entropias = ewma(vetor_ewma,janela)  # EWMA - Calcula
EWMA dos valores normalizados

        ewma_timestamp_epoch = first_timestamp_epoch

        for item_ewma_entropias in range(len(ewma_entropias)):       #
EWMA - Loop p/ calcular EWMA
            resultado_ewma.append( (ewma_timestamp_epoch,
ewma_entropias[item_ewma_entropias] ) ) # EWMA - Guarda EWMA no vetor
            ewma_timestamp_epoch = ewma_timestamp_epoch + float(300) #
EWMA - Preenche tuplas com timestamp

        rrd_file_ewma = rrd_file + '-ewma.rrd'
        # EWMA – cria o arquivo RRD para cada metrica:


        createrrdhw(rrd_file_ewma, '200801010000') # EWMA - Comando p/
criar o arq. RRD

        for item_ewma in resultado_ewma:           # EWMA - Para cada
tupla do vetor EWMA
```

```
            ewma_timestamp = str(item_ewma[0])      # EWMA - armazena o
timestamp da tupla
            ewma_value = str(item_ewma[1])          # EWMA - Valor do
EWMA da tupla
            rrd_line_ewma = "%s:%s" % (ewma_timestamp, ewma_value) #
EWMA - Tupla a ser gravada no RRD
            # rrd_line_ewma = "%s:%s" % (str(item_ewma[0],
str(item_ewma[1])   # EWMA - Tupla a ser gravada no RRD
            os.system('rrdtool update %s %s' % (rrd_file_ewma,
rrd_line_ewma)) # EWMA - Grava tupla no RRD

        #graphtitle_ewma = rrd_file_ewma + " EWMA" # EWMA - Titulo do
grafico EWMA
        #startdate_ewma = '200811270000' # EWMA - Data inicial do
grafico EWMA
        #enddate_ewma = '200811302355'   # EWMA - Data final do
grafico EWMA
        #rrdgraph(rrd_file_ewma, graphtitle_ewma, startdate_ewma,
enddate_ewma) # EWMA - Gera grafico EWMA

        return rrd_line
```

8.4 Script algorithms.py

```
# coding=UTF-8

from math import log
from scipy import array, cumsum, mean
import sys, os, time, string, re

def shannon(histogram, to_be_normalized = False):

    """
    Implementa o algoritmo de 'Shannon'.

    Parâmetros:
      - histogram: 'Dicionário' na forma 'ip': num_pacotes
        Ex: {'a.a.a.a': 43, 'b.b.b.b': 17, 'c.c.c.c': 37}

      - to_be_normalized: 'Booleano' que especifica se o devemos
normalizar

    Retorno:
      O resultado (float), normalizado ou não, do algoritmo de
Shannon.
    """

    shannon = 0

    for value in histogram.values():
      shannon += value * log(value, 2)   # shannon = shannon + value *
log(value, 2)

    shannon *= -1

    if to_be_normalized:
      num_of_ips = len( histogram.keys() )
      if num_of_ips > 1:
         shannon /= log(num_of_ips, 2)   # shannon =
shannon/log(num_of_ips, 2)
```

```
    return shannon

def ewma(s, n):
    """
    returns an n period exponential moving average for the time series
s

    s is a list ordered from oldest (index 0) to most recent (index -
1)
    n is an integer

    returns a numeric array of the exponential moving average
    """
    ewma = []
    j = 1
    #get n sma first and calculate the next n period ewma
    sma = sum(s[:n]) / n
    # multiplier = 2 / float(1 + n)
    multiplier = 0.01
    ewma.append(sma)
    #EMA(current) = ( (Price(current) - EMA(prev) ) x Multiplier) +
EMA(prev)
    ewma.append(( (s[n] - sma) * multiplier) + sma)
    #now calculate the rest of the values
    for i in s[n+1:]:
        tmp = ( (i - ewma[j]) * multiplier) + ewma[j]
        j = j + 1
        ewma.append(tmp)

    return ewma

def createrrd(rrd_filename, startdate):
    """
    Creates RRD database file
    """
    data_inicio  = time.mktime( time.strptime( startdate, "%Y%m%d%H%M"
))
    data_inicio  = str(int(data_inicio))
```

```
    step = "300"
    row_count = "2016"
    alpha = "0.01"
    beta = "0.0035"
    #gamma = "0.1"
    period = "288"
    rrdtool_cmd = "rrdtool create " + rrd_filename + " --start " +
data_inicio + " --step " + step + " DS:entropia:GAUGE:600:0:1
RRA:AVERAGE:0.5:1:600 RRA:AVERAGE:0.5:6:700 RRA:AVERAGE:0.5:24:775
RRA:AVERAGE:0.5:288:797 RRA:MAX:0.5:1:600 RRA:MAX:0.5:6:700
RRA:MAX:0.5:24:775 RRA:MAX:0.5:288:797 "
    try:
        os.system(rrdtool_cmd)
    finally:
        #print "Arquivo %s criado com sucesso!" % (rrd_filename)
        print ''
    return

def createrrdhw(rrd_filename, startdate):
    """
    Creates RRD database file
    """
    data_inicio  = time.mktime( time.strptime( startdate, "%Y%m%d%H%M"
))
    data_inicio  = str(int(data_inicio))
    step = "300"
    row_count = "2016"
    alpha = "0.01"
    beta = "0.0035"
    period = "288"
    rrdtool_cmd = "rrdtool create " + rrd_filename + " --start " +
data_inicio + " --step " + step + " DS:entropia:GAUGE:600:0:1
RRA:AVERAGE:0.5:1:600 RRA:AVERAGE:0.5:6:700 RRA:AVERAGE:0.5:24:775
RRA:AVERAGE:0.5:288:797 RRA:MAX:0.5:1:600 RRA:MAX:0.5:6:700
RRA:MAX:0.5:24:775 RRA:MAX:0.5:288:797 RRA:HWPREDICT:" + row_count +
":" + alpha + ":" + beta + ":" + period
    try:
        os.system(rrdtool_cmd)
    finally:
```

```
        #print "Arquivo %s criado com sucesso!" % (rrd_filename)
        print ''
    return

def createrrdmhw(rrd_filename, startdate):
    """
    Creates RRD database file
    """
    data_inicio  = time.mktime( time.strptime( startdate, "%Y%m%d%H%M"
))
    data_inicio  = str(int(data_inicio))
    step = "300"
    row_count = "2016"
    alpha = "0.01"
    beta = "0.0035"
    period = "288"
    rrdtool_cmd = "rrdtool create " + rrd_filename + " --start " +
data_inicio + " --step " + step + " DS:entropia:GAUGE:600:0:1
RRA:AVERAGE:0.5:1:600 RRA:AVERAGE:0.5:6:700 RRA:AVERAGE:0.5:24:775
RRA:AVERAGE:0.5:288:797 RRA:MAX:0.5:1:600 RRA:MAX:0.5:6:700
RRA:MAX:0.5:24:775 RRA:MAX:0.5:288:797 RRA:MHWPREDICT:" + row_count +
":" + alpha + ":" + beta + ":" + period
    try:
        os.system(rrdtool_cmd)
    finally:
        #print "Arquivo %s criado com sucesso!" % (rrd_filename)
        print ''
    return

def ewmaold(series, window):
    """
    returns an n period exponential moving average for the time series
s

    series is a list ordered from oldest (index 0) to most recent
(index -1)
    window is an integer

    returns a numeric array of the exponential moving average
```

```
    """
    series = series
    ewma = []
    window = 1

    # Primeiro obtem a janela sma e calcula o proximo periodo do ewma
    sma = sum(s[:window]) / n
    alpha = 2 / float(1 + n)
    ewma.append(sma)

    # EWMA(atual) = ((Preco(atual)-EWMA(anterior)) x Multiplicador) +
EWMA(anterior)
    ewma.append(( (series[window] - sma) * alpha) + sma)

    # agora calcula os demais valores
    for i in series[window+1:]:
        tmp = ( (i - ewma[window]) * alpha) + ewma[window]
        window = window + 1
        ewma.append(tmp)

    return ewma

def movavg(s, n):
    ''' returns an n period moving average for the time series s

        s is a list ordered from oldest (index 0) to most recent
(index -1)
        n is an integer

        returns a numeric array of the moving average

        This should run in near constant time with regard to n (of
course,
        O(n) to the length of s).  At least one person has said yuk
because
        of the numerical issue of losing precision in the cumsum, but
for
        small n's, and values like you will see in stock prices and
indices,
```

```
        I don't think this is too much of a problem.  Someone may have
        a more numerically stable version, but then you could just
implement
        the c-code version and wrap it for python.
    '''
    s = array(s)
    c = cumsum(s)
    return (c[n-1:] - c[:-n+1]) / float(n)


def sma(series, window):
    ''' returns an unweighted mean of the previous "window" values
(data points) for the time series "series".

        series is a list ordered from oldest to most recent
        window is an integer, representing the amount of past values
used to calculate the mean

        returns a numeric value of the mean for the provided "window"
    '''
    series = array(series)
    constant = cumsum(series)
    return (constant[window-1:] - constant[:-window+1] /
float(window))

def sma(valores, janela):
    serie = valores[0:janela]
    return scipy.mean(serie)

def rrdgraph(rrdfile, graphtitle, data_inicio, data_fim):
    '''
    Gerador de graficos de dados em base RRD '
    '''
    arq_imagem        = string.rsplit(rrdfile,'.', maxsplit=1)
    arq_imagem        = arq_imagem[0]+'.png'
    data_inicio_epoch = time.mktime( time.strptime( data_inicio,
"%Y%m%d%H%M" ))
    data_fim_epoch    = time.mktime( time.strptime( data_fim,
"%Y%m%d%H%M" ))
```

```
    comentario1       = time.ctime(data_inicio_epoch)+' to
'+time.ctime(data_fim_epoch)
    comentario1       = comentario1.replace(':','\:')
    comentario1       = '"'+comentario1+'\c"'
    comentario2       = '"'+"\\n"+'"'
    vertical          = '"'+'Grau de entropia'+'"'
    data_inicio_epoch = str(data_inicio_epoch).split(".")
    data_inicio_epoch = data_inicio_epoch[0]
    data_fim_epoch    = str(data_fim_epoch).split(".")
    data_fim_epoch    = data_fim_epoch[0]
    graphtitle        = re.escape(graphtitle)

    os.system('rrdtool graph %s \
    --imgformat=PNG \
    --start %s \
    --end %s \
    --title=%s \
    --upper-limit=1.0 \
    --lower-limit=0.0 \
    COMMENT:%s \
    COMMENT:%s \
    --rigid \
    --base=1000 \
    --height=120 \
    --width=500 \
    --vertical-label=%s \
    --slope-mode \
    --font TITLE:9: \
    --font AXIS:8: \
    --font LEGEND:8: \
    --font UNIT:8: \
    DEF:entropia=%s:entropia:AVERAGE \
    LINE1:entropia#FF0000:"Entropia"' % (arq_imagem,
data_inicio_epoch, data_fim_epoch, graphtitle, comentario1,
comentario2, vertical, rrdfile))
    return

def rrdgraphhw(rrdfile, graphtitle, data_inicio, data_fim):
    '''
```

```
    Gerador de graficos de dados em base RRD '
    '''
    arq_imagem        = string.rsplit(rrdfile,'.', maxsplit=1)
    arq_imagem        = arq_imagem[0]+'.png'
    data_inicio_epoch = time.mktime( time.strptime( data_inicio,
"%Y%m%d%H%M" ))
    data_fim_epoch    = time.mktime( time.strptime( data_fim,
"%Y%m%d%H%M" ))
    comentario1       = time.ctime(data_inicio_epoch)+' to
'+time.ctime(data_fim_epoch)
    #comentario1      = '"'+re.escape(comentario1)+'\c"'
    comentario1       = comentario1.replace(':','\:')
    comentario1       = '"'+comentario1+'\c"'
    comentario2       = '"'+"\\n"+'"'
    vertical          = '"'+'Grau de entropia'+'"'
    data_inicio_epoch = str(data_inicio_epoch).split(".")
    data_inicio_epoch = data_inicio_epoch[0]
    data_fim_epoch    = str(data_fim_epoch).split(".")
    data_fim_epoch    = data_fim_epoch[0]
    graphtitle        = re.escape(graphtitle)

    os.system('rrdtool graph %s \
    --imgformat=PNG \
    --start %s \
    --end %s \
    --title=%s \
    --upper-limit=1.0 \
    --lower-limit=0.0 \
    COMMENT:%s \
    COMMENT:%s \
    --rigid \
    --base=1000 \
    --height=120 \
    --width=500 \
    --vertical-label=%s \
    --slope-mode \
    --font TITLE:9: \
    --font AXIS:8: \
    --font LEGEND:8: \
```

```
    --font UNIT:8: \
    DEF:entropia=%s:entropia:AVERAGE \
    DEF:pred=%s:entropia:HWPREDICT \
    DEF:dev=%s:entropia:DEVPREDICT \
    DEF:season=%s:entropia:SEASONAL \
    DEF:devseason=%s:entropia:DEVSEASONAL \
    DEF:fail=%s:entropia:FAILURES \
    CDEF:dev_lower=pred,dev,2,*,- \
    CDEF:dev_upper=pred,dev,2,*,+ \
    TICK:fail#a4a4a4:1.0:"Anomalia" \
    LINE2:dev_lower#6e6e6e:"Limiar de confianca" \
    LINE2:dev_upper#6e6e6e \
    LINE1:entropia#FF0000:"Entropia"' % (arq_imagem,
data_inicio_epoch, data_fim_epoch, graphtitle, comentario1,
comentario2, vertical, rrdfile, rrdfile, rrdfile, rrdfile, rrdfile,
rrdfile))

def rrdplothw(rrdfile, graphtitle, data_inicio, data_fim):
    '''
    Gerador de graficos de dados em base RRD
    '''
    arq_imagem        = string.rsplit(rrdfile,'.', maxsplit=1)
    arq_imagem        = arq_imagem[0]+'.png'
    data_inicio_epoch = time.mktime( time.strptime( data_inicio,
"%Y%m%d%H%M" ))
    data_fim_epoch    = time.mktime( time.strptime( data_fim,
"%Y%m%d%H%M" ))
    comentario1       = time.ctime(data_inicio_epoch)+' to
'+time.ctime(data_fim_epoch)
    comentario1       = comentario1.replace(':','\:')
    comentario1       = '"'+comentario1+'\c"'
    comentario2       = '"'+"\\n"+'"'
    vertical          = '"'+'Grau de entropia'+'"'
    data_inicio_epoch = str(data_inicio_epoch).split(".")
    data_inicio_epoch = data_inicio_epoch[0]
    data_fim_epoch    = str(data_fim_epoch).split(".")
    data_fim_epoch    = data_fim_epoch[0]
    graphtitle        = re.escape(graphtitle)
```

```
    os.system('rrdtool graph %s \
    --imgformat=PNG \
    --start %s \
    --end %s \
    --title=%s \
    --upper-limit=1.0 \
    --lower-limit=0.0 \
    COMMENT:%s \
    COMMENT:%s \
    --rigid \
    --base=1000 \
    --height=120 \
    --width=500 \
    --vertical-label=%s \
    --slope-mode \
    --font TITLE:9: \
    --font AXIS:8: \
    --font LEGEND:8: \
    --font UNIT:8: \
    DEF:entropia=%s:entropia:AVERAGE \
    DEF:pred=%s:entropia:HWPREDICT \
    DEF:dev=%s:entropia:DEVPREDICT \
    DEF:season=%s:entropia:SEASONAL \
    DEF:devseason=%s:entropia:DEVSEASONAL \
    DEF:fail=%s:entropia:FAILURES \
    LINE1:pred#FF0000:"Holt Winters"' % (arq_imagem,
data_inicio_epoch, data_fim_epoch, graphtitle, comentario1,
comentario2, vertical, rrdfile, rrdfile, rrdfile, rrdfile, rrdfile,
rrdfile))
    return

def tunerrdgamma(rrd_filename, gamma):
    """
    Modifies an RRD database Holt-Winters gamma parameter:
    Alter the seasonal coefficient adaptation parameter for the
SEASONAL RRA and alter the seasonal deviation adaptation parameter for
the DEVSEASONAL RRA. This parameter must be between 0 and 1.
    """
    rrdtool_tune_cmd = "rrdtool tune " + rrd_filename + " --gamma " +
```

```
    gamma + " --gamma-deviation " + gamma
        os.system(rrdtool_tune_cmd)
        return
```

8.5 Script utils.py

```
# coding=UTF-8

import re
import sys

class Configuration:

  """
  Represents a '.ini'-like configuration file.
  """

  def __init__(self, file):

    self.file = file
    self.data = {}

    self.__parse()


  def __parse(self):

    regexp_section = re.compile(
        '^[\s]*'              # any number of 'white space' character
        '\['                  # literal '['
        '(?P<section>[^\]]+)' # section name
        ' \]'                 # literal ']'
        , re.VERBOSE | re.IGNORECASE
    )

    regexp_key_value = re.compile(
        '(?P<key>^[^=]+)'# key (any group of characters not including
'=')
        '\s*=\s*' # literal '=' surrounded by any number of space
characters
        '(?P<value>.+)'# value (any group of characters not including
'=')
```

```
        , re.VERBOSE | re.IGNORECASE
    )

    section = None

    for line in open(self.file):
      line = line.strip()
      if len(line) > 0 and not line.startswith('#'):
        match_section = regexp_section.match(line)
        match_key_value = regexp_key_value.match(line)
        if match_section:
          section = match_section.group('section')
          self.data[section] = {}
        elif match_key_value:
          if section != None:
            self.data[section][match_key_value.group('key').strip()] =
match_key_value.group('value').strip()

class progressbar(object):
    def __init__(self, finalcount, block_char='.'):
        self.finalcount = finalcount
        self.blockcount = 0
        self.block = block_char
        self.f = sys.stdout
        if not self.finalcount: return
        self.f.write('\n------------------ % Progress ----------------
---1\n')
        self.f.write('    1    2    3    4    5    6    7    8    9
0\n')
        self.f.write('----0----0----0----0----0----0----0----0----0---
-0\n')
    def progress(self, count):
        count = min(count, self.finalcount)
        if self.finalcount:
            percentcomplete = int(round(100.0*count/self.finalcount))
            if percentcomplete < 1: percentcomplete = 1
        else:
            percentcomplete=100
        blockcount = int(percentcomplete//2)
```

```
            if blockcount <= self.blockcount:
                return
            for i in range(self.blockcount, blockcount):
                self.f.write(self.block)
            self.f.flush( )
            self.blockcount = blockcount
            if percentcomplete == 100:
                self.f.write("\n")
```

8.6 Script injector.py

```
# coding=UTF-8

import re
from glob import glob
from shutil import copy

from flows.flowssample import FlowsSample


class Injector:

      def __init__(self, configuration):

            """
            Inicializa um objeto Injector com informações contidas no
dicionário 'configuration'.

            As seguintes variáveis devem estar definidas no dicionário
'configuration', passado como parâmetro:

              starts_at  : O data-hora de início da injeção, no formato
AAAAMMDDHHMM.
              samples    : Diretório que contém os arquivos de
'amostras'.
              anomalies  : Diretório que contém os arquivos de
'anomalias'.
              injected   : Diretório que contém os arquivos
'injetados'.
              scale      : Fator de escala a ser aplicado nos registros
de anomalias. Escalas usadas no paper para o SBRC 2009: 0.02919 (para
10%), 0.14595 (50%), 0.5838 (20%) e 0.073 (25%)
              filename_re: Expressão regular que define o padrão de
nome dos arquivos. Ela define os seguintes agrupamentos:
                - timestamp
                - pop
                - type
```

```
                - interface
            """

            self.configuration = configuration


      def __get_fields(self, file):
            fields = {}
            filename_re = re.compile(self.configuration['filename_re'])
            filename_match = filename_re.match( file.split('/')[-1] )
            if filename_match:
                  fields['timestamp'] =
filename_match.group('timestamp')
                  fields['pop'] = filename_match.group('pop')
                  fields['type'] = filename_match.group('type')
                  fields['interface'] =
filename_match.group('interface')
            return fields

      def inject(self):
            head, middle, tail = [], [], []
            fields = {}
            position = 0

            anomalies = glob( self.configuration['anomalies'] + '/*' )
            anomalies.sort()

            samples = glob(self.configuration['samples'] + '/*')
            samples.sort()

            fs = FlowsSample(samples[0]) # obj. p/ mudar estado no loop
            for k, sample in enumerate(samples): # copying 'head' files
                  fields = self.__get_fields(sample)
                  if fields != {}:
                        if fields['timestamp'] <
self.configuration['starts_at']:
                              copy(sample,
self.configuration['injected'])
                        else:
```

```
                              head = samples[0:k]
                              position = k
                              break
            print "Escala: ", float(self.configuration['scale'])
            # applying anomalies
            if len(samples) - position < len(anomalies):
                  raise RuntimeError, "Existem mais anomalias a inserir
do que a quantidade de amostras."
            else:
                  middle = samples[position : position +
len(anomalies)]
                  i = 0 # índice para o vetor de anomalias
                  for k in range( position, position + len(anomalies)
):
                        fs.file = samples[k]
                        fs.inject_anomaly(anomalies[i],
float(self.configuration['scale']), self.configuration['injected'] +
'/' + fs.file.split('/')[-1])
                        i += 1
                  position += len(anomalies)
            tail = samples[position : ] # Copiando arquivos 'tail'
            for k in range( position, len(samples) ):
                  copy(samples[k], self.configuration['injected'])

            print " ===== Fim do Processamento ===== "
```

8.7 Script flowssample.py

```
# coding=UTF-8

from __future__ import with_statement
import shutil


class FlowsSample:

	"""
	Representa uma amostra de fluxo, da qual podemos extrair um
histograma, etc.
	"""

	def __init__(self, file):
		self.file = file



	def __records(self):

		"""
		Um gerador, que provê um dicionário para cada linha
processada do arquivo de amostra.
		"""

		for line in open(self.file, 'r'):
			if len(line.strip()) > 0:
				ip, num_packets = line.split(',')
				ip = ip.strip()
				num_packets = int( num_packets )
				yield {'ip': ip, 'num_packets': num_packets}
			else:
				yield {}
```

```
	def histogram(self):

		"""
		Calcula o histograma normalizado do arquivo de amostra
'self.file' e o retorna através de um dicionário de chaves que
representam IPs e valores que representam o total de número de
pacotes.
		"""

		h = {}

		for record in self.__records():
			if record != {}:
				if record['ip'] in h:
					h[record['ip']] += record['num_packets']
				else:
					h[record['ip']] = record['num_packets']

		total_num_packets = sum(h.values())

		for key, value in h.iteritems():
			h[key] = float( h[key] ) / total_num_packets

		return h



	def inject_anomaly(self, anomaly_file, factor, injected_file):

		"""
		Insere a anomalia descrita no arquivo "anomaly_file",
utilizando-se o fator de escala "factor", e salva o resultado no
arquivo "injected_file".
		"""
		shutil.copy(self.file, injected_file)

		with open(injected_file, 'a') as injected:
			for line in open(anomaly_file, 'r'):
				ip, num_packets = line.split(',')
```

```
                        ip = ip.strip()
                        num_packets = int( num_packets ) * factor
                        num_packets = int( num_packets + 1 )
                        injected.write(ip + ', ' + str(num_packets) +
'\n')
```

8.8 Script plota_graficos.py

```
#!/usr/bin/env python
import sys, os, time, string, re
rrdfile1          = 'PE-if26-porta-dest-HW.rrd'
graphtitle        = "BA-PE | Entropias: Porta Destino | Interface
SNMP-ID 26 | Estimativa: Holt-Winters" # Titulo para cada grafico.
data_inicio       = '200907090000'
data_fim          = '200907130000'
arq_imagem        = string.rsplit(rrdfile1,'.', maxsplit=1)
arq_imagem        = arq_imagem[0]+'.png'
data_inicio_epoch = time.mktime( time.strptime( data_inicio,
"%Y%m%d%H%M" ))
data_fim_epoch = time.mktime( time.strptime( data_fim, "%Y%m%d%H%M" ))
comentario1       = time.ctime(data_inicio_epoch)+' to
'+time.ctime(data_fim_epoch)
comentario1       = comentario1.replace(':','\:')
comentario1       = '"'+comentario1+'\c"'
comentario2       = '"'+"\\n"+'"'
vertical          = '"'+'Grau de entropia'+'"'
data_inicio_epoch = str(data_inicio_epoch).split(".")
data_inicio_epoch = data_inicio_epoch[0]
data_fim_epoch    = str(data_fim_epoch).split(".")
data_fim_epoch    = data_fim_epoch[0]
graphtitle        = re.escape(graphtitle)
os.system('rrdtool graph %s \
--imgformat=PNG \
--start %s \
--end %s \
--title=%s \
--upper-limit=1.0 \
--lower-limit=0.0 \
COMMENT:%s \
COMMENT:%s \
--rigid \
--base=1000 \
--height=120 \
--width=500 \
```

```
--vertical-label=%s \
--slope-mode \
--font TITLE:9: \
--font AXIS:8: \
--font LEGEND:8: \
--font UNIT:8: \
DEF:entropia=%s:entropia:AVERAGE \
DEF:pred=%s:entropia:HWPREDICT \
DEF:dev=%s:entropia:DEVPREDICT \
DEF:season=%s:entropia:SEASONAL \
DEF:devseason=%s:entropia:DEVSEASONAL \
DEF:fail=%s:entropia:FAILURES \
CDEF:dev_lower=pred,dev,2,*,- \
CDEF:dev_upper=pred,dev,2,*,+ \
LINE3:pred#00FF00:"Holt-Winters" \
LINE2:dev_lower#6e6e6e:"Limiar de confianca (Holt-Winters)" \
LINE2:dev_upper#6e6e6e \
LINE1:entropia#FF0000:"Entropia"' % (arq_imagem, data_inicio_epoch,
data_fim_epoch, graphtitle, comentario1, comentario2, vertical,
rrdfile1, rrdfile1, rrdfile1, rrdfile1, rrdfile1, rrdfile1))
print ' '
print 'Arquivo RRD.........: ', rrdfile1
print 'Arquivo imagem......: ', arq_imagem
print 'Titulo..............: ', graphtitle
print 'Data inicio.........: ', data_inicio
print 'Data fim............: ', data_fim
print "Data inicio em epoch: ", data_inicio_epoch
print "Data fim............: ", data_fim
print "Data fim em epoch...: ", data_fim_epoch
print 'Comentario1.........: ', comentario1
print 'Comentario2.........: ', comentario2
```

8.9 Arquivo de configuração flows.ini

```
[entropy]
algorithm = shannon
samples   = /amostras/trafego/ataque/injetado
types   = sa: /amostras/rrds/PE-IP-origem-hw.rrd, \
da: /amostras/rrds/PE-IP-destino-hw.rrd, \
sp: /amostras/rrds/PE-Porta-origem-hw.rrd, \
dp: /amostras/rrds/PE-Porta-destno-hw.rrd

[injection]
starts_at   = 200907100000
samples     = /amostras/trafego/normal
anomalies   = /amostras/trafego/ataque
injected    = /amostras/trafego/ataque/injetado
scale       = 0.073
filename_re = (?P<timestamp>[0-9]{12})-(?P<pop>[a-zA-Z]{2})-
(?P<type>[a-zA-Z]{2})-(?P<interface>[a-zA-Z0-9]+)\.txt
```