AVALIAÇÃO E SELEÇÃO DE MODELOS DE PREDIÇÃO DE TRÁFEGO IP

OTÁVIO AUGUSTO ARAÚJO SILVA[1] ,RIVALINO MATIAS JUNIOR[2]
Projeto nº.: a-022/2009

Resumo. *Este trabalho tem como objetivo a avaliação e seleção de modelos de predição de tráfego IP em um ambiente real, assim como a modelagem de um software que poderá definir a melhor predição, dentre os modelos disponíveis, para o ambiente de produção. O trabalho possui embasamento na análise de modelos de séries temporais, tais como, Naïve, Naïve ajustado, Regressão linear simples (RLS), RLS com Sen's Slope, Alisamento Exponencial, Holt-Winters, o modelo Autorregressivo(AR) , e na utilização do índice de erro médio percentual absoluto, MAPE ,para a seleção do melhor modelo de predição.*

Palavras Chaves: Modelos de Predição, Tráfego IP, Séries Temporais, RRD.

Abstract. *The objective of this work is the evaluation and selection of prediction models for IP traffic in a real environment, as well as the modeling of a software that can determine the best prediction, among the models evaluated, for the production environment. This work has grounding in the analysis of time series models, such as, Naïve, Naïve Set, Simple Linear Regression, RLS with Sen's Slope, Exponentially Smoothing, Holt-Winters, and Autoregressive (AR), and in the use, by a software, of the Mean Absolute Percentage Error(MAPE) for the selection of the best forecasting model .*

Key Word: Predict Model, IP Traffic, Time Series, RRD.

[1] Faculdade de Computação (FACOM) – UFU, Av. João Naves de Ávila 2121 - Bloco B - Campus Santa Mônica, Uberlândia, 38400-902, octopos@comp.ufu.br

[2] Faculdade de Computação (FACOM) – UFU, Av. João Naves de Ávila 2121 - Bloco B - Campus Santa Mônica, Uberlândia, 38400-902, rivalino@facom.ufu.br

INTRODUÇÃO

Em diversos ambientes onde existem requisições e a prestação das mesmas por meio de serviços, como em uma rede de computadores, a previsibilidade da demanda por aqueles serviços é de grande valia. Isto ocorre pois podem ser usados, por exemplo, para o planejamento dos recursos disponíveis em relação aos que são necessários.

Tal previsibilidade permite, além do planejamento sobre a demanda de recursos, a correta distribuição dos mesmos.

Para tanto é de grande interesse a geração de predições sobre o tráfego IP[26] , pois o mesmo é predominantemente usado como protocolo de endereçamento nas redes de computadores, como a Internet.

Dessa forma faz-se necessário a avaliação e a seleção de modelos de predição de tráfego IP, que são baseados na análise temporal, sendo assim foi utilizado neste trabalho, modelos de séries temporais.

A necessidade da utilização de séries temporais se dá, em suma, pela sua abrangência com relação às componentes de tendência, sazonalidade, ciclo e variação aleatória.

Podemos citar como exemplos de modelos de séries temporais uni-variadas o Naïve, o processo Autorregressivo (AR), o processo de Média Móvel (MA), o processo Autorregressivo e de Média Móvel (ARMA)[1].

Além de todo trabalho desenvolvido sobre as séries temporais, o escopo do mesmo inclui a modelagem de um sistema autônomo, independente da interação humana após o início das operações, que tem a principal função de escolher o melhor modelo de predição dentre todos os modelos de séries temporais, e a partir desse modelo, gerar séries preditas de acuracidade satisfatória.

Para escolha do melhor modelo de predição, e obtenção de sua acuracidade, será usado um modelo estatístico aplicado sobre os valores preditos e os valores coletados para um determinado instante.

A modelagem desse software incluirá conceitos de sistemas operacionais, na construção de um *daemon*, assim como um paradigma não relacional de banco de dados, os quais serão explanados nas próximas seções.

Durante todo o trabalho, foi-se utilizado, como ambiente de desenvolvimento e de testes, o sistema operacional GNU/Linux[27] , seguindo, na medida do possível, o *Linux Stand Base*(LSB)[28] nas configurações do ambiente de desenvolvimento, evitando assim particularidades que possam existir em distribuições específicas.

MATERIAL E MÉTODOS

1. Séries Temporais

No estudo de séries temporais utilizamos modelos de predições clássicos, os quais são mostrados na Tabela I.

TABELA I

MODELOS DE PREDIÇÃO USADOS NO TRABALHO

Modelos
Naïve e Naïve ajustado [3]
Regressão linear (RLS) [2]
RLS + Sen's slope [10]
Alisamento Exponencial ($0{,}1 \leq \alpha \leq 0{,}9$) [4]
Holt-Winters [4]
Autoregressivos (AR) [4]

Inicialmente foi realizado um estudo teórico a cerca destes modelos, tal estudo levou em conta o modelo em si, sua utilização, seu sistema de previsão e o resultado das previsões em termos de acuracidade, ou qualidade em relação a aproximação de série predita com os valores da série observada.

A fim de analisar qual dos modelos obteria um rendimento consideravelmente aceitável, ou seja, uma boa acuracidade, foi utilizado um índice estatístico denominado MAPE (*Mean Absolute Percentage Error*).

A Equação 1 representa a fórmula do MAPE, o qual representa em termos percentuais o nível de acuracidade do modelo para uma determinada série.

Na Equação (1) temos $Q_{pre(t)}$ que representa o valor previsto no tempo *t*, $Q_{obs(t)}$ que representa o valor observado no tempo *t* e *N* que representa a quantidade de termos da série.

$$MAPE = \frac{\sum_{i=1}^{N} \frac{abs(Q_{pre(t)} - Q_{obs(t)}) \times 100}{Q_{obs(t)}}}{n} \qquad (1)$$

Foram elaborados algoritmos de testes e de parametrização de modelos de séries temporais (ex. AE com $0,1 \leq \alpha \leq 0,9$) , de forma a obter a melhor parametrização para o ajuste do modelo e a predição para uma determinada série temporal. Essa parametrização é atualizada periodicamente de forma que os parâmetros sejam dinâmicos e se relacionem com as séries de forma mais concreta e otimizada.

O conceito de séries temporais é definido em [7] como sendo um conjunto de observações ordenadas no tempo (não necessariamente espaçadas), e que apresentam dependência serial (isto é, dependência entre instantes de tempo).

Podemos assim definir formalmente uma série temporal como sendo a realização de um processo estocástico[9].

O estudo dos modelos de séries temporais citados é de fundamental importância ao projeto como um todo. Por isso, os resultados dos estudos dos mesmos serão apresentados adiante.

Antes de discorrer sobre os modelos devemos conhecer a função de autocorrelação (FAC), mostrada na Equação (2).

$$\rho k = \frac{\mathrm{Cov}(Y_t, Y_{t-k})}{\mathrm{Var}(Y_t)\mathrm{Var}(Y_t)} = \frac{\mathrm{Cov}(Y_t, Y_{t-k})}{\mathrm{Var}(Y_t)} \quad (2)$$

Temos também a função de autocorrelação parcial (FACP), que faz referência à correlação entre duas variáveis de modo a eliminar o efeito de outras variáveis. A FACP é dada pela estimação dos coeficientes por meio de uma regressão.

A FAC e a FACP serão de uso importante na identificação dos modelos AR, MA e ARMA, assim como mostra a Tabela II.

TABELA II
IDENTIFICAÇÃO DO MODELO

Processo	FAC	FACP
AR(p)	Declinante	Truncada em p
MA(q)	Truncada em q	Declinante
ARMA(p,q)	Declinante	Declinante

1.1 Modelos de predição

Para geração de predições, inicialmente , foi utilizado o módulo estatístico MINITAB. O trabalho levou em conta o estudo dos modelos mostrados na Tabela I, aos quais foram submetidas amostras reais de tráfego IP com o intuito de verificar quais dos modelos obteriam os melhores resultados.

Essas amostras foram coletadas em parceria com o Departamento de Engenharia da Computação da Duke University, sob supervisão do Prof. Kishor Trivedi e Rivalino Matias Jr. As amostras em questão são representativas de tráfego WAN e LAN, obtidas em três campi da Duke University (NC/USA), bem como da estrutura administrativa dos hospitais da Duke.

As amostras foram coletadas em diversas localizações das redes supracitadas, utilizando-se um método de coleta passiva de tráfego em redes comutadas desenvolvido pelo orientador desse projeto [5] [6].

A estratégia de amostragem está considerando períodos de coleta que garantam observar o tráfego típico, comportamentos cíclicos e possíveis eventos sazonais que possam caracterizar o tráfego analisado.

Os modelos de predição analisados no escopo desse trabalho, assim como sua definição foram os seguintes:

i) Naive:

Também conhecido como método de previsão ingênuo (*naive*), descrito por um *random walk* (passeio aleatório).

É um modelo relativamente simples e que em determinadas séries apresenta uma boa acurracidade se comparado aos demais modelos.

O modelo é dado pela Equação 3 abaixo, onde a predição no tempo *t* é dada pelo último valor observado

$$\hat{Y}_t = Y_{t-1} \qquad (3)$$

ii) Naive Ajustado:

O modelo Naive ajustado, mostrado na Equação 4, é um modelo ingênuo também, porém ele tenta ajustar, ou melhor, suavizar as discrepâncias que ocorrem no Naive, pois utiliza para a predição em um momento *t+1* e um fator multiplicador ao valor observado Y_t, que tem o intuito de proporcionar uma predição mais embasada na série em questão do que o Naive. Isso acontece pois este fator multiplicador utiliza os valores observados no tempo atual *t* e no tempo *t-1*.

$$\hat{Y}_{t+1} = Y_t \left(1 + \frac{T_t - Y_{t-1}}{Y_{t-1}} \right) \qquad (4)$$

iii) Regressão Linear Simples (RLS):

O modelo de Regressão Linear Simples (RLS), como é descrito em [8] consiste basicamente em estimar uma reta que melhor represente a série em questão. Essa reta é obtida, por exemplo, pelo critério dos mínimos quadrados, na qual a reta é descrita pela Equação 5 de forma que *a* e *b* minimizem a Equação 6.

$$\mathbf{y}_i = \mathbf{a} + \mathbf{b}\ \mathbf{x}_i \qquad (5)$$

$$\mathbf{Q} = \sum_{i=1} (\mathbf{y}i - (\mathbf{a} + \mathbf{b}\ \mathbf{x}i))^2 \qquad (6)$$

Temos na Equação 5 a variável dependente representada por y_i, e x_i que é denominada variável independente.

$$b = \frac{\sum xy - \frac{\sum x \sum y}{n}}{\sum x^2 - \frac{(\sum x)^2}{n}} \qquad (7)$$

$$a = \frac{\sum y - b\sum x}{n} \qquad (8)$$

A regra de otimização baseada na segunda derivada permite prover recursos para demonstrarmos que os valores de *a* e *b* que minimizam a Equação 6 são dados pelas Equações 7 e 8 respectivamente.

iv) RLS + Sen's slope:

O método Sens Slope é referenciado como um método não paramétrico para a estimação de tendência, neste caso, ajustado a um modelo linear [TRIVEDI, 2001].

O modelo de Sen´s Slopes é semelhante ao modelo de Regressão Linear Simples (RLS) porém ele tende a melhorar a acuracidade das predições em relação ao RLS.

No modelo de Sen's Slopes, temos uma equação do tipo $y_i = a + Qx_i$ que representa o modelo. Onde o parâmetro *a* é estimado do mesmo modo que no modelo de RLS. O parâmetro Q, é estimado pela mediana entre os $n*(n-1)/2$ termos de Q_k, ilustrados pela Equação 10.

Em Q_k, Equação 9, temos $1 \leq k \leq n*(n-1)/2$, de tal forma Q_k é o calculo de slope entre os dados $x_{i'}$ e x_i. O valor da série no instante i' é representado por $x_{i'}$, e no instante i é representado por x_i. Onde i' é um tempo anterior a i.

$$Q_k = \frac{x_{i'} - x_i}{i' - i} \qquad (9)$$

$$\frac{X_t - X_1}{t-1}, \frac{X_t - X_2}{t-2}, \ldots, \frac{X_t - X_{t-2}}{2}, \frac{X_t - X_{t-1}}{1} \qquad (10)$$

O algoritmo para este modelo consiste em:

1) Calcular o parâmetro *a* da RLS, que será o mesmo do Sen's Slopes;
2) Calcular os Q_k para a série em questão;
3) Organizar os Q_k em ordem crescente;
4) Localizar a mediana, que no caso irá representar o parâmetro Q do modelo.

v) Alisamento Exponencial(AE):

É um modelo sofisticado, em relação aos modelos anteriores, pois leva em conta os pesos de médias móveis, porém os pesos das observações decrescem ao passo que se distanciam no passado.

Existem diversas variações deste modelo tanto para séries sazonais como não-sazonais, porém a abordagem mais comum é a que se encontra ilustrada na Equação 11 abaixo.

$$\hat{Y}_{t+1} = \hat{Y}_t + \alpha(Y_t - \hat{Y}_t) \qquad (11)$$

Neste modelo, α é um parâmetro que representa o peso ou constante de alisamento que está entre 0 e 1. O intuito dessa constante é de atenuar os efeitos da aleatoriedade.

O modelo é fortemente adotado por organizações empresariais que fazem uso de predições em cima de séries temporais, isso por causa da simplicidade de implementação e por sua capacidade de gerar resultados satisfatórios em geral.

Porém uma das grandes dificuldades relacionadas a este modelo está em estimar o valor do parâmetro α, uma vez que é necessária a análise de uma gama de valores no intervalo entre 0 e 1 na verificação de qual dos valores para α apresenta uma maior acuracidade. Esse processo é trabalhoso e pode demandar alto custo computacional dependendo do tamanho e do tipo de dado da série.

vi) Holt-Winters(HW):

O modelo de Holt-Winters é o modelo a se considerar quando se trata de uma série com nível, ou nível acompanhado de tendência com ou sem fator sazonal.

Existem dois tipos de distinções do Holt-Winters a serem consideradas, denominadas de forma aditiva e forma multiplicativa. A escolha da forma a ser usada depende das características da série [NEWBOLD & BOS, 1993].

Ambos os modelos, tanto o aditivo como o multiplicativo possuem as Equações de L_t que representam o nível, T_t que representam a tendência e F_t que representam a sazonalidade.

As predições são obtidas pelas Equações 15 e 16 para forma aditiva e Equações 20 e 21 para forma multiplicativa. Os parâmetros α, β e γ variam no intervalo de 0 e 1.

Aditivo:

$$L_t = \alpha(Y_t - F_{t-s}) + (1-\alpha)(L_{t-1} + T_{t-1}) \qquad (12)$$

$$T_t = \beta(L_t - L_{t-1}) + (1-\beta)T_{t-1} \qquad (13)$$

$$F_t = \gamma(Y_t - L_t) + (1-\gamma)F_{t-s} \qquad (14)$$

$$\hat{Y}_t(h) = L_t + hT_t + F_{t+h-s} \quad h = 1,2,\ldots,s \qquad (15)$$

$$= L_t + hT_t + F_{t+h-2s} \quad h = s+1, s+2,\ldots,2s \qquad (16)$$

Multiplicativo:

$$Lt = \alpha\frac{Y_t}{T_{t-1}} + (1-\alpha)(L_{t-1} + T_{t-1}) \qquad (17)$$

$$Tt = \beta(L_t - L_{t-1}) + (1-\beta)T_{t-1} \qquad (18)$$

$$Ft = \gamma\frac{Y_t}{L_t} + (1-\gamma)F_{t-s} \qquad (19)$$

$$\hat{Y}_t(h) = (L_t + hT_t)F_{t+h-s} \quad h = 1,2,\ldots,s \qquad (20)$$

$$= (L_t + hT_t)\,F_{t+h-2s} \quad h = s+1, s+2,\ldots,2s \qquad (21)$$

O valor de *s* representa o número de períodos, por exemplo, por ano. Então se quisermos períodos trimestrais teremos *s=4*.

No modelo de Holt-Winters a determinação dos valores iniciais pode ocorrer de várias formas [CHATFIELD 1978].

vii) Médias móveis(MA):

Dizemos que uma série temporal Y_t caracteriza-se por um processo de médias móveis quando o processo é uma combinação de choques passados, no qual podemos definir um processo de médias móveis de ordem *q*, ou MA(q).

$$Y_t = \mu + \varepsilon_t - \varphi_1\varepsilon_{t-1} - ... - \varphi_q\varepsilon_{t-q} \qquad (22)$$

Como $\hat{Y}_t = Y_t - \mu$, teremos:

$$\hat{Y}_t = (1 - \varphi_1 B - ... - \varphi_q B^q)\varepsilon_t = \theta(B)\varepsilon_t \quad (23)$$

Onde:

$$\theta(B) = 1 - \varphi_1 B - ... - \varphi_q B^q \qquad (24)$$

Um MA de 1ª ordem, ou MA(1), pode ser descrito pela Equação 25.

$$Y_t = \mu + \varepsilon_t - \varphi\varepsilon_{t-1} \qquad (25)$$

O processo de estimação de um modelo MA(q) é feito através de métodos iterativos, assim como mostra [9].

Um algoritmo para estimação do modelo MA(q) é mostrado abaixo:

1) Estimar o parâmetro independente da variável ε_{t-k} , no caso μ , que representa um valor inicial através da média amostral.
2) Estima-se um AR com várias defasagens para obter os parâmetros de φ.
3) Calcula-se a tabela de ε_{t-k}.
4) Fazer uma Regressão Linear Múltipla (RLM) em cima da tabela de ε_{t-k} para obter os novos parâmetros de φ.
5) Voltar ao passo 3 enquanto a diferença entre os parâmetros de φ não seja insignificante, por exemplo, uma diferença de 0,01.

viii) Autorregressivos médias móveis(ARMA):

O processo de previsão utilizando o modelo ARMA é muito utilizado na área de econometria, por exemplo, na predição do PIB, da inflação, do desemprego, da variação da taxa de câmbio, das taxas de juro. A pressuposição para estas séries é que elas são estacionárias.

Neste modelo são analisadas propriedades estocásticas das séries temporais levando-se em conta os *logs* das próprias variáveis e o erro (*white-noise*).

O modelo ARMA é uma combinação dos modelos AR e MA, sendo assim representado por ARMA(*p*,*q*) onde *p* é referente à ordem do AR e *q* é referente à ordem do MA.

A forma geral do ARMA(*p*,*q*) é dado pela Equação 26.

$$Y_t = \theta + + \alpha_1 Y_{t-1} + ... + \alpha_q Y_{t-q} + \varepsilon_t - \varphi_1 \varepsilon_{t-1} — ... - \varphi_q \varepsilon_{t-q} \quad (26)$$

Um ARMA(1,1) é dado pela Equação 27.

$$Y_t = \theta + \alpha_1 Y_{t-1} + \varepsilon_t - \varphi_1 \varepsilon_{t-1} \quad (27)$$

O processo de estimação do ARMA é iterativo e trabalhoso, além de ser de alto custo computacional para séries de tamanho relativamente grande. Sua estimação envolve o processo de minimização para estimação dos parâmetros e recálculo dos parâmetros a cada iteração, até o momento em que sejam encontrados os valores dos parâmetros do modelo que minimizam a soma dos quadrados dos resíduos.

Um algoritmo para estimação dos parâmetros do ARMA é descrito abaixo:

1) Estimação inicial dos parâmetros. O *software* MINITAB inicialmente atribui todos os valores dos parâmetros como 0,1.
2) É feita a minimização (processo iterativo) de S*(η) [7].
3) Caso os parâmetros sejam tais que o modelo convirja, então retornar os parâmetros estimados. Senão voltar ao passo 2).

2. ENGINE ESTATÍSTICO

Após o estudo dos modelos, implementou-se um *engine* estatístico para a seleção automática dos modelos de predição. Este *engine* foi desenvolvido na linguagem procedural C, o qual foi especializado para a execução em um sistema operacional GNU/Linux com kernel mínimo 2.6.27[25].

O principal objetivo deste *engine* é a automação da geração de séries temporais utilizando o tráfego colhido, a partir do melhor modelo de predição selecionado. As métricas desse tráfego, assim como as séries temporais serão armazenadas em um banco de dados não relacional, que será descrito na seção 2.2.

O processo de criação deste engine, como um *daemon*, a partir de um processo corrente no sistema não será detalhado de forma intrínseca, pois foge do escopo desse trabalho.

2.1 Engine (Daemon)

Um *daemon*, *Disk and Execution Monitor*[11], é um processo que após passar por algumas particularidades, torna-se independente de terminais e pouco dependente de outros processos já em execução. Seu principal objetivo é executar atividades de monitoramento do sistema e prover serviços para outros processos.

O termo se originou com o Unix, mas a maioria dos sistemas operacionais utiliza *daemons* de alguma forma. No Unix, os nomes dos *daemons* convencionalmente terminam em "d", como inetd, httpd, nfsd.

Os *daemons* nunca deveriam ter comunicação direta com o usuário, um daemon não deve ter nenhuma relação com terminais [13]. Dessa forma, um *daemon* não deve se comunicar diretamente com o usuário, toda a comunicação deve passar por algum tipo de interface, que pode ser tão complexo como uma interface gráfica GTK+, ou tão simples como um *process signal*. Um *process signal* é uma forma de comunicação entre processos utilizados no Unix e em outros sistemas operacionais compatíveis com POSIX. Essencialmente é uma notificação assíncrona enviada para um processo a fim de notificá-lo de um evento que ocorreu. Uma melhor definição sobre *process signals* encontra-se em [11] e [12].

Foi-se adotado no trabalho a comunicação do daemon para saída de estados, isso é, o daemon apenas notificaria em quais processos de execução o mesmo está a partir da passagem pelos mesmos, e a entrada por *process signals*, agora chamados genericamente de sinais. Para suportar essa forma de entrada, o *daemon* teria *handlers*[11] específicos para um conjunto de

sinais, permitindo que aquele pudesse receber instruções para que se deslocasse para um determinado estado de execução.

Em um sistema Unix baseado no SystemV[11] todo processo, processos filhos, é fruto de um outro processo, processo pai, que por início foi lançado pelo processo *init* (processo lançado diretamente pelo kernel e o qual é pai de um processo quando o pai deste morre em detrimento da execução). Dessa forma os processos filhos tem várias relações de dependência com seus pais, sendo os f*iles descriptors*[11], diretório corrente, terminal relacionado, entre vários outros. Essas relações de dependência não seriam atrativas em um *daemon*, para tanto, este após ser lançado por algum processo corrente, tem várias das relações de dependência mencionadas alteradas, como:

1. Desligamento do terminal relacionado, através da troca de sessão do processo, não sendo mais influenciado por sinais enviados aos terminais.

Em sistemas Unix-like[14] todos os processos são relacionados a uma seção, por padrão na mesma seção de seu pai, e cada seção é relacionada a um, ou nenhum, terminal. Todo sinal direcionado a um terminal, também é perpassado para todos os processos em *foreground*[15] de uma seção.

2. Troca do diretório corrente pelo diretório raiz.

Após um processo entrar em execução, seu diretório corrente é o mesmo que o do pai, portanto a troca pela raiz permite que o *daemon* continue sua execução mesmo se o diretório de onde ele foi executado seja desmontado ou movido.

3. Troca dos *files descriptors* hereditários.

Há três *files descriptors* de entrada e saída (I/0) padrão, o *stdin* para entrada, *stdout* para saída e *stderr* para saída de erro, os quais são relacionados aos terminas e consequentemente herdados do processo pai. As bibliotecas padrões do sistema, como glibc[16], executam as chamadas de I/O para aqueles *files descriptors* padrões, porém com o *daemon* não relacionado a um terminal, tais chamadas resultariam em erro, para tanto, aqueles descritores são alterados

para apontar arquivos nulos, como */dev/null* [11] ou são fechados, não apontando para arquivo algum.

4. Mudar a máscara de criação de arquivo.

Uma máscara de criação de arquivo é associada a cada processo criado. Ela especifica como as permissões de arquivo são restritas para cada arquivo criado pelo processo. Assim como os outros atributos, esse é herdado pelo filho, e é alterado de forma implícita. A mudança dessa máscara para um valor coerente, oferece uma maior segurança quanto ao acesso de outros processos aos arquivos criados pelo *deamon*. Essa lista de atributos dependentes e herdados não é uma lista exaustiva, tendo vários outros atributos que devam ser observados e devidamente alterados, mas que fogem do foco do presente trabalho e portando não serão mencionados.

A escolha de um *daemon* para a execução dos modelos de predição, assim como a escolha do melhor modelo para futuras geração de séries temporais, foi feita em detrimento das qualidades e características de um *daemon* já mencionadas anteriormente, possibilitando que não se necessite ter um usuário logado no sistema durante toda a execução do *deamon*, assim como permite uma maior robustez do sistema quanto às influências externas decorrentes da concorrência de atividades em um sistema operacional moderno.

Tendo em vista essa escolha, toda a arquitetura do sistema foi altamente influenciada, pois é de extrema importância que as atividades de predição e escolha sejam feitas, completamente ou parcialmente, independentes de iniciativa humana. Desse modo, como um comportamento esperado, o *daemon* será inicializado durante o carregamento do sistema operacional e seus serviços, e o mesmo deverá ser terminado quando o sistema operacional for desligado.

Sendo assim, todos os eventos do sistema são salvos pelo *daemon* em seus arquivos de log, possibilitando que apenas em casos de exceção seja necessário auditorar problemas de execução. Da mesma forma é possível, através de sinais, fazer com que o sistema reavalie dentre todos os modelos de predição, aquele que mais se adeque ao regime do tráfego corrente.

Em seu funcionamento o *daemon engine* depende que um outro daemon, chamado de *probe*, o qual escreve os dados, colhidos da rede, em um banco de dados. Esse *probe daemon*

não é de escopo desse projeto, o mesmo teve seu desenvolvimento feito em cooperação com a Duke University.

2.2 Armazenamento de dados

Tendo em vista a necessidade de manipular dados, resultado da captura e processamento do tráfego, ou como resultado dos modelos de predição, utilizou-se um banco de dados (DataBase).

Em suma, um banco de dados é uma abstração genérica de algo capaz de armazenar uma quantidade de dados, de forma a recuperar os dados de acordo com algum regime previsto, sejam os dados como foram gravados ou após algum processamento, pós gravação ou pré leitura, dos mesmos.

Neste trabalho houve a necessidade de se manipular duas bases de dados distintas, pois em uma base de dados encontram-se os dados colhidos pelo probe e na outra base de dados, o resultado dos modelos de predição. Uma melhor visão da arquitetura geral do sistema

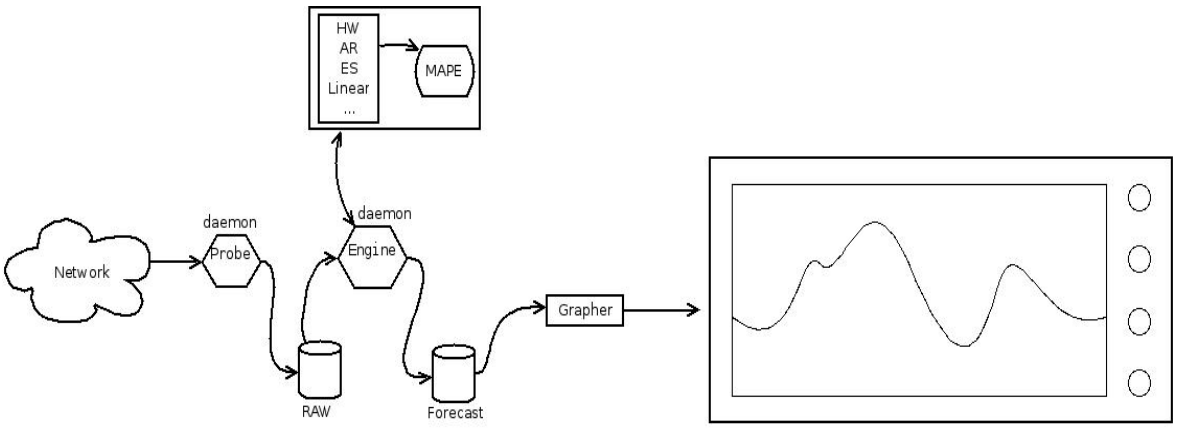

Figura 1: Visão da Arquitetura do sistema

encontra-se na Figura 1.

Salientando-se que o escopo desse trabalho se limita aos elementos: engine, raw, forecast, métodos de predição e MAPE, tendo os outros elementos são fruto de avanços no projeto em cooperação. E o *daemon engine* em questão, lê os dados providos na base de dados *RAW*, gera as predições, com todas as operações que isso acarreta, e salva-as na base de dados *Forecast.*

2.2.1. Round Robin DataBase

Dentre os bancos de dados existem diversos paradigmas que ilustram as necessidades compreendidas em cada classe de problemas. Podemos ilustrar como alguns dos paradigmas de banco de dados, os bancos de dados relacionais e os round robin.

Em um banco de dados relacional(relational database[17]), os dados são encontrados por meio de características comuns encontradas no conjunto de dados, isso é, o conjunto de dados compartilha relações diretas, seja por suas características, seja por sua interação com um outro conjunto de dados.

Como exemplo, um banco de dados relacional que deseja representar os alunos na Universidade Federal de Uberlândia (UFU), deverá entre vários outros atributos, armazenar o nome, CPF e seu curso. Porém um curso também tem seus atributos, e a ligação entre um aluno e a qual curso ele pertence é feito por uma relação de atributos.

Os grupos resultantes dos dados são organizados, pois compartilham atributos, podendo ser ordenados pelos mesmos, seguindo um determinado critério de ordenação.

Na Figura 2 temos um exemplo de 2 tabelas, que representam algum fator do mundo real, que compartilham uma relação através de seus atributos.

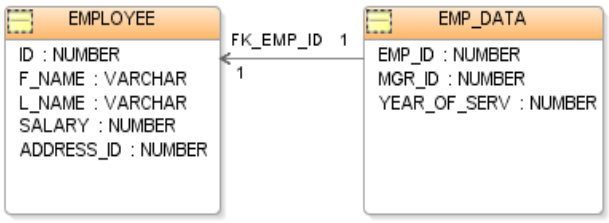

Figura 2: Schema de um Relational DataBase

Além das características mencionadas, um banco de dados relacional armazena dados de forma linear incremental, isso é, a cada dado inserido em sua base de dados, o mesmo é armazenado em uma nova posição, tendo um crescimento do banco de dados proporcional ao tamanho de um dado do tipo inserido. Além de que, essa forma de armazenamento permite a disponibilidade de todos os dados armazenados, desde que os mesmos não sejam explicitamente removidos.

Por outro lado, temos um paradigma de banco de dados, o Round Robin Database(RRD)[18] , cuja afinidade é maior com séries temporais.

O conceito de Round Robin(RR) genericamente dá-se pelo uso de estruturas indexadas ou ligadas, e circulares, isso é, todas as estruturas sobre um regime RR possuem um campo capaz de localizar ou apontar, os termos de acordo com suas posições dentro da estrutura, e todas as estruturas estão dentro de uma lista circular. A Figura 3 demonstra uma lista simples e a Figura 4 uma lista circular.

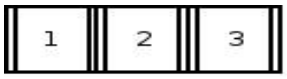

Figura 3: Lista simples

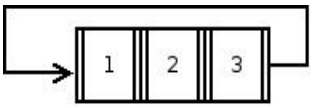

Figura 4: Lista circular

A partir do último elemento de uma lista circular, consegue-se acessar o primeiro elemento. Além de se usar uma lista circular, o uso de uma estrutura RR é feito através de um conjunto determinístico de elemento dentro das listas, de tamanho conhecido, e o regime RR é aplicado aos elementos da lista.

Esse regime simplesmente é a manipulação da lista, a partir do primeiro elemento, de forma a atender ou preencher todos os elemento da lista.

Tendo em vista essa definição genérica, um RRD é uma base de dados onde os elementos são inseridos nas posições de uma lista, de tamanho previamente definido, e no caso onde todas as posições estejam ocupadas, a inserção será na primeira posição da lista. Logo em um RRD o banco de dados não cresce além do esperado, pois existe um máximo de posições de armazenamento definidas.

Uma representação de um RRD seria um círculo com alguns pontos plotados na borda. Estes pontos são os locais onde os dados podem ser armazenados. Quando os dados atuais são lidos ou escritos, um ponteiro, para a posição corrente, se move para o próximo elemento. Como estamos em um círculo, que não tem início ou fim, o conjunto de dados não

irá crescer em tamanho, pois as posições mais antigas são sobrescritas a medida que os dados são inseridos. A Figura 5 demonstra esse conceito.

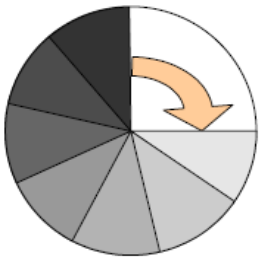

Figura 5: RRD Schema

Diferindo-se de um banco de dados relacional, os dados armazenados em um RRD não são tuplas, os mesmos são valores que podem ser armazenados, na linguagem C, em um *long int* ou *double* na arquitetura do sistema operacional.

Dessa forma, uma entrada de dados terá como chave de busca a data que o dado foi inserido, um número de 32 bytes, baseado em uma data da década de 1970, chamada de Unix-time[19]. Essa chave é definida na inserção do dado, não precisando ser necessariamente a data corrente.

O uso de um RRD neste trabalho deu-se, além da não necessidade de monitorar o uso em disco do banco de dados, mas também pela alta afinidade que um RRD tem com séries temporais, pois todos os dados nesse tipo de banco de dados são armazenados seguindo duas premissas, a data que o dado foi inserido, e o valor do dado em si. Assim, o resultado de uma série temporal pode ser diretamente gravado em disco, tendo o seu valor colocado como um dado, e o tempo que esse valor representa, como chave de busca na entrada do RRD.

Para a manipulação de dados em um RRD, foi utilizada uma *Application Programming Interface,* API[21], para a linguagem C, provida pela ferramenta RRDTool[20], a qual pode ser encontrada para baixar na mesma referência.

Essa API fornece diversas funções, que vão desde funções de inserção e busca de dados a plotagem dos mesmos em várias plataformas gráficas. Porém serão explanadas apenas as funções e aplicações utilizadas durante o desenvolvimento do *daemon engine*, o qual tem também como requisito, a manipulação dos dados gravados nas bases de dados.

2.2.2. API RRD

Para a manipulação de um RRD, a priori necessita-se de um database nesse paradigma, para tanto, foi necessário utilizar as funções providas na API para tal criação.

Como um dos argumentos para a criação, um RRD necessita de um Unix-time inicial, o qual fixa aquele tempo como sendo o menor tempo da base, portanto não terá nenhum dado inserido antes desse tempo. Também é necessário determinar um *step* em segundos, sendo o período de tempo o qual o database será atualizado com novos valores, sendo que o RDD ignora a entrada de valores de tempo entre dois *steps*, e no caso onde não haja valores para serem informados, será armazenado um valor nulo, que varia entre as plataformas operacionais.

Além dos argumentos mencionados, também é necessário definir um conjunto de Data Source, DS, que em suma é a fonte de dados a ser inserido. Em um RRD pode-se ter várias fonte de dados, cada fonte recebe um nome pelo usuário, e a cada inserção é definida de qual fonte aquele dado provem.

Um DS é composto de acordo com a Figura 6.

```
DS:variable_name:DST:heartbeat:min:max
```

Figura 6: Def. de um Data Source

Uma DS é identificada pelo nome, *variable_name* na Figura 6, assim na busca ou na inserção de valores, além de um Unix-time, também se especifica o nome da DS.

Um RRD permite gravar os dados através de várias funções, Data Source Type, identificado como DST na Figura 6. Estas funções permitem que antes dos dados serem gravados na base de dados, eles sofram alguns processos aritméticos. Para tal, basta definir o DST como uma das seguintes funções:

1. Counter: salvar a taxa de variação dos valore ao longo do tempo, supondo que o valor é sempre crescente, a taxa é sempre igual ou maior que 0.
2. Derive: semelhante ao contador, mas permite que os valores sejam negativos.
3. Absolute: armazena o valor atual dividido pelo step, sempre admitindo que o valor anterior é 0.
4. Gauge: não faz nenhuma mudança no valor, salva-o como tal.

Diferentemente de um banco de dados relacional, um banco de dados *round robin* apenas aceita dados em intervalos pré definidos e apenas um dado por DS em cada intervalo.

Portanto, no caso de um determinado intervalo de tempo não ter dados para um DS, é possível especificar um tempo máximo, antes de armazenar um valor nulo na base de dados. Esse valor de tempo máximo , que é definido como *heartbeat* na Figura 6, deve ser sempre maior que o *step*, pois ele limita o tempo total até a obtenção do dado e não o tempo de espera após um *step*. Assim o *heartbeat* possibilita que qualquer dado direcionado a um DS seja armazenado como um dado provindo no Unix-time com *step* segundos, com um atraso máximo.

Esse atributo de atraso é muito útil em ambientes onde os dados provém de um universo de dados não previsível. Entretanto, nesse trabalho não foi utilizado atraso, já que a coleta de tráfego sempre obtém algum resultado.

Por último, na definição de um Data Source, temos os valores mínimos e máximos que um DS pode armazenar, sendo que tudo fora desse intervalo é desconsiderado. Estes valores podem ser definidos como U, ou universo, não tendo nenhuma limitação.

Além da definição de um DS, uma base de dados *round robin* precisa de um último argumento, *Round Robin Archive* (RRA), o qual em suma, limita o tamanho máximo da base de dados, em valores de *rows*, aplicando uma função algébrica em um grupo de *nsteps* entradas, ou n entradas que estão em intervalos de *steps* segundos.

Pode-se ter vários RRA diferentes em um banco de dados seguindo o paradigma round robin, mas um RRA é universal a todos os DS. Um RRA é composto de acordo com a Figura 7.

Figura 7: Def. Round Robin Archive

Na definição de uma RRA, inclui-se o nível de consistência dos dados, chamado de *XFiles Factor* , xff na Figura 7. Este fator, que varia de 0(nenhum) a 1(todos) , os valores dentro do intervalo compreendido pelo RRA podem ser nulos ou desconhecidos. E na hipótese de que aquele fator seja maior do que o valor colhido, toda a fonte de dados, ou DS, é considerada inconsistente.

Além do parâmetro de consistência, também há outros como Consolidation Function(CF), que é uma função, aplicada ao conjunto de dados de tamanho nstep, que determina que a cada n ocorrências de um step, o conjunto será submetido àquela função.
As CFs podem ser as seguintes:

1. AVERAGE.
2. MINIMUM.
3. MAXIMUM.
4. LAST.

O resultado de um CF é armazenado como uma posição das *rows*. Com isso o banco de dados na verdade aceita qualquer número de entradas de dados, mas para um grupo de entrada, apenas um certo número delas serão aplicadas nas CFs, e é o resultado dessas CFs que podem ser recuperados nas buscas. Portanto, tendo um nome de uma DS e um intervalo em Unix-time, recupera-se o resultado das CFs inclusas naquele intervalo.

Para tanto, a função de *fetch* ou busca de um RRD, utiliza o intervalo em Unix-time, os nomes dos DS e uma CF, retornando o resultado de todos os elementos que correspondem a busca dentre todos os elementos pertencentes a todos os DS dentro do banco de dados.

2.3 Integração Daemon e RRD

Após descrito os conceitos e métodos para o entendimento tanto do daemon engine e da base de dados Round Robin, passamos para a integração dos mesmos no presente trabalho.

O engine utilizou da API do RRD, disponível em [22] na versão 1.4.4, através de compilação *buildin*, ou seja, o daemon foi compilado com as instruções de manipulação da base de dados incluídas em seu binário. As duas bases de dados, raw e forecast na Figura 1, foram criadas com os mesmos *DataSource* e com as mesma RRA.

Utilizou-se 6 *DataSources* nos bancos de dados, pois como mencionado, foi escolhido o tráfego DHCP como métrica. Esse tráfego é composto por 6 tipos de mensagem no protocolo[23], para tanto 1 DS para cada tipo de mensagem.

O desenvolvimento do daemon engine supõe que o trafego de DHCP será colhido a cada hora, para tanto o RRD raw terá novos dados a cada hora, finalizando o dia com 6 séries de 24 termos cada. Portanto ambas as bases de dados devem armazenar 6 seríes de 24 termos,

sendo assim foi-se definido que os RRD usados teriam apenas 1 RRA cada, o qual foi definido como segue a Figura 8.

Figura 8: Def. RRA para D.B Raw e Forecast

Desta forma, os RRD utilizados gravarão 1 RRA a cada hora, contendo o último valor inserido na base de dados, tolerando, antes de definir o DS inconsistente, no máximo 50% de valores desconhecidos ou nulos, e comportando no máximo 24 RRA. Assim é possível recuperar, através da função *fetch*, todos os dados inseridos, e portando os 24 termos das 6 séries numéricas, definidos como 6 DS distintos em cada base.

Definido tais suposições ao engine, o seu funcionamento se baseia na leitura da base Raw de 24 termos de 6 séries diferentes, aplicando as mesmas ao melhor modelo, quando já definido, e gravando o resultado na base Forecast. Esse resultado é a predição do tráfego do próximo dia, hora a hora, de acordo com o melhor modelo.

Na definição desse trabalho, o melhor modelo seria escolhido após 48 horas da inicialização do daemon, pois só após esse período de tempo seria possível alimentar os modelos de predição com as séries numéricas e comparar os dados preditos com os colhidos nas outras 24 horas, podendo assim aplicar o MAPE e obter, dentre todos os modelos, aqueles que mais se aproximaram dos dados colhidos.

Também foi definido que seria possível forçar a leitura de Raw com um modelo já especificado, através de sinais enviados ao daemon engine, fazendo-o ir para um estado de execução específico, pulando a escolha do melhor modelo.

Assim como é possível pular a escolha do melhor modelo, também é possível forçar a escolha do mesmo, através de sinais direcionados só processo deamon.

Dentre uma variedade de sinais, disponíveis no sistema operacional GNU/Linux, convencionou-se que o sinal de número 10, SIGUSR1[24] , é destinado a forçar o engine a recalcular e encontrar o melhor modelo de predição, e o envio dos sinais de número 36 ao 42 forçam o daemon a definir o modelo representado pelo sinal como o melhor modelo, onde :

36. Naive
37. Naive Ajustado
38. Regressão Linear Simples (RLS)

39. RLS + Sen's slope
40. Alisamento Exponencial(AE)
41. Holt-Winters(HW)
41. Médias móveis(MA)
42. Autorregressivos médias móveis(ARMA)

RESULTADOS

A fim de se analisar os modelos de predições estudados teoricamente, colheu-se tráfego em um ambiente real, e na busca de boas métricas para tráfego IP coletado, adotou-se a filtragem de tráfego para o protocolo *Dynamic Host Configuration Protocol*(DHCP [23]).

De todo o tráfego colhido, realizou-se uma análise mais especifica em relação às requisições DHCP_DISCOVER[23] obtidas do tráfego de um setor específico da Duke University no intervalo do dia 09 de Agosto de 2009 a partir das 04h00min ao dia 11 de Agosto de 2009 às 14h00min.

As séries de dados, utilizadas como entrada para os modelos de predição, colidas durante esse período foram coletadas em dias normais sem anomalias, as quais poderiam influenciar no real desenvolvimento das séries em questão.

A Tabela III mostrada abaixo deixa evidente que os modelos AE(0.3), ARMA(1,1), HW-Multiplicativo e HW-Aditivo, utilizados com seus respectivos parâmetros indicados na mesma tabela, apresentaram os melhores desempenhos dentre os modelos selecionados para testes.

Esse desempenho tem se mostrado constante com outras séries provenientes da mesma fonte, porém em dias e horários diferentes.

TABELA III
ACURACIDADE DOS MODELOS

DHCPDISCOVER	MAPE %	Acuracidade %	
Naive	32,56	67,43	
Naive Set	61,60	38,39	
AE(0.3)	28,10	71,89	
RLS	34,49	65,50	
Sen`s Slopes	33,74	66,25	
ARMA(1,1)	28,34	71,66	
MA(1)	33,03	66,97	
HW-Multiplicativo	27,00	73,00	alpha=0,3
HW-Aditivo	28,00	72,00	delta=0,1
			gama=0,1
			Seasonal length: 0,6

CONCLUSÃO

Durante o desenvolvimento desse trabalho conclui-se que a escolha pela captura do tráfego DHCP é uma boa métrica para ser utilizada nos modelos de predições, pois representa a rede em termos de demanda de serviços, já que todo novo usuário ganha um novo IP na rede, provido pelo servidor de DHCP. Assim as requisições DHCP representam o estado de demanda na rede em níveis de usuários, sendo uma boa métrica para a avaliação como determinante no tráfego IP que terá-se na rede durante um intervalo de tempo.

Após avaliações experimentais, como já mencionado, os modelos AE(0.3), ARMA(1,1), HW-Multiplicativo e HW-Aditivo destacam-se pela ótima acuracidade em relação aos valores preditos e aos capturados. Esse resultado é de maior importância pois foi obtido através da captura e análise de um tráfego real e em um ambiente onde deseja-se aplicar os modelos, assim como o software desenvolvido para tal.

Observou-se que durante os testes no desenvolvimento do software, o mesmo se comportou bem fidedigno quando a necessidade temporal da captura, isso é, a escolha pela utilização de *daemon*, além dos vários motivos já citados, também não interferiu na temporalização dos dados, pois esse software não influenciava nos tempos de captura, sendo assim, um determinado valor colhido era aplicado aos modelos de predição sem sofrer atrasos discretos perceptíveis, por tanto não interferindo nos resultados.

Conclui-se que os objetivos foram alcançados através da obtenção de uma série de métodos para uma predição de alta acuracidade, assim como a automatização da predição. Também esse processo não influencia em modo algum a obtenção dos resultados, e não

somente isso, mas como toda a predição é armazenada em um paradigma de banco de dados com afinidade a séries temporais. Isso possibilita que as mesmas predições sejam aplicadas a outros modelos e métodos que poderiam avaliar o comportamento da rede ou mesmo como dados para outras predições, sem perder o caráter temporal obtido no momento da coleta ou fornecido por algum dos modelos estudados e descrito no escopo desse trabalho.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Brockwell, P. e Davis, R. (1996) "Introduction to Time Series and Forecasting", Springer.

[2] Fox, J. (1997) "Applied Regression Analysis, Linear Models, and Related Methods", SAGE Publications, California/USA.

[3] Hanke J., Reitsch, A. e Wichern, D. (2001) "Business Forecasting", Prentice Hall.

[4] Makridakis, G. S., Wheelwright, S. C., Hyndman , R. J. (1997) "Forecasting: Methods and Applications", 3rd Edition, Wiley, USA.

[5] Matias Jr., R. ; Brasil, R. G. (2005) "Uma Solução para Análise de Tráfego em Redes Comutadas Baseada em Linux Bridging e Ntop", VIII Workshop de Software Livre/Fórum Internacional de Software Livre (WSL 2005), Brasil.

[6] Matias Jr., R., Schutz, F. (2007) "Avaliação de Modelos de Predição de Tráfego Utilizando um Analisador de Protocolos in-line", VIII Workshop de Software Livre (WSL 2007), Brasil.

[7] Morettin P. A, Clélia M. C. Toloi, "Análise de Séries Temporais", Blucher, pp. 186, 2ed, 2006.

[8] MORAIS, J.F, "Estatística Aplicada a Administração", http://jfmorais2009.vilabol.uol.com.br/ELE09_REG.pdf, Acessado em 15 dez 2009

[9] Sartoris .A, "Estatística e Introdução à Econometria", Saraiva, 2003.

[10] Trivedi, K. S. (2001) "Probability and Statistics with Reliability, Queueing, and Computer Science Applications", 2nd Edition, Wiley-Interscience, USA.

[11] Vahalia, U. (1996) "UNIX internals: the new frontiers" , Prentice Hall, USA

[12] Simsek, B. (2005) , "Signals", http://www.enderunix.org/docs/signals.pdf, Acessado em: 3 jul 2010

[13] IBM Corporation (2008) ,"POSIX Terminal Interactions", z/VM V5R2.0 CMS Application Development Guide SC24-6069-01.
Disponível em: http://publib.boulder.ibm.com/infocenter/zvm/v5r3/topic/com.ibm.zvm.v53.dmsa3/hcsd0b1020.htm , Acessado em 3 jul 2010.

[14] Disponível em: http://www.linfo.org/unix-like.html
Acessado em 04 jul 2010.

[15] Disponível em:
http://linux.about.com/library/glossary/bldef/bldef_fgprocess.htm
Acessado em 04 jul 2010 .

[16] Disponível em: http://www.gnu.org/software/libc/
Acessado em 04 jul 2010

[17] Ramakrishnan, R. e Gehrke, J. (2003) "DataBase Management Systems", 3rd Edition, McGraw-Hili, USA

[18] Disponível em: http://oss.oetiker.ch/rrdtool/tut/rrdtutorial.en.html ,
Acessado em 04 jul 2010 .

[19] Disponível em: http://unixtime.info , Acessado em 04 jul 2010 .

[20] Disponível em: http://oss.oetiker.ch/rrdtool/tut/rrd-beginners.en.html , Acessado em 04 jul 2010.

[21] Disponível em: http://www.pcmag.com/encyclopedia_term/0,2542,t=application+programming+interface&i=37856,00.asp , Acessado em 05 jul 2010.

[22] Disponível em: http://oss.oetiker.ch/rrdtool/, Acessado em 05 jul 2010.

[23] Disponível em: http://tools.ietf.org/html/rfc2131 , Acessado em 07 jul 2010.

[24] Disponível em: http://www.fromdual.com/operating-system-signals , Acessado em 07 jul 2010.

[25] Disponível em: http://www.kernel.org , Acessado em 07 jul 2010.

[26] Disponível em: http://www.cisco.com/en/US/docs/internetworking/technology/handbook/Internet-Protocols.html#wp4145 , Acessado em 12 jul 2010.

[27] Disponível em: www.linux.com , Acessado em 28 ago 2010.

[28] Disponível em: http://www.linuxfoundation.org/collaborate/workgroups/lsb , Acessado em 28 ago 2010.