Instituto de Matemática e Estatística

Universidade de São Paulo

Estudo e Desenvolvimento de Analisadores Estatísticos para Especificações de SLA

MAC 0499 - Trabalho de Formatura Supervisionado

Aluno: Gabriel Henrique Pugliese

Orientador: Manoel Marcilio Sanches

1 de fevereiro de 2010

Sumário

Parte I

Parte Objetiva

1 Introdução

No mundo em que as redes de computadores estão cada vez maiores e crescendo muito rápido com o passar do tempo, é necessário métodos que garantam a disponibilidade ou pelo menos ajude a controlar o que acontece delas. Isto é o que um Acordo de Serviço de Nível (SLA - *Service Level Agreement*) se propõe a fazer. O SLA é um acordo contratual entre duas ou mais entidades, na qual uma das partes se propõe a garantir para a parte que contrata:

- **Disponibilidade**;
- **Performance**;
- **Previsão/incidência de erros**.

Para a garantia de performance e/ou disponibilidade de serviços, a parte que é contratada geralmente se utiliza de métodos como redundância de servidores, tanto para a garantia de disponibilidade de programas como para a performance de máquinas - como uma duplicação de servidor que previne que haja uma parada no uso de uma máquina que está em uso, exemplo: servidor de *backup*. Também pode ser usado equipamentos para prevenção de queda de força, como geradores etc.

A parte que é a previsão/incidência de erros é, geralmente, feita através de programas. Tais programas podem ser monitores em tempo real ou monitores coletores de informações para geração de relatórios para análise futura.

1.1 Objetivo

O objetivo deste trabalho é desenvolver uma ferramenta que possa ser utilizada como um programa adicional aos meios de monitoração de uma rede, que faça a monitoração através de informações coletadas e gere relatórios com dados suficientemente claros para uma análise do andamento da rede.

2 Conceitos

Para melhor entender o funcionamento do programa desenvolvido, esta sessão irá descrever alguns conceitos de rede: os principais protocolos resumidamente e o funcionamento dos programas utilizados.

2.1 Tipos de nós de uma rede

Uma rede pode ser constituída de vários tipos de nós, os quais são diferenciados pela sua atuação dentro dela. Como são muitos, apenas os dividirei em 3 tipos: nós finais, intermediários e primários, separando por 2 grupos: os nós finais, os quais não interferem diretamente na rede - servidores, impressoras, câmeras etc - e os nós primários/intermediários, os quais são essenciais e podem afetar a rede diretamente caso aconteça algum problema com eles - *switches*, roteadores etc.

2.2 Protocolos de rede

Um protocolo de rede, em termos gerais, é um conjunto de regras que interconectam computadores pela rede para a troca de informações entre si. Essas regras são abstratamente separadas em um modelo com estilo de camadas chamado TCP/IP (*Internet Protocol Suite*). Esse modelo de camadas possui esse nome devido aos dois principais protocolos dentro dele que são o TCP (*Transmission Control Protocol*) e o IP (*Internet Protocol*).

O TCP/IP se divide em cinco camadas, nesta ordem:

- **Camada Física**;

- **Camada de Enlace**;
- **Camada de Internet**;
- **Camada de Transporte**;
- **Camada de Aplicação**.

Abaixo há uma breve explicação dos protocolos essenciais para o entendimento do funcionamento do programa.

2.2.1 Protocolo IP

O IP é um protocolo que se encontra na camada de internet e basicamente garante que os dados enviados de um computador se direcione a outro, ponto a ponto, pelo seus respectivos endereços.

2.2.2 Protocolo TCP

O TCP é um protocolo da camada de transporte e garante que os dados que são direcionados pelo IP sejam enviados. Ele atua entre o IP e um protocolo da camada de aplicação. O TCP necessita autorização para efetuar a conexão orientada, além de possuir correção de erros no envio dos dados, isto é, retransmite os dados que podem ser perdidos por algum motivo durante o processo de transmissão e também controla o congestionamento na hora da transmissão de dados. Isto faz com que o TCP seja um protocolo seguro para a transmissão de dados, porém mais custoso que outros, como por exemplo, o protocolo UDP.

2.2.3 Protocolo UDP

O UDP (*User Datagram Protocol*) é também um protocolo da camada de transporte, porém diferentemente do UDP não necessita confirmação de conexão para o seu destino, não possui controle de tráfego dos dados e não há correção de erros. Porém, estas faltas de qualidades fazem com que seja um protocolo de transporte rápido.

2.3 Protocolo SNMP

O protocolo SNMP (*Simple Network Managment Protocol*) se localiza na camada de aplicação e, como diz o nome, fora desenvolvido para a gerência de redes. Este é o protocolo com mais enfoque no projeto. Ele tem funcionamento juntamente ao UDP para obtenção de melhor desempenho entre as comunicações entre nós.

2.3.1 Arquitetura básica

O SNMP funciona, basicamente, da seguinte maneira: existe um nó da rede que se dá o nome de gerente, o qual pode requisitar informações(dados) e recebê-las do agente. O agente é o nó o qual o gerente deseja monitorar. A monitoração se constitui de informações variadas sobre o agente. Pode-se monitorar se um nó está acessível na rede e até mesmo se algum programa está rodando no sistema operacional daquele nó, se no caso for um nó servidor.

2.3.2 SNMP Walk

Dá-se o nome de SNMP Walk o processo de coleta de informações de um nó agente pelo gerente, onde este manda uma requisição para que envie dados específicos. Esses dados, do lado do nó agente fica armazenado em uma tabela chamada MIB (*Management Information Base*). São enviadas sequências de números que facilmente são achadas na tabela e o retornam informações em *strings* ou inteiros/hexadecimais, que são as informações necessárias para a monitoração do nó. O nó agente pode ser de qualquer tipo (primário, intermediário ou final).

2.3.3 SNMP Trap

De forma diferente do Walk, o gerente tem uma interação indireta com os seus nós quando a monitoração é feita por SNMP Traps, pois não são feitas requisições diretas aos nós monitorados e sim é uma passividade na monitoração. Isto é, não são os nós monitorados que enviam as informações ao gerente, mas sim os nós intermediários e primários - os roteadores - que fazem este trabalho.

Roteadores possuem interfaces onde os nós da rede são ligados fisicamente. Os roteadores que possuírem suporte ao SNMP, possuem uma tabela MIB com informações de cada nó e estas são enviadas a qualquer servidor gerente da rede. Isso foi desenvolvido para que se uma rede for muito grande, com diversos nós, o gerente sozinho não poderia requisitar informações de todos eles, ou se o fizesse, causaria um congestionamento ou atraso nas informações coletadas - o que realmente acontece. Assim, cada roteador é configurado

para enviar mensagens com as informações de suas interfaces ao invés de deixar esta trabalho aos nós.

No exemplo de *trap* abaixo, um roteador o envia para um servidor que aloca esses dados em um arquivo texto. Como é possível observar, esses dados aparecem demasiados embaralhados e de difícil leitura:

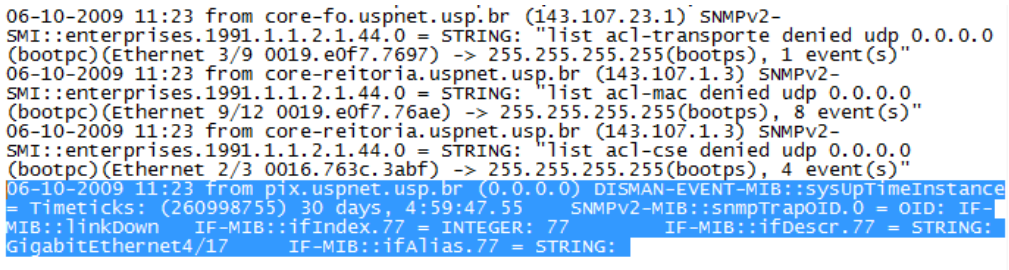

Figura 1: Trap com informação de status down destacado em azul

3 Arquitetura e funcionamento do programa

Nesta parte, será explicado o funcionamento do programa desenvolvido e também como foi utilizado o protocolo SNMP, da linguagem escolhida e como foram utilizados os programas de banco de dados, servidor *web* e para desenho de gráficos. O projeto foi desenvolvido e testado apenas para a plataforma Linux Debian embora a portabilidade para outros sistemas operacionais não seja de difícil acesso. Abaixo segue um esquema superficial do funcionamento:

Figura 2: Esquema resumido da arquitetura do programa

3.1 Subdivisão do código

O código é dividido em cinco arquivos em Perl dentro da pasta /trapd/bin. Foram criados dois módulos:

- **Constants.pm** - Possui constantes como usuários e senhas e caminhos

dos programas utilizados;

- **Status.pm** - É o núcleo do programa onde os principais métodos estão.

Os outros três arquivos são executáveis:

- **trapd.pl** - Este roda periodicamente, configurado no cron (agendado de tarefas do Linux) do Debian, para efetuar a coleta de informações de um arquivo com *traps*. Como os *traps* não são fácilmente legíveis, efetua a coleta, trata as informações e põe em uma tabela de um banco de dados;

- **webgui.cgi** - Interface inicial do usuário onde este escolhe as funcionalidades do programa;

- **st_report.cgi** - Após o usuário escolher as funcionalidades este arquivo apresenta os resultados: representação gráfica e tabela com informações pós-tratadas, ou seja, de um jeito legível.

3.2 Linguagem Perl

A linguagem Perl com orientação a objetos foi escolhida para o desenvolvimento do programa. Além da maior familiaridade com a mesma, apresenta vantagens para o tratamento das informações coletadas com fácil manipulação a expressões regulares, além de poder manipular com funções do banco de dados e criar páginas web utilizando CGI(*Common Gateway Interface*) - tecnologia para gerar páginas onde há interação do usuário por um navegador onde passa-se parâmetros para um programa em um servidor web - facilmente.

3.2.1 Pacotes do Perl utilizados

Foram utilizados os seguintes pacotes do Perl:

- **DBI (Perl's Database Interface)** - utilizado para a fácil manipulação com o banco de dados;
- **CGI** - funções que são utilizadas para a criação da interface web;
- **Socket** - é utilizado para realizar a conexão entre o programa e algum nó da rede;
- **Sys::Hostname** - possui funções que resolvem endereços de IP que são mandados pelos SNMP Traps, isto é, descobrem o *hostname* (nome de rede) de nós atráves do IP dos mesmos;
- **strict** - restringe construções no Perl que não são seguras.

3.3 SNMP Trap Daemon

Este programa é a chave dos *traps* SNMP. Este fica rodando constantemente no sistema esperando por novas informações de *traps* que são direcionadas pelos roteadores para o servidor onde está rodando este daemon e este fica aguardando mensagens na porta 162 UDP (Conexões ou envio de dados TCP/IP são sempre orientados, além de um indereço IP, também à uma porta). Quando uma mensagens é recebida ela é alocada em um arquivo texto para ser posteriormente tratada.

A configuração é para que mande mensagens de *trap* em apenas uma linha contínua e não haja quebras de linha para um mesmo *trap*, para facilitar a diferença entre eles. Sua inicialização é a seguinte:

```
[bash:~]\$ snmptrapd -F "%#02l-%#02m-%#04y %#02h:%#02j \
from %B (%A) %v\n" -Lf "ArquivoDeSaida"
```

A opção -F e o argumento pode ser traduzido por: uma expressão regular que gera mensagens de *trap* diferentes por linha, inicializando pela data do evento no formato DD-MM-AAAA seguido pelo horário no formato HH:MM. Após isso, mostra as informações técnicas do *trap*. O argumento de -Lf é o arquivo de saída do programa.

3.3.1 Tratamento do SNMP Trap

Os SNMP Traps podem trazer vários tipos de informações diferentes, como data e horário do envio da informação, número de identificação onde se encontra a informação na tabela MIB etc. Porém um *trap* contém apenas uma informação chave sobre cada nó. Sabendo disso, foi escolhido dois tipos de *traps* específicos, que trazem a informação de quando uma interface está com o estado *down* ou *up*, ou seja, esta interface não está mais apresentando alguma conexão com o nó no caso de *down* e quando alguma interface faz uma nova conexão com algum nó no caso de *up*.

Estes *traps* são tratados com uma expressão regular bem simples no código, colocando os dados em um vetor para inserção em uma tabela de um banco de dados.

3.4 Banco de Dados SQL

Para o armazenamento dos dados e consulta rápida foi utilizado um banco de dados MySQL. A estrutura criada é bem simples onde é criado um banco

de dados com nome **snmptrap** e possui uma tabela, nomeada **trap_status**, com os campos:

- **id** - contador que auto incrementa;
- **date** - data do *trap*;
- **time** - horário do *trap*;
- **router** - nome ou endereço IP do roteador que enviou o *trap*;
- **ip** - endereço IP da interface indicada pelo *trap* - é a interface gerenciadora do roteador, a que envia os dados;
- **interface** - nome da interface;
- **status** - estado da interface que varia de *down* e *up*.

Essa tabela é populada toda vez que é executado o arquivo **trapd.pl** pelo cron periodicamente e é lida toda vez que um usuário da web faz uma requisição de relatório novo.

3.5 Servidor e Interface Web

É utilizado o Apache2 como servidor *web* e precisa ser configurado para executar scripts na pasta onde o programa está instalado, pois é utilizado CGI para as operações.

A interface de interação com o usuário possui um campo onde pode escolher o roteador que enviou os *traps* . O segundo campo é a escolha do intervalo de tempo do relatório (máximo de trinta dias). O terceiro campo

é a escolha do intervalo de tempo, onde é possível escolher a criação de um gráfico diário, mensal ou anual. Após a submissão dessas variáveis, a interface do usuário torna-se estática, onde aparece apenas o relatório.

O relatório apresenta o número total de incidentes no intervalo de tempo escolhido, o gráfico de barras e uma tabela com as informações do *trap* tratado que está no banco de dados.

Abaixo mostra-se a interface de interação com o usuário e uma exemplo de tabela mostrada no relatório:

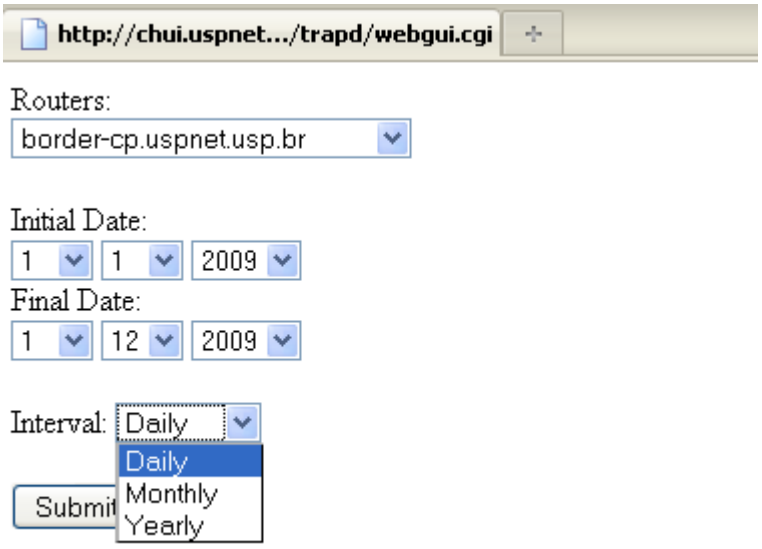

Figura 3: Menu de interação com o usuário

date	time	status	router	interface
2009-11-03	10:16:00	up	core-arusp.uspnet.usp.br	FastEthernet4/3
2009-11-05	08:36:00	down	core-arusp.uspnet.usp.br	FastEthernet3/23
2009-11-05	18:24:00	down	core-arusp.uspnet.usp.br	FastEthernet3/23
2009-11-05	18:24:00	up	core-arusp.uspnet.usp.br	FastEthernet3/23
2009-11-06	06:07:00	down	core-arusp.uspnet.usp.br	FastEthernet3/23
2009-11-06	06:08:00	up	core-arusp.uspnet.usp.br	FastEthernet3/23
2009-11-06	20:04:00	down	core-arusp.uspnet.usp.br	FastEthernet4/3
2009-11-09	11:29:00	down	core-arusp.uspnet.usp.br	FastEthernet3/4
2009-11-09	11:30:00	up	core-arusp.uspnet.usp.br	FastEthernet3/4
2009-11-09	11:40:00	down	core-arusp.uspnet.usp.br	FastEthernet3/4
2009-11-09	11:41:00	up	core-arusp.uspnet.usp.br	FastEthernet3/4
2009-11-09	14:44:00	down	core-arusp.uspnet.usp.br	FastEthernet3/4
2009-11-09	14:45:00	up	core-arusp.uspnet.usp.br	FastEthernet3/4
2009-11-09	17:50:00	down	core-arusp.uspnet.usp.br	FastEthernet3/6
2009-11-10	23:22:00	up	core-arusp.uspnet.usp.br	FastEthernet4/12
2009-11-11	00:35:00	up	core-arusp.uspnet.usp.br	FastEthernet4/6
2009-11-11	00:36:00	down	core-arusp.uspnet.usp.br	FastEthernet3/6
2009-11-11	00:36:00	down	core-arusp.uspnet.usp.br	FastEthernet3/9

Figura 4: Tabela do relatório contendo informações dos traps

3.6 Representação do gráfico

Para o desenho do gráfico do relatório é utilizado o Gnuplot. Ele é executado pelo **st_report.cgi** toda vez que um usuário faz uma requisição de relatório ao enviar as variáveis da interface web. Os gráficos possuem tamanho fixo e são do tipo de barras. O gráfico mostra o número absoluto de incidentes ocorridos por dia, mês ou ano.

O arquivo de estatísticas status.data é criado na pasta **/trapd/images** e a partir deste é gerada a imagem **status.png** com o gráfico feito pelo Gnuplot, onde é inserido no relatório juntamente com a tabela com os detalhes das interfaces e roteadores.

Abaixo alguns exemplos de gráficos traçados:

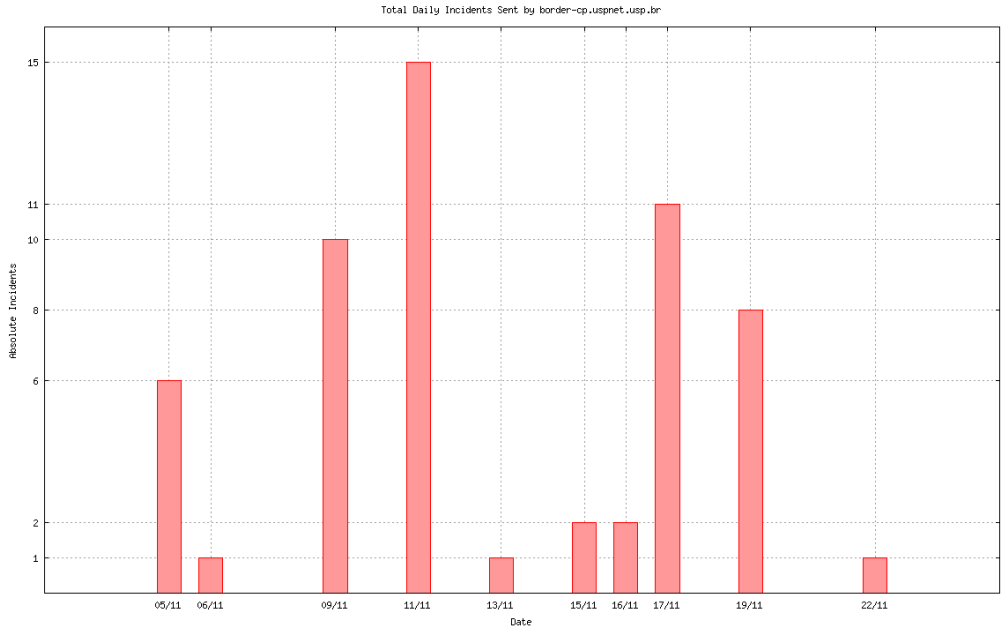

Figura 5: Incidentes em intervalo de tempo diário por 1 roteador

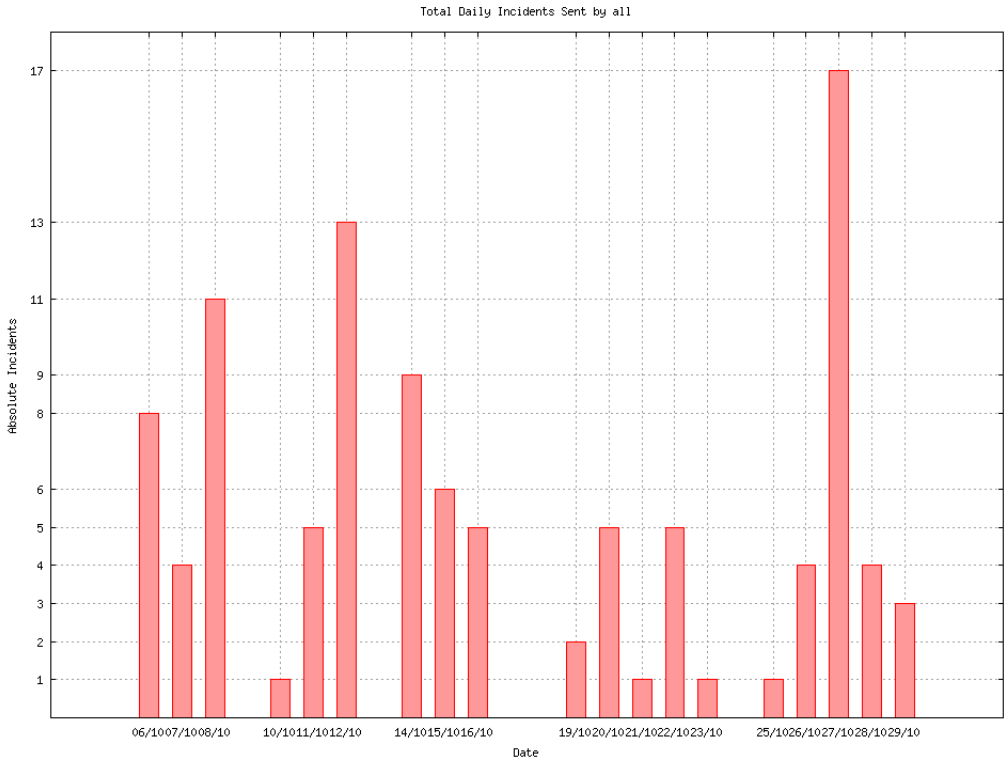

Figura 6: Incidentes em intervalo de tempo diário por todos os roteadores

Parte II

Parte Subjetiva

1 Desafios e Frustrações

O objetivo inicial do projeto era adicionar aos meios de monitoração já existentes à parte de Operações de Redes do Centro de Computação e Eletrônica da USP, o CCE. Além de já ser um desafio grande por si só, tive que adequar e modificar constantemente o programa para melhor atender às necessidades de quem utilizará o programa.

Quanto aos desafios do desenvolvimento, creio que a parte mais difícil foi aprender vários programas com funcionamentos diferentes. Estudei muito como deixar as inserções e consultas SQL mais rápidas. O problema principal foi escolher um programa de desenho do gráfico. Procurei por vários e tive que estudá-los para ver qual usar no programa. Entre alguns o que supriria melhor seria o Gnuplot.

Além disso, tive que estudar o funcionamento dos roteadores com relação aos *traps* enviados e também o funcionamento do snmptrapd do lado do servidor que recebe as mensagens.

Tive também que aprender como funciona a Orientação à Objetos do Perl, pois queria aprender algo novo em cima da linguagem que eu já mexia. Além de aprender a programar com ele, queria fazer de uma forma que deixasse o código elegante e de fácil leitura.

A maior frustração foi que não pude implementar mais funcionalidades ao programa ainda. Perdi bastante tempo na escolha dos programas e no aprendizado básico deles. Também tive que mexer no tipo de representação gráfica que fiz algumas vezes, para atender melhor aos usuários do programa. Também houveram muitas falhas que precisaram ser corrigidas e tomaram

grande tempo.

2 Disciplinas relevantes para o trabalho

Como estou fazendo o TCC e não estarei me formando, creio que alguma outra disciplina poderia estar na lista abaixo se eu tivesse com os créditos em dia. Porém estas me ajudaram bastante no desenvolvimento do projeto e são as que consideram principais:

- MAC110 - Introdução à Computação

Foi a disciplina que pôde dar uma base para a programação orientada à objetos, onde foi utilizada a linguagem Java.

- MAC122 - Princípio de Desenvolvimento de Algoritmos

Esta disciplina foi a qual mais ajudou para o desenvolvimento dos algoritmos utilizados no projeto e a qual eu considero uma das principais que fixam a base do curso.

- MAC323 - Estruturas de dados

A disciplina ajudou para o entendimento de estruturas de dados e algoritmos mais complexos dos que são vistos em MAC122.

- MAC0211/MAC242 - Laboratório de Programação I e II

Além de ajudarem na questão de manipulação com projetos longos e como se organizar para desenvolvê-los, também foi quando aprendi a linguagem Perl e expressões regulares.

- MAC0426 - Sistemas de Bancos de Dados

Esta foi essencial para a manipulação com o banco de dados e as consultas SQL, mesmo o banco de dados não sendo muito grande no projeto ajudou a criá-lo do melhor jeito.

- MAC0448 - Programação para Redes de Computadores e PCS0210 - Redes de Computadores

São as matérias de mais importância, pois fixou os conceitos de redes de computadores envolvidos em todo o projeto.

3 Considerações Finais

O projeto será continuado no CCE por mim para aplicar melhores funcionalidades que e corrigir algumas falhas que venham a existir ainda. Também quero disponibilizar o projeto para a comunidade *open-source* de redes, pois foram programas desse destino que me ajudaram a aprender bastante sobre a área de redes.

O programa atualmente está em fase de testes, porém se mostra fiel às informações sobre as quedas acontecidas na rede da USP.

Também pretendo atuar nesta área de trabalho no futuro, após a formação no bacharelado. Talvez até tentar um mestrado relacionado a redes.