Análise de desempenho da biblioteca libpcap: uma abordagem voltada à gerência de segurança

Ramicés dos S. Silva[1], **Rafael L. Cancian**[1]

[1]Departamento de Automação e Sistemas (DAS)
Universidade Federal de Santa Catarina (UFSC)
Caixa Postal 476 – 88040-900 – Florianópolis – SC – Brasil

{ramices,cancian}@das.ufsc.br

Abstract. *The adoption of security and management tools for network management has important role in those activities related to computer network security and availability. Several tools use passive traffic capturing to measure data from the network traffic and many applications actually use the libpcap to implement it. This work proposes to evaluate which factors have a significant influence on the libpcap performance. Was accomplished applying an experimental methodology called design of experiments in order to achieve reliable results with objective of the identify factors that influence in the performance of this library.*

Resumo. *Ferramentas de segurança e apoio ao gerenciamento de redes, assim como a base destas ferramentas (suas bibliotecas), têm papel importante em se tratando das atividades relacionadas à segurança e disponibilidade das redes de computadores. Muitas dessas ferramentas de apoio fazem uso de captura passiva de tráfego para obter informações da rede. Em muitos casos, essa captura é feita através da libpcap, uma biblioteca que implementa captura passiva de tráfego. Neste trabalho foi feita a análise, em um ambiente específico, de quais fatores influenciam o desempenho da libpcap. Para isso fez-se uso da metodologia de projetos de experimentos fatoriais, garantindo que os resultados obtidos são estatisticamente confiáveis. Com base na técnica de análise e projeto de experimentos foi conduzida a análise da libpcap, tendo alcançado o objetivo de identificar e quantificar os fatores que influenciam no desempenho de tráfego capturado pela biblioteca, além de chegar a um modelo de previsão em função da combinação dos fatores analisados.*

1. Introdução

Muitas aplicações de apoio ao gerenciamento e segurança de redes fazem uso especificamente de captura passiva de tráfego, fazendo uso de uma biblioteca de software chamada libpcap [Libpcap 2007]. Esta é uma biblioteca *open source* portável e que provê funcionalidades para captura de tráfego das interfaces de rede. A ausência de trabalhos sistematizados para a avaliação do desempenho da libpcap motivou a elaboração deste trabalho para servir de referência à comunidade usuária da biblioteca no âmbito acadêmico-científico e comercial seja para gerência ou segurança de redes de computadores ou áreas afins.

Este trabalho realizou um estudo do desempenho da biblioteca libpcap, a partir de um projeto de experimentos, utilizando fatores e níveis baseados em caracterizações de sua utilização em um ambiente de testes com três estações rodando sistema operacional Linux. Fatores como: otimizações de *kernel* e parâmetros de configuração foram

combinados em um ambiente experimental e resultados acerca da quantidade de pacotes capturados foram analisados.

2. Fundamentação Teórica

O gerenciamento de redes pode ser aplicado em função de algumas áreas sempre com o objetivo de garantir a segurança e o perfeito funcionamento da rede [Soares et al. 1997]. [ISO 1988] define à classificação das gerências em áreas funcionais, como: (i) gerenciamento de configuração; (ii) gerenciamento de falhas; (iii) gerenciamento de desempenho; e (iv) gerenciamento de contabilização. A libpcap se aplica as três últimas citadas.

2.1. Captura e Análise de Tráfego

A análise de tráfego tem como base a caracterização e coleta de tráfego das redes. As ferramentas de análise de tráfego são baseadas em princípios de captura de tráfego, ou seja, são coletados os dados que trafegam na rede em pontos estratégicos das mesmas. Tecnologias como RMON, sFlow e NetFlow fazem parte do grupo de agentes baseados em captura passiva de tráfego.[Andreozzi et al. 2005] conceitua a técnica como uma técnica de análise e monitoramento de redes que captura e examina os pacotes que passam através do link monitorado, permitindo, até mesmo análise de impacto de pacotes.

Porém a utilização de agentes além de adicionar tráfego na rede requer que os dispositivos (ativos de rede) implementem tais tecnologias o que acaba inviabilizando sua utilização em função do custo dos equipamentos. Normalmente, só os equipamentos de centro de rede (*core*) possuem tais características o que impede o monitoramento dos segmentos de rede após os *switch*es de borda. [Junior and Gonçalves]. É partindo do problema de custo e *overhead* que muitas soluções de apoio ao gerenciamento de redes e segurança são baseadas em captura passiva de tráfego.

Exemplos de aplicações de análise de tráfego

Como aplicações de apoio ao gerenciamento e segurança de redes que fazem uso da libpcap, têm-se:

- **Ntop** é uma solução de monitoramento de tráfego que possui uma interface com o usuário bastante intuitiva e de fácil adaptação. O Ntop tem sido incorporado em produtos comerciais de empresas como a Cyclades e 3Com.
- **Tcpdump** é definido como sendo uma aplicação que captura o tráfego da rede e imprime na tela os cabeçalhos dos pacotes de uma interface de rede de acordo com expressões boleanas utilizadas como filtro [Libpcap 2007]. O projeto tcpdump faz parte do projeto da libpcap, conseqüentemente este faz uso desta biblioteca para captura dos pacotes.
- **Snort** é uma ferramenta que não é utilizada para gerenciamento mas para segurança de redes, é definido como “uma tecnologia *open source* de detecção e prevenção de intrusos. O Snort tornou-se rapidamente o mais expressivo sistema de detecção e prevenção de intrusão do mundo. [SNORT 2007] Todo o funcionamento do Snort tem por princípio identificar ataques através da análise de padrões do tráfego. Todo tráfego é capturado de forma passiva através da libpcap.
- **Wireshark** é uma ferramenta que é a continuidade do projeto Ethereal. É um analisador gráfico de protocolos de redes. O formato nativo de captura de pacotes é o libpcap, que é utilizado pelo tcpdump e várias outras ferramentas.[Wireshark 2007]

2.2. Bilioteca de Captura Libpcap

O desempenho e eficácia de muitos sistemas de apoio ao gerenciamento e segurança de redes dependem diretamente da *performance* do seu mecanismo de coleta passiva de pacotes. Tal dependência motivou a realização deste trabalho no que tange a realização da avaliação de desempenho de uma biblioteca de captura de pacotes de rede, neste caso a libpcap.

O projeto libpcap nasceu no NRG (Network Research Group - Grupo de Pesquisa de Redes) na Califórnia [LBNL 2007] e atualmente faz parte do projeto libpcap/tcpdump. Seu desenvolvimento é atribuído a Van Jacbson, Crig Leres e Steven MacCanne e permite a captura de tráfego de rede e a recuperação dos frames. [Libpcap 2007].

A libpcap, em sua implementação original, apesar de fazer uso do mecanismo de captura de pacotes do *kernel*, seu funcionamento em modo usuário está sujeito à concorrência por utilização dos recursos da máquina com outras aplicações. O que pode representar um problema quanto ao seu desempenho.

A estrutura de funcionamento dos métodos e chamadas para a biblioteca libpcap é simples e bastante transparente. Sua utilização se dá na captura dos dados que estão chegando na camada de enlace de dados (*data link layer*). Alguns parâmetros devem ser informados ao criar uma captura como, por exemplo, o modo de operação da biblioteca (promíscuo ou não promíscuo) ou quanto ao número de iterações de processamento e recepção dos pacotes.

2.3. Projeto de Experimentos

Experimento é definido formalmente por [Montgomery 2001] como um teste ou uma série de testes nas quais se fazem inferências e alterações nas variáveis de entrada de um processo ou sistema onde a partir de observações são identificadas as razões das mudanças observadas nas variáveis de saída. Para que estas observações sejam feitas de forma eficiente e seus resultados realmente demonstrem a realidade, técnicas e modelos de projeto de experimentos foram criados. O objetivo de um projeto de experimento é obter o máximo de informação consistente com o mínimo de experimentos [Jain 1991] ou ainda desenvolver um processo robusto proporcionando o mínimo de variabilidade em função dos fatores externos [Montgomery 2001]. O objetivo de se utilizar uma metodologia de projeto de experimentos para os testes propostos neste trabalho, é buscar uma consistência das análises dos resultados obtidos durante os testes de desempenho da libcap, estando em conformidade com a técnica DOE (*Design of experiments*).

[Jain 1991] e [Montgomery 2001] conceituam alguns termos normalmente utilizados: Variável resposta: é a variável que representa a resposta que se investiga no processo. Fatores: São as variáveis ou constantes que afetam a resposta do experimento; Níveis: É o conjunto de valores, considerados, que cada fator pode assumir; Tratamentos: São as combinações de fatores e níveis e suas interações; Replicação: É a repetição da execução de um dado tratamento, sendo todas as replicações sendo executadas nas mesmas condições ambientais; Interação: É a condição onde dois ou mais fatores são avaliados em um tratamento de forma combinada; Projeto experimental: Consiste na etapa de especificação do experimento. Repetição: O termo é usado associado à medição repetida dos fatores ou variáveis resposta.

Para que o objetivo do experimento seja alcançado, o projeto do experimento deverá apresentar uma estratégia de experimentação. Uma das estratégias sugeridas por [Montgomery 2001] é o **projeto fatorial** (*Factorial design*): Em se tratando de experimentos com dois ou mais fatores, os quais são considerados interativos, a abordagem fatorial geralmente é a mais eficiente. Esta abordagem permite que a cada avaliação completa ou replicação seja possível avaliar os efeitos de todas as combinações de fatores e níveis definidos, e foi utilizada no estudo apresentado neste artigo.

A partir do modelo de projeto fatorial, cinco abordagens podem ser encontradas em [Jain 1991] e [Montgomery 2001]. A abordagem de projeto fatorial 2^k é utilizado para determinar o efeito de k fatores, cada um com dois níveis. Este modelo matemático foi desenvolvido de forma a ajudar na tarefa de combinar e analisar a influência dos fatores [Jain 1991]. Uma replicação completa necessita de uma combinação de k fatores e seus níveis, tomados dois a dois, sendo por esse motivo chamado de projeto fatorial 2^k. [Montgomery 2001]. O projeto fatorial $2^k r$ é utilizado quando a análise do experimento exige que o erro experimental seja mensurado, com isso passa-se a considerar como parte do valor medido não só os efeitos de cada fator mas também o efeito do erro experimental [Jain 1991].Como o objetivo deste trabalho é obter dados consistentes para análise de desempenho da libpcap, far-se-á o uso deste modelo de projeto para que também sejam levados em consideração os erros experimentais dos testes de desempenho realizados.

Cada tratamento é replicado r vezes com intuito de obter os diversos valores associados à variável resposta para cada replicação. Com base nestes valores, foi estimado o erro experimental (e_{ij}) através da seguinte expressão de regressão não linear:

$$y_{ij} = q_0 + q_A.x_{Ai} + q_B.x_{Bi} + q_{AB}.x_{Ai}.x_{Bi} + ... + e_{ij} \quad (1)$$

onde:
y representa a média dos tratamentos (para iteração i,j); q o coefificente de efeitos dos fatores, sendo q_0 a média dos coeficientes; x a variável independente; e e o erro experimental (para iteração i,j).

[Jain 1991], demonstra o cálculo da soma quadrática total (SST), apresentada na equação seguinte.

$$SST = 2^k.({q_A}^2 + {q_B}^2 + {q_C}^2 ... + {q_{AB}}^2 + {q_{AC}}^2 + ...) + SSE \quad (2)$$

onde SSE é a soma dos quadrados dos erros experimentais, dado por:

$$SSE = \sum_{i=1}^{2^k} \sum_{j=1}^{r} (y_{ij} - \hat{y}_i)^2 \quad (3)$$

onde:
y_{ij} representa a medida de cada iteração; e $\hat{y}_i$ é a média de y em função das replicações.

3. Desenvolvimento

3.1. Montagem e Configuração do Ambiente de Testes

A montagem e configuração do ambiente de testes consistiu na configuração do sistema operacional em estações de laboratório, compilação e instalação dos *kernels* modificados, com suporte a PF-RING e NAPI com *device polling* na estação responsável pela captura do tráfego. Também foram criados os scripts de automatização de parte do processo de execução e coleta de dados do experimento. O experimento contou com 3 computadores com placas de rede RTL8169S, denominados A, B, e C, conectados a um *switch*. A arquitetura desse experimento é apresentada na Figura 1.

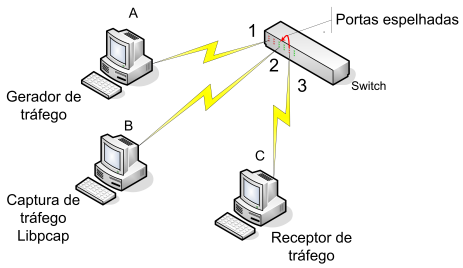

Figura 1. Arquitetura do experimento

O *switch* utilizado foi um SMC modelo 8024L2 Gigabit ethernet gerenciável.

3.2. Testes de Geração de Tráfego

Inicialmente, com objetivo de alcançar um *throughput* maior tentou-se identificar gargalos que pudessem limitar a geração de tráfego. A partir destas observações, o teste que foi executado foi com a geração de tráfego diretamente na camada 2, para isso usou-se um módulo de testes do *kernel* Linux denominado pktgen.

Como resultado da geração de tráfego com o pktgen, obteve-se um *throughput* de 796Mbps, o que significa aproximadamente 80% do valor nominal da interface adquirida. A partir deste teste optou-se por utilizar o netperf com o teste para UDP-STREAM (protocolo UDP). O resultado obtido com este teste foi de um *throughput* de 790Mbps com MTU de 1500 e 895Mbps com um MTU de 7200 (valor máximo suportado pela placa de rede).

3.3. Aplicação de Captura de Tráfego

Foi desenvolvida uma ferramenta para captura de tráfego utilizando a biblioteca libpcap. O desenvolvimento foi realizado sobre uma ferramenta mais simples já desenvolvida por [Deri 2004] e que já implementava o mecanismo de captura de tráfego. Foram implementadas melhorias, como a implementação de função para gravação dos resultados; solução de alguns problemas com unidades de tempo; mudança com a forma que os resultados eram apresentados; e alteração do controle de parada com base no intervalo de confiança. Todas as alterações foram testadas em relação à aplicação original e através de verificações da comparação do tráfego capturado pela ferramenta em relação ao mesmo tráfego capturado com a ferramenta wireshark.

3.4. Experimentação

O ambiente de testes foi composto por três computadores os quais foram interconectados através de um *switch* gigabit ethernet. Um dos computadores foi utilizado na geração de tráfego utilizando a ferramentas netperf. O tráfego gerado foi consumido por um segundo computador. Um espelhamento de portas permitiu que um terceiro computador pudesse fazer a captura do tráfego. Os experimentos foram executados de acordo com o fluxograma que pode ser observado a Figura 2. Todo o processo de replicação da captura de tráfego foi automatizado por um script de execução, o que proporcionou grande economia de tempo para a troca de configução entre as replicações.

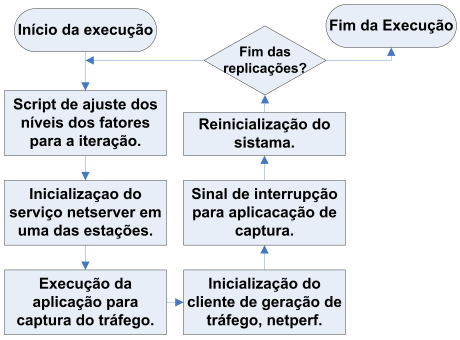

Figura 2. Fluxograma de execução dos experimentos

Inicialmente, criou-se o ambiente com a possibilidade de variar todos os fatores definidos a partir do ambiente configurado. Para facilitar o processo de combinação de fatores foi atribuído a cada fator uma letra:

- A: PF-RING - Otimização das estruturas de armazenamento em nível de *kernel*.
- B: NAPI - Otimização a partir da implementação de *device polling*.
- C: MTU - Tamanho máximo para o *frame* ethernet
- D: Rx-check - Checagem dos *frames* pela interface de rede.

Para os fatores A e B, o nível -1 representa a não aplicação do patch de otimização enquanto 1 representa que a melhoria foi aplicada. No fator D, o nível 1 representa ausência da verificação dos frames no recebimento e -1 o padrão da interface de rede que é com a verificação habilitada. Foram dezesseis experimentos diferentes, em função do número de combinação resultante do projeto fatorial (quatro fatores $\rightarrow 2^4 = 16$), sendo que cada experimento foi replicado 30 vezes, com o objetivo de obter uma amostra do erro experimental. Assim, o experimento final ficou com quatrocentas e oitenta (480) rodadas divididas em dezesseis combinações com trinta replicações cada.

4. Resultados

Foram analisadas métricas em relação à adequação dos dados do experimento ao modelo escolhido. Para a variável de resposta escolhida (bytes capturados). Para a métrica quantidade de bytes que efetivamente foram capturados pela aplicação através do uso da

libpcap, a primeira análise foi baseada na análise de variância. Foram escolhidos apenas os fatores que realmente eram significantes a análise, ou seja, que o percentual de contribuição era pelo menos duas vezes maior que o percentual o qual o erro experimental representou. Da análise da variância conclui-se que tem-se apenas 0,01% de chance do modelo ter sido definido por ruído ou erro experimental, o que garante que o modelo é significativo; ainda partir desses resultados, determinou-se os índices para equação de regressão linear para previsão do modelo de dados em função do intervalo de confiança. Os intervalos de confiança dos coeficientes da equação de previsão são apresentados na Tabela 1. Uma vez que o intervalo de confiança é bastante estreito, conseguiu-se coeficientes bastante confiáveis, pois a distância entre o valor escolhido e os limites inferior e superior ficam sempre abaixo de 5% de sua média. Isso garante a precisão do modelo matemático de previsão (regressão linar múltipla) representado na Equação 4. Esse modelo pode ser utilizado para prever situações diversas em função da combinação dos fatores utilizados num caso qualquer.

Tabela 1. Coeficientes para equação de previsão - GB capturados

Fator	Coeficiente	Erro	95% inferior	95% superior
A: PF-RING	-0,59	0,01	-0,61	-0,57
B: NAPI	0,86	0,01	0,84	0,88
C: MTUC	0,25	0,01	0,23	0,27
AB	0,44	0,01	0,42	0,46
AC	-0,22	0,01	-0,24	-0,20

$$GBcap = 1,89 - 0,59.A + 0,86.B + 0,25.C + 0,44.A.B - 0.22.A.C \quad (4)$$

Além da equação de previsão, tem-se como resultados da análise de variância o percentual de contribuição de cada fator para a métrica de desempenho em questão. Considerando como melhor resultado uma quantidade elevada de GB capturados pela biblioteca. A Tabela 2 representa os percentuais de influência de cada fator.Pode-se observar a grande influência do fator que representa a otimização voltada ao controle da situação de livelock implementado pela NAPI. Aproximadamente 51% da variação na quantidade de dados capturados foi em função do NAPI; contudo, a melhoria PF-RING representa também um fator de peso. Observa-se também, já na análise desta primeira métrica, que o fator RX-check, por apresentar percentual de influência muito baixo não aparece na análise.

Tabela 2. Coeficientes para equação de previsão - GB capturados

Fator	% Contribuição
A: PF-RING	23,86
B: NAPI	51,46
C: MTU	4,47
AB	13,41
AC	3,39
Erro	1,03

Além das observações feitas para adequação ao modelo e ao método escolhido utilizaram-se outras três verificações que consistem na avaliação da distribuição dos dados

e na verificação da aleatoriedade dos erros. A Figura 3 mostra o gráfico Box-Cox que traz como resultado a necessidade ou não de transformação dos dados.

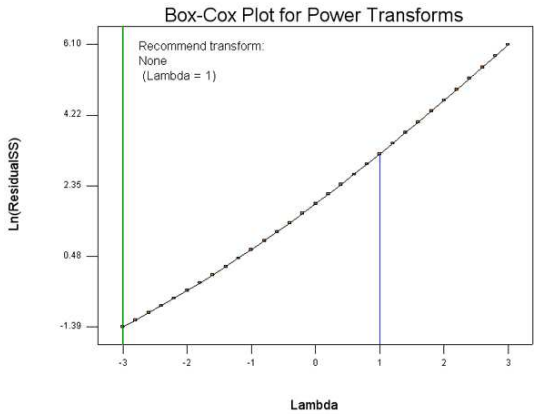

Figura 3. Box-Cox

O objetivo da transformação Box-cox é identificar, através do Lambda demonstrado no gráfico da Figura 3, a necessidade de alguma transformação nos dados. Neste caso nenhuma transformação foi necessária. A próxima análise feita foi quanto a distribuição dos dados. Confrontou-se a distribuição dos dados experimentais com a reta de probabilidade normal teórica e realmente os dados se aproximam muito deste modelo de distribuição, conforme pode ser observado na Figura 4.

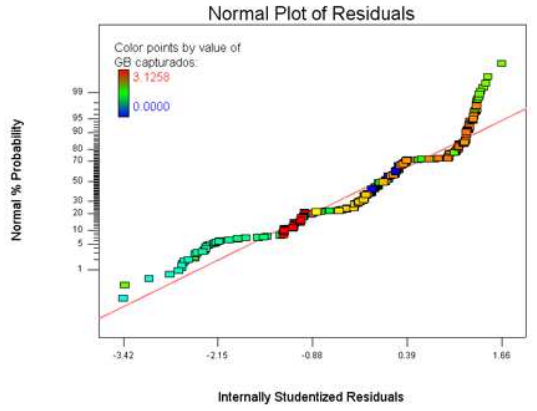

Figura 4. Gráfico da distribuição dos dados da métrica GB capturados em relação à normal

Após verificar a adequação dos resultados a uma distribuição normal foi realizada a etapa de verificação da curva que relaciona os dados reais aos previstos pela equação de previsão. Como resultado observou-se que os dados reais realmente se aproximam da

curva resultante da equação de previsão.

Outra análise feita foi em relação à interação dos fatores, ou seja, como o modelo prevê influência de fatores associados, pode-se analisar também sua influência conjunta com os gráficos de interação, conforme mostra a Figura 5

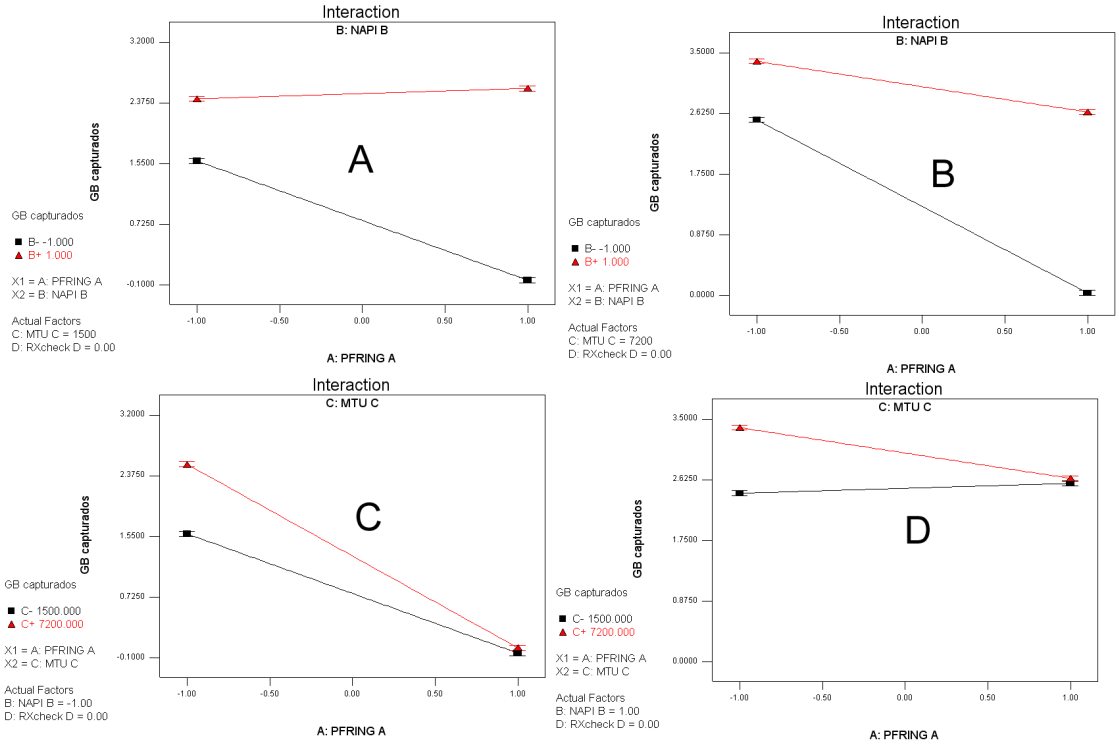

Figura 5. Gráficos de interação (A) Interação AB com MTU de 1500, (B) Interação AB com MTU de 7200, (C) Interação AC com NAPI -1 e (D) Interação AC com NAPI

Pode-se observar em A a grande influência do fator NAPI na interação AB onde fica claro que a melhoria PF-RING é praticamente um ajuste fino no desempenho. Conseguiu-se melhorar ainda mais o número de GB capturados aplicando a melhoria NAPI em conjunto com MTU alto (7200) em B; contudo, pode-se observar em C e D, a ligação entre o desempenho da libpcap e a situação de livelock interrupt . Em C, situação sem a otimização NAPI, verifica-se um baixo desempenho da biblioteca com frames de tamanho pequeno, ou seja, alta freqüência na interface, e essa situação já fica um pouco melhor com a NAPI habilitada em D. Além das interações entre fatores gerou-se a curva de otimização para a função de previsão dos dados. As quatro variáveis que estão previstas no modelo são representadas, de forma a maximizar o resultado de GB capturados.

O gráfico da Figura 6 representa a situação ótima para o ambiente testado em função da combinação dos fatores, sendo que a região vermelha da superfície representa o máximo desta função. A combinação de fatores para tal situação foi conseguida com a aplicação da otimização NAPI e a utilização da MTU em 7200 bytes sem a utilização de PF-RING, conforme mostram os eixos de coordenadas NAPI e PFRING.

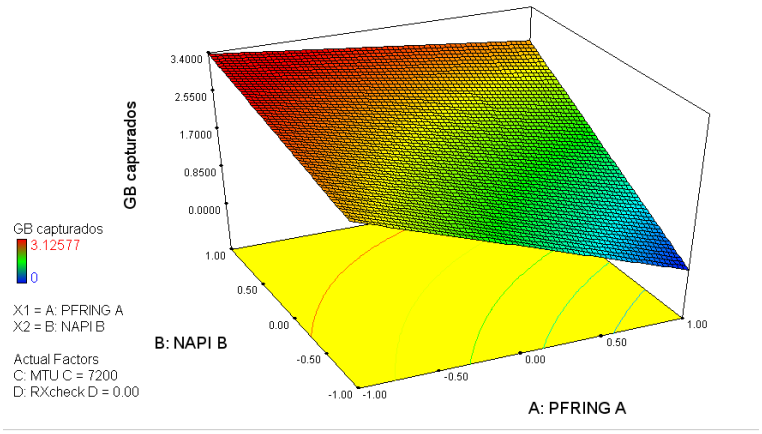

Figura 6. Gráfico de superfície da equação de previsão para GB capturados

A utilização dos fatores NAPI e PF-RING em nível alto também representa um bom desempenho, em torno de 20% abaixo da situação máxima.

5. Conclusões

O presente trabalho consistiu na avaliação de desempenho da biblioteca libpcap para captura de tráfego de redes de computadores, contudo levou-se em consideração a busca por resultados consistentes e confiáveis conseguidos através a utilização da metodologia de projeto de experimentos (DOE). A quantificação, através da análise de variância, dos efeitos dos fatores identificados foi o principal resultado deste trabalho, pois a partir dos percentuais encontrados para cada métrica de desempenho conseguiu-se uma análise completa que pode ser utilizada como parâmetro para projetos de gerência e segurança de redes.

Através da análise das métricas de desempenho pode-se concluir que a biblioteca libpcap apresenta um bom funcionamento desde que cuidados com otimizações e configurações de redes sejam observados. Os resultados deixaram bem claros quanto a necessidade da aplicação, principalmente, da melhoria de controle do *device polling*, sendo o fator de maior influência no desempenho da biblioteca analisada. Problemas de segurança graves podem ser ocasionados quando se faz um projeto de redes e aplica-se ferramentas como um IDS ou analisador de protocolos e este não tem a eficácia que deveria. É isso que pode acontecer se o projetista não se atentar aos fatores envolvidos no desempenho principalmente da biblioteca que faz a captura do tráfego utilizado pelas ferramentas.

Dos quatro fatores analisados, o NAPI foi o que apresentou maior influência da quantidade de dados capturados, sendo responsável por uma variação de 51% nessa variável de resposta. O uso de PF-RING têm influência em 23,86% nessa mesma variável, enquanto a influência conjunta desses dois fatores explica uma variação de 13,41%. O tamanho da MTU tem influência pequena, de apenas 4,47%. Apenas um dos fatores escolhidos, RX-check, não apresentou influência ao desempenho da biblioteca libpcap,

contudo isso só foi possível de observar em função dos experimentos e da metodologia utilizada.

Referências

Andreozzi, S., Antoniades, D., Ciuffoletti, A., Ghiselli, A., Markatos, E. P., Polychronakis, M., and Trimintzios, P. (2005). Issues about the Integration of Passive and Active Monitoring for Grid Networks. Bologna. Proceedings of CoreGRID Integration Workshop.

Deri, L. (2004). Improving passive packet capture: beyond device polling. Amsterdam. Proceedings of 2nd SCAMPI Workshop.

ISO (1988). Information processing systems - open systems interconnection. ISO-7698.

Jain, R. (1991). *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*. Wiley.

Junior, R. M. and Gonçalves, R. B. Uma solução para análise de tráfego em redes comutadas baseada em linux bridging e ntop.

LBNL (2007). Network resarch group. `http://www.ee.lbl.gov`.

Libpcap (2007). Documentação. `http://www.tcpcump.org`.

Montgomery, D. C. (2001). *Design and Analysis of Experiments*. John Wiley and Sons.

SNORT (2007). Overview. `http://www.sourcefire.com`.

Soares, L. F. G., Lemos, G., and Colcher, S. (1997). *Redes de Computadores: das LANs, MANs e WANs às Redes ATM*. Campus.

Wireshark (2007). Documentation. `http://www.wireshark.org/docs`.