Resumo

Atualmente, observa-se um número cada vez maior e diversificado de equipamentos interconectados nas redes das organizações, dentre eles podemos citar: estações de trabalho, impressoras, modems e roteadores. Ao mesmo tempo, os administradores e projetistas das redes necessitam de informações atualizadas sobre cada um destes elementos. Estas informações podem ser divididas em dois grupos: informações de gerenciamento e informações de desempenho. As informações de gerenciamento são importantes para se verificar o estado atual do equipamento em termos de, por exemplo: nome, descrições de software/hardware, contato do responsável. As informações de desempenho refletem o comportamento dos serviços prestados pelo equipamento durante um determinado período, baseando-se em: taxa de entrada e saída de pacotes IP, consumo de memória dos processos em execução, utilização de CPU, etc. Para realizar tais tarefas, os profissionais da área de redes são auxiliados por ferramentas de gerenciamento, majoritariamente proprietárias. Programas proprietários são pouco flexíveis, requerem alto poder computacional do hardware e têm custo elevado. Para solucionar estas limitações, desenvolvemos uma ferramenta de código aberto para auxiliar o gerenciamento e análise de desempenho. Como estudo de caso, iremos monitorar um servidor que provê conteúdos de áudio e vídeo para telefones celulares em forma de streaming que chamamos de servidor de streaming. Baseando-se nos resultados obtidos pela ferramenta, observamos que a mesma é capaz de disponibilizar informações de gerenciamento em tempo real. A análise dos gráficos gerados indicou que, apesar de o servidor estar bem dimensionado para atender aos clientes internos, o projetista da rede precisa considerar a largura de banda que deve ser dedicada ao serviço de streaming para clientes externos na Internet, já que a taxa de transferência máxima observada em rede local atingiu valores significantes.

Abstract

The increasing number and complexity of interconnected devices is clearly noticeable in today's networks. Among them, we can mention desktops, printers, modems and routers. At the same time, network professionals need real time information regarding each of these devices. The type of information required can be divided into two groups: management and performance. Network management is used to monitor the actual state of the devices by gathering information such as name, hardware and software description, contact of the person responsible for that equipment. Performance information reflects the service behavior provided by the device based on some statistics, like inbound/outbound packet flow, memory consumption, CPU utilization, etc. To perform such tasks, network administrators use technologies known as management tools. The majority of these tools are proprietary. These kind of softwares are usually not flexible enough, require high hardware computing power and are expensive. To overcome these problems, we have developed an open source tool which helps the network administrator in management and performance analysis. As a study case, we chose to manage a device that provides multimedia content to mobile clients, known as streaming server. As the result of using this tool, we were able to receive real time management information from the streaming server. Although the performance analysis showed that the streaming server is reliable to serve the internal clients, we noticed that the network engineer should be concerned on the network bandwidth allocated to the server if multimedia content is going to be provided to external clients using the Internet since, the maximum transfer rate was observed to achieve considerable values in local network.

Sumário

Índice de Figuras

Índice de Tabelas

Tabela de Símbolos e Siglas

SNMP – *Simple Network Management Protocol*
IETF – *Internet Engineering Task Force*
CPU – *Central Processing Unit*
RFC – *Request for Comments*
SNMPv2 – *Simple Network Management Protocol version 2*
SNMPv3 – *Simple Network Management Protocol version 3*
MIB – *Management Information Base*
OSI – *Open Systems Interconnection*
ASN.1 – *Abstract Sintax Notation One*
ISO – *International Organization for Standardization*
ORG – *Organization*
DOD – *Department of Defense*
MGMT - *Management*
MIB-II – *Management Information Base II*
IP – *Internet Protocol*
ICMP – *Internet Control Message Protocol*
TCP – *Transmission Control Protocol*
UDP – *User Datagram Protocol*
EGP – *Exterior Gateway Protocol*
RAM – *Random Access Memory*
PDU – *Protocol Data Unit*
RRDTOOL – *Round Robin Database Tool*
HTML – *Hipertext Markup Language*
PHP – *Hipertext Preprocessor*
W-CDMA – *Wide-Band Code-Division Multiple Access*
SIM – *Subscriber Identify Module*
RPN – *Reverse Polish Notation*
SMS – *Short Message Service*

Agradecimentos

- Agradeço aos meus pais, Alcides Bezerra e Maria de Conceição, pelo incentivo não apenas na minha graduação em Engenharia da Computação, mas principalmente por me ensinar, baseados em si mesmos, que o sucesso é uma conseqüência de muito esforço com os estudos.
- Ao meu Orientador, Professor Renato Mariz de Moraes, pelo esforço dedicado ao desenvolvimento de um trabalho de conclusão de curso que pode gerar bons frutos no futuro, assim como, pela compreensão nos momentos difíceis.
- À minha namorada, Valma Gondim, pela força e incentivo para a realização deste trabalho, além do auxílio na formatação do mesmo.
- Aos eternos amigos da faculdade: Rodrigo Gomes, Rodrigo Cursino, Rafael Bandeira, Gabriel da Luz, Reinaldo Melo, Tiago Lima, Fernando Antônio, Diogo Costa, Adilson Arcoverde, Hilton Lupus, Cláudio Cavalcanti, Rodrigo Brayner, Juliana Lima, José Guilherme, Gabriel Alves, Allan Bruno, Lívia Brito, Laureano Montarroyos.
- Aos professores de graduação em Engenharia da Computação que me ensinaram muito do que sei hoje como profissional e ser humano. Em especial aos professores Fernando Buarque, Adriano Lorena, Abel Guilhermino, Carlos Alexandre e Ricardo Massa.

Capítulo 1

Introdução

Atualmente, a diversidade e a complexidade dos equipamentos interconectados nas organizações tornam difícil ao administrador e projetista da rede executar as tarefas de gerenciamento e análise de desempenho dos equipamentos [10]. O gerenciamento visa saber se os equipamentos - roteadores, servidores e estações de trabalho - estão funcionando, enquanto que a análise de desempenho destina-se a verificar se os mesmos estão trabalhando de acordo com os requisitos do projeto.

Antigamente, o gerenciamento de redes era feito de forma re-ativa, ou seja, se o tempo de resposta a uma requisição para um determinado equipamento se tornasse excessivamente grande, o administrador da rede enviaria um comando `ping` [13] para o equipamento e o problema poderia ser localizado analisando o pacote de resposta ao comando. Esta solução era viável, uma vez que a quantidade de equipamentos interconectados era pequena.

Quando a Internet se tornou mundialmente difundida, com diversos *backbones* e provedores de acesso, essa solução deixou de ser adequada havendo a necessidade de criação de melhores ferramentas de gerenciamento de redes. A criação do protocolo SNMP (*Simple Network Management Protocol*) [10], em 1990, pela IETF (*Internet Engineering Task Force*) [15], permitiu que o gerenciamento de redes se tornasse pró-ativo, possibilitando que o administrador da rede fosse notificado assim que o servidor começasse a apresentar comportamentos que pudessem induzi-lo à falhas, tais como, cargas elevadas de processamento de memória e CPU (*Central Processing Unit*), e taxas elevadas de entrada e saída de pacotes.

Com a disseminação do SNMP, começaram a surgir mais técnicas para monitoramento e os fabricantes dos equipamentos de rede sentiram a necessidade de criar novas tecnologias de gerenciamento. Iniciou-se o lançamento dos softwares proprietários específicos para este propósito.

Os softwares proprietários são, em sua maioria, pouco flexíveis, requerem alto poder computacional de hardware e têm custo elevado. Considerando a diversidade de fabricantes dos equipamentos interconectados nas redes, a utilização de softwares desta natureza causou problemas aos administradores.

Então começaram a surgir as comunidades de softwares livres que defendiam a disponibilização do código fonte dos programas, tornando-os mais flexíveis e principalmente mais baratos (muitos deles por nenhum custo). Com o advento do conjunto de protocolos TCP/IP [13] e devido à liberdade de criação e modificação do código fonte dos programas, a filosofia do software livre se tornou adequada para o desenvolvimento de aplicações servidoras de rede, tais como, servidores de aplicações Web, de Email, de DNS (*Domain Name Service*) [13] e Gerenciamento.

Seguindo a temática de software livre, o nosso trabalho visa desenvolver uma aplicação de gerenciamento utilizando ferramentas sob a licença GPL (*General Public License*) [5] ou Licença Pública Geral, dentre elas vamos utilizar o sistema operacional Linux RedHat [16] e a ferramenta RRDTool [11].

A ferramenta de gerenciamento consiste de uma aplicação na Web, desenvolvida em HTML *(Hipertext Markup Language)* [2] e PHP *(Hipertext Preprocessor)* [14], que apresenta informações a respeito de um conjunto de variáveis que serão utilizadas para representar o comportamento de um servidor de streaming em dois níveis. O primeiro, é o nível de monitoramento, onde o administrador deve ser capaz de verificar se os equipamentos de sua rede estão funcionando ou não, como também, coletar informações descritivas destes nós da rede. O segundo, é o nível de análise de desempenho, onde o administrador deve ser capaz de julgar o desempenho de seus servidores, tendo como base, gráficos apresentando as estatísticas de diversas variáveis, tais como, consumo de memória, tráfego de entrada e saída de pacotes, utilização de CPU, etc.

Neste trabalho utilizamos duas ferramentas principais: SNMP e RRDTool.

O SNMP faz parte da pilha de protocolos TCP/IP [13]. Seguindo o modelo de camadas, o SNMP está na camada de aplicação, sendo utilizado, em nosso trabalho, para coletar dados do servidor utilizando a rede local.

O RRDTool é a ferramenta utilizada para armazenar e exibir os dados coletados.

A Figura 1 apresenta um diagrama simplificado da ferramenta que foi desenvolvida. Na primeira etapa, foram coletados os dados das variáveis de gerenciamento e desempenho do servidor em estudo, utilizando o SNMP. O servidor selecionado para monitoramento é chamado de servidor de streaming [12], uma vez que a principal função do mesmo é prover conteúdos de áudio e vídeo sob demanda. Na segunda fase, estes dados foram armazenados utilizando o RRDTool. Por último, foram gerados os gráficos para exibição da informação ao administrador da rede.

Figura 1. Diagrama simplificado da ferramenta de monitoramento.

As principais contribuições que oferecemos no trabalho são:

- Uma nova abordagem para o desenvolvimento de ferramentas de gerenciamento de redes utilizando SNMP para coleta e RRDTool para armazenamento e apresentação de dados.
- Um método para um método para validação da função para cálculo da média do RRDTool.
- Um método para cálculo da variância utilizando a notação RPN [11].

Capítulo 2

SNMP(*Simple Network Management Protocol*)

O modelo SNMP de uma rede gerenciada é constituído por quatro componentes que seguem:

1. Nós gerenciados;
2. Estações de gerenciamento;
3. Informações de gerenciamento;
4. Um protocolo de gerenciamento.

Os nós gerenciados são as estações de trabalho, roteadores, modems, impressoras ou qualquer outro dispositivo que seja capaz de comunicar informações de seu estado atual para o mundo externo. Em nosso caso, o nó gerenciado é o servidor de Streaming, onde deve estar sendo executado um processo de gerenciamento SNMP chamado de agente. O agente possui um banco de dados local contendo diversas variáveis que descrevem seu estado atual.

O gerenciamento propriamente dito é realizado na estação de gerenciamento, de onde são enviadas requisições a respeito do estado atual das variáveis dos nós gerenciados e as respostas são obtidas. A ferramenta proposta neste trabalho é uma estação de gerenciamento capaz de solicitar dados para o agente em execução no servidor de Streaming e disponibilizar a informação recebida de forma gráfica.

O modelo do protocolo SNMP pode ser entendido como um ambiente cliente-servidor conforme exibido na Figura 2. Neste ambiente, o servidor é responsável por enviar solicitações aos clientes e os mesmos respondem às requisições. Na literatura do SNMP, chamamos o servidor de *estação de gerenciamento* e os clientes de *agentes SNMP*.

A ferramenta que desenvolvemos é uma estação de gerenciamento responsável por solicitar os dados aos agentes SNMP.

Os agentes SNMP são responsáveis por responder às requisições do servidor. Através de um conjunto de operações que permite a leitura do estado atual de diversas variáveis, também chamadas de objetos SNMP, dos equipamentos monitorados, o servidor poderá coletar os dados necessários.

No entanto, podem acontecer eventos não esperados nos nós de gerenciamento, tais como, falhas nas conexões de rede, reinicializações não planejadas dos nós, linhas congestionadas e assim por diante. Quando um agente percebe que algum destes eventos significativos ocorreu, a estação de gerenciamento é automaticamente notificada por ele através de mensagens assíncronas chamadas historicamente de Trap [10] conforme exibido na Figura 2.

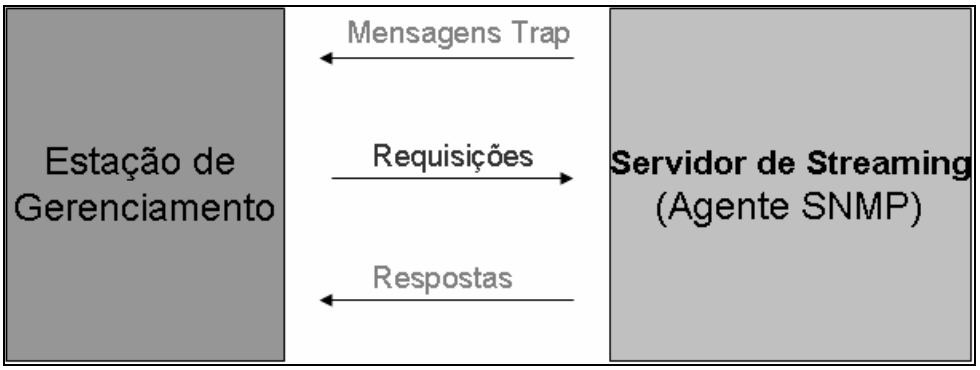

Figura 2. Modelo Cliente-Servidor do protocolo SNMP

Para permitir que uma estação de gerenciamento se comunique com a diversidade de equipamentos interconectados, a informação das variáveis mantidas por todos estes dispositivos deve ser rigorosamente especificada. No SNMP, essas variáveis são chamadas de objetos. O conjunto formado por todos os objetos permitidos para gerenciamento em um nó é determinado por uma estrutura de dados denominada MIB (*Management Information Base*) [13].

2.1 MIB (*Management Information Base*)

A identificação dos objetos SNMP dos nós gerenciados é definida pela estrutura de dados chamada MIB, seguindo uma sintaxe definida pela OSI (*Open Systems Interconnection*) chamada de ASN.1 (*Abstract Sintax Notation One*) [13]. O mecanismo utilizado para identificar unicamente os objetos é realizado através de uma árvore de padrões, onde cada um dos objetos é inserido em um local único da árvore. A parte da árvore que inclui a MIB do SNMP é exibida na Figura 3.

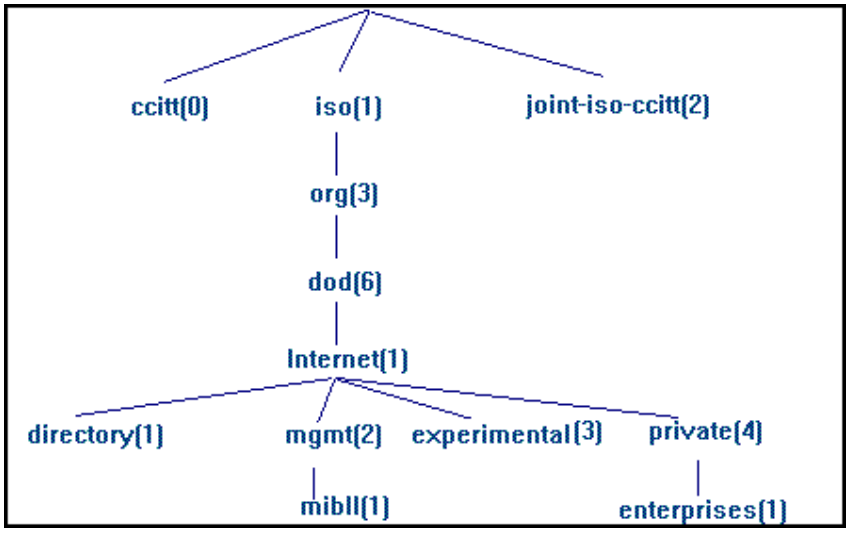

Figura 3. Árvore de atribuição de nomes dos objetos da ASN.1 [13]

Conforme exibido na figura 3, todos os arcos possuem tanto um rótulo como um número. Portanto, os objetos podem ser identificados por uma lista de rótulos, uma lista de números ou até mesmo uma lista mista como segue:

a. iso(1).org(3).dod(6).Internet(1).mgmt(2).mibII(1)
b. 1.3.6.1.2.1
c. Internet(1).2.1

Dentre as diversas estruturas MIB disponíveis no protocolo SNMP, serão utilizadas as seguintes em nosso trabalho:

1. MIB-II: Implementada por padrão por todos os agentes SNMP, esta tabela define variáveis de gerenciamento do sistema, tais como: nome do sistema,

contato do responsável pelo sistema, descrição do sistema. Também define informações de desempenho do sistema, tais como: número de pacotes enviados e recebidos, taxa de datagramas recebidos com erros.

Na estrutura MIB-II, os objetos são agrupados em nove categorias resumidas na Tabela 1.

Tabela 1. Descrição das categorias da MIB-II

Grupo	Descrição
System	Nome, local e descrição do sistema
Interfaces	Interfaces de rede e seus tráfegos
IP	Estatísticas de pacotes IP
ICMP	Estatísticas das mensagens ICMP
TCP	Estatísticas de tráfego TCP
UDP	Estatísticas de tráfego UDP

Em nosso trabalho estamos utilizando os objetos da categoria System para coleta de informações de gerenciamento do servidor de Streaming, tais como: descrição e local do sistema, contato do responsável. Assim como, os objetos das categorias Interfaces, IP, TCP e UDP para coleta de informações de desempenho, tais como: fluxo de entrada/saída de pacotes IP, tráfego de entrada/saída em bps(bits por segundo), etc.

2. HOST-RESOURCES-MIB: Define as variáveis para gerenciamento dos recursos do equipamento monitorado. Informações como a taxa de consumo de memória, utilização de CPU e espaço em disco serão utilizadas no projeto.

Na estrutura HOST-RESOURCES-MIB, os objetos são todos agrupados em uma única categoria chamada Host.

Em nosso trabalho estamos utilizando apenas dois objetos desta categoria: `hrSWRunPerfMem` e `hrSWRunPerfCPU`. Estes objetos retornam o consumo de memória RAM e o processamento de CPU, respectivamente, do nó gerenciado.

2.2 Modos de Operação do SNMP

O modo como o SNMP opera é normalmente aquele em que a estação de gerenciamento, utilizando como base a sintaxe ASN.1, envia solicitações a um agente requisitando informações sobre os diversos objetos dos nós gerenciados. O agente simplesmente responde a requisição com as informações solicitadas.

As mensagens trocadas entre agentes e estações de gerenciamento são chamadas de PDU (*Protocol Data Unit*). Existe uma mensagem PDU padrão para cada um dos seguintes modos de operação do SNMP que estamos utilizando no trabalho:

- Modo `get`: Requisições `get` são enviadas pela estação de gerenciamento para o agente. O agente recebe, processa a requisição e transmite uma mensagem `get-response` para a estação de gerenciamento. Por exemplo, em nosso trabalho, a ferramenta Net-SNMP do Linux [6] disponibiliza mecanismos de coleta de informações para os agentes. O comando abaixo é utilizado para consultar o número de pacotes IP de entrada no endereço IP do servidor de Streaming: 172.27.4.195.

```
root@manager> snmpget –v 1 –c public 172.27.4.195 IP-
                    MIB::ipInDelivers.0
```

- Modo `get-next`: Requisições `get-next` também são enviadas pela estação de gerenciamento para o agente, só que desta vez, um grupo de objetos SNMP são solicitados. O agente percorre a árvore da MIB do nó gerenciado e retorna os valores encontrados em cada arco percorrido. Por exemplo, em nosso trabalho, o comando abaixo é utilizado para consultar o consumo de memória de todos os processos em execução no endereço IP do servidor de Streaming.

```
root@manager> snmpwalk –v 1 –c public 172.27.4.195
                              hrSWRunPerfMem
```

- Modo `trap`: É a forma com a qual os agentes reportam às estações de gerenciamento que algum evento inesperado, como por exemplo, parada das interfaces de rede, parada de serviços, sobrecarga de discos rígidos, ocorreu no equipamento. Em nosso trabalho, o agente SNMP no servidor de streaming foi configurado para reportar mensagens `trap` para a estação de gerenciamento.

Capítulo 3

RRDTOOL (*Round Robin Database Tool*)

O RRDTool é uma ferramenta de código aberto escrita por Tobias Oetiker [11] sobre licença GNU GPL [5] que gerencia um banco de dados do tipo Round Robin [11]. Sua principal função é armazenar séries de dados temporais. Por esse motivo, esta ferramenta é amplamente utilizada para armazenar dados como utilização de banda em uma rede, fluxo de entrada/saída de pacotes, taxa de requisições a um servidor, etc. Round Robin é uma técnica que trabalha com uma quantidade fixa de dados e um ponteiro para o elemento atual conforme exibido na Figura 4, onde *t0* até *t5* são os elementos do banco dispostos em intervalos fixos no tempo. O elemento atual é lido ou escrito, e após um intervalo predefinido de tempo, o ponteiro é movido para o próximo elemento. Assim, na criação da base de dados, é especificado o tamanho fixo da mesma que não sofrerá alteração durante seu tempo de vida. Ou seja, os dados são inseridos nas posições livres e quando não houver mais espaço, os dados mais antigos são sobrescritos, numa espécie de fila circular. Os dados são armazenados neste tipo de banco para que as leituras sejam regulares ao longo do tempo.

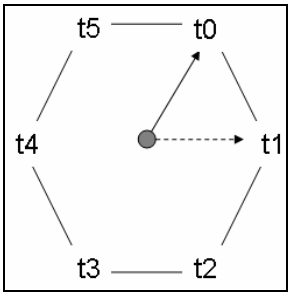

Figura 4. Técnica Round Robin

3.1 Características

As principais características do RRDTool que o classificam como uma ferramenta ideal para coleta e apresentação de dados em intervalos fixos de tempo são:

Aquisição de dados. A ferramenta RRDTool pode ser utilizada para a coleta e armazenamento de qualquer tipo de dados que se apresentam em séries temporais, ou seja, é um mecanismo capaz de capturar o valor de uma variável em diversos intervalos de tempo. Os dados capturados são automaticamente interpolados com os dados anteriores utilizando funções de consolidação, onde um valor resultante é armazenado.

Funções de Consolidação. Se quisermos obter a média do consumo de memória de um equipamento nas últimas cinco horas, podemos configurar o RRDTool para armazenar o consumo de memória em intervalos fixos de tempo, de cinco em cinco minutos, por exemplo. Para que possamos obter esta média, devemos configurar a ferramenta para aplicar uma função de consolidação (média, valor total, valor mínimo, valor máximo, último valor) após cinco horas e este valor consolidado será armazenado em um RRA (*Round Robin Arquive*) [11]. Podemos definir diversas funções de consolidação em um único banco que serão mantidos a medida em que novos dados são armazenados.

RRA (*Round Robin Arquives*). Os valores dos elementos com as mesmas configurações de consolidação são armazenados em um mesmo RRA. Esta é uma forma eficiente de armazenar os valores, uma vez que sabemos ser fixo o tamanho de armazenamento. Por exemplo, se quisermos armazenar 1000 (mil) valores em um intervalo de 5 minutos, a ferramenta RRDTool irá alocar espaço para os mil

valores mais uma área para cabeçalho. Este cabeçalho armazena o ponteiro que contém o endereço do último valor que foi escrito no banco. Os novos valores são então armazenados através do mecanismo de Round Robin. A utilização de RRAs garante que o banco de dados não cresça ao longo do tempo e que dados antigos são automaticamente eliminados. A utilização de funções de consolidação nos permite armazenar os dados que realmente interessam, tais como: o número máximo de bits por segundo de entrada em um roteador, a média de utilização de CPU de um servidor de streaming, número máximo de requisições a um servidor Web, etc.

Unknown Data (dados desconhecidos). Como mencionado anteriormente, o RRDTool armazena dados em intervalos fixos de tempo. Pode acontecer de não haver informação disponível no momento em que os dados devem ser inseridos no banco. Neste caso, a aquisição de dados não é possível. Comumente, estes problemas ocorrem por queda momentânea da rede. O RRDTool trata esses casos inserindo um valor `unknown` no banco de dados. Na consolidação de um conjunto de dados, a quantidade de valores `unknown` é calculada e quando um novo valor consolidado está pronto para ser inserido no banco RRA, é verificado se a porcentagem destes valores `unknown` está acima de um limite configurável. Se não estiver, um valor `unknown` é armazenado. Esta característica do RRDTool assume que é melhor armazenar um valor desconhecido do que um valor 0 (zero) ou qualquer outro valor válido no banco, uma vez que um valor válido poderia alterar as métricas no momento da consolidação dos dados.

Gráficos. O RRDTool permite a geração de relatórios tanto de forma numérica como gráfica tendo como base os dados armazenados em 1 (um) ou vários bancos RRD. Um dos nosso objetivos é apresentar os dados sobre todas as variáveis que iremos monitorar utilizando esta ferramenta gráfica. Através de scripts shell, podemos definir tamanho, cor e legendas dos gráficos. Também, esta função nos permite redefinir os valores das entradas de dados, para que assim possamos transformar bits em bytes, por exemplo, ou até realizar operações mais específicas como calcular a variância dos dados de entrada.

3.2 Funções do RRDTool

Dentre as diversas funções disponibilizadas pela ferramenta RRDTool, estamos utilizando algumas em nosso trabalho, apresentadas abaixo.

3.2.1 Create

É a função de criação de novos banco de dados RRD (*Round Robin Database*) [11]. Devemos utilizar a sintaxe abaixo para criação dos arquivos RRD:

```
rrdtool create nome_do_arquivo
[--start-time tempo_inicial]
[--step intervalo]
[DS: nome-ds:DST:heartbeat:min:max]
[RRA:CF:xfiles:step:rows]
```

Para o melhor entendimento das opções acima, apresenta-se abaixo o código para criação do banco RRD, utilizado no presente trabalho, para armazenar os valores da memória consumida pelo servidor de streaming:

```
1 #! /bin/sh
2
3 $tempo_atual = $(date +%s)
4
5 ./rrdtool create memoria.rrd --start $tempo_atual –step 60
6 DS:mem_tot:GAUGE:120:U:U RRA:AVERAGE:0.5:1:1440
```

Na linha 3, a variável `$tempo_atual` recebe a hora atual da estação de gerenciamento em segundos. O RRDTool utiliza o *Unix Epoch* [6] como referencial no tempo que representa o número de segundos transcorridos desde a meia-noite do primeiro dia do mês de janeiro de 1970.

Na linha 6, definimos a fontes de dados chamada `DS` (*Data Source*). Podemos definir várias fontes de dados em um mesmo banco RRD. Por exemplo, uma fonte de dados para armazenar o tráfego de entrada e outra para armazenar o tráfego de saída no servidor. Utilizamos as seguintes opções para definir a fonte de dados do banco `memoria.rrd`:

1) `mem_tot`: Nome da fonte de dados.

2) GAUGE: Tipo da fonte de dados. Os tipo disponíveis são: COUNTER, DERIVE, ABSOLUTE, GAUGE.

- O tipo COUNTER irá armazenar a taxa de alteração do valor ao longo do intervalo especificado em step, assumindo que este valor está sempre crescendo e o valor anterior é maior do que zero. Os contadores de tráfego dos roteadores são um bom exemplo de utilização do tipo COUNTER.
- O tipo DERIVE é semelhante ao COUNTER, mas este também permite a utilização de valores negativos. Se desejarmos obter a taxa de alteração no espaço livre do disco rígido do servidor, podemos utilizar o tipo DERIVE.
- O tipo ABSOLUTE também armazena a taxa de alteração, mas sempre considera que o valor anterior é zero. Ou seja, a diferença entre o valor atual e o valor anterior é sempre igual ao valor atual. Resumindo, o banco armazena apenas o valor atual divido pelo valor especificado em step.
- O tipo GAUGE apenas salva o valor em si próprio, sem realizar nenhum processamento. O consumo de memória é um exemplo típico de utilização deste tipo de fonte de dados.

A Tabela 2 apresenta as diferenças entre os tipos de fontes de dados apresentados acima considerando um step de 60 segundos. A primeira linha apresenta exemplos de valores coletados via SNMP. As três últimas linhas apresentam os valores que são efetivamente armazenados no banco.

Tabela 2. Diferenças entre os tipos de fontes de dados.

Valores	60	120	180	240
COUNTER	1	1	1	1
DERIVE	1	1	1	1
ABSOLUTE	1	2	3	4
GAUGE	60	120	180	240

3) 120: É conhecido como *heartbeat* (batida de coração). Em nosso exemplo, este argumento é utilizando para informar ao RRDTool que se o banco não receber dados no intervalo definido em step, 60 segundos, o mesmo deve aguardar mais 60

segundos (um total de 120 segundos). Se nenhum dado for recebido durante esse tempo, um valor unknown será armazenado.

4) U:U : Estes dois parâmetros são utilizados para limitar os valores máximo e mínimo, respectivamente, que são permitidos para armazenamento, de forma que qualquer valor recebido que esteja fora destes limites será armazenado como unknown. Utilizamos a opção "U" para indicar que qualquer valor pode ser armazenado no banco.

A última opção da linha 6 define um RRA (*Round Robin Arquive*). Estes arquivos contém os valores ou estatísticas de cada uma das *Data Sources* (fontes de dados) definidas. Quando novos dados devem ser inseridos em um banco RRD, o RRDTool define um espaço livre no banco para armazenamento de acordo com o valor step e cada valor inserido é chamado de PDP (*Primary Data Point*). Utilizamos as seguintes opções para definir o arquivo RRA do banco memoria.rrd:

i) AVERAGE: É a função de Consolidação. As funções de consolidação podem ser AVERAGE (Média), MINIMUM (Mínimo), MAXIMUM (Máximo) e LAST (Último). Estas funções são utilizadas para consolidar os PDPs formando os CDPs (*Consolidated Data Points*).

ii) 0.5: É conhecido como xfiles e define qual parte do intervalo de consolidação pode ser considerado como unknown. Limitado entre 0 (zero) e 1 (um), este argumento é a taxa de valores PDPs unknown permitidos em relação ao número total de PDPs no intervalo.

iii) 1: É conhecido como steps e define quantos dos PDPs podem ser utilizados no cálculo dos CDPs.

iv) 1440: define quantos CDPs podem ser armazenados no RRA.

Resumindo, em nosso código de criação do banco RRD, estamos especificando que o RRA deve utilizar 1 (um) valor PDP para formar um CDP. Ou seja, estamos armazenando os valores inseridos no banco RRD a cada intervalo de 60 segundos. Também, estamos definindo que 1440 valores de CDP devem ser armazenados no RRA. Como cada PDP ocorre em 60 segundos e estamos

armazenando 1440 CPDs, reservamos espaço suficiente para 1 (um) dia de monitoramento, conforme o cálculo abaixo:

```
1 PDP =  60 segundos
1 PDP = 1 CPD
1440 CDPs = 1440 min = 24hs
```

3.2.2 Update

É a função de armazenamento de dados no banco RRD. Devemos utilizar a sintaxe abaixo para armazenar dados nos bancos:

```
rrdtool update nome_do_arquivo
[tempo_1]:[valor_1]
[tempo_2]:[valor_2]
[tempo_n]:[valor_n]
```

Para melhor entendimento da sintaxe acima, podemos exibir um trecho do código utilizado para armazenamento dos dados que representa o consumo de memória pelo servidor de streaming:

```
1 #! /bin/sh
2
3 $tempo_atual = $(date +%s)
4
5 ./rrdtool update memoria_stream.rrd $tempo_atual:$memoria
```

Na linha 5, `memoria_stream.rrd` é o nome do arquivo RRD. O próximo parâmetro define os valores de consumo de memória, em `KBytes`, contidos na variável `$memoria`, que devem ser armazenados nos espaços disponíveis do banco determinados pela variável `$tempo_atual`.

3.2.3 Fetch

É a função que consulta dados armazenados no banco RRD em um intervalo de tempo. Devemos utilizar a sintaxe abaixo para consultar dados nos bancos:

```
rrdtool fetch nome_do_arquivo
[Funcao_Consolidacao]
[--start tempo_inicial]
[--end tempo_final]
```

O exemplo abaixo consulta os valores armazenados no banco `memoria.rrd` utilizando a função de consolidação AVERAGE (média) entre 13/10/2006 às 12h51min (`1160749980` segundos) e 13/10/2006 às 18h34min (`1160773440` segundos):

```
root@manager>rrdtool fetch memoria_stream.rrd AVERAGE --start 1160749980 --
end 1160773440
```

Segue a saída do comando acima:

```
                    mem_tot

1160750040: 1.6632800000e+05
1160750100: 1.6632800000e+05
1160750160: 1.6632800000e+05
1160750220: 1.6633200000e+05
1160750280: 1.6640800000e+05
1160750340: 1.6640800000e+05
      :           :
      :           :
1160773440: 1.6650200000e+05
```

Observando a saída acima, a primeira coluna refere-se aos elementos do banco identificados pelo tempo, em segundos desde *Epoch* [6], e a segunda coluna contém os valores armazenados utilizando o comando `update`. Nota-se que o intervalo entre os elementos é 60 segundos conforme especificado na criação do banco RRD.

3.2.4 Graph

É a função de criação de gráficos tendo como base os dados armazenados em diversos bancos RRD. Esta função é a mais frequentemente utilizada em nosso trabalho uma vez que estamos disponibilizando gráficos de todas as variáveis monitoradas, além de nos possibilitar realizar cálculos aritméticos nos dados

originais do banco. As três principais características desta função são: definição de dados e definição de variáveis e expressões RPN (*Reverse Polish Notation*) [11].

Definição de dados. A função `graph` necessita primeiramente de uma fonte de dados. Então, devemos utilizar as chamadas definições de dados para indicar o banco `RRD` onde devem ser coletados os dados. Segue a sintaxe para definição de dados:

```
DEF:[nome_var]=[arquivo_rrd]:[nome-ds]:[CF]
```

A opção `nome_var` é o nome da variável que representa a fonte de dados. As opções `rrdfile, ds-name` e `CF` são: o nome do banco RRD, o nome da fonte de dados do banco e a função de consolidação, respectivamente

No exemplo abaixo, estamos definindo a variável `memoria_total` como sendo a fonte de dados que representa o consumo de memória no banco `memoria.rrd`:

```
DEF:memoria_total=memoria.rrd:mem_tot:AVERAGE
```

Definição de variáveis. É importante salientar que ao definirmos a variável `memoria_total` acima, este mesmo nome pode ser utilizado em todo o script de geração do gráfico. Se quisermos exibir no gráfico valores como médias, máximos, mínimos, percentuais, podemos fazê-lo utilizando a definição de variáveis de acordo com a seguinte sintaxe:

```
VDEF:[nome_var]=expressao RPN
```

A opção `nome_var` é o nome da variável que contém o valor resultante da `expressão RPN` que é a notação utilizada pelo RRDTool para efetuar cálculos aritméticos.

Expressões RPN. *Reverse Polish Notation* é um método para representação de expressões aritméticas em que o operador é colocado após os operandos [11]. Na prática, o método para avaliação das expressões RPN pode ser entendido como um algoritmo que efetua operações de inserção e remoção, conhecidas como *push* e *pop*, em uma estrutura de dados de uma pilha do tipo LIFO (*Last In, First Out*) [3].

O algoritmo percorre a expressão da esquerda para direita e a seguinte seqüência de iterações é realizada:

1) Se um operando (valor numérico) for encontrado, o mesmo é inserido (*push*) na pilha.

2) Quando o operador (soma, multiplicação, divisão, etc.) for encontrado, dois operandos no topo da pilha são retirados (*pop*), a operação aritmética/booleana é realizada e o resultado da mesma é inserido (*push*) na pilha. As interações continuam até que a pilha só contenha um único valor. Este valor final deve ser o resultado da expressão.

Por exemplo, a expressão "`1,2,3,+,+`" é avaliada da seguinte forma: primeiro, o valor 1 (um) é inserido na pilha seguido dos valores 2 (dois) e 3 (três). Quando o algoritmo encontra o operador "+", os dois operandos do topo da pilha são retirados, a operação de soma é realizada entre eles e o resultado desta operação é inserido no topo da pilha. O algoritmo varre novamente a expressão até que todos os operadores tenham sido utilizados, restando apenas o valor final de expressão conforme apresentado na Figura 5.

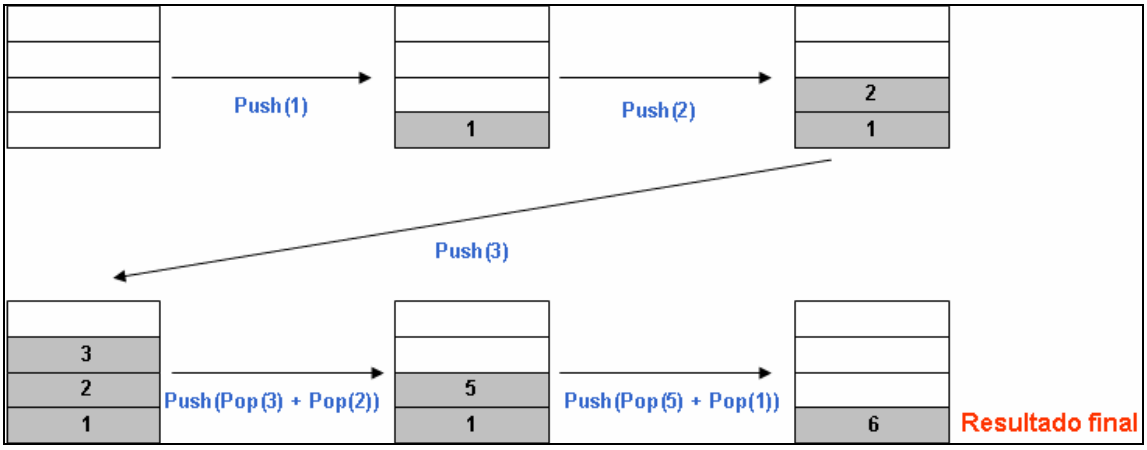

Figura 5. Avaliação da expressão RPN "1,2,3,+,+"

No exemplo abaixo, estamos, primeiro, transformando a memória de `KBytes` (*KiloBytes*) para `MBytes` (*MegaBytes*) apenas dividindo o valor original de memória por `1024` e representando este novo conjunto de dados por `memMB`. Logo após, estamos calculando a média deste valor.

```
CDEF:memMB=memoria_total,1024,/

VDEF:media=memMB,AVERAGE
```

Uma vez que temos as fontes de dados na escala que desejamos, `MBytes` por exemplo, podemos proceder à exibição da informação final. Dentre as diversas opções disponíveis no RRDTool, vamos apresentar as que estamos utilizando para exibir os dados no gráfico:

- `LINE[width]:value:[#color]:[legend]` – Desenha uma linha com a espessura especificada por `width`. `Value` recebe o nome da variável especificada pelas definições dos dados (`DEF`, `CDEF` e `VDEF`). `Color` é a cor da linha que utiliza o padrão *RGB (Red-Green-Blue)* [4]. `Legend` é a legenda que será exibida abaixo do gráfico.
- `AREA:value:[#color]:[legend]` – Semelhante à linha, só que a área entre os pontos no eixo-x e a linha y=0 estará preenchida.
- `COMMENT:texto` – Um texto é exibido na área reservada para legendas no gráfico.
- `GPRINT:vname:format` – Utilizado para exibir os valores definidos apenas pelas opções `VDEF`. Ou seja, é utilizado para exibir médias, máximos e mínimos.

Segue como exemplo de utilização da função `graph`, o gráfico da Figura 6 do consumo de memória de todos os processos em execução no servidor de streaming chamado `memoria.png`.

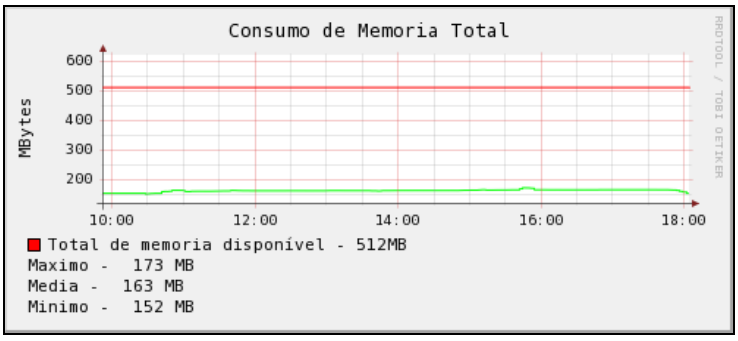

Figura 6. Gráfico de consumo de memória do servidor de streaming.

Capítulo 4

Metodologia

A metodologia empregada em nosso trabalho é experimental, ou seja, iremos realizar experimentos com o servidor de streaming utilizando uma infra-estrutura de rede local, conforme exibida na Figura 7. Os experimentos, em nosso trabalho, são as coletas de dados de diversas variáveis que foram realizadas nos períodos em que o servidor de streaming esteve em sua maior atividade. As variáveis, objetos SNMP, selecionadas para monitoramento estão apresentadas na Tabela 3. Estas variáveis podem representar *strings*, que são utilizados para coletar informações de gerenciamento do servidor, e contadores que estão sempre se atualizando para melhor descrever o comportamento atual dos componentes (hardware/software) e dos serviços do nó gerenciado, utilizado para a análise de desempenho do servidor. Nossos experimentos consistem em consultar todos estes contadores, utilizando o SNMP, em intervalos regulares (minuto a minuto) por um determinado período, e em seguida armazenar e apresentar estes dados utilizando uma ferramenta apropriada, o RRDTool. Primeiro, para conseguirmos realizar estes experimentos precisávamos garantir que poderíamos instalar uma estação de gerenciamento na infra-estrutura existente da empresa. Segundo, para coletar dados de todas as variáveis propostas, precisaríamos monitorar o nó gerenciado nos períodos em que o servidor de streaming estivesse em plena atividade. Terceiro, armazenamos os dados coletados em bancos de dados RRD com capacidade para 1 (um) dia de monitoramento. Quarto, exibimos a informação das diversas variáveis monitoradas em forma de gráficos, disponibilizando-os na ferramenta de monitoramento que desenvolvemos para a Web.

Tabela 3. Objetos SNMP selecionados para monitoramento

Objeto SNMP	Grupo	Descrição
hrSWRunPerfMem	HOST-RESOURCES-MIB	Memória alocada para o processo
hrSWRunPerfCPU	HOST-RESOURCES-MIB	Tempo de CPU alocado para o processo
ifInOctets	INTERFACES	Entrada de bytes pela interface
ifOutOctets	INTERFACES	Saída de bytes pela interface
ipInDelivers	IP	Entrada de pacotes IP
ipOutRequests	IP	Saída de pacotes IP
tcpInSegs	TCP	Entrada de pacotes TCP
tcpOutSegs	TCP	Saída de pacotes TCP
udpInDatagrams	UDP	Entrada de pacotes UDP
udpOutDatagrams	UDP	Saída de pacotes UDP
sysDescr	System	Descrição do sistema(Hardware e Software)
sysContact	System	Contato responsável pelo sistema
sysName	System	Nome do sistema

4.1 Infra-estrutura dos experimentos

Deve-se garantir que a estação de gerenciamento e que os nós, em nosso caso o servidor de Streaming, estejam logicamente acessíveis entre si. Isto foi feito colocando-os na mesma rede, ou seja, em mesmo domínio de colisão conforme exibido na Figura 7. Deve-se salientar que este ambiente é simulado, ou seja, toda a infra-estrutura de acesso ao servidor foi construída para ser acessível apenas dentro da própria organização. É importante considerar esta restrição no projeto, uma vez que, normalmente, os servidores de streaming são disponibilizados e acessíveis na Internet.

Os equipamentos estão interconectados via cabos de par-trançado por rede local Ethernet 100BaseT [13] com o auxílio de um HUB como ilustrado na Figura 7.

Para acessar os conteúdos multimídia do servidor de streaming, o telefone celular, exibido na Figura 7, precisa primeiramente obter conectividade com o simulador de rede W-CDMA (3G) [17], o equipamento Agilent [1] exibido na Figura 6(b). Para que isso ocorra, é necessário que um cartão SIM [8] do fabricante Agilent

seja inserido no telefone e que o telefone seja conectado ao simulador via cabo coaxial. Uma vez que o telefone esteja recebendo o sinal 3G do simulador, podemos iniciar as requisições ao servidor.

Levando-se em consideração que o serviço de streaming tenha sido inicializado no servidor e que os conteúdos multimídia tenham sidos disponibilizados, o telefone está apto para acessar os conteúdos através do endereço: `rtsp://172.27.4.195/nome_do_arquivo`.

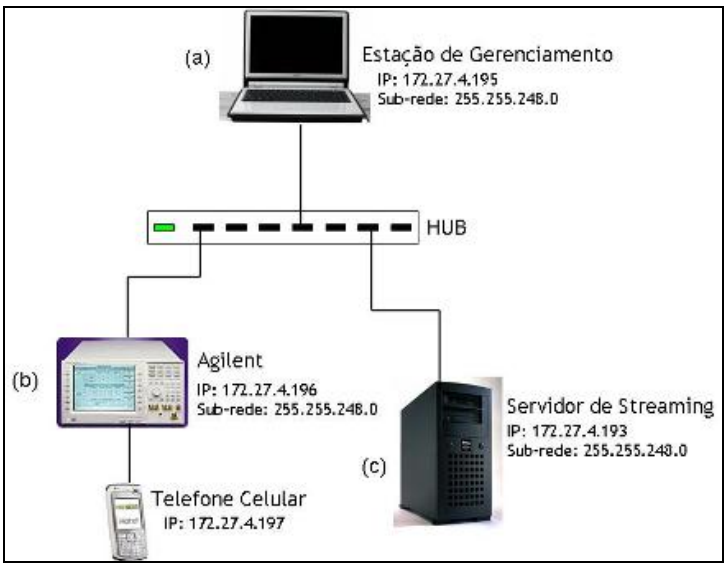

Figura 7. Infra-Estrutura

4.1.1 Especificações do Hardware

a) **Na Figura 7(a), temos a Estação de Gerenciamento.** Fabricante: AMAZONPC. Processador: Pentium 4 Intel 1.7GHz. Memória RAM: 512MB. Placa de rede: Realtek Gigabit Ethernet. Endereço IP: 172.27.4.195.

b) **Na Figura 7(b), temos o Simulador de rede W-CDMA(3G).** Fabricante: Agilent Technologies [1]. Modelo: 8960. Endereço IP: 172.27.4.196.

c) **Na figura 7(c), temos o Servidor de Streaming.** Fabricante: Positivo. Processador: Pentium 4 Intel 2.6GHz. Memória RAM:512MB. Placa de rede: Realtek Gigabit Ethernet. Endereço IP: 172.27.4.193.

d) **Na figura 7(d), temos o Telefone Celular.** Fabricante: Motorola. Modelo: E390. Endereço IP: 172.27.4.197

e) **Na figura 7(e), temos o HUB.** Fabricante: D-Link. Hub 8 portas 10/100Mbps. Modelo DES-1008D.

4.1.2 Especificações do Software

a) **Na Figura 7(a), temos a Estação de Gerenciamento**. Sistema Operacional: Linux Red Hat versão 7.2 [16]. Linguagem dos Scripts: `Shell` [6]. Processos utilizados: `httpd, crond, smnpd`.
Site na web: Desenvolvido utilizando as linguagens PHP e HTML.

b) **Na Figura 7(c), temos o Servidor de Streaming**: Sistema Operacional: Microsoft - Windows 2003 Server.
Aplicação de Streaming**:** Real Networks - *Helix Mobile Streaming Server* [12].

4.2 Coleta de dados

Como mencionado, deve-se garantir que a estação de gerenciamento e que o servidor de streaming estejam logicamente acessíveis entre si. Normalmente, essa verificação é feita enviando comandos `ping` ao equipamento de destino. O equipamento irá responder confirmando o recebimento do pacote.

Após esta verificação inicial, o agente SNMP deve ser configurado no equipamento gerenciado.

O SNMP utiliza a noção de comunidades para estabelecer confiança entre gerentes a agentes. Um agente é configurado com três nomes de comunidade: `read-only, read-write e trap`. Os nomes das comunidades podem ser entendidos como senhas para acessar as informações dos agentes. Por exemplo, a senha da comunidade `read-only` permite a leitura dos dados, mas não permite a

escrita. A senha da comunidade `read-write`, permite leitura e escrita dos dados. A senha da comunidade `trap` permite o recebimento de mensagens `trap` do agente.

Em nosso trabalho, estamos utilizando a senha padrão da comunidade `read-only` chamada `public` no agente SNMP do servidor de Streaming. Ou seja, estamos apenas consultando os dados do servidor e recebendo mensagens `trap`.

Após a configuração do SNMP e suas comunidades, podemos proceder às primeiras coletas de informações ao servidor para fins de testes.

Estamos utilizando dois mecanismos para coleta de informações:

1) snmpget: Referente ao modo de operação `get` do SNMP, este método é enviado da estação de gerenciamento para o agente que, por sua vez, responde com o valor do objeto SNMP solicitado. Utilizando este mecanismo conseguimos coletar o consumo de memória atual do processo da aplicação de streaming em execução no servidor, como segue:

```
root@manager> snmpget -v 1 -c public 172.27.4.195
hrSWRunPerfMem.4048
```

Onde:

`-v 1`: versão do SNMP.

`-c` : nome da comunidade.

`172.27.4.195`: endereço IP do servidor de streaming.

`hrSWRunPerfMem:` Objeto SNMP que representa o consumo de memória.

`4048:` identificador da porta do processo em execução de streaming.

Então, o agente no servidor de streaming processa o comando e retorna o valor do objeto SNMP solicitado como segue:

```
HOST-RESOURCES-MIB::hrSWRunPerfMem.4048 = INTEGER: 25656
KBytes
```

2) snmpwalk: Referente ao modo de operação *get-next* do SNMP, este método é também enviado da estação de gerenciamento para o agente que, por sua vez,

responde com um conjunto de valores referente ao objeto SNMP. Utilizando este mecanismo conseguimos coletar o consumo de memória atual de todos os processos em execução no servidor de streaming, como segue:

```
root@manager> snmpwalk -v 1 -c public 172.27.4.195
hrSWRunPerfMem
```

Onde:

`-v 1:` versão do SNMP.

`-c`: nome da comunidade.

`172.27.4.195`: endereço IP do servidor de streaming.

`hrSWRunPerfMem:` Objeto SNMP que representa o consumo de memória

Então, o agente processa o comando e retorna o conjunto de valores do objeto SNMP solicitado como segue:

```
HOST-RESOURCES-MIB::hrSWRunPerfMem.3602 = INTEGER: 648 KBytes
HOST-RESOURCES-MIB::hrSWRunPerfMem.3607 = INTEGER: 428 KBytes
HOST-RESOURCES-MIB::hrSWRunPerfMem.3642 = INTEGER: 3592 KBytes
HOST-RESOURCES-MIB::hrSWRunPerfMem.3651 = INTEGER: 5092 KBytes
HOST-RESOURCES-MIB::hrSWRunPerfMem.3661 = INTEGER: 812 KBytes
HOST-RESOURCES-MIB::hrSWRunPerfMem.3672 = INTEGER: 6560 KBytes
HOST-RESOURCES-MIB::hrSWRunPerfMem.3681 = INTEGER: 572 KBytes
                              .
                              .
                              .
HOST-RESOURCES-MIB::hrSWRunPerfMem.4127 = INTEGER: 1520 Kbytes
```

Uma vez que os agentes estão respondendo às solicitações da estação de gerenciamento, podemos iniciar a coleta de dados propriamente dita.

Primeiramente, criamos um script shell, chamado `atualiza_banco.sh`, para coletar e armazenar, ao mesmo tempo, os dados de todos os objetos SNMP selecionados para monitoramento. Este script foi executado diversas vezes

manualmente para se garantir que os dados estavam realmente sendo coletados. Para melhor entendimento, vamos exibir um trecho do código deste script abaixo:

atualiza_banco.sh

```
1 #!/bin/sh
2 SNMP_BASE=/usr/local/bin
3 RRD_BASE=/root/rrdtool-1.2.15/bin
4 IP=172.27.4.195
5 tempo_atual=$(date +%s)
6 $SNMP_BASE/snmpwalk -v 1 -c public $IP hrSWRunPerfMem >
7 $RRD_BASE/snmp_mem_tot
8 $SNMP_BASE/snmpget -v 1 -c public $IP hrSWRunPerfMem.4048 >
9 $RRD_BASE/snmp_mem_stream
10 $SNMP_BASE/snmpwalk -v 1 -c public $IP hrSWRunPerfCPU >
11 $RRD_BASE/snmp_CPU_tot
```

As quatro primeiras linhas estão declarando quatro variáveis que são utilizadas ao longo do script. Para executar os comandos SNMP e RRDTool, o Linux necessita conhecer o caminho dos arquivos executáveis dos comandos. A variável `$SNMP_BASE` da linha 1 (um) armazena o endereço das funções SNMP e a variável `$RRD_BASE` linha 2 (dois) armazena o endereço das funções RRDTool. A variável `$IP` da linha 3 (três) armazena o endereço IP do servidor de streaming e a quarta variável armazena a hora atual da estação de gerenciamento em segundos desde Epoch [9]. Os três comandos das linhas 6, 8 e 10 coletam os dados em si dos objetos `hrSWRunPerfMem` (memória consumida por todos os processos), `hrSWRunPerfMem.4048` (memória consumida pelo processo de streaming) e `hrSWRunPerfCPU` (tempo de CPU alocado para todos os processos em centi-segundos), respectivamente. O símbolo '>' redireciona a saída dos comandos para os arquivos-texto `snmp_mem_tot`, `snmp_mem_stream` e `snmp_CPU_tot`.

Então, se quisermos verificar se os dados estão sendo realmente coletados, precisamos verificar o conteúdo dos arquivos-texto gerados pela execução do script `atualiza_bancos.sh`.

Lembrando que configuramos os arquivos RRD para armazenar dados no intervalo de 60 segundos, precisamos automatizar o script de coleta para ser

executado nesse mesmo intervalo. A ferramenta utilizada para automatizar tarefas no Linux se chama cron [6]. O `cron` é um daemon (processo) que pode ser utilizado para agendar tarefas recorrentes de acordo com uma combinação de minuto, hora, dia do mês, mês e dia da semana. Para agendar uma nova tarefa devemos seguir o formato abaixo:

```
minuto   hora dia  mes  dia-da-semana  comando
```

Onde:
`minuto` – inteiro de 0 até 59
`hora` – inteiro de 0 até 23
`dia` – inteiro de 1 até 31
`mes` – inteiro de 1 até 12
`dia-da-semana` – inteiro de 0 até 7
`comando` – Comando a ser executado. Em nosso caso, o script de coleta de dados.

Já que nosso objetivo é coletar dados de minuto a minuto e o processo `cron` pesquisa tarefas agendadas neste mesmo intervalo, devemos utilizar o asterisco (*) em todos os campos, exceto `comando`, indicando que a tarefa deve ser executa em qualquer intervalo, como segue:

```
*    *    *    *    *    /root/rrdtool/bin/atualiza_banco.sh
```

Desta forma, estamos agendando nosso script *atualiza_banco.sh* para ser executado de minuto em minuto.

4.3 Armazenamento de dados

Nota-se que a resposta do agente SNMP utiliza uma notação específica como segue:

```
HOST-RESOURCES-MIB::hrSWRunPerfMem.3602 = INTEGER: 648 KBytes
```

Figura 8. Resposta do agente SNMP

Como estamos interessados apenas no valor da variável solicitada, tivemos que utilizar, em nosso script `atualiza.bancos.sh`, uma linguagem para manipulação de arquivos-texto chamada AWK [7] para que apenas o valor da variável fosse coletado para armazenamento no banco de dados. A função básica do AWK é pesquisar padrões nas linhas dos arquivos. Quando um padrão é encontrado em uma linha, o AWK realiza ações nesta linha. Este processamento continua até que todas as linhas do arquivo sejam processadas.

Um código AWK consiste de uma série de regras, onde cada regra especifica um padrão para pesquisa e uma ação a ser tomada quando o padrão é encontrado. Sintaticamente, as regras consistem de um padrão seguido de uma ação. A ação é delimitada por chaves para separá-la do padrão, como segue:

```
padrao { acao }
padrao { acao }
     ⋮
```

Para executarmos um programa AWK, devemos seguir o formato abaixo:

```
awk ‘programa’ arquivo1 arquivo2 ...
```

A linguagem AWK nos permite selecionar campos específicos em cada linha do arquivo de entrada. Por exemplo, se quisermos selecionar apenas o valor em *KBytes* do texto da Figura 8, utilizamos o seguinte comando:

```
`awk '{memS =$4} END {print memS}' snmp_mem_stream`
```

O comando acima utiliza como base o arquivo-texto `snmp_mem_stream` e armazena em uma variável temporária `memS`, o valor contido no quarto campo da linha ($4), imprimindo a variável temporária para a saída padrão.

Utilizando a mesma idéia, podemos somar os valores do quarto campo de cada uma das linhas do arquivo texto `snmp_mem_tot` como segue:

```
`memoria_tot=`awk '{memT +=$4} END {print memT}' $RRD_BASE/snmp_mem_tot`
```

O comando acima atribui a variável `memoria_tot,` a soma dos valores do quarto campo de todas as linhas do arquivo-texto `snmp_mem_tot`. Então, a variável `memoria_tot` é a soma do consumo de memória de todos os processos em execução no servidor de streaming.

Já que conseguimos, com o auxílio do AWK, selecionar os valores em si das variáveis monitoradas, podemos proceder ao armazenamento dos dados nos bancos RRD que criamos no trabalho apresentados na Tabela 4 (próxima página).

Segue o trecho final do código do script `atualiza_bancos.sh` para armazenamento dos dados:

```
1 $RRD_BASE/rrdtool update $RRD_BASE/memoria.rrd
2 $tempo_atual:$memoria_tot
3 $RRD_BASE/rrdtool update $RRD_BASE/memoria_stream.rrd
4 $tempo_atual:$memoria_stream
5 $RRD_BASE/rrdtool update $RRD_BASE/cpu.rrd
6 $tempo_atual:$cpu_tot
                    :
                    :
```

Na linha 1, `memoria.rrd` é o nome do banco de dados RRD que armazena a soma de valores de consumo de memória de todos os processos. Na linha 3, `memoria_stream.rrd` é o nome do banco de dados RRD que armazena apenas o consumo de memória do processo de streaming. Na linha 5, `cpu.rrd` é o nome do banco de dados RRD que armazena a soma dos tempos de CPU alocados para todos os processos.

Tabela 4. Bancos de dados RRD

Nome	Tipo de fonte de dados	Descrição
Memoria.rrd	GAUGE	Armazena a soma de valores de memória de todos os processos em execução.
Memoria_stream.rrd	GAUGE	Armazena o valor de memória apenas do processo em execução de streaming.
cpu.rrd	GAUGE	Armazena a soma dos tempos de CPU alocados para todos os processos em execução
cpu_stream.rrd	GAUGE	Armazena o tempo de CPU alocado para o processo de streaming.
trafego.rrd	COUNTER	Armazena o tráfego de entrada e saída de bytes na interface de rede.
bandaES.rrd	COUNTER	Armazena o tráfego de entrada e saída de pacotes IP.
tcp.rrd	COUNTER	Armazena o tráfego de entrada e saída de pacotes TCP.
udp.rrd	COUNTER	Armazena o tráfego de entrada e saída de pacotes UDP.

Para garantir que os dados estão sendo armazenados corretamente, utilizamos a função `fetch`, utilizando como argumentos, o tempo inicial e final da coleta e armazenamento de dados. Por exemplo, se o script `atualiza_bancos.sh` estava sendo executado no período entre 13/10/2006 às 14h33min e 13/10/2006 às 18hs33min, devemos utilizar o seguinte comando:

```
root@manager> rrdtool fetch memoria.rrd AVERAGE --start
1160749980 --end 1160773440
```

Onde:

`1160749980` – referente ao tempo inicial da coleta: 13/10/2006 às 14h33min

`1160773440` – referente ao tempo final da coleta: 13/10/2006 às 18hs33min

Segue abaixo o retorno do comando acima, o que nos garante que os dados de fato foram coletados e armazenados no banco RRD:

```
1160749980: 1.6632800000e+05
1160750040: 1.6632800000e+05
1160750100: 1.6632800000e+05
1160750160: 1.6632800000e+05
1160750220: 1.6633200000e+05
1160750280: 1.6640800000e+05
            .
            .
            .
1160773440: 1.5790000000e+05
```

4.4 Apresentação dos dados

Após garantirmos que os métodos de coleta e armazenamento de dados estão funcionando, podemos proceder aos métodos de apresentação da informação coletada em forma de gráficos utilizando a função `graph`.

Para gerar os primeiros gráficos das variáveis monitoradas, criamos o script `gera_graficos.sh`. Inicialmente criamos um gráfico para banco RRD e os apresentamos em uma mesma página na estação de gerenciamento para disponibilizar uma idéia geral sobre o comportamento de todas as variáveis monitoradas.

Após isso, geramos gráficos de média e variância de cada uma das variáveis em separado para possibilitar uma análise mais específica do comportamento de cada uma destas características do servidor.

Para melhor visualização da informação coletada, devemos considerar as unidades de medida em que os dados foram coletados. Por exemplo, os dados de consumo de memória RAM foram coletados originalmente pelo SNMP em `KBytes`. Uma vez que, atualmente, a capacidade de memória RAM dos equipamentos é medida em `MBytes` (MegaBytes), é interessante apresentar a informação em

unidades de MBytes. Para realizar esta transformação, utilizamos as opções CDEF e a notação RPN, como segue:

```
CDEF:memMB=memoria_total,1024,/
```

Outro exemplo é a unidade utilizada para apresentar os dados do gráfico de tráfego de entrada e saída de dados. Os dados são coletados originalmente em Bps (Bytes por segundo), mas achamos conveniente, para o trabalho, transformar o tráfego em unidades de MBps (MegaBytes por segundo), já que estamos trabalhando em rede local. Neste caso, utilizando a notação RPN, dividimos o valor original por 1048576, como segue:

```
CDEF:entr_MBps=entrada,1048576,/
```

Visando melhorar a visualização da informação coletada também estamos apresentando os valores máximos e mínimos de cada uma das variáveis, utilizando as opções VDEF e GPRINT, como segue:

```
'VDEF:maximo_saida=saida_MBps,MAXIMUM'
'COMMENT:Maximo -'
'GPRINT:maximo_saida:%2.2lf MBps\n'
```

Onde o primeiro comando acima, declara uma variável maximo_saida, como sendo o valor maximo encontrado na fonte de dados representada pela variável saida_MBps. O segundo comando imprime o texto "Maximo – " na área reservada para legendas do gráfico e o terceiro comando imprime o valor máximo em si com duas casa decimais e o texto "MBps" em seguida.

A Figura 9 apresenta o gráfico do tráfego de entrada e saída de dados em MegaBytes por segundo utilizando as opções citadas acima.

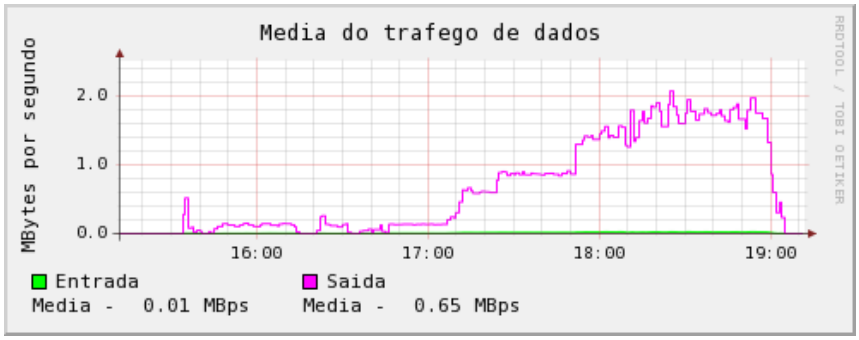

Figura 9. Tráfego de Entrada e Saída de dados em MBps.

As opções `CDEF` e `VDEF` do RRDTool nos permitiu também a criação de gráficos que representam a variância dos dados coletados. A fórmula da variância está apresentada abaixo:

$$Var(x) = (x - E(x))^2$$

Onde E(x) é a média dos valores x.

Baseando-se nesta fórmula, criamos as seguintes definições:

```
'VDEF:media=memMB,AVERAGE'
'CDEF:diferenca=memMB,media,-'
'CDEF:variancia_mem=diferenca,diferenca,*'
```

A primeira opção define uma variável `media` como sendo a média dos valores representados pelo variável `memMB`. A segunda opção define uma variável `diferenca` como sendo a diferença entre os valores representados por `memMB` e a variável `media`. A terceira opção define a variável `variancia_mem` como sendo a multiplicação da variável `diferenca` por ela mesma. Ou seja a variável `variancia_mem` é a variância de `memMB`.

A Figura 10 apresenta a variância da memória utilizando as opções citadas acima.

Figura 10. Variância do Consumo de memória.

Estes gráficos e todos os outros das demais variáveis, além de informações de gerenciamento, foram disponibilizados na ferramenta on-line que chamamos de Estação de Gerenciamento, descrita adiante.

4.5 Validação de média

O RRDTool disponibiliza um método para cálculo da média dos valores armazenados em um banco RRD utilizando a opção VDEF conforme apresentado a seguir.

VDEF:[nome_var]:[nome_ds],**AVERAGE**

Para fins de validação, julgamos conveniente calcular a média utilizando a notação RPN[1]. Primeiramente, selecionamos um intervalo de 4 (quatro) horas do banco memoria.rrd, período em que o servidor de streaming esteve em plena atividade. Em seguida, criamos 4 (quatro) bancos RRD – `memoria1.rrd`, `memoria2.rrd`, `memoria3.rrd` e `memoria4.rrd` – contendo espaço limitado para armazenamento de apenas 1 (uma) hora cada um. Para a população do banco `memoria1.rrd` selecionamos os dados da primeira hora do banco `memoria.rrd`. No banco `memoria2.rrd` armazenamos os valores da segunda hora do banco `memoria.rrd` e assim por diante. No final, temos os valores do banco `memoria.rrd` divididos em quatro bancos distintos. Uma vez que temos 4 fontes de

[1] Ver página 25

dados distintas, podemos realizar operações aritméticas utilizando a notação RPN entre elas, como segue:

```
CDEF:media_calculada=memoria1,memoria2,memoria3,memoria4,+,+,+,4,/
```

O código acima realiza a soma entre as quatro fontes de dados e o resultado desta operação é dividido por 4 (quatro), ou seja, estamos calculando a média de valores armazenados nos quatro bancos.

As Figuras 11 e 12 abaixo apresentam as médias obtidas utilizando o método padrão do RRDTool e o método descrito acima, respectivamente. Verificando os gráficos, podemos notar que os valores são semelhantes, o que nos garante que podemos utilizar o método padrão do RRDTool sem maiores preocupações ao longo do trabalho.

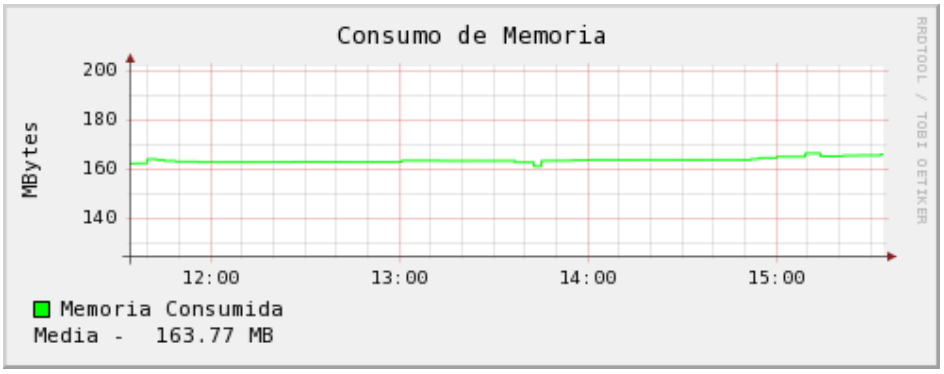

Figura 11. Média calculada pelo RRDTool.

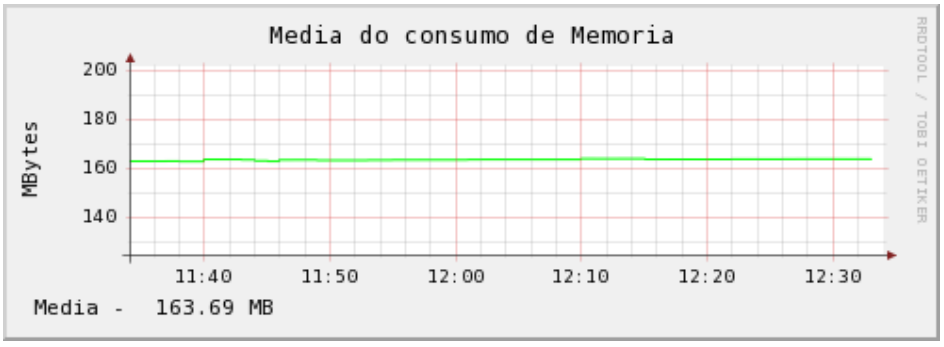

Figura 12. Média calculada utilizando o nosso método.

Capítulo 5

Estação de Gerenciamento

Em função do tamanho e complexidade da rede que será gerenciada, a instalação de uma estação de gerenciamento pode requerer um poder computacional elevado do hardware [10]. Já que estamos requisitando informações de apenas um nó da rede e a metodologia utilizada para coletar os dados é através de scritps Shell em oposição a utilização de softwares proprietários que sobrecarregam o servidor, uma estação de trabalho convencional pode ser utilizado para este propósito. Baseado em MAURO,D. [10], uma estação de trabalho com no mínino 300MHz de capacidade de processamento, 128MB de Memória RAM e 500MB de espaço em disco é suficiente para se instalar uma aplicação de gerenciamento. Nota-se, consultando o capítulo anterior na seção de infra-estrutura, que o equipamento que está sendo utilizado como estação de gerenciamento atende aos requisitos mínimos de hardware.

Um dos objetivos de nosso trabalho é disponibilizar as informações de gerenciamento e desempenho do servidor na Web. Para atingir este objetivo, criamos a aplicação da estação de gerenciamento em um site desenvolvido com o auxílio de duas linguagens. A primeira delas é a linguagem HTML (*Hipertext Markup Language*) [2] responsável pelo layout da aplicação. A segunda é a linguagem PHP (*Hipertext Preprocessor*) [14] responsável pelo conteúdo da página, ou seja, é a ferramenta utilizada para realizar as requisições de informações de gerenciamento.

É importante salientar que a página na Web foi desenvolvida para obter informações de gerenciamento, tais como, verificação do estado de funcionamento do servidor (funcionando ou não) e as consultas da descrição, contato e nome do sistema em tempo real. As coletas das informações de desempenho, o armazenamento destes dados coletados pelo RRDTool e a geração dos gráficos foram realizados de forma manual em background, ou seja, sem intervenção da aplicação na Web, e os resultados - os gráficos de desempenho - foram disponibilizados na página.

A ferramenta está disponível em: `http://172.27.3.148/index.php`. A página de Login do sistema está apresentada na Figura 13:

Figura 13. Login do sistema

Conforme exibido na Figura 13, o sistema solicita um usuário e uma senha de acesso, ambos cadastrados como "`gerente`". Ao pressionar o botão `Login`, o sistema efetua uma operação simples de autenticação utilizando PHP. O código PHP direciona o sistema para a página inicial se a operação retornar verdadeiro ou volta para a página do Login caso contrário, exibindo a seguinte mensagem para o usuário: "`Usuário/senha inválidos. Tente novamente`".

Na página inicial da ferramenta, criamos o script em PHP para o monitoramento dos diversos equipamentos interconectados na rede local. O passo inicial na construção deste script foi definir um cabeçalho utilizado para recarregar a página em intervalos fixos de tempo. Podemos fazer isso utilizando a `tag meta` do HTML, como segue:

```
<META HTTP-EQUIV=”Refresh” CONTENT=60>
```

Nesse caso, o nosso script será recarregado a cada 60 segundos. Em seguida definimos um array que contém informações sobre os equipamentos que desejamos monitorar, adiante:

```
1 <?php
2 $servidores = array (
3 "streaming" => "172.27.4.195",
4 "cesar" => "172.27.3.148"
5 "casa" => "192.168.0.32"
6 );
7 ?>
```

Onde as tags “<?php” e “?>” das linhas 1 e 7 são os delimitadores do código PHP.

O trecho mais importante do nosso script é o código exibido abaixo. Na linha 2, o script percorre cada linha do array de servidores, executa um comando `ping` para cada um deles nas linhas 3 e 4, e finalmente imprime em uma tabela o texto **“On-line”** se a expressão **“bytes from”** for retornada pelo comando ou imprime o texto **“Off-line”,** caso contrário.

```
1 <?php
2 while (list($site,$ip) = each($servidores)){
3 $comando = "/bin/ping -c 1 ".$ip;
4 $saida = shell_exec($comando);
5 if (ereg("bytes from",$saida)){
6 echo "<TD><FONT COLOR=green>On-line</FONT></TD>"; }
7 else {
8 echo "<TD><FONT COLOR=red>Off-line</FONT></TD>"; }
9 ?>
```

Apresentamos a página inicial da ferramenta na Figura 14.

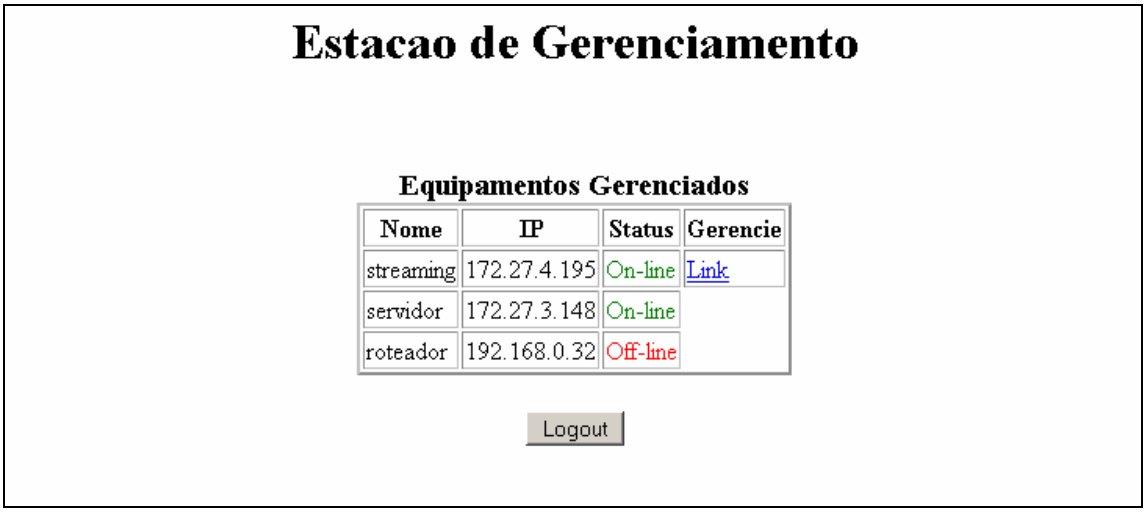

Figura 14. Página inicial do sistema

Ao acessar o link exibido na coluna **"Gerencie"** referente ao servidor de streaming, iremos ser direcionados para a página que apresenta as informações de gerenciamento e desempenho do servidor. Esta página é divida em duas partes. A primeira, coleta em tempo real as informações de gerenciamento do servidor. Para coletar tais informações, criamos um script PHP que solicita os dados dos objetos SNMP utilizando o comando `snmpget` conforme exibido a seguir:

```
1 <?php
2 $comando_descricao = "/usr/local/bin/snmpget -v 1 -c public -Ov
3 172.27.4.195 SNMPv2-MIB::sysDescr.0";
4 $comando_name = "/usr/local/bin/snmpget -v 1 -c public -Ov
5 172.27.4.195 SNMPv2-MIB::sysName.0";
6 $comando_contato = "/usr/local/bin/snmpget -v 1 -c public -Ov
7 172.27.4.195 SNMPv2-MIB::sysContact.0";
8 $descricao = shell_exec($comando_descricao);
9 $name = shell_exec($comando_name);
10 $contato = shell_exec($comando_contato);
11 ?>
```

Como podemos observar no trecho do código acima, as variáveis `$comando_descricao`, `$comando_name` e `$comando_contato` nas linhas 2, 4 e 6, armazenam os comandos `snmpget` para coletar a descrição do sistema, o nome do sistema e o contato responsável pelo sistema, respectivamente. As variáveis `$descricao`, `$name` e `$contato` nas linhas 8, 9 e 10, armazenam o resultado da execução dos comandos shell armazenados pelas variáveis correspondentes.

A segunda parte da página exibe os gráficos de desempenho que tinham sido gerados anteriormente dos objetos SNMP selecionados para monitoramento. Ao clicar nos gráficos, o sistema é redirecionado para a página que contém os gráficos de média e variância da variável selecionada. Por exemplo, ao clicarmos no gráfico que representa o comportamento da variável memória consumida, o sistema é redirecionado para a pagina que exibe os gráficos contendo média e variância do consumo de memória para que uma análise mais aprofundada possa ser efetuada.

Capítulo 6

Resultados Obtidos

O principal objetivo do nosso trabalho é auxiliar o gerenciamento e análise de desempenho do servidor de streaming com a utilização de uma ferramenta on-line que chamamos de estação de gerenciamento.

Observamos que as informações de gerenciamento foram corretamente coletadas em tempo real utilizando a linguagem PHP como observado no capítulo anterior.

As informações de desempenho do servidor foram coletadas nesta ordem: coleta dos dados de desempenho utilizando o SNMP, armazenamento dos dados coletados, e finalmente a geração de gráficos utilizando o RRDTool. Para atingir este objetivo fizemos quatro coletas de dados nos dias 07/10/2006, 09/10/2003, 11/10/2006 e 13/10/2006.

A seguir vamos apresentar os resultados obtidos, utilizando os gráficos gerados no dia 11/10/2006 no intervalo entre 14h11min e 19h11min, das variáveis selecionadas para analise de desempenho.

Vamos apresentar dois gráficos para cada objeto SNMP monitorado, onde o primeiro exibe o comportamento do objeto ao longo do tempo, assim como a média dos valores exibidos. O segundo exibe a variância do objeto ao longo do tempo.

6.1 Consumo de memória

A Figura 15 apresenta o gráfico da soma da memória consumida por todos os processos em execução no servidor de streaming. Podemos observar que o consumo médio de memória foi 152 MB. Uma vez que o servidor possui 512 MB de memória RAM disponível, podemos concluir que a configuração de memória atual é suficiente para atender aos requisitos do serviço de streaming.

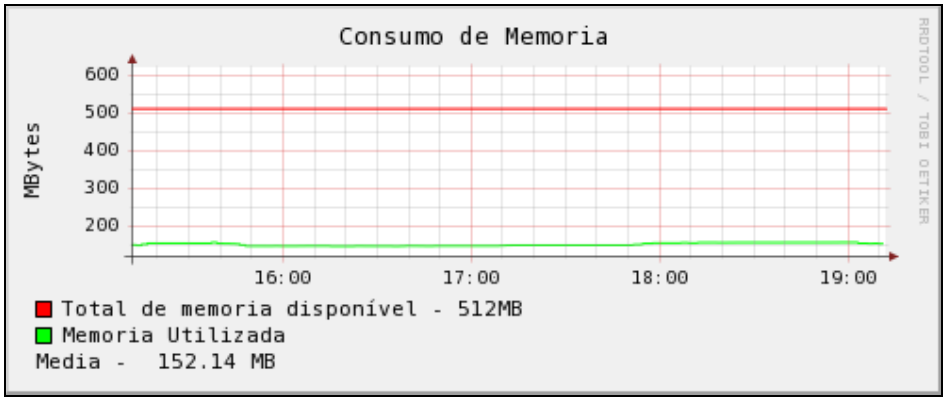

Figura 15. Gráfico do consumo de memória de todos os processos em execução.

A Figura 16 apresenta o gráfico da variância do consumo de memória de todos os processos em execução.

Figura 16. Gráfico da variância do consumo de memória de todos os processos em execução.

Nota-se que o gráfico da variância destaca os valores do consumo de memória que estão fora da média. Também, podemos perceber que no período entre 18 e 19 horas, ocorreu um aumento na variância, indicando que neste período,

houve um aumento de atividade no servidor. No entanto, entre 17 e 18hs, o acesso ficou em torno da média do período.

6.2 Utilização de CPU

A Figura 17 apresenta o gráfico da soma do tempo de CPU, em centi-segundos, alocado para os processos em execução no servidor monitorado.

Observa-se que há um aumento constante do tempo que o processador aloca para os processos em execução no servidor.

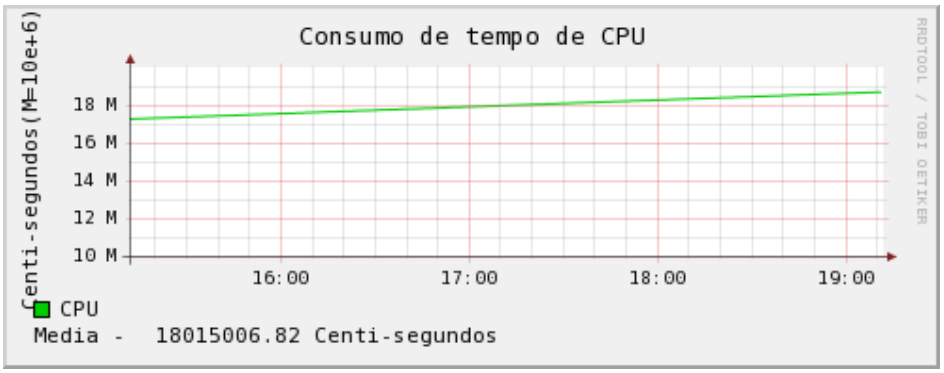

Figura 17. Gráfico de alocação de tempo de CPU para os processos.

A Figura 18 apresenta o gráfico da variância da alocação de tempo de CPU para todos os processos em execução.

Figura 18. Gráfico da variância da alocação de tempo de CPU para os processos.

Nota-se que fora das proximidades de 17hs, a alocação de tempo de CPU dos processos esteve fora da média, reforçando a idéia de crescimento linear deste valor. Relacionando este comportamento com o gráfico da Figura 16, podemos

concluir que antes das 17hs, o tempo alocado pela CPU para os processos em execução foi menor do que a média. Assim como, depois das 17hs, este mesmo tempo foi maior do que a média.

6.3 Tráfego de Entrada e Saída de dados

A Figura 19 apresenta as variáveis que representam o tráfego de entrada e saída de dados, em MegaBytes por segundo (MBps), na interface de rede utilizada pelo servidor. Nota-se que a maior parte do tráfego do servidor compõe-se de dados de saída, como esperado, já que o servidor provê conteúdos multimídia aos clientes.

É importante verificar também na Figura 19 que a taxa máxima de saída foi aproximadamente 2 MBps, equivalendo a 16 Mbps. Como a taxa de transferência máxima da placa de rede do servidor e da rede local dos experimentos são respectivamente, 1000 Mbps (Megabits por segundo) e 100 Mbps [13], podemos concluir que a velocidade no meio de transmissão utilizado atualmente atende às necessidades do servidor.

Por outro lado, se desejarmos prover conteúdos multimídia para clientes externos através da Internet, devemos nos preocupar com a largura de banda que deve estar disponível ao servidor.

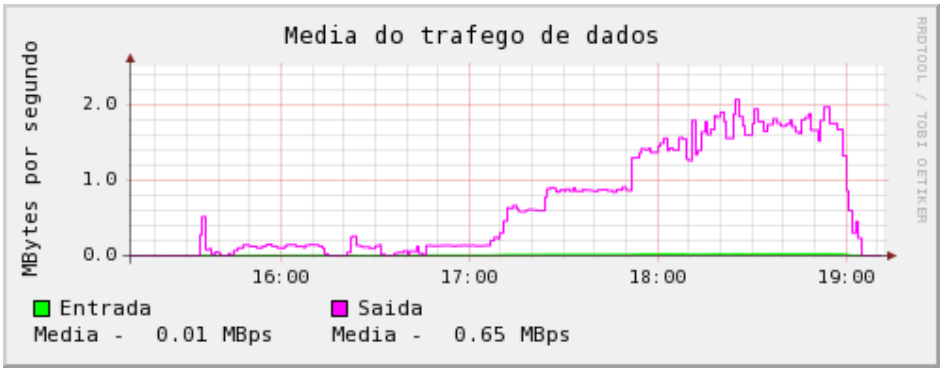

Figura 19. Tráfego de entrada e saída de dados em MBps.

A variância do tráfego de entrada e saída de dados estão ilustrados nas Figuras 20 e 21, respectivamente. Podemos observar que há um aumento da variância no período entre 18 e 19hs, reforçando a conclusão de que nesse intervalo, o servidor de streaming proveu uma quantidade maior de dados.

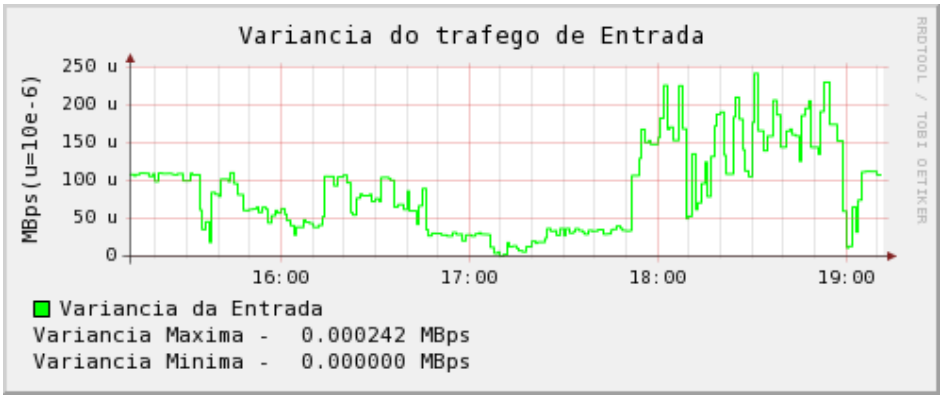

Figura 20. Variância do tráfego de entrada de dados.

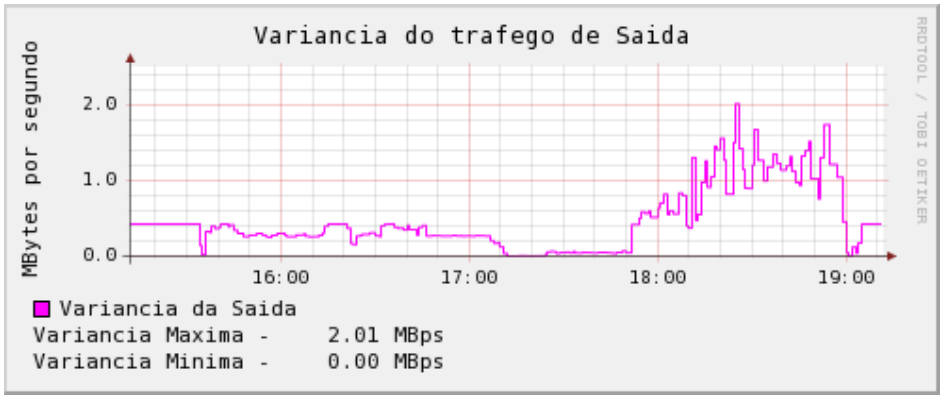

Figura 21. Variância do tráfego de saída de dados.

6.4 Tráfego de Entrada e Saída de Pacotes IP

A Figura 22 apresenta o tráfego de entrada e saída de pacotes IP, incluindo, pacotes TCP, UDP, ICMP e outros. Observamos que os gráficos das Figuras 22 e 19 são semelhantes. Isto também era esperado, uma vez que o tráfego de entrada e saída de dados está diretamente relacionado ao tráfego de entrada e saída de pacotes IP, conseqüência do serviço de streaming oferecido.

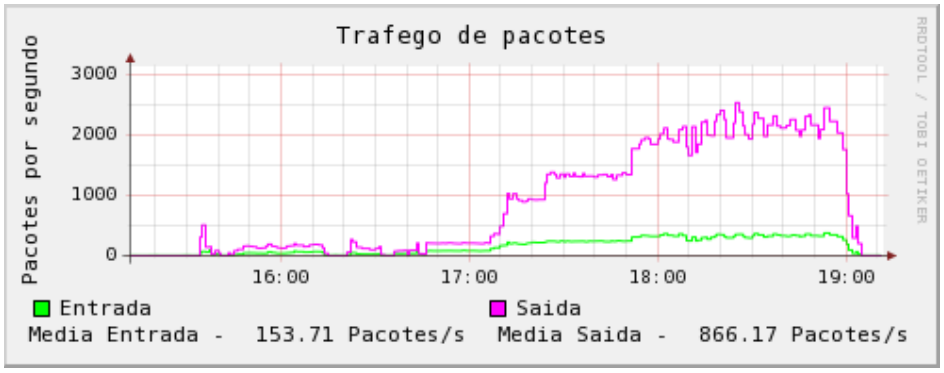

Figura 22. Tráfego de entrada e saída de pacotes IP.

Também achamos conveniente apresentar a variância das variáveis entrada e saída de pacotes em dois gráficos apresentados nas Figuras 23 e 24.

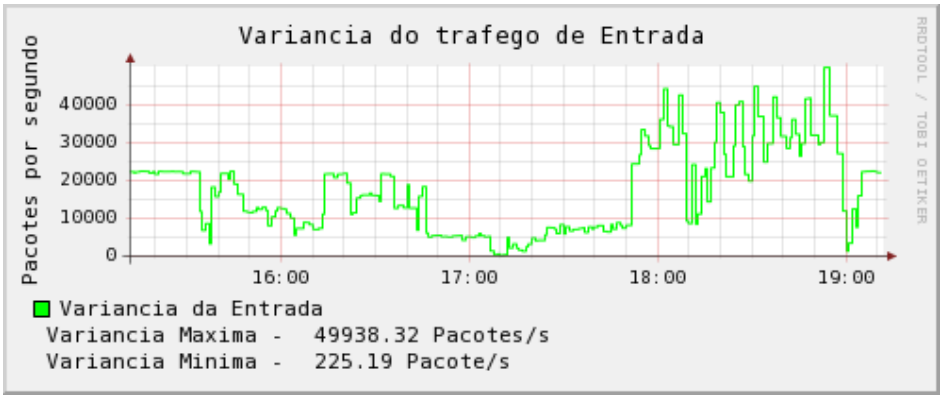

Figura 23. Variância do tráfego de entrada de pacotes IP.

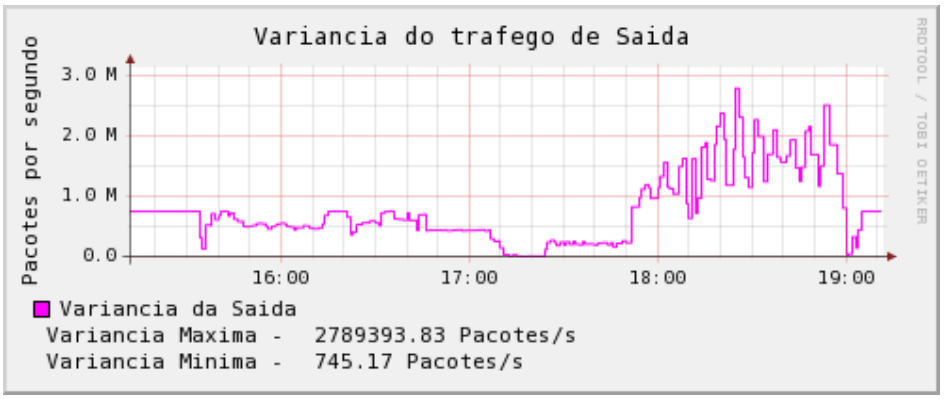

Figura 24. Variância do tráfego de saída de pacotes IP.

Novamente, os gráficos de variância das Figuras 23 e 24 assemelham-se aos gráficos das Figuras 20 e 21, conforme esperado.

6.5 Tráfego de Entrada e Saída de Pacotes UDP

A Figura 25 apresenta o gráfico do tráfego de entrada e saída de pacotes UDP, além da taxa de recebimento de pacotes com porta de destino inválida.

Relacionando esta figura com a Figura 22, observamos que a maior parte dos pacotes enviados pelo servidor utilizam como serviço de transporte o protocolo UDP. Este fato ocorre devido ao fato do servidor de streaming utilizar o protocolo TCP apenas para estabelecer as conexões e o protocolo UDP para trafegar os dados em si. Este é um mecanismo tipicamente utilizado em aplicações que possuem restrições de entrega de dados em tempo real, como o caso do streaming.

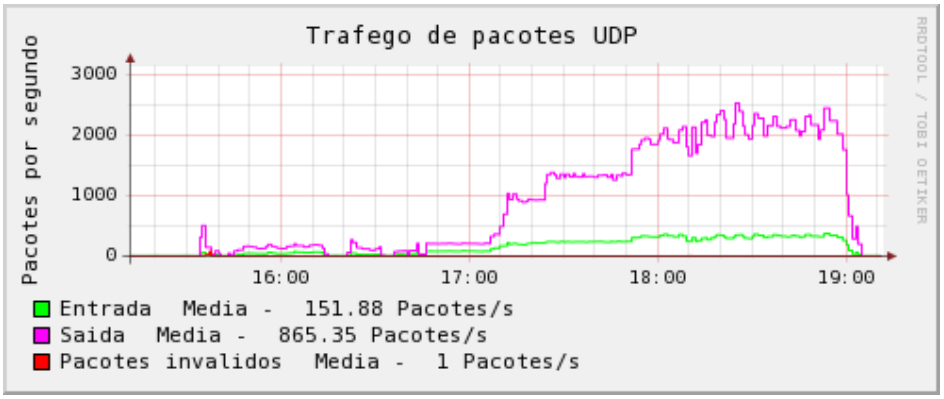

Figura 25. Tráfego de entrada e saída de pacotes UDP.

Nas Figuras 26 e 27 estamos apresentando a variância do tráfego de entrada e saída de pacotes UDP, respectivamente. Observamos uma maior variância de entrada e saída de pacotes UDP no período entre 18 e 19hs, indicando que o servidor proveu uma quantidade de dados acima da média neste período.

Figura 26. Variância do tráfego de entrada de pacotes UDP.

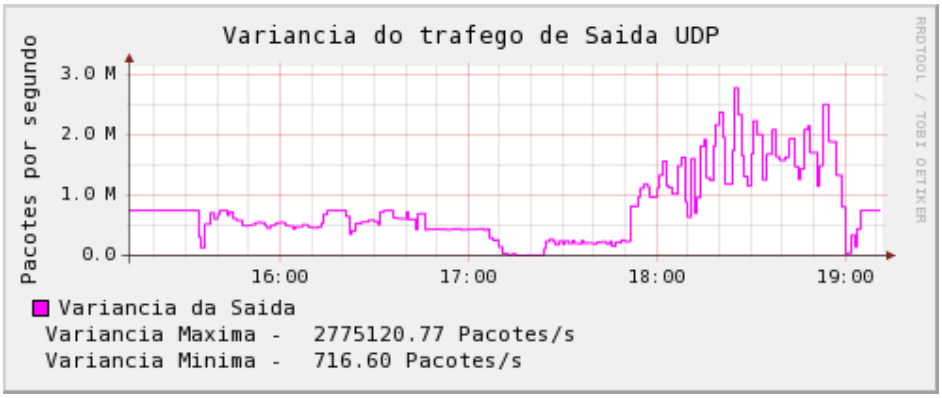

Figura 27. Variância do tráfego de saída de pacotes UDP.

Na Figura 28 estamos apresentando a variância da taxa de entrada de pacotes UDP com porta de destino inválida. Notamos que em torno das 16hs ocorreu um aumento da variância desta taxa, o que nos levou a crer que neste período, o servidor recebeu pacotes UDP endereçados para portas de outros serviços, exceto streaming.

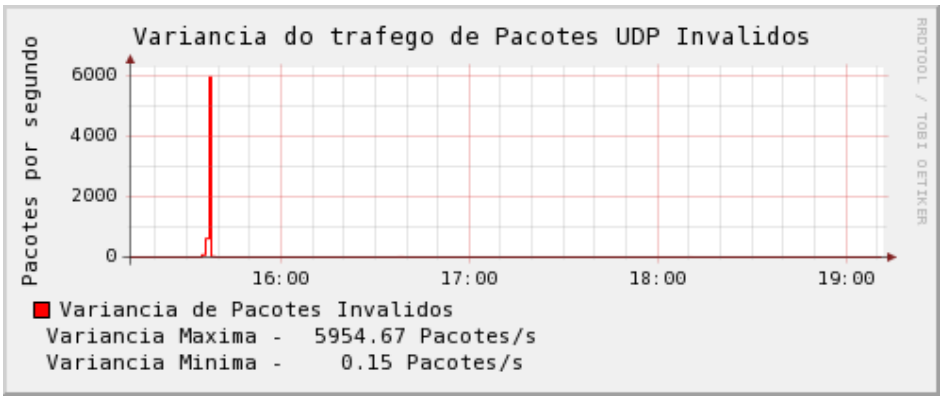

Figura 28. Variância da taxa de recebimento de pacotes com porta inválida de destino.

6.6 Tráfego de Entrada e Saída de Pacotes TCP

A Figura 29 apresenta o gráfico do tráfego de entrada e saída de pacotes TCP além da taxa de retransmissão destes pacotes. Comparando as escalas das ordenadas das Figuras 25 e 29, podemos verificar que a taxa de pacotes TCP é inferior a taxa de pacotes UDP devido aos motivos mencionados anteriormente.

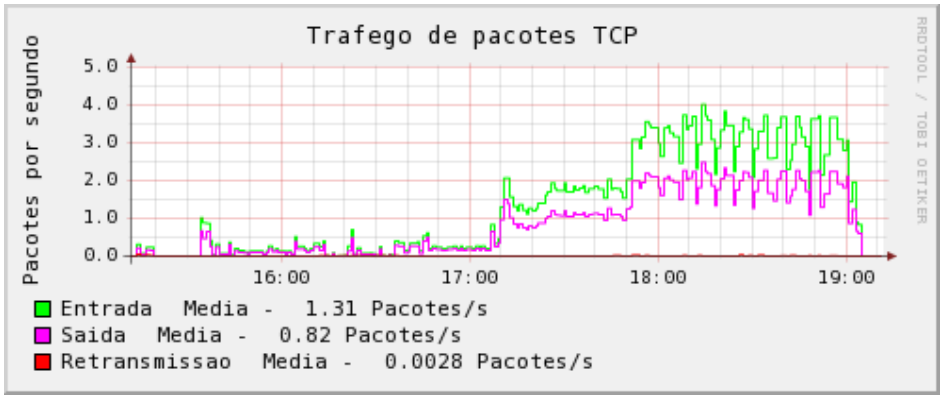

Figura 29. Tráfego de entrada e saída de pacotes TCP

Nas Figuras 30 e 31 estamos apresentando a variância do tráfego de entrada e saída de pacotes TCP, respectivamente. Analisando os dois gráficos, confirmamos que no período entre 18 e 19hs houve um aumento da variância desta taxa devido a uma quantidade maior de requisições dos clientes.

Figura 30. Variância da entrada de pacotes TCP

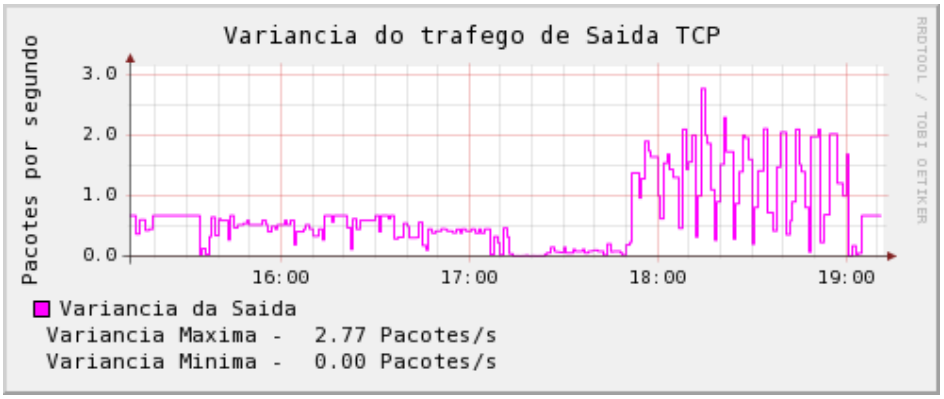

Figura 31. Variância da saída de pacotes TCP

Na Figura 32, estamos apresentando a variância da taxa de retransmissão de pacotes TCP. Notamos que em torno de 14hs, ocorreu um aumento da variância

desta taxa em conseqüência do estabelecimento inicial das conexões com os clientes.

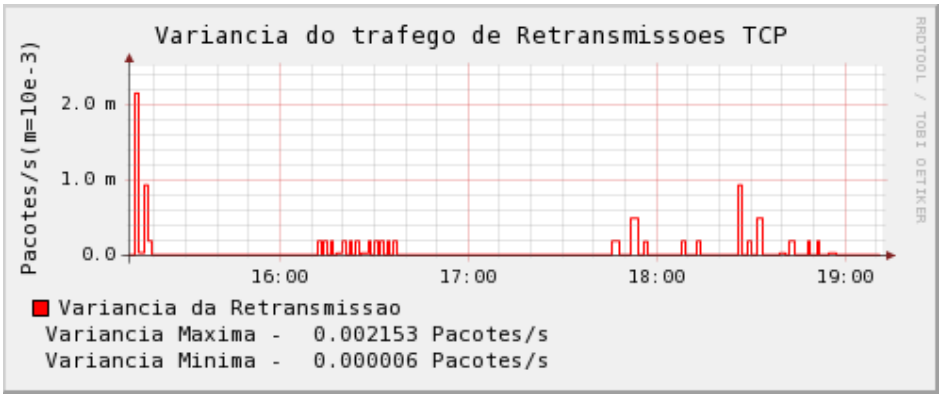

Figura 32. Variância do tráfego de retransmissões de pacotes TCP

Capítulo 7

Conclusões

Atualmente, a diversidade de equipamentos interconectados nas organizações torna difícil ao administrador e projetista de rede executar as tarefas de gerenciamento e análise de desempenho dos equipamentos [10]. O gerenciamento visa saber se os equipamentos - roteadores, servidores e estações de trabalho - estão funcionando, enquanto que a análise de desempenho destina-se a verificar se os mesmos estão trabalhando de acordo com os requisitos do projeto.

Este trabalho apresentou o desenvolvimento de uma ferramenta, utilizando programas de código aberto sob licença GPL, para auxiliar os profissionais da área de redes na realização de tarefas de gerenciamento.

A ferramenta desenvolvida tem duas características principais. Primeiro, coletar, em tempo real, as informações descritivas de gerenciamento dos equipamento, tais como: nome e descrição de hardware/software do componente da rede, contato do responsável. Segundo, coletar informações de desempenho em um período determinado e apresentar a informação coletada para o usuário final em forma de gráficos. A ferramenta foi desenvolvida para ser acessível pela web, de forma que o gerenciamento e análise dos gráficos possa ser realizado de qualquer máquina conectada na rede.

O objetivo principal do trabalho foi descrever uma forma de analisar o comportamento do servidor de streaming através de uma análise da média e variância de consumo de memória, utilização de CPU, tráfego de pacotes, assim como, verificar correlações entre os diversos gráficos.

Baseando-se nas informações apresentadas pelos gráficos gerados, uma análise do comportamento atual do servidor de streaming foi realizada. O resultado do consumo de memória RAM mostrou que estão sendo utilizados, em média, 152 MB de memória, ou seja, aproximadamente 30% de 512 MB disponíveis, indicando que o servidor possui memória suficiente para atender aos clientes. Por outro lado, apesar de observarmos que existe largura de banda suficiente para o servidor atender aos clientes internos, vimos que a taxa máxima de transferência de dados em rede local atingiu aproximadamente 2 MBps, valor este que deve ser levado em consideração no cálculo da banda disponível ao servidor para atender clientes externos utilizando a Internet. A análise do tráfego de dados e de pacotes TCP e UDP foi importante para verificarmos comportamentos característicos de um servidor de streaming, tais como, um maior tráfego de dados na saída do que na entrada da interface, e uma taxa maior de pacotes utilizando o protocolo UDP em comparação ao protocolo TCP. Os gráficos de variância foram importantes para identificarmos os períodos em que o processamento do servidor esteve acima ou abaixo da média, como por exemplo, o gráficos da variância da entrada e saída de dados que apresentam maior variação no período entre 18 e 19hs, indicando que neste período o servidor atendeu a uma quantidade maior de clientes.

Esperamos, no futuro, realizar melhorias na ferramenta desenvolvida explorando mais profundamente as capacidades da linguagem PHP com o objetivo de aperfeiçoar a interação com o usuário de forma que todo o processo de monitoramento e análise de desempenho seja controlado pelo administrador da rede. Por exemplo, o usuário poderá selecionar um subconjunto de seus equipamentos para monitoramento, assim como, definir um intervalo para monitoramento dos objetos SNMP. Alertas enviados por e-mail ou SMS (*Short Message Service*) [9] são outras sugestões para uma próxima versão da ferramenta.

Bibliografia

[1] AGILENT TECHNOLOGIES. Agilent 8960 Series – Application Notes. Disponível em: <http://www.home.agilent.com/agilent/redirector.jspx?action=ref&cname=AGILENT_EDITORIAL&ckey=809370&lc=eng&cc=US> Acesso em: 25 Set. 2006.

[2] MARAM, R. Creating Web Pages with HTML Simplified. 2. ed. IDG Books Worldwide. p. 232.

[3] Tenenbaum, Aaron M.; Langsam, Y. Estruturas de dados usando C. 1. ed. Makron Books. p. 904.

[4] Gomes, J; Velho, L. Computação Gráfica: Imagem. 1. ed. IMPA. p. 421.

[5] GNU General Public License. Disponível em: <http://www.gnu.org/copyleft/gpl.html> Acesso em: 05 Nov. 2006.

[6] GRAIG, H. Linux Network Servers. 1. ed. John Wiley Consumer, 2002. p. 643

[7] ROBBINS, A. Effective Awk Programming. 3. ed. O´reilly, 2001. p. 448.

[8] INTERNATIONAL ENGINEERING CONSORTIUM. Cellular Communications. Disponível em: <http://iec.org/online/tutorials/cell_comm> Acesso em: 14 Nov. 2006.

[9] INTERNATIONAL ENGINEERING CONSORTIUM. Wireless Short Message Service. Disponível em: <http://www.iec.org/online/tutorials/wire_sms/glossary.html> Acesso em: 15 Nov. 2005.

[10] MAURO, D.; SHMIDT, K. Essential SNMP. 1. ed. O´reilly, 2001. p. 291

[11] OETIKER, T. RRDTool – About RRDTool. Disponível em: <http://oss.oetiker.ch/rrdtool/> Acesso em: 19 Set. 2006.

[12] REALNETWORKS. Helix Streaming Server. Disponível em: <http://www.realnetworks.com/products/media_delivery.html> Acesso em: 05 Nov. 2006.

[13] TANENBAUM, ANDREW S. Redes de Computadores. 3. ed. Campus, 1997. p. 875.

[14] SKLAR, D.; TRACHTENBERG, A. PHP Cookbook. 1. ed. O´reilly, 2002. p. 624.

[15] THE INTERNET ENGINEERING TASK FORCE. Disponível em: <http://www.ietf.org> Acesso em: 05 Nov. 2006.

[16] RED HAT. Disponível em: <http://www.redhat.com> Acesso em: 05 Nov. 2006.

[17] UMTS WORLD. WCDMA Specification Page. Disponível em: <http://www.umtsworld.com/technology/wcdma.htm> Acesso em: 05 Nov. 2006.