BRUNO ARRIEL REZENDE

ANÁLISE DE DESEMPENHO DE TERMINAIS LEVES EM LABORATÓRIOS DE INFORMÁTICA

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

LAVRAS
MINAS GERAIS – BRASIL
2008

BRUNO ARRIEL REZENDE

ANÁLISE DE DESEMPENHO DE TERMINAIS LEVES EM LABORATÓRIOS DE INFORMÁTICA

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

Área de Concentração:

Redes de Computadores

Orientador:

Prof. PhD. Luiz Henrique A. Corrreia

LAVRAS
MINAS GERAIS – BRASIL
2008

Ficha Catalográfica

Rezende, Bruno Arriel

Análise de Desempenho de Terminais Leves em Laboratórios de Informática / Bruno Arriel Rezende. Lavras – Minas Gerais, 2008. 77p : il.

Monografia de Graduação – Universidade Federal de Lavras. Departamento de Ciência da Computação.

1. Introdução. 2. Referencial Teórico. 3. Metodologia. 4. Experimento. 5. Resultados e Discussão. 6. Conclusão. I. REZENDE, B. A. II. Universidade Federal de Lavras. III. Título.

BRUNO ARRIEL REZENDE

ANÁLISE DE DESEMPENHO DE TERMINAIS LEVES EM LABORATÓRIOS DE INFORMÁTICA

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

Aprovada em 20 de Junho de 2008

_____
Prof. PhD. José Monserrat Neto

_____
Prof. PhD. André Luiz Zambalde

_____
Prof. PhD. Luiz Henrique A. Corrreia
(Orientador)

LAVRAS
MINAS GERAIS – BRASIL

“Cada segundo é tempo para mudar tudo para sempre”
(Charles Chaplin).

Agradecimentos

Agradeço primeiramente a Deus por ter me dado o Dom da vida e força para seguir nos momentos de desânimo.

Agradeço aos meus pais que mesmo nos momentos mais difíceis me apoiaram. Pelos ensinamentos e lições de vida que me proporcionaram.

Agradeço ao meu irmão e aos amigos pelos momentos de alegria e descontração proporcionados, e pelos desafios que me ajudaram a ultrapassar.

Agradeço aos professores que me ensinaram e motivaram meu desenvolvimento. Em especial ao meu orientador Luiz Henrique pela paciência.

E finalmente a todos aqueles que de alguma forma contribuíram para este trabalho

ANÁLISE DE DESEMPENHO DE TERMINAIS LEVES EM LABORATÓRIOS DE INFORMÁTICA

RESUMO

Este trabalho visa demonstrar o uso de terminais leves como alternativa ao método tradicional no qual se utiliza computadores pessoais para fazer a montagem de laboratórios de informática. O estudo constitui-se da avaliação da montagem e do desempenho do LTSP - *Linux Terminal Server Project*. Ele é uma ferramenta de código aberto que possibilita a montagem desses laboratórios utilizando computadores obsoletos como terminais aumentando assim sua vida útil e também facilitando a administração do laboratório, pois todos os programas são instalados em apenas uma máquina denominada de servidor de terminais. Os resultados mostram a viabilidade de se utilizar LTSP para montar laboratórios de informática utilizando como terminais computadores obsoletos.

Palavras-chave: Redes de Computadores, Terminais leves, Computação Centralizada.

ANALYSIS OF PERFORMANCE OF THIN CLIENT IN LABORATORIES INFORMATICS

ABSTRACT

The present work aims to demonstrate the use of thin client as an alternative to the traditional method of using of personal computers in order to assemble computer laboratories. The study is constituted of the assembling evaluation and the performance of LTSP - Linux Terminal Server Project. This last one is an open source tool which makes it possible to assemble these laboratories utilizing obsolete computer as terminals, therefore, increasing its life usefulness and also facilitating the laboratory management, since all programs are installed in just one machine called of terminal server. The results present the viability of using the LTSP to assemble computer laboratories utilizing the obsolete computers as terminals.

Keywords: *Computer networks, thin client, centralized computer.*

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

1 INTRODUÇÃO

A humanidade tem testemunhado que as tecnologias da informação e comunicação se desenvolvem a uma velocidade impressionante, trazendo benefícios a praticamente todas as áreas de atuação do ser humano. Nesse contexto o computador aparece como uma das principais ferramentas para permitir o acesso dos usuários a essas novas tecnologias.

Os computadores hoje são encontrados em todos os lugares e para seu funcionamento são utilizados *softwares*, que juntamente com *o hardware* vem se desenvolvendo a uma velocidade espantosa. Esses *softwares* exigem mais e mais poder de processamento dos computadores à medida que passam a realizar cada vez mais tarefas, que antes eram atribuídas exclusivamente aos seres humanos. Essa evolução tornou o computador uma das ferramentas essenciais para diversas atividades em vários setores das empresas, da produção industrial, da divulgação e acesso a cultura, da educação e informação.

A evolução, no entanto, vem tornando menor o tempo de vida útil dessas máquinas, surgindo então uma série de problemas sociais e ambientais:

Obsolescência das Máquinas – o que fazer com as máquinas que apesar de seu pouco tempo de uso se tornaram antigas. Já que no paradigma de computação distribuída não atendem mais aos requisitos mínimos de poder de processamento exigidos para o funcionamento dos programas mais recentes.

Custo de atualização – os parques computacionais de empresas, universidades, órgãos públicos e privados foram formados ao longo de vários anos de investimentos. Porém o surgimento de novos programas gera a necessidade de uma atualização dos equipamentos desses parques. Mas essa atualização total tem um custo muito elevado.

Impacto ambiental – ao se fazer o descarte dessas máquinas deve-se levar em consideração o impacto ambiental provocado pelos materiais utilizados na sua composição – basicamente plástico e metais pesados – serem altamente tóxicos e levarem muitos anos para se decompor.

Exclusão digital – como prover acesso aos benefícios trazidos por essas novas tecnologias a uma parcela considerável da população, que devido a sua condição financeira não pode adquirir um computador e tampouco acompanhar a constante necessidade de atualização dos mesmo.

Outro problema que pode ser levantado é a dificuldade que empresas, universidades e outras organizações têm para gerenciar um número crescente de computadores distribuídos em seus vários setores. Como cada computador, na computação distribuída, possui seu próprio sistema operacional e sua unidade de armazenamento o serviço de gerenciamento e manutenção é repetido para cada máquina existente em uma organização.

Uma possibilidade de se prolongar a vida útil dessas máquinas antigas e reduzir os gastos com gerenciamento e manutenção é utilizá-las como terminais leves. Como tais, essas máquinas são ligadas por uma rede a outra máquina chamada de servidor que possuirá um poder de processamento maior, a qual ficará encarregada de realizar todo o processamento. Dessa forma, torna-se possível utilizar programas atuais através dessas máquinas obsoletas, pois as mesmas possuirão, na configuração mais básica, a função de comunicação com o servidor e de exibir os resultados do processamento do servidor ao usuário.

Além disso, como a computação fica centralizada em uma única máquina, o trabalho de gerenciamento passa a ser concentrado nesse computador, eliminando assim o retrabalho de configuração, *backup e* instalação e remoção de *software*.

1.1 Motivação

Nas empresas o acesso às tecnologias da informação e comunicação através dos computadores é muitas vezes essencial para a sobrevivência dessas organizações. Esse acesso também facilita a vida cotidiana através de novas formas de comunicação, facilidade e agilidade em se obter informações diversas, como canal de ligação entre o cidadão e órgãos e setores governamentais.

Nas escolas e universidades o acesso tanto de estudantes quanto de professores aos computadores tem se tornado essencial como ferramenta de auxílio a pesquisa contribuindo assim para uma boa formação.

Porém, esse acesso aos computadores para grande parte das pessoas ainda é reduzido, pois há um pequeno número de máquinas com poder de processamento suficiente para suportar os requisitos dos *softwares* atuais. Isso se deve aos elevados custos de investimentos necessários para se adquirirem esses equipamentos e à falta de políticas que destinem verbas para mantê-los atualizados. Além disso, há um grande número de computadores obsoletos acumulados ao longo dos anos por grandes organizações tais como escolas, universidades e empresas.

Uma idéia natural e muito interessante para aumentar a vida útil dessas máquinas obsoletas seria utilizá-las como terminais leves. Entretanto, a falta de parâmetros para se determinar as configurações mínimas para a utilização de terminais leves, seus custos de implantação, seu desempenho e fatores limitantes, faz com que essa solução não seja devidamente considerada e descartada a princípio perdendo-se assim uma possibilidade de se ampliar os pontos de acesso às novas tecnologias da informação e comunicação.

Este trabalho apresenta uma avaliação da utilização de terminais leves em laboratórios de informática.

1.2 Objetivos

O principal objetivo desse trabalho é montar e avaliar o desempenho de um laboratório baseado no conceito de terminais leves, utilizando o LTSP (*Linux Terminal Server Project*). Neste trabalho serão analisados aspectos como a facilidade de instalação e configuração, e o real funcionamento da solução. Para isso foram analisadas métricas de desempenho como consumo de processamento e memória, largura da banda utilizada, número de processos ativos, que servirão como base para se avaliar os reais benefícios da utilização do LTSP, quais os requisitos mínimos, custos de implantação e manutenção, fatores limitantes.

A montagem desse laboratório possibilita aumentar o número de pontos de acesso dos computadores, reaproveitando máquinas já existentes que seriam descartadas.

1.3 Organização do Trabalho

Este trabalho apresenta a implantação de terminais leves em laboratórios de informática e está organizado da forma descrita a seguir:

No capítulo 1 é apresentada a construção do trabalho realizado, descrevendo o seu contexto, sua motivação e seus objetivos.

No capítulo 2 descreve-se o referencial teórico conceituando computação distribuída e centralizada. São apresentados quais são as soluções existentes para se aproveitar a capacidade ociosa dos computadores atuais. E por fim é realizado um aprofundamento sobre o projeto de servidores de terminais Linux destacando os programa e protocolos utilizados como base para seu funcionamento.

O capítulo 3 apresenta a metodologia de desenvolvimento do trabalho, o tipo de pesquisa, os procedimentos metodológicos utilizados, a descrição dos experimentos realizados e os passos para a montagem do laboratório.

No capítulo 4 é discutida a praticidade da montagem de laboratórios que utilizam a solução proposta, sendo também demonstrados o resultado da avaliação das métricas de desempenho colhidos durante o desenvolvimento do trabalho.

E por último, o capítulo 5 apresenta as conclusões sobre o desenvolvimento do trabalho e propõe novos trabalhos.

2 REFERENCIAL TEÓRICO

Ao longo dos últimos anos a computação tem utilizado dois paradigmas principais: a computação centralizada e a computação distribuída. No início da popularização da computação surgiram os sistemas de tempo compartilhado, em que muitos usuários compartilhavam um grande computador central através de terminais. Em seguida vieram os computadores pessoais - PC e as redes de alta velocidade e cada usuário passou a ter um computador independente em sua mesa, que compartilhava informações e dispositivos através das redes (Tanenbaum, 2001).

2.1 Computação distribuída

A computação distribuída surgiu na década de oitenta decorrente de avanços da eletrônica, nas tecnologias de desenvolvimento de microprocessadores, e da necessidade de compartilhamento de recursos de armazenamento e comunicação que deram origem às redes de computadores. Os microprocessadores surgidos nessa época tinham o mesmo poder de processamento de um *mainframe*, que custava dezenas de milhões de dólares, porém custando apenas pouco mais que mil dólares (Tanenbaum, 1995).

Na computação distribuída o usuário intererage com o sistema na mesma máquina em que o processamento é realizado e quando é necessário ele utiliza uma rede de computadores para acessar outra máquina em busca das informações desejadas (Souza Filho, 2003).

Um sistema de computação distribuído, possui no mínimo um dos seguintes requisitos(Ghosh, 2007):

Múltiplos processos – mais de um processo seqüencial, seja ele de sistema ou de usuário, pode ser executado cada um deles possuindo um caminho e um controle independentes.

Comunicação interprocessos – um processo pode se comunicar com outro por mensagens através de um canal em um período finito de tempo.

Separação dos espaços de endereçamento – Os processos não devem possuir espaços de memória compartilhados.

Metas coletivas – os processos devem poder interagir uns com os outros a fim de se obter um resultado comum. Por exemplo, se um processo calcula a largura de um quadrado e outro calcula a altura, os dois processos podem interagir para se obter a área do quadrado.

2.2 Computação centralizada

No início da década de sessenta, pesquisadores do MIT e de outros lugares desenvolveram sistemas operacionais que permitiam que vários programadores se comunicassem diretamente com um computador central através de terminais remotos conectados por linhas telefônicas ou por redes seriais.

Esses sistemas eram denominados sistemas de tempo compartilhado. Neles cada usuário possuía uma fatia do tempo do processador para que seus processos fossem executados, possibilitando assim um melhor aproveitamento da capacidade de processamento dos grandes *mainframes* existentes na época (Tanenbaum, 2007).

Esse paradigma está voltando ao foco das pesquisas, como forma de se ter um melhor aproveitamento da capacidade ociosa de processamento dos microprocessadores atuais. O crescimento do poder de processamento foi tão grande nas últimas décadas que pequenos computadores de hoje são capazes de executar várias tarefas concorrentemente para vários usuários (Oliveira et. al., 2006).

Existem várias abordagens para o aproveitamento dessa capacidade ociosa dos processadores, a seguir serão demonstradas algumas dessas abordagens, porém nesse trabalho será dada ênfase à utilização dessas máquinas como servidores de terminais leves.

2.3 Aproveitamento da capacidade ociosa dos processadores

As soluções mais conhecidas e estudadas sobre como aproveitar a capacidade ociosa do *hardware* atual são máquinas virtuais, multiterminais descritos em detalhes a seguir e terminais leves que é o foco desse trabalho. Os terminais leves são vistos em detalhes na seção 2.4.

Máquinas Virtuais.

Uma das soluções que mais tem despertado interesse de pesquisadores para o aproveitamento da capacidade ociosa dos computadores é a utilização de máquinas virtuais (*Virtual Machine* - VM). Esta tecnologia permite se fazer uma abstração do *hardware* real através de uma representação desse por uma máquina virtual. Com a utilização de máquinas virtuais é possível executar diversos sistemas operacionais sobre um mesmo equipamento de *hardware* real (Koslovski et. al., 2007).

Uma VM é um ambiente criado por um monitor de máquinas virtuais (*Virtual Machine Monitor* – VMM). É o VMM que permite criar e controlar uma ou mais máquinas virtuais abstraindo o *hardware* real subjacente. Dessa forma, cada VM pode fornecer as facilidades necessárias para uma aplicação ou "sistema convidado" que "acredita" ser executado por um *hardware* real (Laureano et. al., 2003).

Existem duas abordagens para a construção de sistemas virtuais, as quais se diferenciam pela localização do monitor de máquinas virtuais:

Virtualização clássica - modelo em que o MMV se encontra entre o *hardware* e os sistemas convidados, ilustrado pela Figura 2.1 (Laureano et. al., 2003). Essa abordagem faz com que o MMV tenha o nível mais alto de privilégio de execução das instruções. Nesse modelo, todas as chamadas de sistema são interceptadas pelo MMV para posteriormente serem repassadas ao *hardware*.

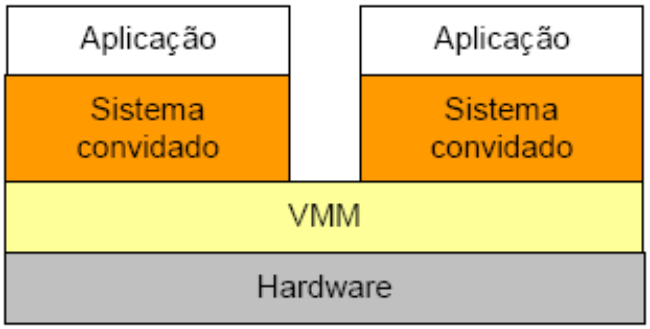

Figura 2.1: MMV entre o hardware e o Sistema Convidado.

Um exemplo de utilização do modelo de virtualização clássico é a ferramenta de código aberto desenvolvida pela Universidade de Cambridge denominada *Xen*. A existência de limitações nas implementações utilizando a arquitetura *Intel IA-32* permite aos sistemas operacionais virtualizados executarem instruções com maior privilégio. Essa limitação faz com que o MMV *Xen* utilize a técnica de paravirtualização para contornar o problema impedindo assim que os sistemas operacionais virtualizados acessem o *hardware* diretamente. Porém a utilização dessa técnica requer pequenas modificações nos sistemas operacionais a serem virtualizados.

Os sistemas operacionais atualmente adaptados a virtualização utilizando o *Xen* são: *Linux, NetBSD, FreeBSD* e *Windows XP*.

Virtualização hospedada – nesse modelo o MMV é implementado como um processo de um sistema operacional real subjacente denominado *sistema anfitrião* sendo ilustrado pela Figura 2.2 (Laureano et. al., 2003). Por ser executado sobre um outro sistema operacional o MMV utiliza o nível mais baixo de privilégios, utilizando tradução binária em tempo de execução para adaptar as instruções que necessitam de um nível de privilégio maior.

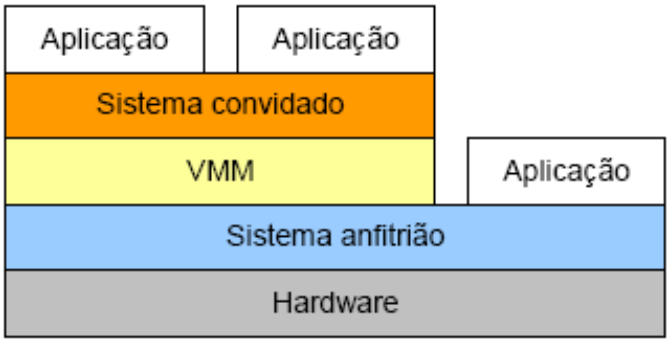

Figura 2.2: MMV sobre um sistema anfitrião.

Um exemplo de ferramenta de virtualização que utiliza o modelo hospedado é um produto desenvolvido pela empresa *VMware Inc*. denominado *VMware*. A ferramenta é dividida em duas partes a *VMapp* é a máquina virtual que executada no espaço de usuário do sistema operacional hospedeiro e a *VMDriver* é um *software* que executa junto ao sistema operacional hospedeiro e processa as requisições feitas pelo *VMapp*.

Sempre que o sistema hospedado faz um acesso a algum dispositivo virtual, o MMV salva o contexto da MV e muda para o seu contexto, faz o acesso ao dispositivo real e retorna ao contexto da máquina virtual assim que é completada a requisição. Utilizando essa arquitetura, a *VMware* pode ser utilizada sem necessidade de alterações nos códigos da maioria dos sistemas operacionais existentes no mercado.

Utiliza-se máquinas virtuais como ferramenta para diversos propósitos os quais podemos citar (Koslovski, 2006 e Laureano et. al., 2003):

Apoio ao ensino e a pesquisa – permitindo que se faça simulações de falhas, configurações virtuais de *hardware* e *software*, execução simultânea de diversos sistemas operacionais sobre um mesmo *hardware*.

Consolidação de servidores – permitindo a criação de diversas máquinas com configurações específicas para aplicações diferentes sobre um mesmo *hardware*. Garantindo a confiabilidade na medida que possibilita a migração ou replicação de uma máquina virtual de

um *hardware* real para outro sem a necessidade de parar o serviço oferecido pela máquina virtual.

Garantir portabilidade – permitindo que aplicações em softwares legados possam ser executados em máquinas novas através de VM que simulam o *hardware* e *software* originais dessa aplicação.

Multiterminais.

A computação pessoal passou a ser o modelo mais difundido a partir da década de oitenta devido ao seu menor custo em relação a computação centralizada. No modelo de computação distribuída ou pessoal o usuário passa a interagir com o sistema operacional - SO no mesmo equipamento onde o processamento é realizado. Entretanto, o SO desses equipamentos manteve a capacidade de uso simultâneo por muitos usuários por meio de outros equipamentos e PC conectados através de uma rede de computadores, características da computação centralizada, e dessa forma é possível aproveitar o tempo ocioso do processador.

Uma outra forma de aproveitar esse tempo ocioso do processador é montar um multiterminal. Ele é um computador pessoal em que são conectados vários conjuntos de monitor, teclado e mouse de forma a permitir a utilização simultânea e independente desse PC por vários usuários em um mesmo local (Oliveira et. al., 2006).

A montagem de um multiterminal exige algumas modificações no *hardware* de um PC normal para que esse possibilite a conexão de mais de um conjunto de monitor, teclado e mouse. Essa alteração é realizada adicionando-se mais placas de vídeo como mostrado na Figura 2.3 (Kawashima, 2006) e utilizando adaptadores USB que facilitam a conexão dos fones de ouvido, microfone, teclado e mouse como apresentado na Figura 2.4 (Kawashima, 2006).

Além das modificações no *hardware,* também é necessário que se façam algumas mudanças nos sistemas operacionais construídos com base no modelo de computação pessoal, para que os mesmos possam dar suporte à utilização de multiterminais (Zanoni et. al., 2008). No centro de computação científica e software livre – C3SL da Universidade Federal do Paraná - UFPR há um projeto de pesquisa para o desenvolvimento da tecnologia de multiterminais para o sistema operacional GNU/Linux (C3SL, 2008).

Figura 2.3: Placas de vídeo PCI adicionais (1, 2, 3, 4).


Figura 2.4: Adaptador USB/PS2 e Placa de Som usb.

As implementações do modelo de multiterminais já desenvolvidas ou em desenvolvimento no C3SL foram realizadas com o SO GNU/Linux e podem ser portadas facilmente para outro SO. Há dois grupos de soluções para resolver o problema de atribuir um conjunto de dispositivos de entrada (mouse, teclado, câmera) ao dispositivo de saída correspondente (monitor) comentados a seguir (Zanoni et. al., 2008):

Utilização de várias instâncias do servidor Xorg – utilizando várias instâncias do servidor X (Programa que provê a interface gráfica em SO GNU/Linux) tem-se uma solução com um melhor desempenho. No entanto, apresenta pouca estabilidade funcionando apenas para um conjunto reduzido de *harware*.

Utilização de Servidores X aninhados – com servidores X aninhados resolve-se o problema da instabilidade, porém há uma piora no desempenho. A primeira solução

multiterminal que utilizava servidores X aninhados foi criada no final de 2005 e utilizava um novo *driver* de entrada para o servidor *Xnest* (alternativa ao servidor X) o qual recebia os eventos de entrada diretamente da interface *evdev* do *Kernel* ao invés de recebê-los do servidor base. Além disso, um cursor de *mouse* próprio foi implementado, pois o *Xnest* original utiliza o cursor do servidor base, que geralmente é um servidor X.

Após essa implementação várias outras foram criadas com a finalidade de se aumentar o desempenho e a facilidade de manutenção do código para os multiterminais. Os multiterminais são indicados para serem utilizados em laboratórios, escolas, locais públicos com acesso à internet, ou seja, locais nos quais o computador serve com ferramenta no auxílio a atividades administrativas que não requerem uso intenso de CPU (Multiseat, 2008).

Um caso de uso interessante de ser mencionado é o projeto de inclusão digital do governo do Paraná, denominado Paraná Digital, onde 11 mil multiterminais estão sendo instalados em todas as escolas públicas estaduais do Paraná, totalizando 44 mil pontos de trabalho (C3SL, 2008).

2.4 Terminais leves ou *Thin Clients*.

Um terminal é um tipo especial de computador que não possui CPU, nem armazenamento próprio; ele é apenas um dispositivo de entrada e saída que age como uma janela que provê acesso a outro computador que se encontra em algum outro local.

Esse conceito se relaciona à configuração de minicomputadores e *mainframes* que realizam operações com grandes quantidades de informações de entrada e saída (Norton, 1997).

Um outro conceito utilizado atualmente, é geralmente aplicado a computadores pessoais usados como clientes de um servidor de aplicativos. Em geral, rodando um servidor de terminais *Linux* - LTSP, *Windows Terminal Services* – WTS ou outra solução em que o servidor fica encarregado de fazer todo processamento e armazenamento de dados, enquanto os clientes apenas exibem imagens na tela e enviam de volta os movimentos do *mouse* e caracteres digitados. Pode-se dizer que *Thin Client* ou terminais leves são versões modernizadas dos antigos "terminais burros". A diferença é que agora ao invés de aplicativos de texto simples, são usados aplicativos gráficos quase sem limitações e com suporte a placas de som, impressoras e *drives* de dispositivos locais nos terminais (Morimoto, 2008a).

O conceito de terminal leve consiste em um sistema computacional formado por clientes (terminais) e um ou mais servidores que utilizam um protocolo sobre uma rede de computadores para exibir os resultados do processamento do servidor nas estações clientes (Lai & Nieh, 2002).

Existem diversas implementações do modelo de terminais leves entre elas podemos destacar: O *Windows Terminal Service ou WTS*.

Esta solução é de propriedade da Microsoft. Os conceitos de funcionamento do *Windows Terminal Service* são os mesmo utilizados no LTSP, solução que será estudada neste trabalho e explicada em detalhes na seção 2.5. Porém o WTS utiliza protocolos de propriedade da Microsoft, tendo como principal protocolo utilizado o *Remote Desktop Protocol* - RDP.

Para utilização de terminais ligados a um WTS, é necessário a compra de licenças adicionais as do servidor. Há duas formas de licenciamento para os clientes WTS: Numa a licença é definida pelo número de máquinas que serão utilizadas. Na outra é definida pelo número de usuários que podem se conectar de qualquer máquina que tem acesso ao servidor WTS (Morimoto, 2008b).

2.5 O Projeto de Servidores de Terminais Linux

LTSP é a sigla para *Linux Terminal Server Project* ou projeto de servidores de terminais leves*,* um projeto de código aberto, criado e mantido por James McQuillan nos Estados Unidos em 1996. Hoje o projeto conta com a contribuição de vários desenvolvedores ao redor do mundo, inclusive do Brasil. O projeto visa reunir um conjunto de ferramentas administrativas para facilitar a utilização de estações de trabalho de baixo custo, máquinas obsoletas ou *thin client*, como terminais de caracteres ou gráficos de um servidor GNU/Linux (Ferreti, 2004).

Nas soluções baseadas em terminais leves o servidor fica com a parte pesada do trabalho, que é executar os programas, armazenar os dados e enviar para os clientes as instruções para montar as janelas que serão exibidas. Os clientes montam as janelas e enviam os movimentos do *mouse* e as teclas digitadas no teclado (Morimoto, 2006). A Figura 2.5 ilustra a estrutura básica para o funcionamento de uma solução baseada em LTSP.

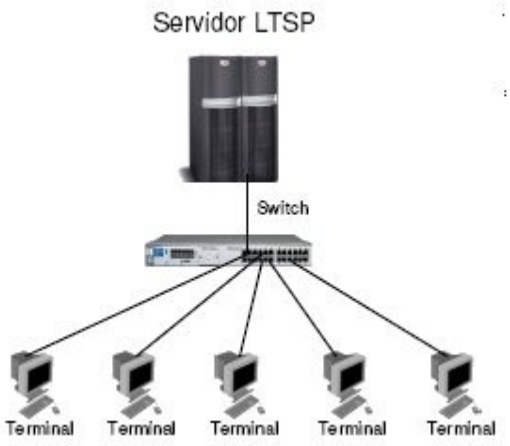

Figura 2.5: Estrutura básica para o funcionamento de uma solução baseada em LTSP.

fonte: (Ferreti, 2004)

O LTSP é um projeto de software livre sob a licença GPLv2 que possui a característica de prover acesso via terminais com quase todas as funcionalidades existentes em um sistema operacional GNU/Linux. Se uma aplicação pode ser executada sobre a plataforma GNU/Linux, certamente poderá ser utilizada via terminais através do LTSP.

Uma outra característica de destaque é a de facilitar as operações de manutenção, instalação e gerenciamento de ambientes formados por terminais leves, pois todo o processamento e armazenamento se encontra no servidor bastando realizar uma modificação no servidor para que esta tenha efeito em todos os terminais clientes (Martins, 2004).

Outro ponto positivo é observado em caso de falhas ou quebra dos terminais, em que a substituição do terminal com problemas é necessária. Nesse caso não será preciso realizar qualquer configuração ou instalação no novo terminal bastando ligá-lo no lugar do que se encontra com problemas.

O LTSP é uma solução extremamente promissora para a criação de terminais leves com o sistema operacional GNU/Linux. Ele utiliza uma combinação de serviços fornecidos pelos protocolos *DHCP, TFTP* e *XDMCP* para permitir que as estações não apenas rodem aplicativos instalados no servidor, mas realmente dêem *boot* via rede, baixando todos os softwares de que precisam diretamente do servidor. Não é preciso dispositivos de armazenamento nas estações, apenas a memória *ROM* da placa de rede ou então um disquete de *boot* (Morimoto, 2006).

Figura 2.6: *Thin Client.*
Fonte:(TecnoWorld, 2008)

Podemos ter as seguintes opções de terminais gráficos para um sistema baseado em LTSP.

- Utilização de *Thin Client, como os da Figura 2.6*, que são máquinas projetadas especificamente para funcionar como terminais gráficos, sendo assim possuem como características baixo poder de processamento, baixo consumo de energia, tamanho e custo de manutenção reduzidos;

- Estações de trabalho montadas com o propósito de serem servidores gráficos, por este motivo geralmente são máquinas capazes de realizarem o *boot* via rede, desprovidas de dispositivos de armazenamento e com poder de processamento reduzido;

- Estações de trabalho obsoletas, entende-se por obsoletas aquelas estações que possuem um poder de processamento insuficiente para executar os *softwares* atuais no modelo tradicional devido ao aumento das necessidades de processamento ocasionados pela evolução desses *softwares.*

É necessária a instalação e configuração dos serviços, listados abaixo, no servidor para dar suporte ao funcionamento dos terminais leves.

Servidor de DHCP – é encarregado de atribuir os endereços de IP para cada terminal juntamente com a localização do *kernel*;

Servidor de TFTP – é encarregado de baixar o *Kernel,* ou núcleo, para os terminais;

Servidor de NFS – faz o mapeamento dos arquivos utilizados nas estações;

Servidor X, XDMCP e Gerenciadores de janelas – responsáveis por disponibilizar o ambiente gráfico para os terminais.

Servidor SSH – responsável pela autenticação e tunelamento dos comandos enviados ao servidor, serviço utilizado a partir da versão 5.0 do LTSP;

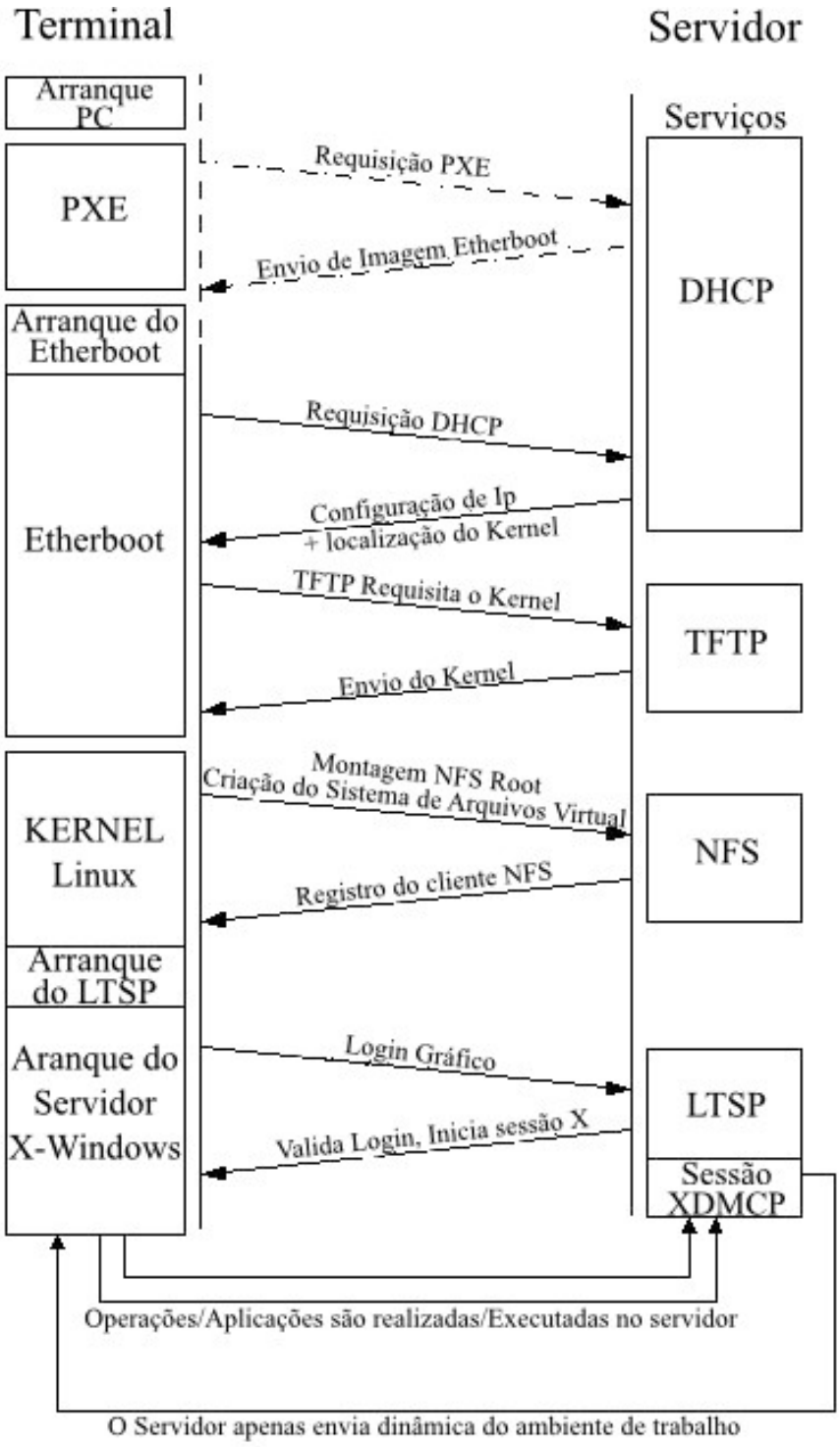

Figura 2.7: Esquema do processo de boot de um terminal.

Nas subseções a seguir é apresentado um detalhamento sobre o funcionamento desses serviços assim como a ordem de sua execução no processo de *boot* de um terminal. Como ilustrado na Figura 2.7

2.5.1 Protocolo DHCP – *Dynamic Host Configuration Protocol*

O DHCP é um protocolo de serviço localizado na camada de aplicação da arquitetura TCP/IP que oferece configuração dinâmica de terminais, com concessão de endereços IP de *host* e outros parâmetros de configuração para clientes de rede. Este protocolo é o sucessor do *BOOTP* que, embora mais simples, tornou-se limitado para as exigências atuais. O DHCP surgiu como padrão em Outubro de 1993. A RFC 2131 contém as especificações mais atuais de Março de 1997 (Droms, 1997). O último padrão para a especificação do DHCP sobre IPv6 (DHCPv6) foi publicado em Julho de 2003 na RFC 3315 (Droms et. al., 2003).

O DHCP usa um modelo cliente-servidor, no qual o servidor DHCP mantém o gerenciamento centralizado dos endereços de IP usados em uma rede.

A implementação, tanto do cliente como do servidor, realizada pelo *Internet Software Consortium* ISC, está presente nas distribuições mais populares como Red Hat, Suse, Debian e seus derivados (Nemeth et. al., 2002).

A Figura 2.8 apresenta o funcionamento do DHCP , que pode ser descrito como:

1. Um cliente envia um pacote em difusão (destinado a todas as máquinas) com um pedido DHCP;
2. Os servidores DHCP que capturarem este pacote irão verificar se o pedido atende aos critérios especificados nos arquivos de configuração.
3. Caso atenda, o servidor responderá com uma mensagem contendo as informações previstas no arquivo de configuração.

O papel do DHCP no processo de inicialização de um terminal no LTSP

Ao ser ligado o terminal passa por um processo de auto teste. Terminado esse processo ele executará o *Etherboot*, que enviará uma mensagem em difusão para a rede contendo o endereço físico da placa de rede (MAC).

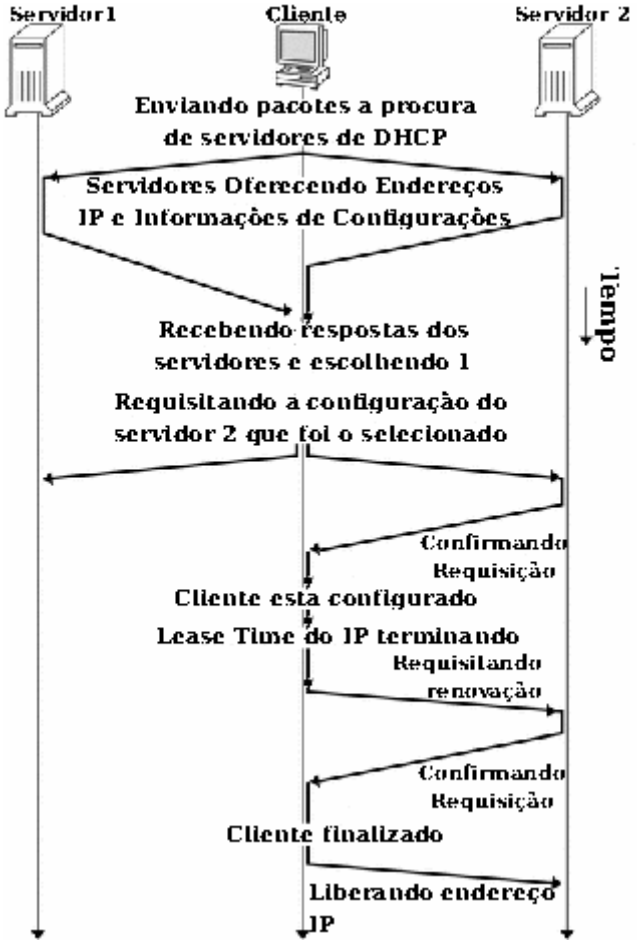

Figura 2.8: Funcionamento do DHCP.

O servidor DHCP que se encontra nessa rede recebe esta mensagem verifica se o endereço físico está descrito no arquivo *dhcpd.conf* ou se há uma resposta padrão.

Se uma das duas alternativas for atendida o servidor retorna uma mensagem contendo: um endereço IP, a máscara de rede, o caminho do *kernel*, o caminho do sistema de arquivos raiz e outros parâmetros opcionais de acordo com o que está especificado no arquivo *dhcp.conf* para aquele endereço físico.

Se não há referência ao endereço físico e também não há opção padrão a mensagem é descartada (Martins, 2004 e ISC, 2008).

2.5.2 Protocolo TFTP – *Trivial File Transfer Protocol*

O *Trivial File Transfer Protocol* ou protocolo trivial de transferência de arquivos é um protocolo que se encontra hierarquicamente acima do protocolo UDP, o TFTP foi projetado para ser um protocolo de transferência de arquivos, simples, pequeno e de fácil implementação que apenas lê ou escreve arquivos (ou correspondências) de um servidor remoto (Javvin, 2005).

O tamanho reduzido da implementação do protocolo TFTP é importante para muitas aplicações, como por exemplo, em *thin clients* no qual o programa baseado nesse protocolo pode ser gravado em ROM e utilizado para obter uma imagem inicial da memória quando a máquina é ligada. Esse programa gravado em ROM recebe o nome de *bootstrap*.

Uma vantagem em se utilizar o TFTP é que ele permite que o código *bootstrap* utilize os mesmos protocolos básicos do conjunto TCP/IP que o sistema operacional irá utilizar quando for totalmente inicializado (Comer, 2006).

Estrutura e funcionamento do TFTP:

O cliente inicializa a comunicação com o servidor TFTP enviando para a porta 69 um pacote de requisição, que pode ser de leitura (o arquivo será transferido do servidor para o cliente) ou de escrita (o arquivo será transferido do cliente para o servidor). Se essa requisição for aceita pelo servidor, a requisição também terá a função de estabelecer uma conexão entre o cliente e servidor, pois apenas nesse pacote há a informação do nome do arquivo e o modo de codificação que será utilizado na transferência do mesmo.

O envio do arquivo é realizado em blocos com tamanho fixado em 512 *bytes* e numerados consecutivamente inicializando-se pelo bloco número um.

Cada pacote de dados é definido da seguinte forma:

2 Bytes	2 Bytes	N Bytes
Código de Operação	Número do bloco	Dados

Figura 2.9: Estrutura do pacote de dados do TFTP.

Cada pacote de confirmação é definido da seguinte forma:

2 Bytes	2 Bytes
Código de Operação	Número do bloco

Figura 2.10: Estrutura do pacote de confirmação do TFTP.

No pacote de confirmação o número do bloco é uma referência ao número do bloco do pacote de dados que está sendo confirmado. No TFTP o bloco $n+1$ só é transmitido após a chegada da confirmação do recebimento do bloco n.

Tanto do lado do servidor quanto do cliente é implementado um contador de tempo limite para a transmissão dos pacotes de dados ou de confirmação.

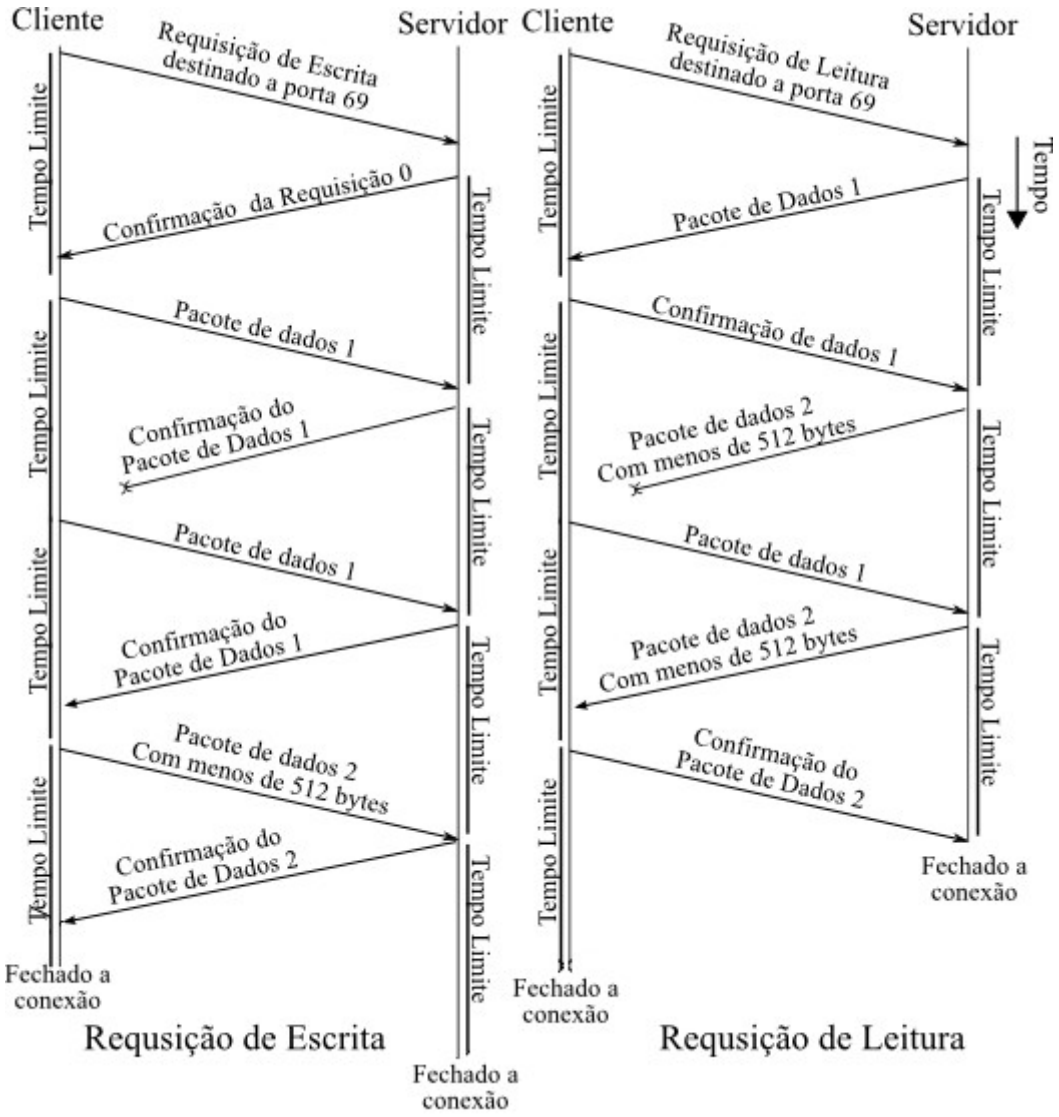

Figura 2.11: Funcionamento do TFTP.

Se o lado que envia os dados chegar a seu limite de tempo, sem receber um pacote de confirmação referente ao último pacote de dados enviado, ele reenviará o último pacote de dados. Se o lado responsável pela confirmação chegar a seu limite de tempo sem receber o próximo pacote de dados, ele retransmite o último pacote de confirmação. Com os dois lados

fazendo retransmissão, há uma certeza que a transferência não irá falhar após um simples pacote perdido.

A transmissão é finalizada quando é transmitido um pacote que informa um erro ou o pacote tem tamanho menor que 512 bytes. O protocolo TFTP se encontra atualmente na versão 2 e está especificado na RFC 1350 (Sollins, 1992).

A Figura 2.11 ilustra o funcionamento do TFTP, para uma requisição de escrita e para uma requisição de leitura com perda pacotes durante a transmissão.

O papel do TFTP no processo de inicialização de um terminal no LTSP.

Após receber a configuração da placa de rede vinda do servidor DHCP, o terminal passa a executar o programa TFTP que será encarregado de baixar do servidor para um local específico da memória do terminal o *kernel*, passando o controle do terminal para ele.

O *kernel* carregado pelo TFTP é responsável por fazer a detecção dos periféricos, carregar o módulo e as configurações finais da placa de rede (Martins, 2004 e LTSP.org, 2008).

2.5.3 Protocolo NFS – *Network File System*

Criado pela *Sun Microsystems* Incorporated em 1984, o sistema de arquivos em rede ou NFS tornou-se um padrão da IETF para acesso de arquivos compartilhados transparente e integrado (Tanenbaum, 2001). A Figura 2.12 é a ilustração de como o NFS é embutido no sistema operacional.

Ao se abrir um arquivo, seja para leitura ou escrita, faz-se uma requisição ao sistema operacional. O mecanismo de acesso a arquivos aceita a requisição e automaticamente a transfere para o software do sistema de arquivos local ou para o cliente do NFS, de acordo com a localização do arquivo. No caso da requisição ao processo cliente NFS, esse repassa a requisição ao servidor NFS, através do protocolo NFS (Comer, 2006). A arquitetura do NFS é constituída de três componentes básicos: o protocolo NFS, um mecanismo de chamada remoto (*Remote Procedure Call* - RPC) e uma representação de dados externa de finalidade geral (*eXternal Data Representation* – XDR). O RPC é o responsável por fazer a comunicação entre o cliente e o servidor, já o XDR faz a conversão da representação dos dados possibilitando que máquinas heterogêneas possam se comunicar.

Para ilustrar o funcionamento do protocolo NFS e será mostrado um exemplo de sua implementação para a plataforma Unix a seguir (Tykhomyrov & Tonkonog, 2007):

O NFS é o serviço mais inteligente que usa o procedimento de chamada remota (RPC). Como exemplo do funcionamento, vamos dizer que há um servidor chamado Yamaha, contendo todos os arquivos $HOME no diretório /home. A partir de sua máquina local, chamada de Honda é digitado o seguinte comando:

```
mount -t nfs yamaha: /home /home
```

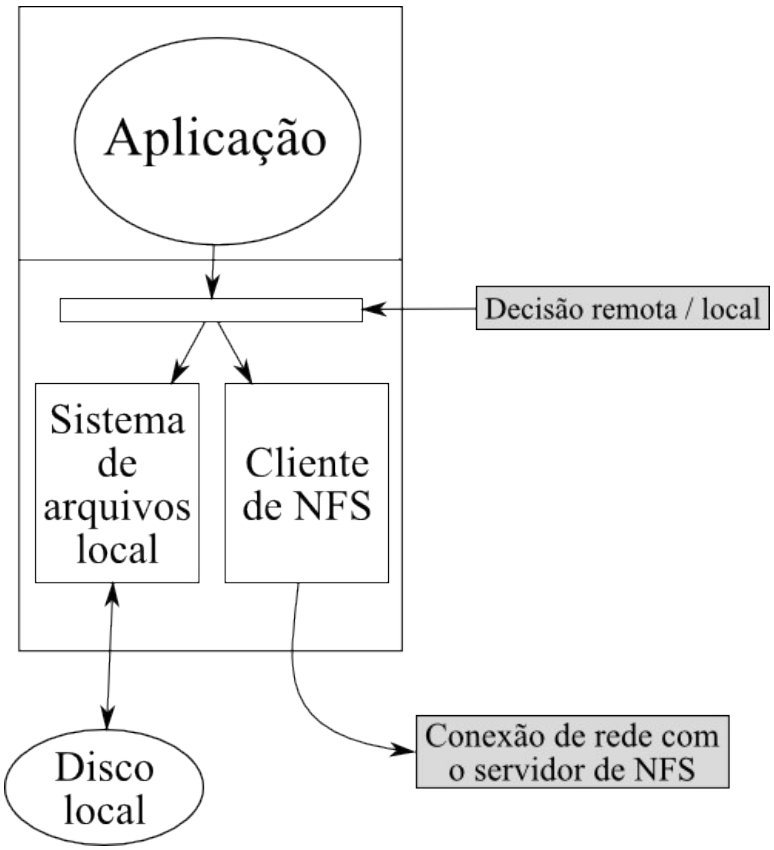

Figura 2.12: Implementação do NFS em um S.O.

De acordo com esse comando, o *mount* irá tentar se conectar ao processo *rpc.mountd* no servidor Yamaha via RPC. O servidor irá checar se a requisição é permitida para *mount* /home vindo de Honda e retornará um arquivo *handle* que, por sua vez, será usado por todos os pedidos de arquivos abaixo de /home. Se a requisição não for permitida, será visto uma mensagem de erro correspondente. Porém se o arquivo *handle* é retornado corretamente, a negociação com Yamaha foi realizada corretamente.

Quando um usuário em Honda tentar acessar um arquivo NFS, o *kernel* fará uma chamada RPC para o processo NFS, usualmente *rpc.nfsd*, passando como parâmetros o nome do arquivo, ID do usuário e do grupo. Assim o servidor Yamaha, pode impedir ou liberar o acesso antes de enviar o *handle,* de acordo com as permissões de usuários configuradas no servidor.

O papel do NFS no LTSP

As estações, em geral, não possuem dispositivos de armazenamento local então, o LTSP utiliza o NFS como sistema de arquivos.

O NFS substitui o sistema de arquivos carregado pelo TFTP, inclusive montando uma nova raiz para o sistema com permissões somente de leitura, esse diretório raiz geralmente se encontra no Servidor no diretório */opt/ltsp/i386* e é compartilhado por todas as estações. O NFS também é o responsável por armazenar os arquivos das estações no servidor (LTSP.org, 2008).

2.5.4 A Interface Gráfica

A arquitetura que permite a utilização de interfaces gráficas no Linux possui três componentes básicos:

- X *Window System* também denominado de X *Window* ou X.
- X *Display Manager Control Protocol* ou XDMCP.
- E um gerenciador de janela, os mais comuns são: GDM e KDM.

Cada um desses componentes serão respectivamente explicados a seguir.

X Window System

O X *Window* é o sistema gráfico utilizado nos S.O. UNIX e seus derivados como as diversas distribuições Linux. O X *Window* utiliza-se de uma arquitetura cliente/servidor para exibir janelas gráficas. Para ilustrar o funcionamento do cliente e do servidor no X *Window* será considerado que estão funcionando em máquinas distintas, porém isso não é necessário. Tanto o cliente quanto o servidor podem funcionar na mesma máquina.

O Servidor X *Window* é um programa executado no *desktop* do usuário, ou terminal, que permite a interação do usuário com os dispositivos de entrada e saída. É o X que exibe textos ou figuras geométricas na tela, responde a requisições realizadas pelo cliente, desenha as janelas, gerencia a entrada realizada pelo *mouse* ou pelo teclado. O servidor X também define qual janela está selecionada e para qual cliente será enviada a entrada realizada pelo usuário.

O Cliente X *Window* é um programa executado em máquina remota que geralmente possui um maior poder de processamento, comumente chamado de "servidor". As atribuições do cliente X *Window* são enviar pedidos e receber eventos do servidor, assim como receber mensagens de erro.

O conceito de cliente e servidor é aplicado à visão do programa e não do usuário, por esse motivo pode parecer estranho à primeira vista. Mas como o principal trabalho do servidor é exibir o retorno do processamento ao usuário através de uma interface gráfica de saída, faz sentido que ele fique mais próximo do usuário.

As principais mensagens de comunicação entre o servidor e o cliente do protocolo X são:

Requisições – Clientes X fazem requisições ao servidor X para ativar uma determinada ação. Por exemplo: Criar Janela. Para melhorar o desempenho, o cliente X normalmente não espera por uma resposta. O pedido é passado para a camada de rede entregar. Requisições X são quaisquer pedidos múltiplos de 4 bytes.

Respostas – O servidor X irá responder a solicitações de clientes X para as quais uma resposta é exigida. Como já mencionado, nem todos os pedidos exigem uma resposta. Repostas X são quaisquer respostas múltiplas, de 4 bytes, com um mínimo de 32 bytes.

Eventos – O servidor X enviará ao cliente X um evento que a aplicação está esperando. Isto pode incluir entrada de teclado ou Mouse. Para minimizar o tráfego de rede, só eventos esperados são enviados para os Clientes X. Eventos X possuem tamanho de 32 bytes .

Erros – O servidor X apresentará um relatório de erros em pedidos ao cliente X. Erros são como um evento, mas são tratados de forma diferente. Erros no X são do mesmo tamanho que os seus eventos para simplificar o manuseio. Eles são enviados para a rotina de manipulação de erros do cliente X.

Como o X não é uma interface gráfica de usuário (GUI) completa, é necessário a utilização de outros componentes para complementá-lo. Isso é uma decisão de projeto que visa dar maior portabilidade e flexibilidade ao X (Javvin, 2005 e Tanenbaum, 2001).

XDMCP

O *X Display Manager Control Protocol* (XDMCP) ou Protocolo de Controle e Gerenciamento de Janelas é projetado para fornecer um acesso autenticado a um serviço de gerenciamento de janelas, a janelas remotas. Um servidor de rede chamado de Gerenciador de Janela usará o XDMCP para se comunicar com a janela e negociar a inicialização das sessões X.

O protocolo disponibiliza serviços que possibilitam a autenticação da janela no gerenciador. Também permite que todas as informações de configuração possam ser centralizadas no gerenciador, isso diminui o trabalho do administrador do sistema em grandes redes (Packard, 2005).

Todo pacote do protocolo XDMCP que transita entre a janela e o gerenciador deve conter um dos seguintes códigos de operação: DifusãoDeConsulta, Consulta, ConsultaIndireta, EncaminhamentoDeConsulta, Disposto, Indisposto, Requisição, Aceito, Rejeitado, Gerenciar, Recusado, Falhou, FiqueAtivo ou Ativo. Abaixo é dada uma descrição de cada código de operação.

Consulta, DifusãoDeConsulta, ConsultaIndireta – os pacotes que contêm esses códigos de operação são enviados da janela para o gerenciador.

Um pacote de *Consulta* é enviado de uma janela para uma máquina específica perguntando se esta máquina está apta a fornecer um serviço de gerenciamento para a janela. A máquina deve responder enviando um pacote *Disposto* se estiver apta e um pacote *Indisposto* caso contrário.

O funcionamento do pacote *DifusãoDeConsulta* é similar ao do pacote *Consulta*. Porém esse se destina a qualquer máquina conectada à rede e as máquinas que não estão aptas a fornecer o serviço apenas ignoram a consulta, não enviam o pacote *Indisposto*.

Um pacote *ConsultaIndireta* é enviado a um gerenciador conhecido que encaminha a requisição a uma coleção de gerenciadores secundários usando um pacote *EncaminhamentoDeConsulta*. Deste modo, a coleção de gerenciadores pode estar agrupada em outra rede; o uso de um gerenciador central reduz a sobrecarga do sistema administrativo. O primeiro gerenciador também pode enviar um pacote *Disposto*.

EncaminhamentoDeConsulta – é um pacote enviado entre gerenciadores.

Quando um primeiro gerenciador recebe um pacote *ConsultaIndireta,* ele fica responsável por enviar um pacote *EncaminhamentoDeConsulta* para uma máquina da lista de gerenciadores disponíveis para fornecer o serviço para a janela usando o mesmo caminho na rede que o pacote de *ConsultaIndireta* original realizou. Os campos de porta e endereço do cliente devem conter um endereço que o gerenciador secundário pode usar para localizar a janela na rede. Cada gerenciador secundário envia um pacote de *Disposto* para a janela se estiver apto a fornecer o serviço. Semelhante ao pacote de *DifusãoDeConsulta* o pacote de *EncaminhamentoDeConsulta* não requer um pacote de *Indisposto* como resposta quando a máquina não está apta a fornecer o serviço.

Disposto – um pacote de *Disposto* é sempre enviado do gerenciador para a janela.

Um pacote *Disposto* é enviado pelos gerenciadores que podem fornecer serviços de conexão para esta janela. É emitido em resposta aos pacotes *Consulta, DifusãoDeConsulta,* ou *EncaminhamentoDeConsulta,* mas não implica em garantia no fornecimento do serviço (por exemplo, o gerenciador pode posteriormente decidir que já aceitou muitas conexões).

Indisposto – um pacote de *Indisposto* é sempre enviado no sentido do gerenciador para a janela.

Um pacote *Indisposto* é enviado em resposta a pedidos diretos de *Consulta* quando o gerenciador não está pronto para fornecer o serviço. Ele geralmente é enviado quando os gerenciadores desejam realizar alguma manutenção, prestar serviços a janela específicas ou que suportam um número limitado de janela simultâneas.

Requisição – um pacote de *Requisição* é sempre enviado da janela para o gerenciadores.

Um pacote de *Requisição* é enviado por uma janela para uma máquina específica requisitando um ID de sessão na preparação para um estabelecimento de uma conexão. Se o gerenciador está pronto para fornecer o serviço de conexão para a janela, ele deve retornar um pacote *Aceito* com um ID de sessão válido e deve estar pronto para a requisição do pacote de *Gerenciar* subseqüente. Caso contrário, ele deve retornar um pacote de *Rejeitado.*

Aceito – um pacote de *Aceito* é sempre enviado do gerenciador para a janela.

Um pacote *Aceito* é enviado por um gerenciador em resposta a um pacote de *Requisção* quando o gerenciador está pronto para aceitar a conexão com a janela. O ID da sessão é usado para identificar esta conexão e as que a precede. Também identificará a janela em seus pacotes *Gerenciar* subseqüentes. O ID de sessão é um número de 32 bits que é incrementado a cada vez

que um pacote *Aceito* é enviado sendo único durante um longo período de tempo. Se a informação de autenticação é inválida, um pacote de *Rejeitado* é retornado com uma mensagem do seu status.

Rejeitado - um pacote de *Rejeitado* é sempre enviado do gerenciador para a janela.

Um pacote *Rejeitado* é enviado por um gerenciador em resposta a um pacote de *Requisição* quando o gerenciador não pode estabelecer a conexão com a janela. Isto é permitido mesmo que o gerenciador tenha previamente respondido com um pacote de *Disposto* a uma consulta.

Gerenciar – um pacote de Gerenciar é sempre enviado da janela para o gerenciadores.

Um pacote *Gerenciar* é enviado para solicitar que o gerenciador comece uma sessão na janela. Se a identificação da sessão está correta o gerenciador deve abrir uma conexão, caso contrário, deve responder com um pacote *Recusado* ou *Falhou,* a menos que o ID da sessão seja encontrado nas sessões em funcionamento ou uma sessão que ainda não abriu a conexão da janela, mas que ainda não fez a tentativa acima. Neste último caso, o pacote *Gerenciar* deve ser ignorado. Isso trabalhará de forma a indicar para ambos os lados do canal que a conexão foi bem sucedida ou em caso de falha será uma indicação para o lado oposto ao da falha que algo saiu errado (Que geralmente irá gerar um pacote de *Falhou*).

Recusado - um pacote de *Recusado* é sempre enviado do gerenciador para a janela.

Um pacote *Recusado* é enviado por um gerenciador quando o ID da sessão recebido no pacote *Gerenciar* não corresponde ao ID da sessão corrente. A janela deve assumir que ela recebeu um pacote de *Aceito* antigo e deve reenviar seu pacote de *Requisição.*

Falhou - um pacote de *Falhou* é sempre enviado do gerenciador para a janela.

Um pacote *Falhou* é enviado pelo gerenciador quando ele tem problemas para estabelecer a conexão X inicial em resposta a um pacote *Gerenciar.*

FiqueAtivo - um pacote de *FiqueAtivo* é sempre enviado da janela para o gerenciador.

Um pacote *FiqueAtivo* pode ser enviado a qualquer tempo durante a sessão por uma janela para descobrir se o gerenciador está funcionando. O gerenciador deve responder com um pacote *Ativo* sempre que receber esse tipo de arquivo.

Isto permite a janela descobrir quando o gerenciador não está mais ativo. Não é exigido que uma janela envie pacotes *FiqueAtivo* e na falta do recibo de pacotes *Ativos, n*ão se exige a execução de nenhuma ação específica.

A utilidade prevista para esse pacote é terminar uma sessão ativa quando a máquina na qual o gerenciador está instalado ou a ligação de rede falharem. A janela deve manter-se a par do tempo decorrido desde o recebimento do último pacote enviado pelo gerenciador e usar um pacote *FiqueAtivo* quando esse intervalo de tempo for muito grande.

Ativo - um pacote de *Ativo* é sempre enviado do gerenciador para a janela.

Um pacote *Ativo* é enviado em resposta a uma *requisição FiqueAtivo*. Se a sessão estiver ativa na janela naquele momento, o gerenciador inclui a ID da sessão no pacote. A janela pode usar esta informação para determinar o status do gerenciador.

Os diagramas de estados que representam o funcionamento de cada um dos componentes do protocolo XDMCP são mostrados pelas figuras 2.13 e 2.14.

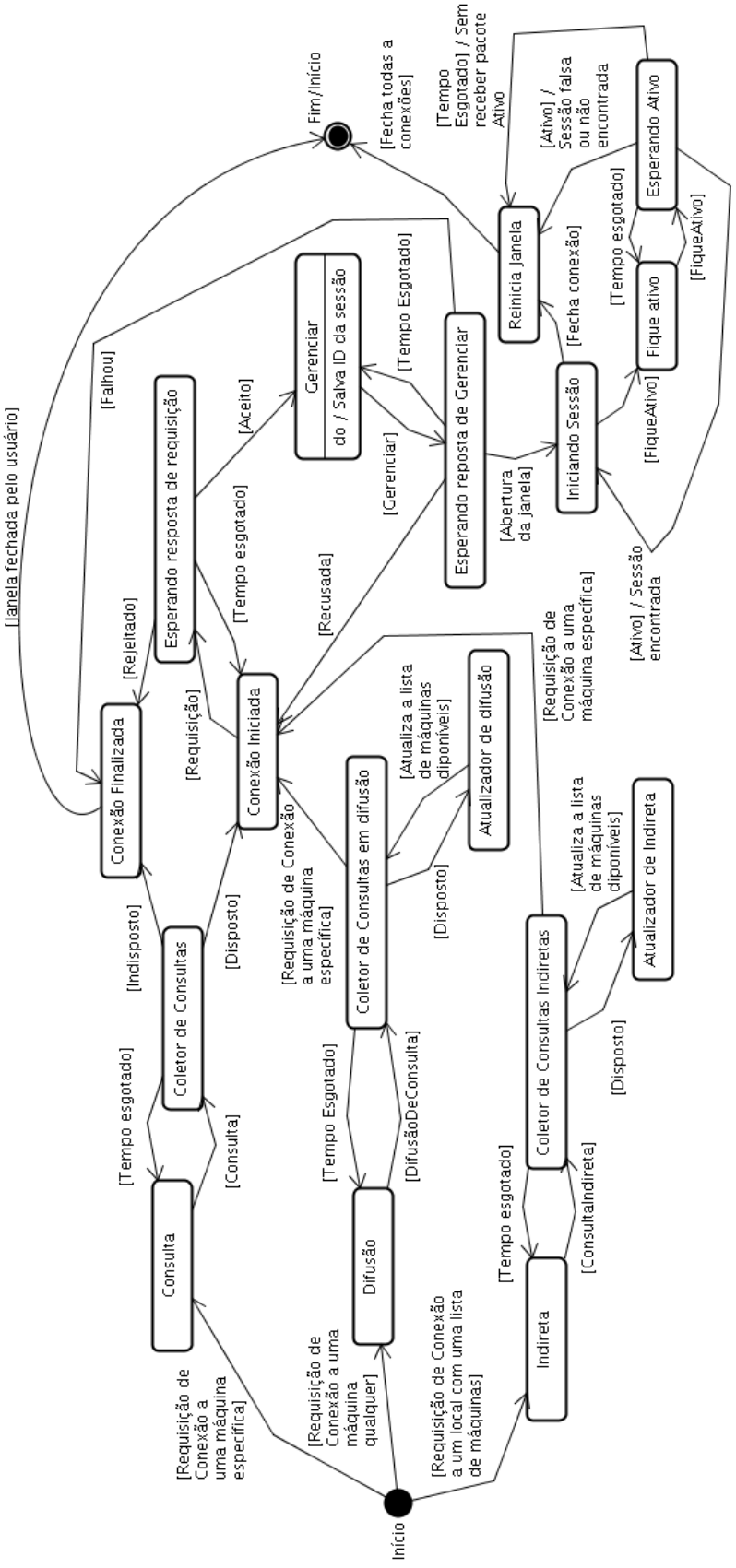

Figura 2.13: Diagrama de estados da Janela do protocolo XDMCP.

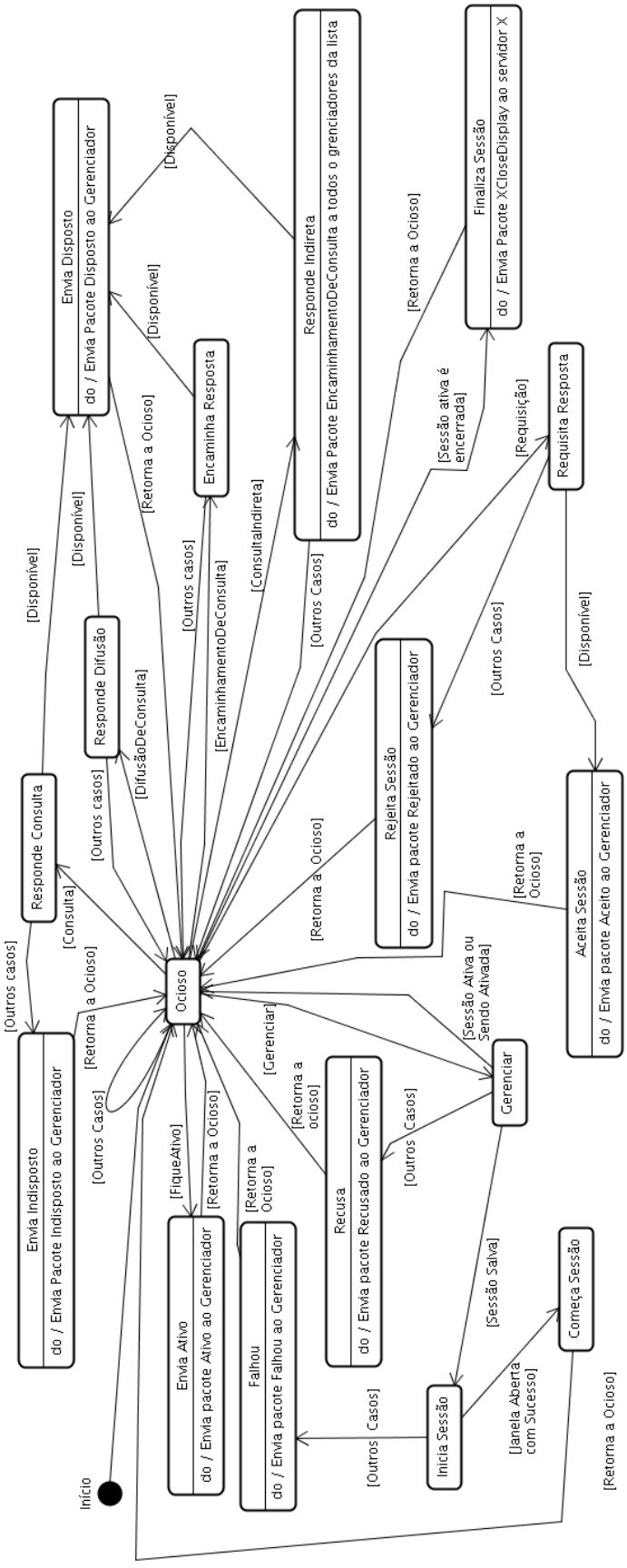

Figura 2.14: Diagrama de estados do gerenciador do protocolo XDMCP.

Gerenciador de Janelas

É através do gerenciador de janela que o usuário inicia sua interação com a GUI. No gerenciador de janela o usuário escolhe em qual máquina vai se conectar, digita seu *login* e senha, realiza a sua autenticação, e passa a ter acesso ao restante da GUI. Quando o usuário finaliza a sessão o gerenciador de janela encerra a conexão com a máquina local ou remota. O gerenciador de janela de uma estação utiliza-se do XDMCP para se comunicar com o servidor X das máquinas remotas (Pachard, 2008 e Petersen et. al., 2007).

Os gerenciadores de janelas implementam todas as características necessárias para o gerenciamento remoto ou local das janelas. Isso inclui autenticar usuários, iniciar e finalizar uma sessão.

O papel da interface gráfica no LTSP

Terminado todo o processo de montagem do sistema de arquivos e reconhecimento dos *hardwares* das estações, o servidor X *Window* na estação faz uma requisição XDMCP para o Gerenciador de janelas no servidor e esse retorna uma mensagem para o X *Window* exibir a tela para o usuário se *logar*. Após estar *logado* o usuário estará executando seus comandos no servidor através da estação (LTSP.org, 2008 e Martins, 2004).

Com a finalidade de se aumentar a segurança da arquitetura LTSP, a partir da versão 5.0 é utilizado o *LTSP Display* Manager (LDM), como gerenciador de janela padrão, no lugar dos gerenciadores mais populares nas distribuições GNU/Linux tais como o GDM – *Gnome Display Manager* e o KDM – *KDE Display Manager*. Com o LDM além da comunicação ser criptografada o *login* e a senha da sessão também passam a trafegar criptografados (LTSP.org, 2008).

2.5.5 A tecnologia SSH – Secure Shell Host

SSH é um protocolo que possibilita *login* remoto seguro entre outros serviços de rede de forma segura sobre uma conexão insegura como a TCP/IP (Javvin, 2005).

Os sistema SSH, escrito por Ttu Ylönen, é um substituto seguro para *rlogin*, *rcp*, e *telnet*. Ele usa autenticação criptografada para confirmar a identidade dos usuários e criptografa toda a comunicação entre duas máquinas (Nemeth et. al., 2002).

A tecnologia SSH se tornou uma alternativa popular para o TELNET, além disso o SSH provê duas melhorias significantes em relação ao TELNET, comunicação segura utilizando protocolos de criptografia e a capacidade de realizar transferência de dados adicionais e independentes pela mesma conexão. Embora sua origem tenha sido comercial, o SSH agora é um padrão proposto pelo IETF – *Internet Engineering Task Force* (Comer, 2006).

O protocolo da tecnologia SSH, descrito pela RFC 4251 é composto de três componentes principais (Ylonen, et. al., 2006a):

- O protocolo da camada de transporte fornece autenticação de servidores, confidencialidade e integridade de dados com privacidade garantida pela criptografia dos dados . Opcionalmente pode oferecer compactação dos dados a serem transportados. A proposta para padronização desse padrão é descrita na RFC 4253 (Ylonen, et. al., 2006b).

- O protocolo de autenticação do usuário é o componente do SSH responsável por autenticar o usuário para o servidor. Ele funciona sobre a camada de transporte e fornece um único túnel para a conexão SSH. A RFC 4252 possui uma proposta de padronização desse protocolo (Ylonen, et. al., 2006c).

- O protocolo de conexão com proposta de padronização encontrada na RFC 4254 (Ylonen, et. al., 2006d). É projetado para executar sobre a camada de autenticação do usuário possibilitando *login* interativo das sessões, execução remota de comandos transmissão de conexões X11 e TCP/IP. Cada um desses serviços é transmitido em um canal de comunicação lógico que será multiplexado em uma única conexão SSH.

O SSH é muito flexível e não exige a utilização de um protocolo específico em todos os aspectos da comunicação. Na autenticação do servidor o SSH utiliza criptografia de chave pública, já na autenticação do usuário ele permite o uso de senhas interativas ou criptografia de chave pública. Após a fase de autenticação é permitido a utilização dos mais diversos protocolos de criptografia para a comunicação.

Além de um serviço de *login* seguro remoto, o SSH fornece uma conexão segura de finalidade geral com o recurso de multiplexador de vários serviços através dessa conexão. Entre esses serviços que podem ser multiplexados estão a transferência segura de arquivos e o encaminhamento de portas.

O papel do SSH no LTSP

Utilizado somente a partir da versão 5.0 do LTSP o SSH provê uma forma segura via tunelamento de se realizar a autenticação do usuário do terminal, fornecer um meio seguro de comunicação entre o terminal e o servidor e a possibilidade de compactação para a transmissão dos dados necessários para a exportação da interface gráfica para os terminais.

A seguir é apresentada a metodologia utilizada no desenvolvimento deste trabalho.

3 METODOLOGIA

A presente pesquisa é de natureza aplicada ou tecnológica. Caracterizada pela utilização de conhecimentos básicos em redes de computadores e sistemas operacionais e suas tecnologias (Jung, 2004). Tendo como objetivo realizar uma pesquisa exploratória que visa oferecer uma alternativa a utilização tradicional dos computadores pessoais.

A pesquisa exploratória tem por finalidade proporcionar uma maior familiaridade com o problema, afim de torná-lo mais claro ou a construir hipóteses. Essas características levam ao aprimoramento de idéias ou a descoberta de intuições (Gil, 1991).

Como referencial bibliográfico utilizou-se informações divulgadas em livros técnicos, anais de eventos, palestras e em materiais disponíveis via *internet* como: históricos de listas de discussões, documentação de protocolos de serviços, e tutoriais. Estes estudos serviram para embasar os conhecimentos preexistentes e auxiliar na inferência de novos conhecimentos.

O procedimento utilizado foi a pesquisa experimental, fundamentada em referências bibliográficas classificada quanto ao local como pesquisa em laboratório e quanto ao tempo como estudo longitudinal (Jung, 2004).

O desenvolvimento experimental é composto de atividades sistemáticas definidas a partir de conhecimentos preexistentes o qual visa a demonstração da viabilidade técnica ou funcional de novos produtos, processos, serviços ou o aperfeiçoamento dos já existentes (Tomiyama, 2004).

A pesquisa experimental deste trabalho iniciou-se com um levantamento de quais eram as soluções e ferramentas existentes no mercado que utilizam o conceito de terminais leves. Conhecidas as soluções partiu-se para a verificação de quais seriam as ferramentas mais indicadas para o reaproveitamento das máquina obsoletas.

Definida a solução a ser utilizada, partiu-se para a montagem de um laboratório no qual foram colocados 10 terminais, com diversas configurações de memória e processadores, em funcionamento. A finalidade da montagem desse laboratório era permitir fazer um levantamento de dados sobre desempenho, facilidade de montagem da solução e eventuais problemas.

Também foi aplicado um questionário aos usuários do sistema para que esses passassem suas impressões a respeito do seu desempenho.

A sub-sessão a seguir apresenta uma descrição detalhada da montagem do laboratório e do método utilizado para se fazer a coleta dos dados sobre o desempenho do sistema de terminais leves.

3.1 Descrição do experimento

Este trabalho iniciou-se com a realização de um levantamento sobre quais eram as soluções existentes no mercado que possibilitavam a utilização de máquinas obsoletas como terminais. Desse levantamento chegou-se a duas soluções que mostraram-se viáveis para essa finalidade o *Windows Terminal Server* - WTS e o *Linux Terminal Server Project* - LTSP.

3.2 Solução escolhida

Entre as duas soluções encontradas foi escolhido o LTSP por ser uma ferramenta de código aberto que não requer qualquer tipo de licença para sua utilização. Também devido a ser amplamente utilizado principalmente em projetos de inclusão digital no Brasil, com destaque para os projetos mantidos pelas prefeituras de São Paulo, Porto Alegre e do Governo do Paraná (Filitto, 2007).

Outro motivo que levou a escolha da solução é que apesar da sua grande utilização encontra-se poucos estudos sobre seu desempenho, requisitos mínimos das máquinas a serem utilizadas tanto para servidores quanto para terminais.

Determinada a ferramenta a ser utilizada, o LTSP, o passo seguinte foi montar um protótipo da solução com a finalidade de se aprender como era seu funcionamento, os programas utilizados e configurações. As configurações de *hardware* desse protótipo estão descritos na Tabela 3.1.

Tabela 3.1: Configuração de Hardware do protótipo.

Tipo	Processador	Memória RAM
Servidor	AMD Atlhon XP 2000+	512 MB
Cliente	Intel Celeron 1000 Ghz	128 MB

Na primeira tentativa de instalação foi utilizado a distribuição Kurumin 7 com a versão 4.2 do LTSP, seguindo o tutorial disponível no site da distribuição (Morimoto, 2006). Essa tentativa não foi bem sucedida devido a problemas encontrados na configuração do pacote *tftpd* que tornava instável o processo de *boot* das estações.

Com o lançamento da versão 5.0 do LTSP para a distribuição Ubuntu optou-se por fazer uma nova tentativa seguindo o tutorial disponível em (UbuntuLTSP, 2007). A instalação dessa versão se mostrou muito prática, pois é totalmente automatizada, bastando digitar apenas poucos comandos.

Além da facilidade de instalação, essa versão se mostrou interessante para o experimento, devido a uma mudança ocorrida na estrutura do LTSP. Até a versão 4.2 os terminais utilizavam para seu funcionamento uma imagem do GNU/Linux Red Hat modificada, porém completa, dessa forma os programas ficavam duplicados. Tínhamos por exemplo, dois servidores de interface gráfica ou dois navegadores um para uso do servidor e outro para uso das estações.

Essa estrutura além de utilizar desnecessariamente espaço em disco com cópias de programas que já existiam no servidor também dificultava a instalação de novos programas, a adição de novas funcionalidades, como por exemplo, a utilização de dispositivos de armazenamento móveis e principalmente a manutenção do código.

A estrutura do LTSP 5.0, por outro lado permite a utilização dos programas nativos do servidor pelos clientes eliminando a duplicação de programas. Opcionalmente essa versão permite também a instalação local de programas, dessa forma pode-se utilizar o processamento das estações mais robustas para executar programas mais pesados. Essas estações com maior poder de processamento são chamadas de *FatClient.*

A versão 5.0 sofreu também modificações que visam um aumento na segurança da solução. Destaca-se entre elas a substituição da maneira como se exporta a interface gráfica para os terminais. Em versões anteriores a interface gráfica era exportada diretamente pelo XDMCP e a partir dessa versão passou a ser utilizado um túnel SSH exportando a interface gráfica. Houve da mesma forma uma troca no sistema de gerenciamento de *login* remoto para o *LTSP Display Manager* - LDM que possibilita as senhas trafegarem somente criptografadas desde o início do processo de *login* o que não ocorre com gerenciadores tradicionais como GNOME e KDE (UbuntuLTSP, 2007 e LTSP.org, 2008).

Uma consequência das modificações de segurança a ser observada no experimento é a possibilidade de que elas acarretem um aumento nos requisitos mínimos de memória e processamento tanto por parte dos clientes quanto do servidor.

Após a fase de aprendizado sobre o funcionamento da solução utilizando o protótipo, procurou-se então um local onde a solução poderia ser colocada em teste, numa escala maior.

3.2.1 Escolha e preparação do laboratório para o teste

O local escolhido foi o antigo laboratório de tecnologia da informação e aprendizagem em administração financiado pelo PROIN - Programa de Apoio à Integração Graduação – Pós-Graduação da CAPES, localizado no prédio do DAE – Departamento de Administração e Economia da Universidade Federal de Lavras.

Esse laboratório foi inaugurado em 1998 sendo destinado a utilização de docentes de graduação e pós-graduação do curso de Administração. A infra-estrutura original do laboratório era constituída de 24 computadores Intel Pentium 200MHz interligados através de uma rede cabeada que suportava uma taxa de transferência de até 10Mbps.

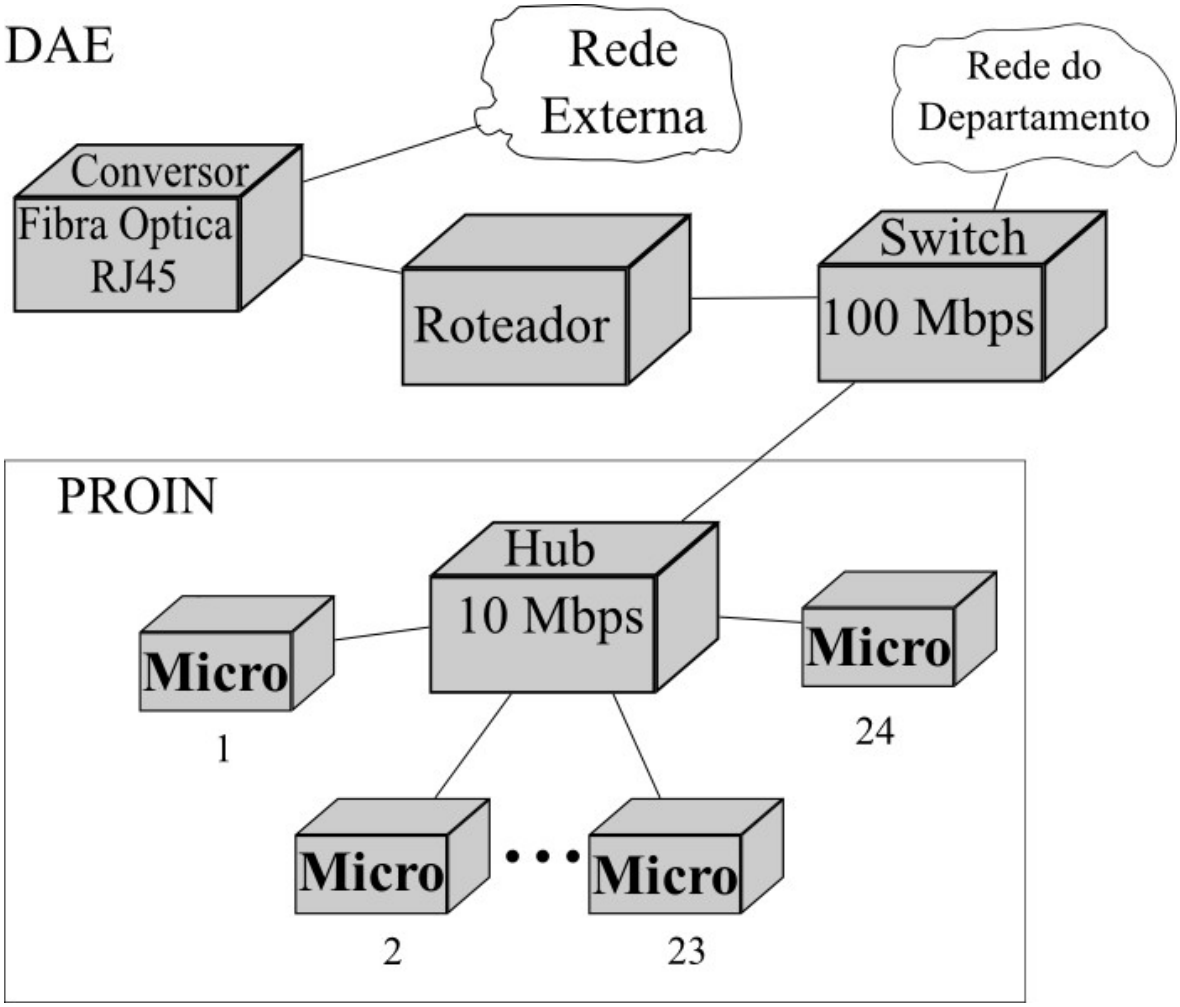

Figura 3.1: Leiaute da rede do PROIN antes de seu fechamento.

O PROIN foi desativado em 2006 devido à obsolescência de seus equipamentos. Após seu fechamento, a sala do antigo laboratório passou a ser utilizada como depósito do DAE para as máquinas obsoletas ou que apresentavam algum defeito. A Figura 3.1 representa um desenho do leiaute da rede do PROIN antes de seu fechamento.

Figura 3.2: Estado do laboratório inicial do laboratório

Para a reativação do laboratório foi necessário verificar o estado que o mesmo se encontrava a .o que é ilustrado pela Figura 3.2

3.2.2 Verificação da infra-estrutura disponível

Durante essa verificação constatou-se que a infra-estrutura da rede cabeada se encontrava bastante comprometida. Entre os problemas da rede podemos citar a impossibilidade de utilização dos ativos de rede como *hub* e *switch* por estarem queimados ou serem obsoletos, muitas das canaletas para o suporte dos cabos estavam quebradas e os cabos partidos ou apresentando mau contato.

Foram encontrados no local 45 computadores, porém muitos deles se destinavam ao descarte. Realizou-se então uma seleção para separar os computadores que funcionavam dos que seriam descartados, após essa seleção obteve-se um total de 17 computadores funcionando. A configuração das máquinas recuperadas é descrita na Tabela 3.2.

Tabela 3.2: Configuração das máquinas recuperadas.

Processador	Frequência em MHz	Tipo de Memória RAM	Quantidade de Memória RAM em MB	Velocidade da placa de rede em Mbps	Número de Computa-dores Semelhantes
Intel Pentium	200	SIMM	16	10	11
Intel Pentium	200	SIMM/DIMM	32	100	3
Intel Pentium	200	SIMM/DIMM	48	100	1
AMD K6-2	500	DIMM	128	100	2

Além desses computadores durante a fase de levantamento dos dados foram utilizados dois computadores novos com a configuração descrita na Tabela 3.3.

Tabela 3.3: Configuração das máquinas novas.

Processador	Frequência em GHz	Tipo de Memória RAM	Quantidade de Memória RAM em MB	Velocidade da placa de rede em Mbps
Intel Pentium 4	3,20	DDR2	2048	100
AMD Athlon 64	1,80	DDR2	1024	100

Finalizada a verificação foi possível estabelecer qual o material deveria ser adquirido para se obter o um laboratório de computação funcionando com LTSP.

3.2.3 Montagem do laboratório utilizando LTSP

A montagem do laboratório foi realizada tendo como meta disponibilizar um mínimo de 10 terminais para o teste. Porém a infra-estrutura de rede e o servidor deveriam suportar uma posterior expansão para 24 terminais, ou seja, o número original de computadores disponíveis no PROIN antes de seu fechamento.

Por meio do levantamento descrito na sub-seção 4.4.2 chegou-se a conclusão que as máquinas existentes teriam poder de processamento suficiente apenas para serem utilizadas como terminais. Foi então necessário adquirir uma máquina para ser utilizada como servidor assim como material para refazer a infra-estrutura da rede.

A Tabela 3.4 mostra quais materiais foram adquiridos, pelo DAE, assim como suas quantidades e valores em agosto de 2007.

Tabela 3.4: Custo da implantação do Laboratório.

Material	Quantidade	Preço em R$
Servidor HP ML115 com 4 GB de Memória RAM	1	4998,91
Régua com 8 Tomadas e disjuntor	1	83,00
Guia de Cabos	4	56,00
Switch Planet 24 Portas 10/100 Mbps Cat 5e	2	730,00
Patch Panel Planet 24 Portas Cat 5e	2	239,80
Rack de parede	1	310,00
Caixa de Cabos Cat 5e Furukawa	2	580,00
kit de Caixa externa + 1 tomada RJ45/IDC 110 Tibix Cat 5e	30	214,00
Patch Cord	52	410,80
Kit (Parafuso philips M5 + porca gaiola)	6	55,20
Total		7677,71

Para o servidor foi escolhido um AMD Opteron Dual Core com freqüência de 1,8 GHz e 4 GB de memória RAM. A quantidade de memória foi dimensionada para suportar até 24 terminais, porém no experimento só foram utilizados 10.

De posse dos materiais necessários partiu-se para a montagem do laboratório. A rede de dados foi recriada obedecendo a norma TIA/EIA 568-B. Como sistema operacional do servidor foi utilizado a distribuição Ubuntu na sua versão 7.04. A instalação padrão do Ubuntu já disponibiliza a maioria dos programas utilizados em ambientes administrativos e de pesquisa tais como: *Firefox* - navegador de *internet*, *Open Office* - suíte de escritório, *Gaim* – Cliente de mensagem instantânea entre outros.

Para se ter uma solução LTSP, foi necessário instalar no servidor os programas *openssh-server* e *ltsp-server-standalone* via *apt-get.* O *apt-get* é um programa que baixa e instala outros programas disponíveis em repositórios na *internet* utilizado principalmente em instalações derivadas da distribuição Debian como é o caso do Ubuntu.

O programa *ltsp-server-standalone* e suas dependências preparam todo o ambiente para o funcionamento da solução LTSP. Após rodar alguns *scripts* que o programa fornece e alterar alguns arquivos de configuração de serviços disponíveis na distribuição, já é possível utilizar a ferramenta LTSP.

Para o funcionamento dos terminais há várias maneiras de se realizar o processo de *boot* através da rede. Em máquinas mais novas o BIOS – *Basic I/O System* já vem preparado para realizar esse processo utilizando o sistema de ambiente de execução de pré-inicialização, PXE – *Preboot Execution Environment.*

Para as máquinas que não possuem essa opção em seu BIOS, como é o caso das máquinas antigas utilizadas nesse experimento, é necessário adquirir uma imagem de inicialização idêntica a de uma memória ROM para a inicialização da máquina (Rom-o-matic, 2008). Essa imagem é específica para cada marca e modelo de placa de rede. Existem também diferenças nas imagens quanto ao tipo de mídia a ser utilizada para armazená-las. Nesse experimento optou-se por utilizar os discos rígidos disponíveis em cada máquina para armazenar as imagens.

Figura 3.3: Terminais em funcionamento com LTSP no laboratório do PROIN

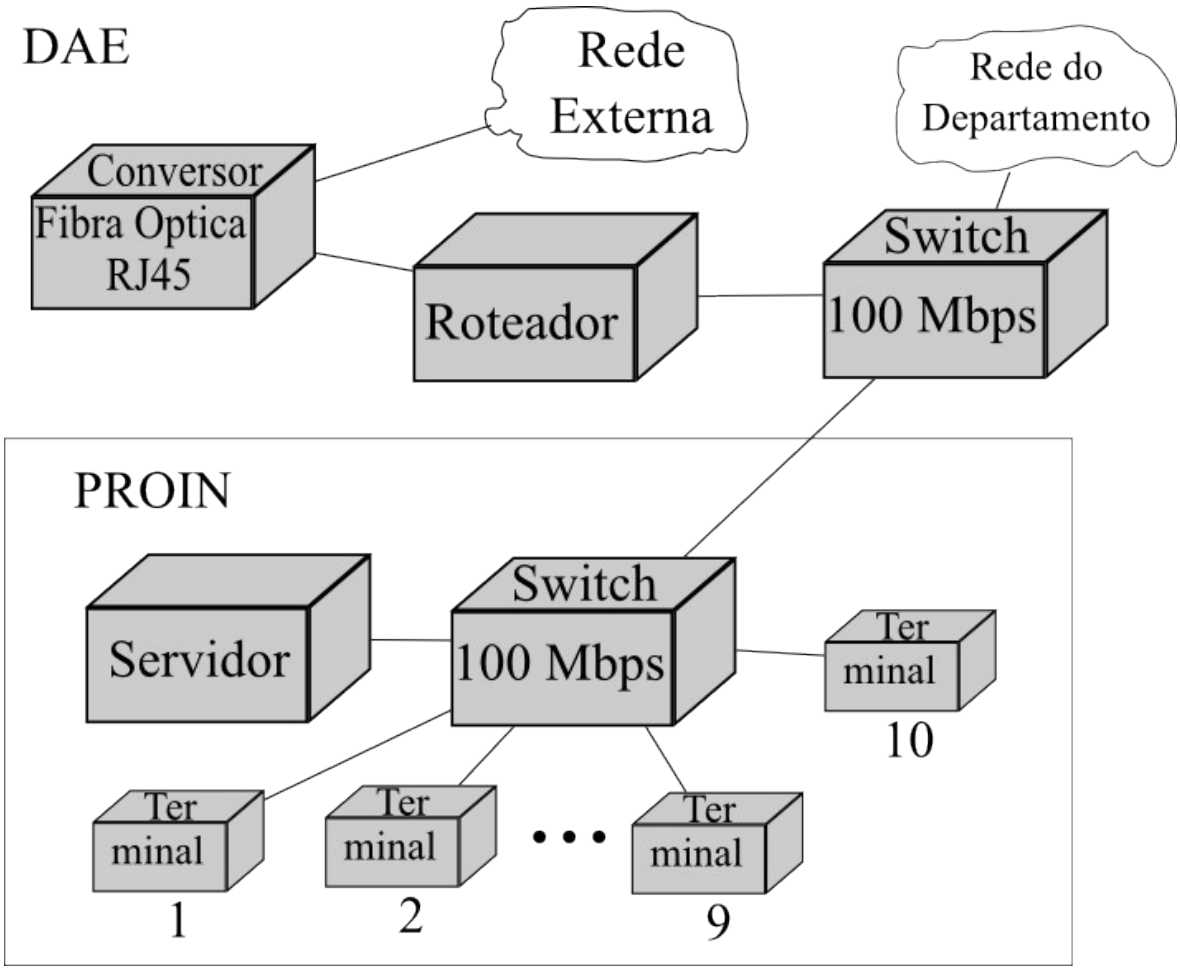

Figura 3.4: Leiaute do laboratório utilizando LTSP.

Com as imagens já gravadas nos discos passamos a ter o sistema em funcionamento. A Figura 3.4 é um esquema atual da montagem do laboratório e a Figura 3.3 mostra os terminais em funcionamento da esquerda para a direita nessa figura temos: o primeiro terminal em processo de inicialização, o segundo com um programa editor de texto aberto, o terceiro com o navegador de *internet* aberto e os demais terminais na tela de *login* esperando a autenticação do usuário.

Para a avaliação das métricas de desempenho como porcentagem de uso da CPU, carga média, uso de memória e rede da solução utilizada neste trabalho é necessário realizar a coleta dos dados a seguir: número de processos em execução, tempo de execução dos processos, número de pacotes enviados e recebidos pela rede.

Na seção a seguir é descrito o *software* e o método utilizado para a coleta dos dados.

3.3 Coleta dos dados sobre o desempenho

Para a coleta de métricas como porcentagem de uso da CPU, carga média, uso de memória e rede, foi utilizado a ferramenta de monitoramento de *clusters* Ganglia que é composta de dois processos o *gmond* e o *gmetad*.

O monitoramento do servidor e de cada terminal é realizado pelo *gmond*. Ele responde a requisições do clientes retornando uma representação em XML dos dados coletados. Por motivos de desempenho o *gmond* só envia os dados coletados quando esses sofrem uma mudança significativa e em períodos aleatórios evitando que vários *gmond* enviem os dados ao mesmo tempo.

Por outro lado *gmetad* é o processo responsável por reunir a informação de todos os *gmond,* e caso exista, de outros *gmetad* encontrados na rede.

Para armazenar e visualizar os dados monitorados o Ganglia utiliza o *RRDtools* (*Round Robin Database*). Ele gera gráficos, métricas pelo tempo, para diversas granularidades de tempo. Esses gráficos podem ser visualizados por meio de uma interface Web em PHP (Neves et. al., 2005).

Com o Ganglia em funcionamento foi realizada uma simulação com o uso de 10 terminais durante uma hora e meia. Dessa maneira foi possível comparar o resultado da coleta das métricas desejadas entre os 10 terminais. Na primeira meia hora do experimento os usuários estiveram livres para realizar tarefas diversas como navegação na *internet,* digitação de textos nas suítes de escritórios e visualização de vídeos em *flash* da forma que desejassem.

No restante do período os usuários foram orientados a seguir um roteiro que simulava uma aula de informática básica, muito comum em ambientes de inclusão digital.

No roteiro, primeiramente deveriam abrir o editor de texto, digitar um pequeno texto, abrir o *browser* procurar uma figura qualquer na *internet* e inseri-la no texto. Realizada essa série de tarefas o usuário deveria salvar o texto em sua área de trabalho e fechar o editor de texto.

A segunda tarefa foi abrir um vídeo, em *flash,* pré-determinado e deixá-lo tocar até o fim. Após o fim do vídeo foi solicitado que os usuários respondessem um questionário a respeito das suas impressões sobre o funcionamento do sistema.

Na seção a seguir são discutidas as diferenças encontradas na maneira de se fazer a instalação das duas versões do LTSP estudadas, a montagem do laboratório e em seguida são mostrados e discutidos os resultados da avaliação das métricas de desempenho do sistema.

4 RESULTADOS E DISCUSSÃO

Durante a fase de aprendizado a instalação da versão 4.2 do LTSP se mostrou uma tarefa um tanto quanto complexa, mesmo existindo um *script* que tem por finalidade facilitar esse processo. O funcionamento desse *script* se restringe a baixar a imagem a ser utilizada como sistema operacional pelos terminais, realizar uma verificação do estado dos serviços necessários ao LTSP e prover alguns modelos de arquivos de configuração para os programas que disponibilizam esses serviços. Porém, grande parte da configuração dos programas que fornecem esses serviços deve ser realizada manualmente.

Os programas que fornecem os serviços e que devem estar instalados e configurados no servidor para a utilização do LTSP 4.2 são: *tftpd, dhcp3-server, nfs-kernel-server, xdmcp*. Este processo de instalação manual dos programas, que servem de base a ferramenta LTSP, exige que o usuário tenha habilidade e conhecimento para realizar as alterações necessárias nas configurações padrões dos programas para que esses se adeqüem à forma que o LTSP utiliza.

Um problema encontrado para a instalação desses serviços é a dificuldade de se localizar um erro de configuração, pois um erro em algum dos programas que oferecem os serviços básicos pode provocar um efeito cascata que prejudica o funcionamento de outro programa, mesmo que este outro programa esteja bem configurado. Este problema só é detectado durante a execução do *script* de instalação do LTSP, ou seja, todos os serviços já foram instalados e configurados. Logo, a resolução do problema requer uma revisão completa de todo o processo de instalação dos programas que provêem esses serviços.

Por outro lado, a instalação do LTSP 5.0 se mostrou bastante prática e mais adequada aos usuários que não possuem um profundo conhecimento sobre o funcionamento das configurações dos serviços em sistemas operacionais GNU/Linux. Na distribuição Ubuntu 7.04, utilizada neste trabalho, é necessário que o usuário saiba como instalar programas por meio do *apt-get* e como fazer para executar *scripts* para se ter um sistema LTSP operacional.

Os passos a serem seguidos para a instalação do LTSP 5.0 em um servidor já utilizando como sistema operacional a distribuição GNU/Linux Ubuntu 7.04 são encontrados em UbuntuLTSP, 2007. O LTSP 5.0 só requer que o usuário edite algum arquivo de configuração quando esse deseja expandir as funcionalidades padrões do sistema como por exemplo, a utilização de disquetes e *drives* de CD-ROM dos terminais.

No decorrer da montagem do laboratório foi observado que colocar para funcionar um terminal que possui a opção de *boot* pela rede é um processo bastante prático. Basta que o

usuário ligue o computador em um ponto da rede e o terminal já estará funcionando. A única configuração que o usuário necessita fazer é habilitar a opção de *boot* pela rede no BIOS da placa mãe caso esta ainda não tenha sido habilitada.

Em contrapartida utilizar como terminal uma máquina que não possui essa opção requer um conhecimento sobre *hardware* mais aprofundado, pois o usuário deve identificar a marca e o modelo da placa de rede do computador. Identificada corretamente essa placa o usuário deve adquirir uma imagem inicializável para a mesma (Rom-o-matic, 2008). Esse processo pode dificultar a montagem e a manutenção de laboratórios utilizando terminais leves caso exista uma variedade muito grande de marcas e modelos das placas de rede desses computadores.

No experimento optou-se por utilizar a imagem de inicialização gravada no disco rígido dos computadores. Com as imagens já gravadas passou-se a fazer os testes de inicialização dos computadores. Nesse ponto notamos que as modificações, visando um aumento na segurança dos dados trafegados entre o servidor e os terminais, acarretaram um aumento do requisito mínimo de memória RAM dos terminais. Conforme a literatura pesquisada durante este trabalho em versões anteriores ao LTSP 5.0, era possível ter um terminal com 16MB de memória RAM em operação. Porém, somente as máquinas que possuíam 24 MB de memória RAM ou mais terminavam o processo de abertura de uma sessão de usuário.

A tentativa de se aumentar a quantidade de memória dos onze computadores que possuíam apenas 16 MB de memória RAM descritos na Tabela 3.2, da qual apenas duas tiveram sua memória efetivamente aumentada, destacou um problema que até então não fora levado em consideração: A dificuldade de se encontrar peças compatíveis com essas tecnologias ultrapassadas. Além do mais mesmo que se encontre essas peças o custo de aquisição quase sempre é inviável quando comparado a uma peça com tecnologia atual que tem função semelhante.

Com o servidor configurado e os dez terminais em funcionamento iniciou-se o levantamento dos dados da simulação do uso efetivo do sistema por meio da ferramenta Ganglia (Neves et. al., 2005). Essa simulação teve seu início às 14h 30' e término às 16h. Ela incluiu a utilização de vídeos em *flash* por serem programas que exigem uma melhor performance dos computadores, navegadores e suítes de escritório como editor de texto e planilha eletrônica que são programas amplamente utilizados em laboratórios de informática em geral.

Pela observação do funcionamento do LTSP durante a simulação verificamos que os vídeos em *flash* não eram transmitidos de forma correta nas estações de baixo poder de processamento e pouca memória. Os vídeos apresentavam perda de muitos quadros ao serem

transmitidos inviabilizando sua utilização por essas estações. No entanto, nas duas estações com melhor performance essa perda de pacotes é imperceptível apesar de ainda existirem.

Já a utilização de *softwares* mais leves como um navegador e um programa editor de texto não apresentou diferença substancial entre os diversos tipos de estações disponíveis. Porém, foi relatado pelos usuários dos terminais mais fracos, com menor poder de processamento e quantidade de memória, que ao abrirem muitas instâncias do mesmo software como por exemplo, várias janelas do navegador, havia um congelamento de algumas dessas janelas.

Esse congelamento obrigava o usuário a finalizar a janela congelada, sendo que as outras janelas e aplicações continuavam funcionando de forma correta. Em momento algum o usuário chegou ao ponto crítico de ter que efetuar o reinício de sua estação. A princípio esperava-se que isso ocorresse com as estações mais fracas ao executarem aplicações multimídia.

Com relação ao questionário todas as respostas dos usuários apontaram que a utilização de suíte de escritório no LTSP é igual ou melhor que com computadores no modelo tradicional e as notas para o desempenho geral da solução ficaram entre bom e ótimo.

As métricas avaliadas nesse trabalho construídas com os dados recolhidos durante a simulação são apresentadas a seguir.

Os gráficos foram gerados automaticamente por um dos componentes da ferramenta Ganglia que utiliza os IP das máquinas como referência a elas. Logo a Tabela 4.1 auxilia no mapeamento entre os gráficos e os computadores aos quais eles se referem.

Tabela 4.1: Relação entre os IP e os computadores.

IP	Frequência em MHz	Memória RAM em MB
192.168.18.88	500	128
192.168.18.106	1800	1024
192.168.18.126	3200	2048
192.168.18.236	200	24
192.168.18.240	500	128
192.168.18.241	200	32
192.168.18.243	200	48
192.168.18.245	200	32
192.168.18.250	200	32
192.168.18.252	200	24

A primeira série de gráficos se referem à utilização da CPU tanto dos terminais quanto do servidor.

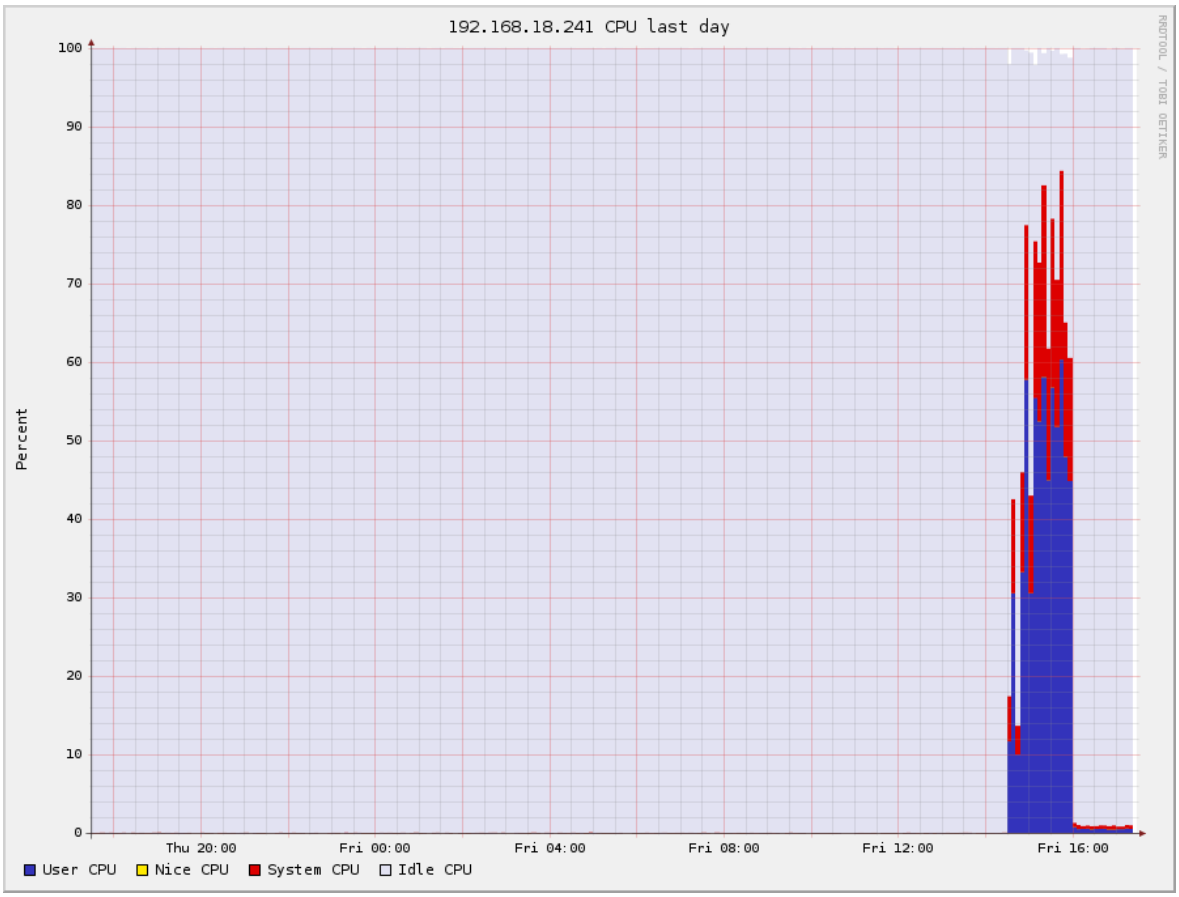

Figura 4.1: Gráfico com o consumo máximo de CPU.

O gráfico da Figura 4.1 é referente aos dados recolhidos do terminal com IP 192.168.18.241, seu gráfico é semelhante ao do terminal com IP 192.168.18.88, ele mostra o maior consumo de CPU da simulação com picos de até 84% de utilização por processos de todo o sistema (System CPU) e de 60% de utilização por processos do usuário (User CPU). Os usuários que estavam nesses terminais usaram na maior parte do experimento aplicações construídas em *flash*.

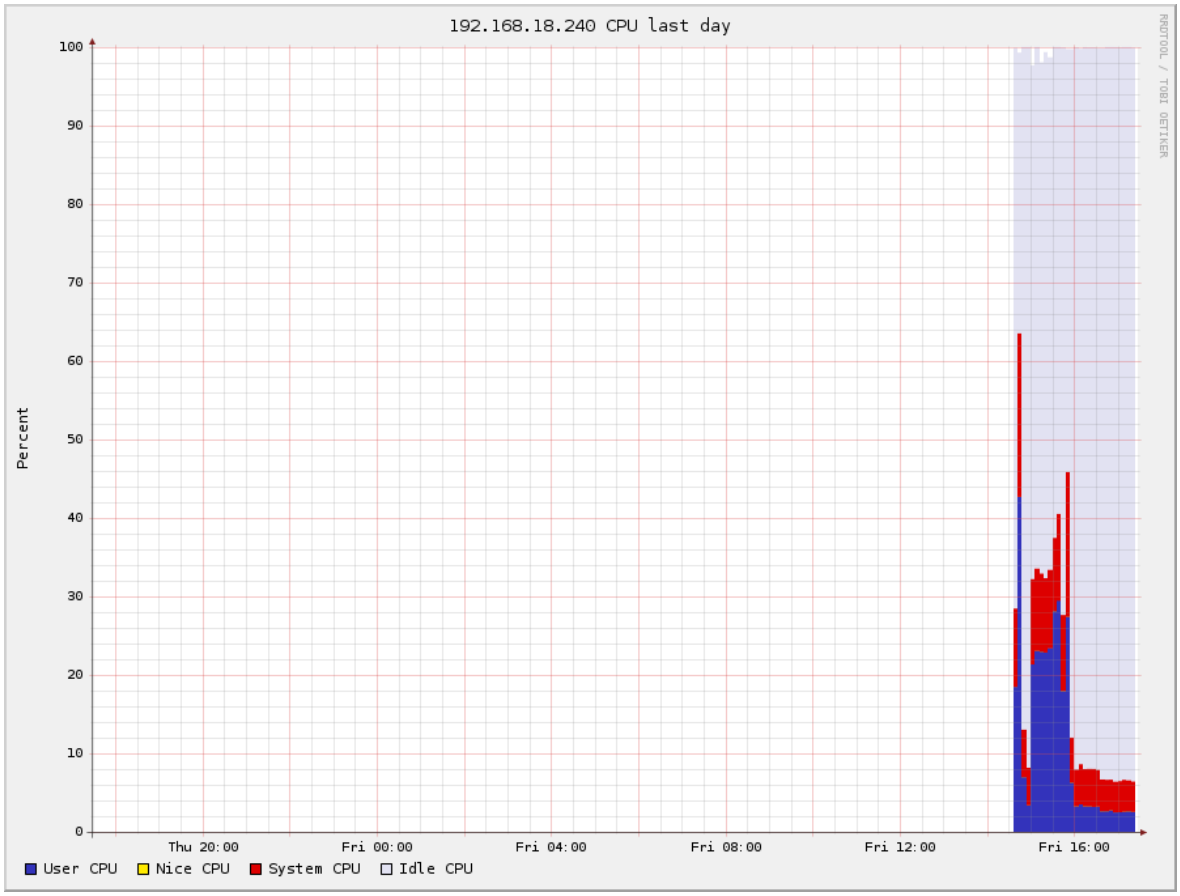

Figura 4.2: Gráfico com o consumo médio de CPU.

No gráfico da Figura 4.2 o consumo de CPU do terminal 192.168.18.240 apresentado é semelhante ao gráfico das máquinas 192.168.18.236, 192.168.18.245 e 192.168.18.250. Os usuários dessas máquinas optaram por utilizar programas mais leves como editores de texto, leitura de correio eletrônico e edição de *blogs* como o Orkut.

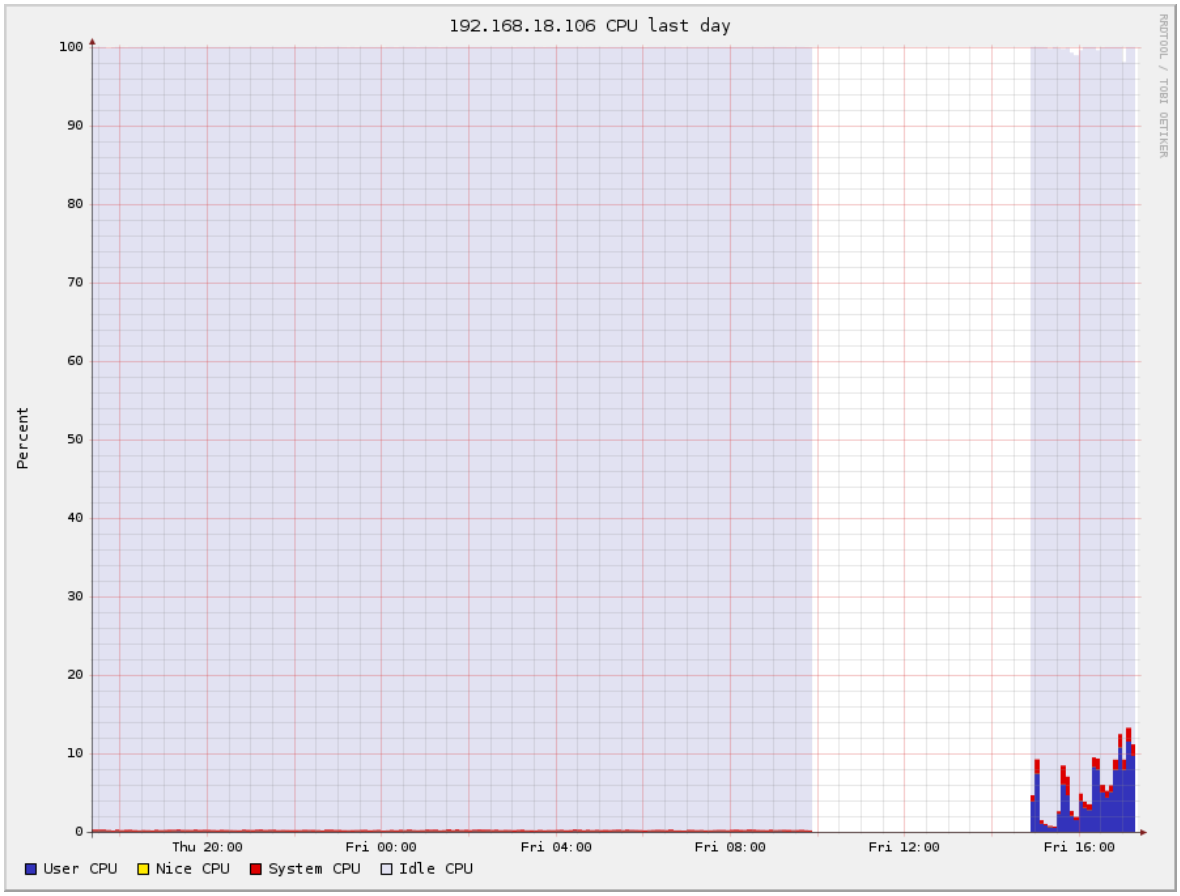

Figura 4.3: Gráfico com o consumo mínimo de CPU.

Os dados originados do terminal com IP 192.168.18.106 são apresentados no gráfico da Figura 4.3 que demonstra o menor consumo de CPU. O usuário dessa máquina passou maior parte da simulação navegando por páginas HTML e eventualmente abriu um vídeo em *flash,* como pode ser notado nos picos de utilização da CPU. Porém, cabe destacar que essa máquina possui poder de processamento suficiente para ser utilizada com uma estação de trabalho tradicional.

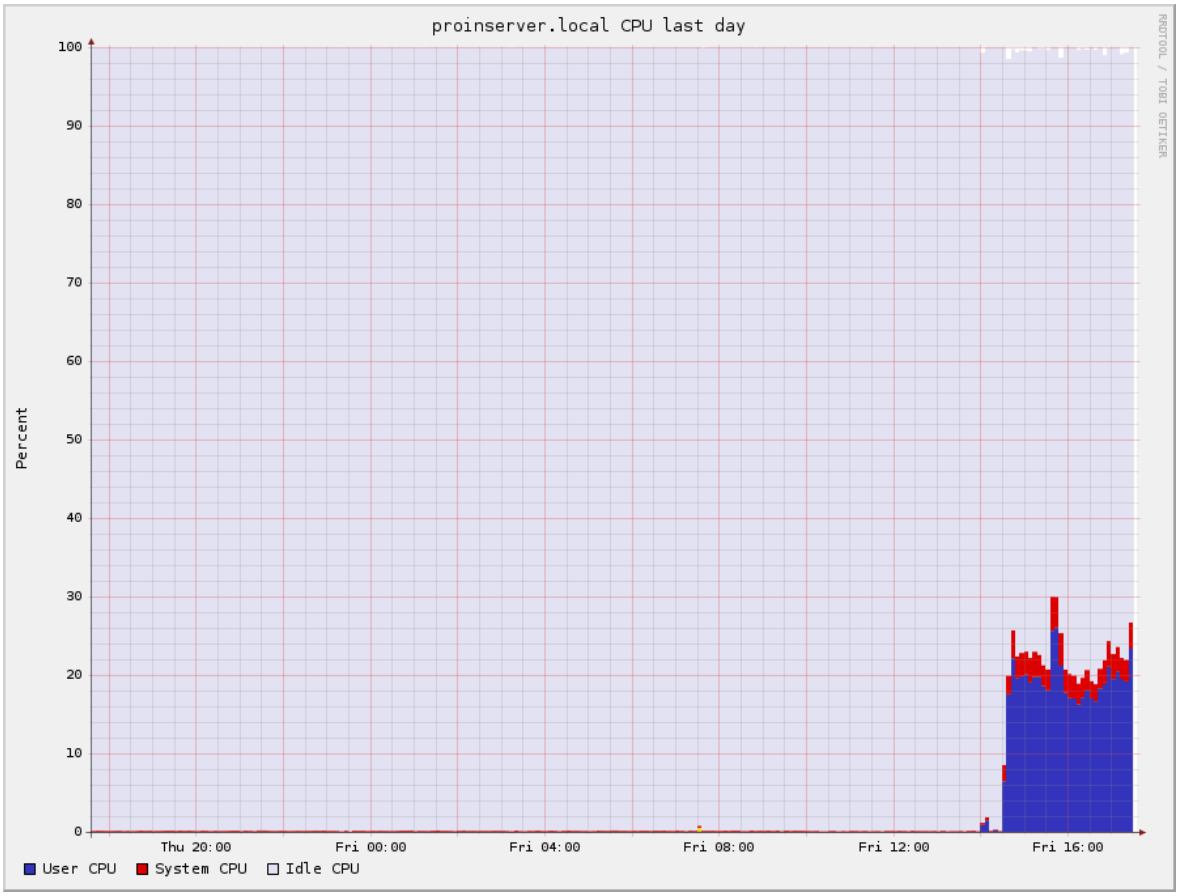

Figura 4.4: Gráfico do consumo de CPU do servidor.

Pelo gráfico da Figura 4.4 nota-se que, durante toda a simulação, o consumo de CPU do servidor foi de no máximo 30% para todos os processos e 22% apenas para processos do usuário. Esses valores são atingidos no momento que os usuários estão seguindo o roteiro de atividades, ou seja, estão realizando as mesmas tarefas ao mesmo tempo. Fora desse período o consumo fica abaixo de de 22%, o que demonstra que a utilização de CPU não pode ser considerada um fator limitante para a solução. Cabe ressaltar que o momento do pico de CPU no gráfico corresponde ao período em que os usuários estão visualizando um vídeo pré-determinado de cinco minutos solicitado no roteiro.

O gráfico da Figura 4.5 resume o consumo de CPU do sistema LTSP como um todo, por meio dele pode ser notada a diferença no consumo de CPU tanto do usuário quanto total em atividades distintas. De 14h 30' às 15h os usuários estavam liberados para utilizarem os terminais da forma que desejassem, mas pediu-se que fosse dado uma maior ênfase à utilização de aplicações multimídia. Nesse período o consumo de CPU por todos os processos no sistema ficou a maior parte do tempo acima de 28% com picos de até 39% enquanto que os processos do usuário ficaram em 16% com picos de 26%.

A partir de 15h os usuários concomitantemente passaram a utilizar o editor de texto para criar um pequeno texto e o navegador para fazer buscas e visualizar um pequeno vídeo. Nesse intervalo de tempo o consumo de CPU tanto de processos do sistema quanto de usuário ficaram próximos aos da primeira etapa.

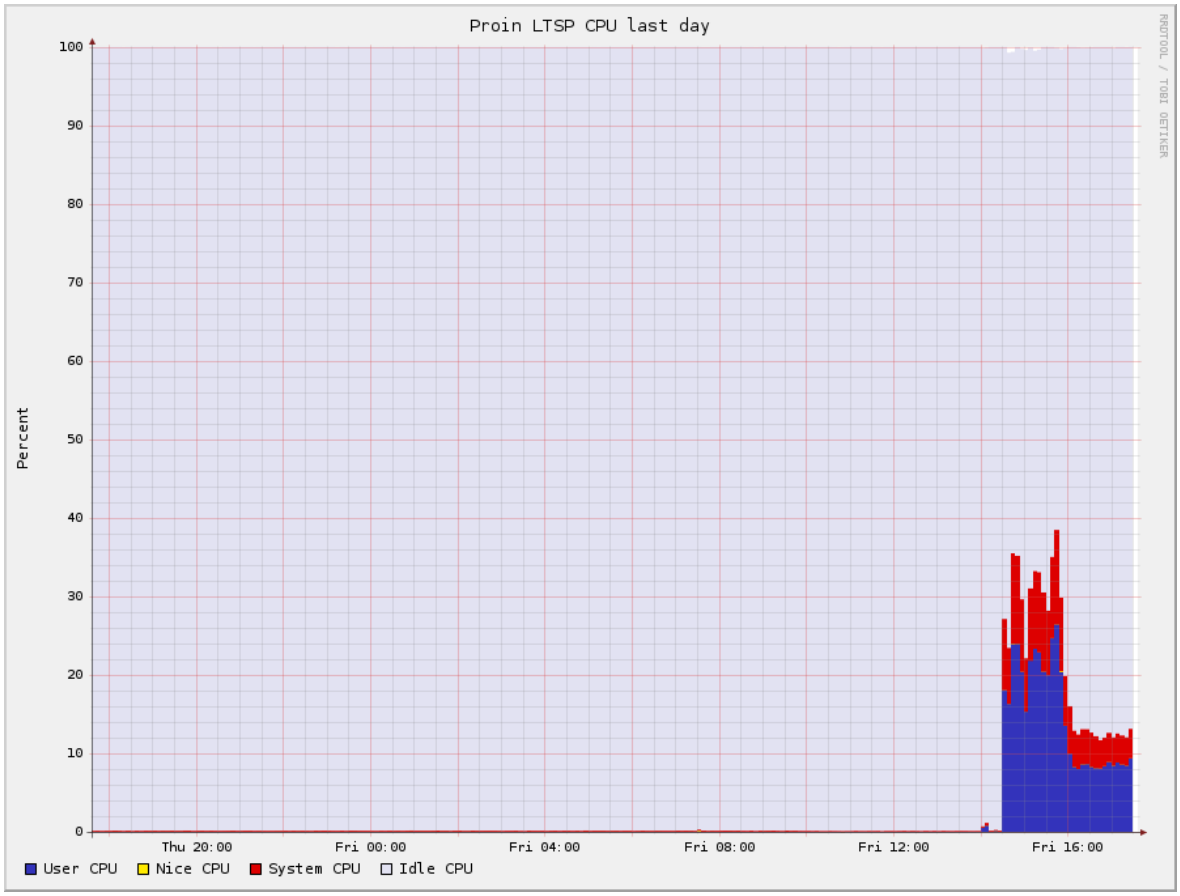

Figura 4.5: Gráfico do consumo de CPU de todo o sistema LTSP.

A próxima série de gráficos mostra a utilização de memória dos terminais e do servidor.

Este gráfico da Figura 4.6 representa um grupo de cinco máquinas, 192.168.18.236, 192.168.18.241, 192.168.18.243, 192.168.18.245 e 192.168.18.250 com configuração semelhante de processador e memória. Nele pode-se observar que devido a baixa quantidade de memória RAM disponível a máquina passa o tempo todo realizando *swap*. Essa atividade causa um atraso na resposta ao usuário, já que a máquina não possui dispositivo de armazenamento próprio e o *swap* é realizado no disco do servidor através da interface de rede. Durante a atividade de *swap* foi notado um pequeno atraso na resposta do movimento do *mouse*, que não chegou a atrapalhar a interação do usuário com o terminal.

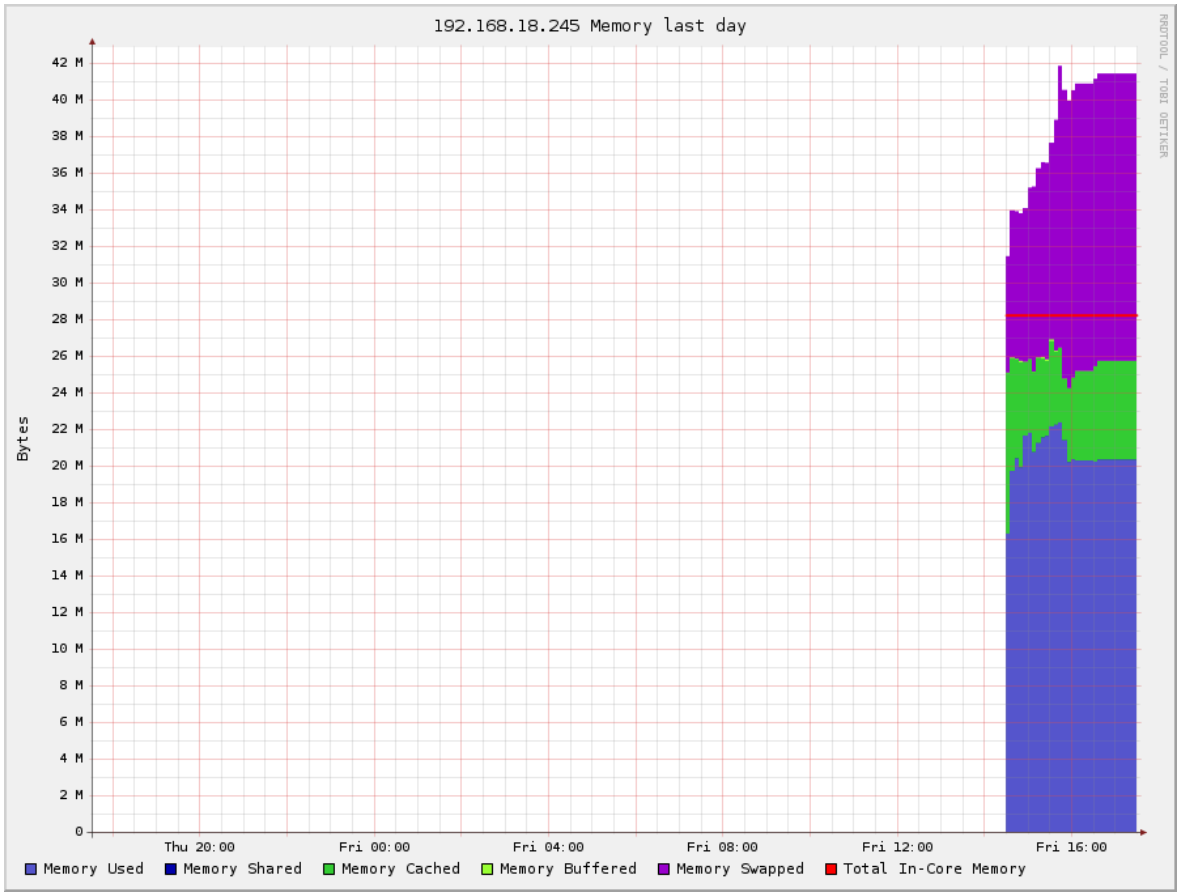

Figura 4.6: Gráfico do consumo de memória das máquinas com menor quantidade de memória.

A Figura 4.7 contém o gráfico que representa as máquinas 192.168.18.88 e 192.168.18.240 que possuem uma configuração média de quantidade de memória. Nele pode ser observado que o processo de *swap* não é realizado, o que evidência que 128 MB de memória RAM é uma quantidade razoável e suficiente a ser utilizada nos terminais.

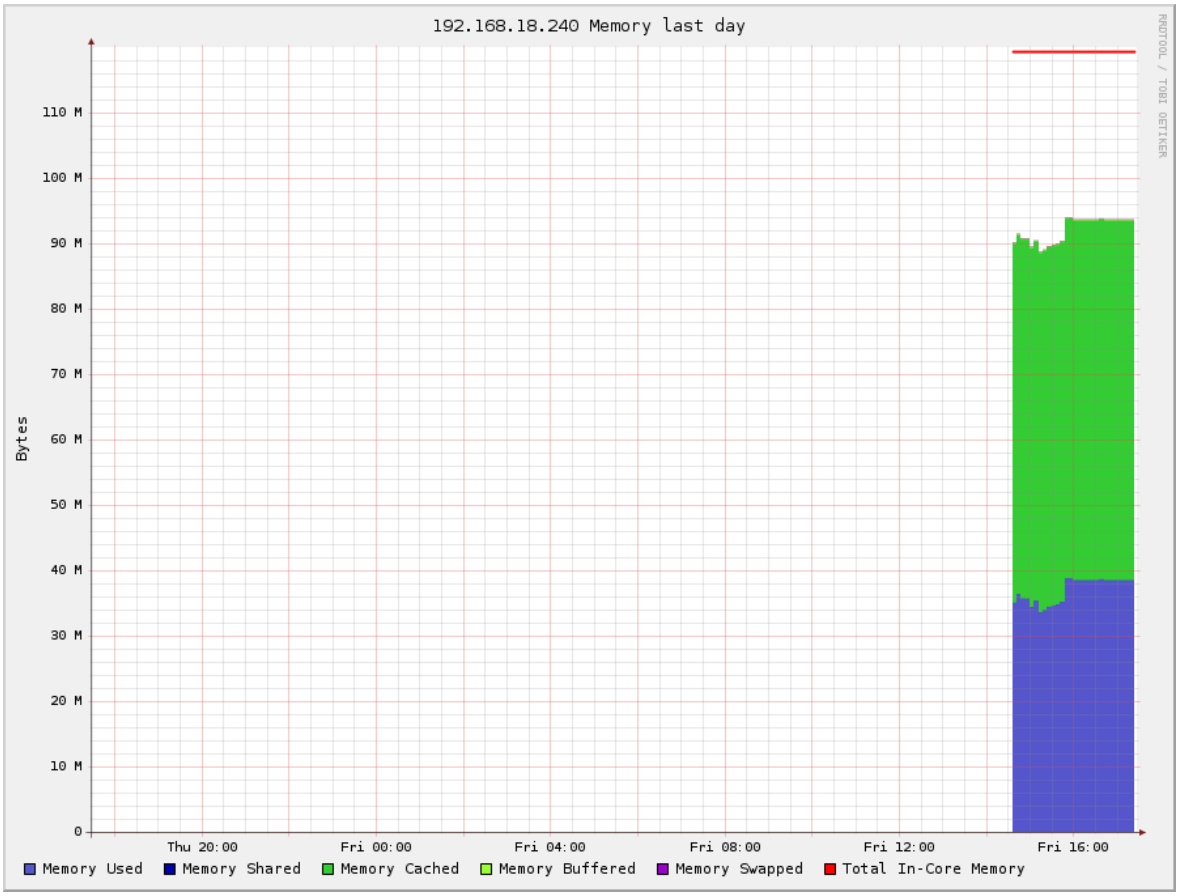

Figura 4.7: Gráfico do consumo de memória da máquinas 192.168.18.88 e 192.168.18.240.

Para comprovar que 128 MB de memória RAM pode ser considerado a quantidade mínima de memória desejada para que os terminais funcionem perfeitamente utilizamos a Figura 4.8 que representa o gráfico do consumo de memória da máquina 192.168.18.106. Por essa figura é possível notar que apesar da memória disponível estar bem acima dos 128 MB (*Total in-Core Memory*), somente ultrapassa essa quantidade em pequenos picos que correspondem ao período durante o qual se faz uso de aplicações multimídia.

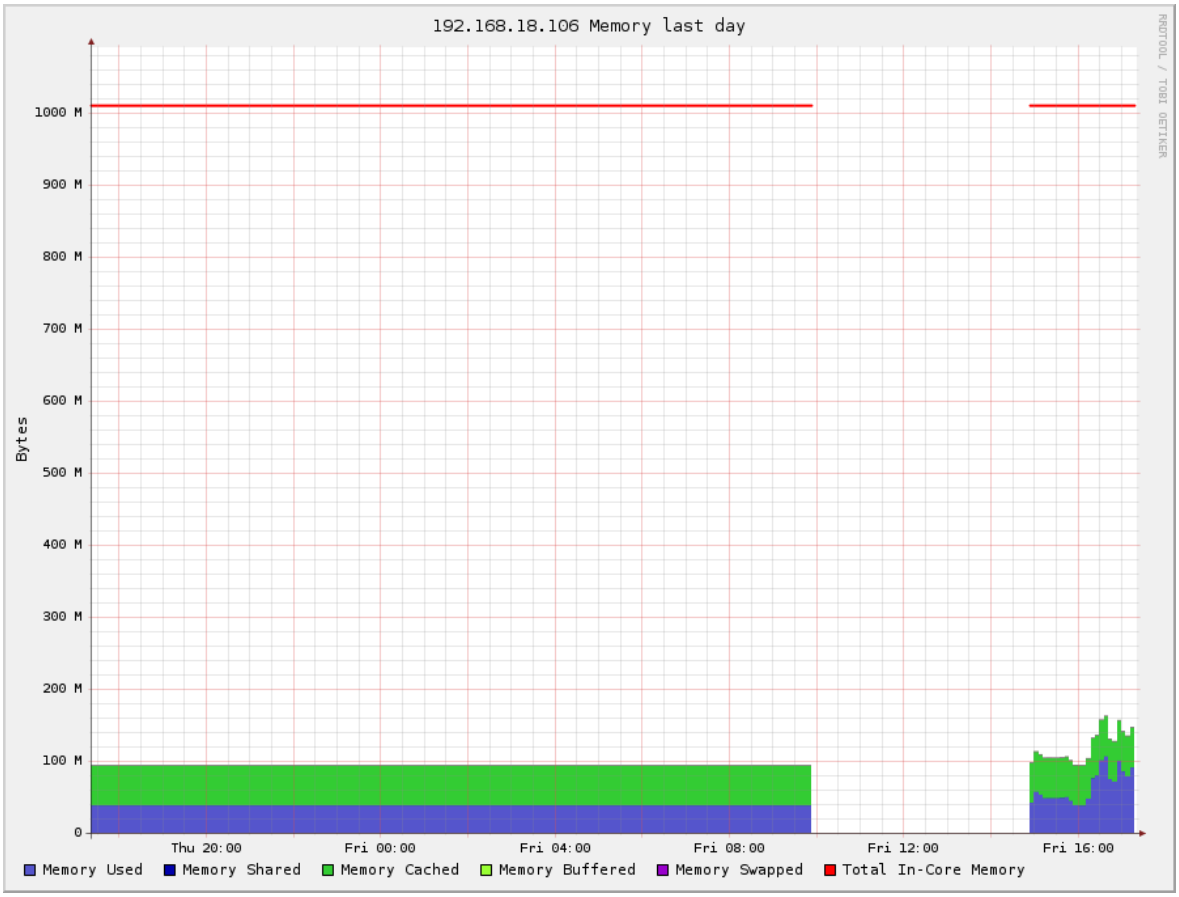

Figura 4.8: Gráfico do consumo de memória da máquina 192.168.18.106.

No gráfico da utilização da memória do servidor na Figura 4.9 ficou demonstrado que 2 GB são suficientes para atender os dez terminais utilizados no teste. É necessário salientar que a utilização efetiva da memória (*Memory Used*) não passou de 1 GB.

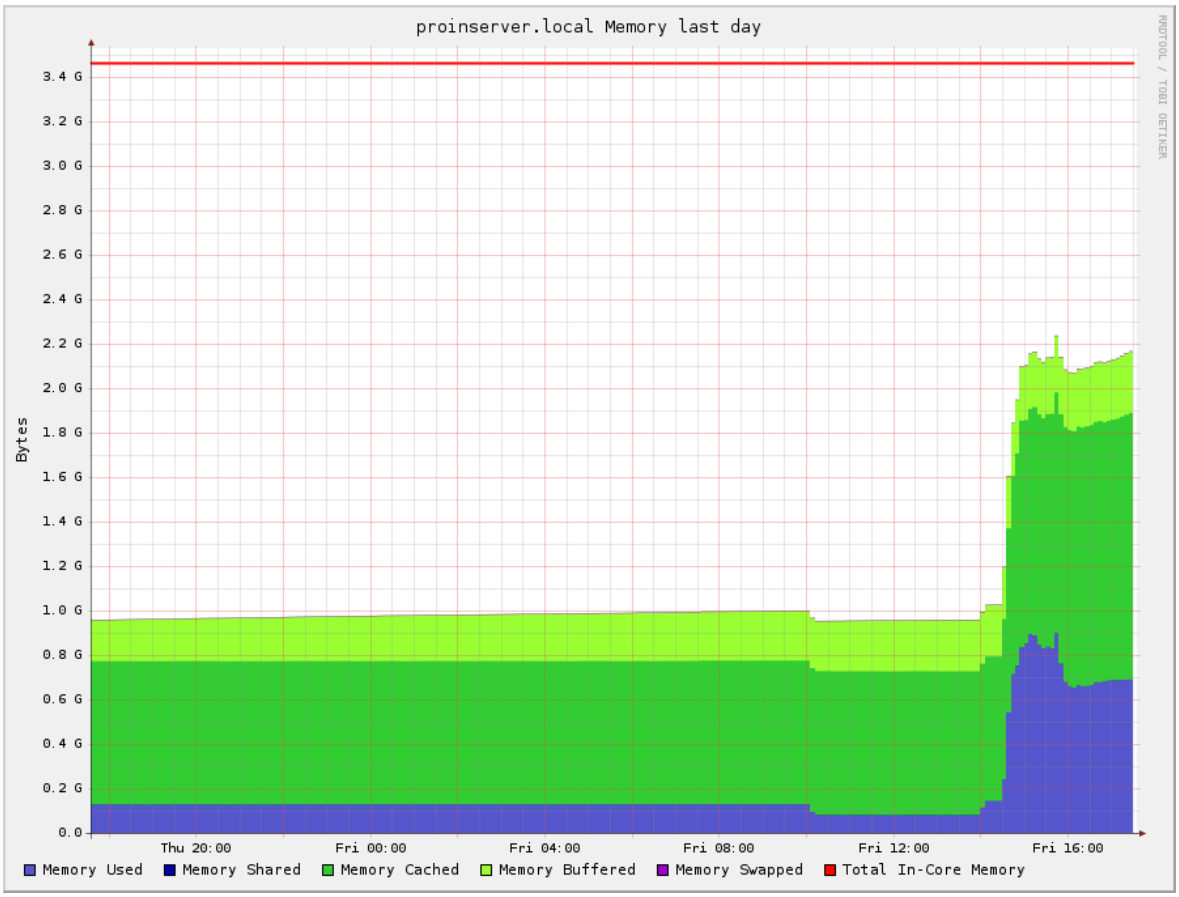

Figura 4.9: Gráfico do consumo de memória do servidor.

A última série de gráficos a serem exibidos correspondem a utilização da rede do sistema LTSP. Essa métrica é apontada como um dos fatores que mais influem na desempenho da solução. Durante a preparação do laboratório foi observado que a utilização de placas de 10Mbps era inviável, pois demora cerca de cinco minutos para o terminal chegar até a tela de *login.*

Nas figuras 4.10 e 4.11 apresentados os gráficos de consumo da banda de rede para um dos terminais exemplificando os demais e do servidor respectivamente. Pelos gráficos é inferido que o fluxo de dados do servidor é bem maior que os das estações. Essa configuração sugere que na impossibilidade de se ter uma rede Gigabit instalada é ideal que pelo menos a ligação entre o servidor e o *switch* suportem a taxa de 1 Gbps, pois dessa forma se reduz um dos gargalos do sistema.

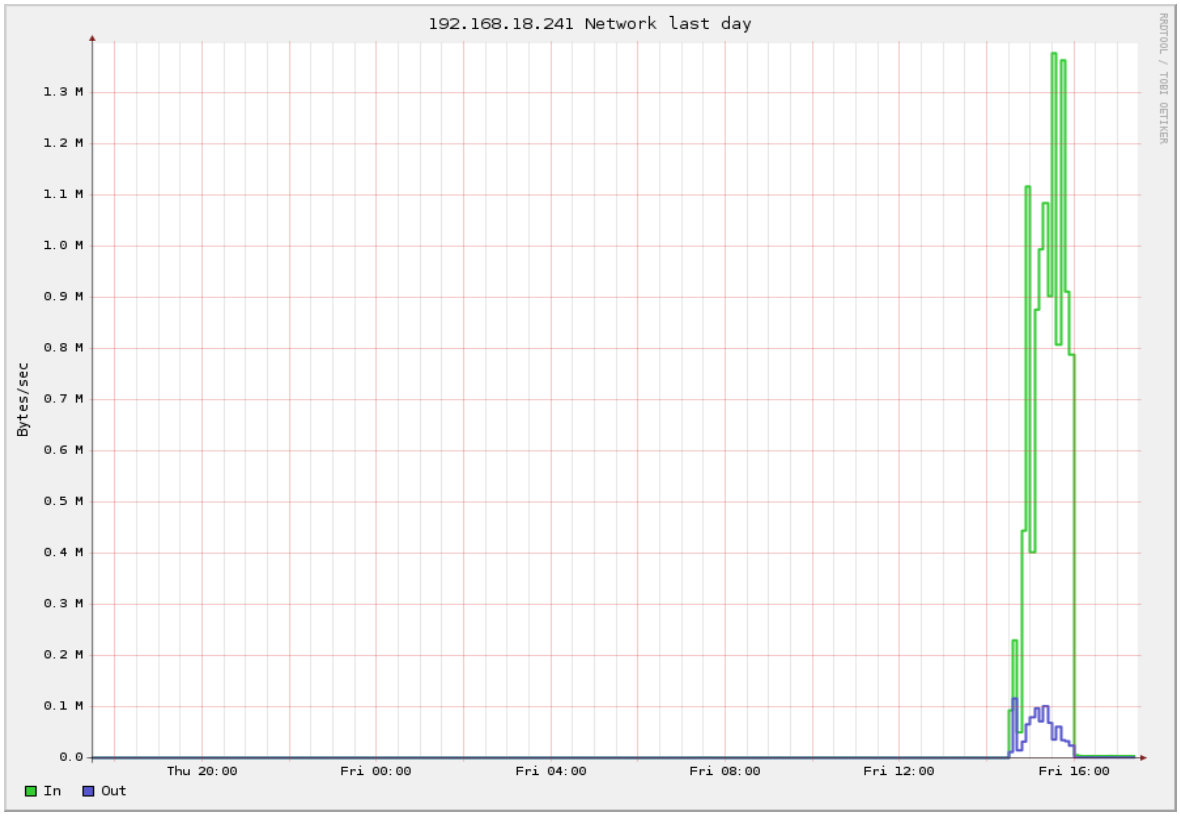

Figura 4.10: Gráfico do consumo de rede de um dos terminais.

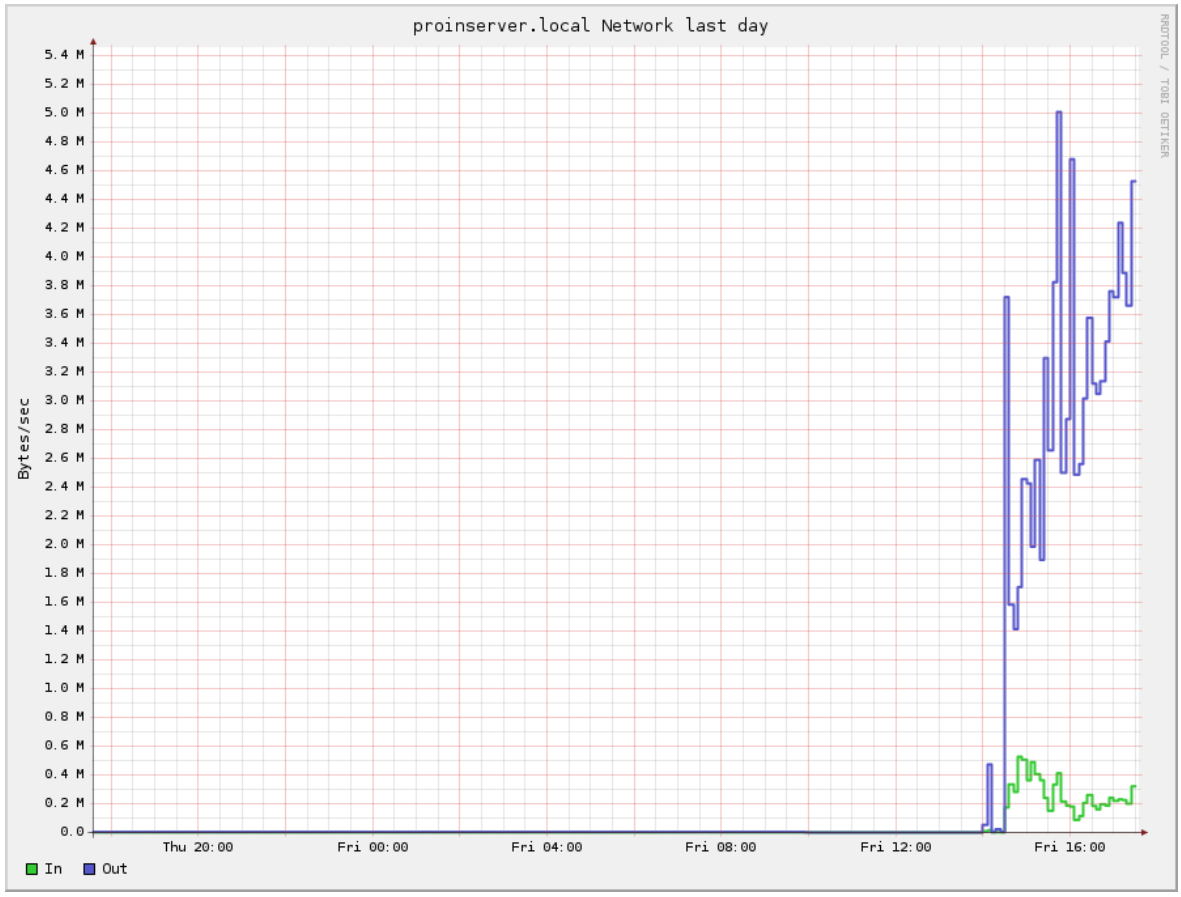

Figura 4.11: Gráfico do consumo de rede do Servidor.

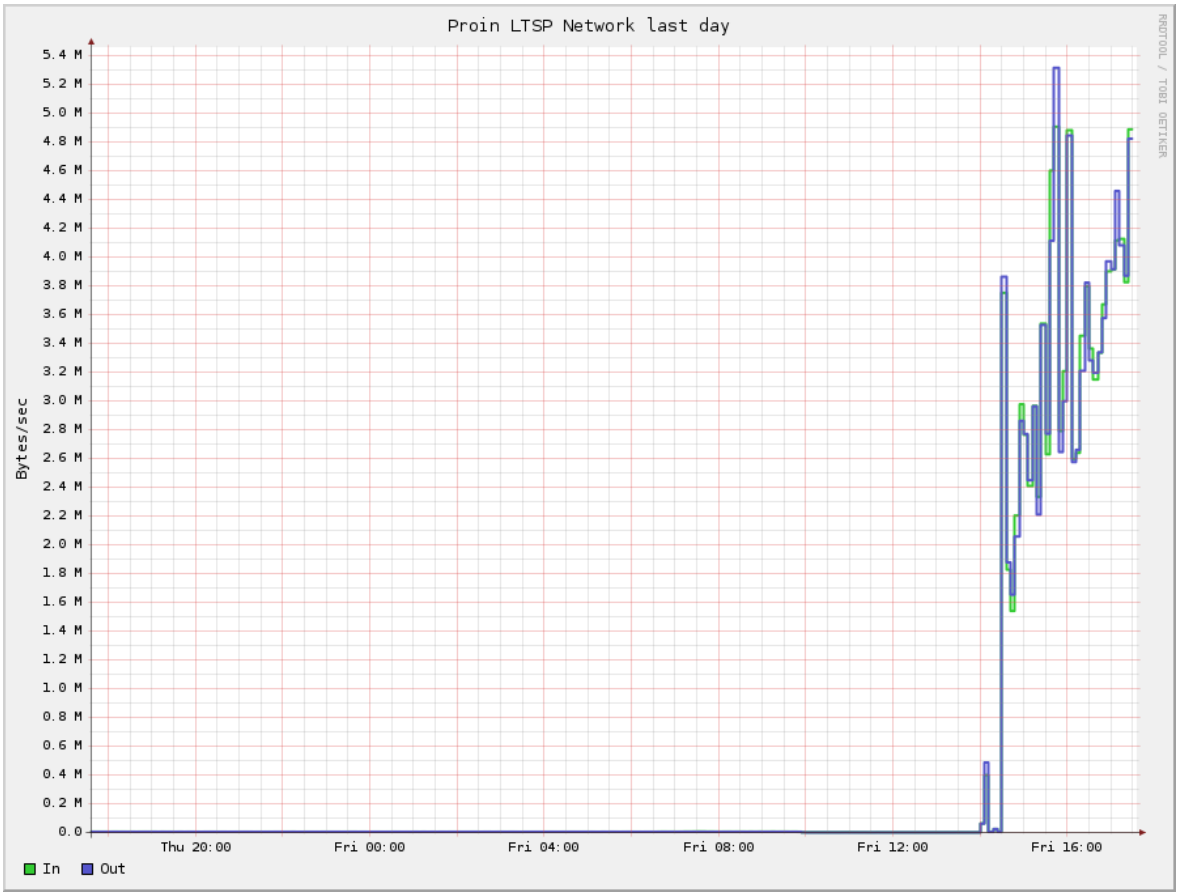

Figura 4.12: Gráfico que resume a utilização da rede pelo LTSP.

No gráfico da Figura 4.12 é apresentado um resumo do consumo da rede por todo o sistema LTSP, durante a simulação. Por ele percebe-se que no período mais crítico da utilização da rede foram gastos 5,3 MB/s de uma banda total de 11,9 MB/s, ou seja, o equivalente a 44% . Esse pico no consumo corresponde ao momento da visualização do vídeo indicado no roteiro. No entanto, durante o período de uso aleatório e o de utilização da suíte de escritório conforme o roteiro, o consumo da rede ficou entre 1,5 MB/s e 3,5 MB/s ou em porcentagem em relação ao total de banda disponível entre 12,6% e 29,44%. Esse resultado indica que se o laboratório for destinado a aplicações que não utilizem multimídia a solução suportaria a expansão para as 24 máquinas conforme foi planejado durante a fase montagem do laboratório.

Na seção a seguir é feito a conclusão deste trabalho assim como apresentadas algumas sugestões de trabalhos futuros.

5 CONCLUSÕES

Este trabalho propôs avaliar a utilização de terminais leves para se montar laboratórios de informática, analisando seu desempenho, como um alternativa ao modelo tradicional que faz uso de computadores pessoais completos, com dispositivos de armazenamento e sistema operacionais próprios, para cada estação de trabalho.

A principal contribuição deste trabalho é possibilitar um reaproveitamento de computadores, considerados obsoletos, como terminais para aumentar assim a sua vida útil. Foi realizada uma análise dos seguintes aspectos da solução proposta: facilidade de instalação do LTSP e de novos programas aos terminais, facilidade de configuração, redução dos gastos com manutenção, problemas encontrados na utilização de hardware obsoletos e por último fez-se uma avaliação de seu desempenho em uma simulação de uso.

Facilidade de instalação – com as mudança estruturais ocorridas entre a versão 4.2 e 5.0 o LTSP tornou-se uma ferramenta muito prática de ser instalada e configurada no GNU/Linux Ubuntu, pois na versão mais recente, sua instalação e configuração é realizada utilizando o programa nativo de instalação de programas do Ubuntu e completada por um *script* responsável por baixar a imagem do *kernel* a ser utilizada pelos terminais de acordo com sua arquitetura, utilizando como padrão a mesma arquitetura do servidor.

Instalação de novos programas - disponibilizar novos programas para todos os terminais também é um processo bastante prático, visto que basta instalar o novo programa no servidor e já estará pronto para ser utilizado pelos usuários dos terminais. Ao fazer essa instalação o administrador do sistema tem a opção de escolher como o novo programa instalado será executado. Ele pode optar pela execução dos novos programas no servidor ou localmente nas estações. Isso permite que se adéqüe o LTSP à utilização que se pretende fazer e aos terminais disponíveis. Dessa forma, se contamos com terminais mais robustos e aplicações que necessitam de maior poder de processamento, podemos optar por instalá-las de modo que executem localmente. Caso contrário, opta-se por deixar o programa executando no servidor.

Manutenção - como os terminais não possuem *softwares* instalados a tarefa de manutenção se reduz a verificação de problemas físicos nos *hardwares* dos terminais ou na estrutura da rede e a manter o servidor em funcionamento. Entretanto essa redução do trabalho, motivada pela centralização do processamento no servidor, implica que em caso de falha no servidor ou em alguma parte da rede que interliga o servidor aos terminais todos os terminais ficariam fora de funcionamento.

Problemas com os hardwares obsoletos – por se tratarem de equipamentos que utilizam uma tecnologia ultrapassada, qualquer tipo de manutenção se torna um problema, pois é difícil de se encontrar novas peças para reposição e quando encontradas o preço cobrado inviabiliza a compra. Esse problema é contornado mantendo-se algumas máquinas de reserva de forma que se substitua o terminal que apresentar um problema por uma das máquinas sobressalentes. Nesse caso pode-se considerar que a vida útil da máquina substituída chegou ao final.

Avaliação do desempenho – pela análise de desempenho pode-se observar que a utilização de terminais leves não é adequada para locais onde se fará uso intensivo de aplicações multimídia, pois essas aplicações além de aumentarem em muito o tráfego de dados na rede, também requerem um poder de processamento muito maior para que sejam visualizadas de forma satisfatória. Também foi observado que 128 MB de memória RAM é uma quantidade razoável e suficiente para os terminais.

Por outro lado o desempenho em aplicativos como suíte de escritório e navegador de internet foi satisfatório. De acordo com o questionário para os usuários o desempenho foi muito semelhante ao de soluções que utilizam computadores pessoais.

Conclui-se então que a viabilidade ou não da utilização de terminais leves para a montagem de laboratórios de informática está ligada ao tipo de aplicação que irá ser utilizada preferencialmente.

Como trabalhos futuros sugere-se: (a) um estudo comparativo entre a utilização de terminais leves e multiterminais; (b) uma avaliação da escalabilidade de soluções baseadas em terminais leves; (c) um estudo sobre a otimização de aplicações multimídia a fim de possibilitar sua utilização de forma satisfatória em terminais.

ANEXO A - ROTEIRO DO EXPERIMENTO

Abra o writer

Digitar o texto abaixo:

LTSP é a sigla para *Linux Terminal Server Project* ou projeto de servidores de terminais leves, um projeto de código aberto, criado e mantido por James McQuillan nos Estados Unidos em 1996. Hoje o projeto conta com a contribuição de vários desenvolvedores ao redor do mundo, inclusive do Brasil. O projeto visa reunir um conjunto de ferramentas administrativas para facilitar utilização de estações de trabalho de baixo custo, máquinas obsoletas ou thin client, como terminais de caracteres ou gráficos de um servidor GNU/Linux

Nas soluções baseadas em terminais leves o servidor fica com a parte pesada do trabalho, que é executar os programas, armazenar os dados e enviar para os clientes instruções para montar as janelas que serão exibidas. Os clientes montam as janelas e enviam os movimentos do mouse e as teclas digitadas no teclado

Abra o browser procurar uma imagem para o texto

Colar imagem no texto e salvar no desktop

Feche o writer.

Abra o Flash localizado em:

http://br.youtube.com/watch?v=um_8LY3KxkM&feature=bz301

Copie para a área de trabalho o arquivo localizado em:

/tmp/questionário.odt

Renomeie o arquivo para o nome da máquina que está utilizando

Abra novamente o writer e responda o questionário

Terminado de responder o questionário salve o arquivo na pasta /media/compartilhado

ANEXO B - QUESTIONÁRIO

Quanto a Máquina utilizada

Qual você estava utilizando?

Ela travou em algum momento?

Foi notado algum problema referente ao hardware?

Quanto ao desenvolvimento do Roteiro

Foi possível realizar todos os testes do roteiro?

Na sua opinião o writer demorou a abrir da primeira vez? E na segunda?

Como foi seu desempenho em relação ao writer utilizado em PC normal?

Pior, Igual, Melhor

Qual o desempenho do flash do roteiro?Demorou carregar, não carregou...

Abra outro flash a sua escolha colocando os dados de duração e link e sua opinião sobre o desempenho do flash em relação ou anterior.

Qual a nota você dá para o desempenho geral do LTSP?

Péssimo, ruim, regular, bom, ótimo

REFERÊNCIAS BIBLIOGRÁFICAS

CENTRO DE COMPUTAÇÃO CIENTÍFICA E SOFTWARE LIVRE. Universidade Federal do Paraná. Desenvolvido por: SANTOS, Douglas G dos; BUENO, Juliana, 2002-2008. Apresenta informações sobre o Centro de Computação Científica e Software Livre. Disponível em: <http://www.c3sl.ufpr.br>. Acesso em: 10 Mai. 2008.

COMER, Douglas. **Interligação de Redes com TCP/IP**. 5. ed. Rio de Janeiro: Editora Campus, 2006. 468 p.

DROMS R. **Dynamic Host Configuration Protocol**. Internet Engineering Task Force (IETF), Março 1997. (Request for Comments: 3121). Disponível em: <http://www.ietf.org/rfc/rfc2131.txt>. Acesso em: 21 Mar, 2008

DROMS, R.; Ed.; BOUND, J.; VOLZ, B.; LEMON, T.; PERKINS, C.; CARNEY, M. **Dynamic Host Configuration Protocol for IPv6 (DHCPv6)**. Internet Engineering Task Force (IETF), Julho 2003. (Request for Comments: 3315). Disponível em: <http://www.ietf.org/rfc/rfc3315.txt>. Acesso em: 21 Mar, 2008

FERRETI, Carlos. **Implantação de uma Rede de Computarores Baseada no Linux Terminal Server Project**. 2004. 72 p. Monografia de Conclusão de Curso de Pós-Graduação - Universidade Federal de Lavras, Lavras.

FILITTO, Danilo; OLIVEIRA JUNIOR, José R. **A relevância da Tecnologia LTSP na Inclusão Digital**. In: Saber Acadêmico - Revista Multidisciplinar da UNIESP. Presidente Prudente: UNIESP, 2007. Disponível em: <http://www.uniesp.edu.br/revista4/publi-art2.php?codigo=5>. Acesso:17 Mai. 2008.

GHOSH, Sukumar. **Distributed Systems: an algorithmic approach**. Boca Raton: Chapman & Hall, 2007. 402 p.

GIL, Antonio Carlos. **Como Elaborar Projetos de Pesquisa**. 3. ed. São Paulo: Editora Atlas, 1991. 159 p.

GNOME DISPLAY MANAGER REFERENCE MANUAL. **Gnome.org**. Desenvolvido por: PETERSEN, Martin K.; LEBL, George; CAMERON, Brian; HANEMAN, Bill, 2007. Site oficial do projeto GNOME. Disponível em: <http://www.gnome.org/projects/gdm/index.html>. Acesso em: 13 Mai. 2008.

INTERNET SYSTEMS CONSORTIUM. **Internet Systems Consortium, Inc**. Desenvolvido por: Richard Adams, 2003-2008. Organização dedicada a dar suporte de infra-estrutura para organização e conexão universal da internet. Disponível em: <http://www.isc.org>. Acesso em: 07 Dez. 2007.

JAVVIN TECHNOLOGIES. **Network Protocols Handbook**. . Saratoga: Javvin Technologies, 2005. 340 p.

JUNG, Carlos Fernando. **Metodologia para pesquisa & desenvolvimento aplicada a novas tecnologias, produtos e processos**. Rio de Janeiro: Axcel Books, 2004. 312 p.

KAWASHIMA, Celso M. **Multi-Terminais Linux**. 2006. 92 p. Monografia de Conclusão de Curso de Pós-Graduação - Universidade Federal de Lavras, Lavras.

KOSLOVSKI, Guilherme P.; BOUFLEUR, Márcio P.; CHARÃO, Andrea S. **Uso de Virtualização de Recursos Computacionais na Administração de Redes**. In: [ERRC] IV - Escola Regional de Redes de Computadores. Passo Fundo: SBC-Sociedade Brasileira de Computação, 2006. 6 p. Disponível em: <http://bibliotecadigital.sbc.org.br/download.php?paper=961>. Acesso:08 Mai. 2008.

KOSLOVSK, Guilherme P.; BOUFLEUR, Márcio P.; CHARÃO, Andrea S. **Módulo de Descoberta Automática de Monitores de Máquinas Virtuais Xen**. In:VIII Workshop sobre Software Livre -WSL 2007, Porto Alegre, p. 31-36, Abril 2007.

LAI, Albert. NIEH, Jason.. **Limits of wide-area thin-client computing**. In: SIGMETRICS '02: Proceedings of the 2002 ACM Sigmetrics International Conference on Measurement and Modeling of Computer Systems. New York: ACM Press, 2002. p. 228-239. Disponível em: <http://portal.acm.org/citation.cfm?doid=511334.511363#>. Acesso:06 Fev. 2007.

LAUREANO, Marcos A. P.; MAZIERO, Carlos A.; JAMHOUR, Edgard. **Detecção de Intrusão em Máquinas Virtuais**. In: 5º SSI - Simpósio de Segurança em Informática. São José dos Campos: , 2003. p. 101-107. Disponível em: <http://www.mlaureano.org/projects/vmids/vmids_ssi.pdf>. Acesso:06 Mai. 2008.

LINUX TERMINAL SERVER PROJECT. **LTSP.org**. Desenvolvido por: LTSP.org, 2008. Site oficial do projeto LTSP. Disponível em: <http://www.ltsp.org>. Acesso em: 21 Mar. 2008.

MARTINS, Jonsue Trapp. **Criação de Ambientes Distribuídos com LTSP**. 2004. 71 p. Monografia de Especialização em Informática - Universidade Federal do Paraná, Curitiba.

MORIMOTO, Carlos E. **Dicionário de Termos Técnicos de Informática**. In: e-book, GDH Press e Sul Editores, 2008a. 397 p. Disponível em: <http://www.guiadohardware.net/e-books/download/Dicionario_de_Termos_de_informatica-3ed.pdf>. Acesso:21 Mar, 2008.

MORIMOTO, Carlos E. **Usando o Windows Terminal Server (WTS)**. In: Tutorial, Guia do Hardware, 2008b. Disponível em: <http://www.guiadohardware.net/tutoriais/wts/>. Acesso:21 Mar, 2008.

MORIMOTO, Carlos E. **Terminais Leves com LTSP 4,2**. In: Guia, Guia do Hardware, 2006. Disponível em: <http://www.guiadohardware.net/guias/17/index1.php>. Acesso:28 Mai, 2008.

MULTISEAT - C3SL. **Centro de Computação Científica e Software Livre**. 2008. Wiki com informações sobre o desenvolvimento da tecnologia de multiterminais. Disponível em: <http://wiki.c3sl.ufpr.br/multiseat/index.php/Main_Page/pt-br>. Acesso em: 10 Mai. 2008.

NEMETH, Evi. SNYDER, Garth. HEIN, Trent. **Linux Administration Handbook**. Upper Saddle River: Prentice Hall PTR, 2002. 890 p.

NEVES, Macelo V.; SCHEID, Tiago; CHARÃO, Andrea S. **Monitoração de Clusters com a ferramenta Ganglia: avaliação e adaptação**. In: 6º Workshop sobre Software Livre, Porto Alegre, p. 271-276, Junho 2005.

NORTON, Peter. **Intrudução a informática**. São Paulo: Pearson; Makron Books, 1997. 619 p.

OLIVEIRA, Ander C. de; VIGNATTI, Tiago; WEIGAERTNER, Daniel; SILVA, Fabiano; CASTILHO, Marcos; SUNEY, Marcos. **Um modelo de computação multiusuário baseado em computadores pessoais**. In: VII Workshop sobre Software Livre - WSL 2006, Porto Alegre, p. 135-140, Abril 2006.

PACKARD, Keith. X Consortium. " **X Display Manager Control Protocol**". Laboratory for Computer ScienceMassachusetts Institute of Technology. Disponível em: <http://www.x.org/docs/XDMCP/>. Acesso em: 9 Abr. 2008

ROM-O-MATIC. Rom-o-matic. Desenvolvido por: **Rom-o-matic**, 2008. Site com um gerador de imagens de boot para diversas marcas e modelos de placas de rede. Disponível em: <http://rom-o-matic.net/>. Acesso em: 22 Mai 2008.

SOLLINS, K. **The TFTP Protocol (Revision 2)**. Internet Engineering Task Force (IETF), Julho 1992. (Request for Comments: 1350). Disponível em: <http://www.ietf.org/rfc/rfc1350.txt>. Acesso em: 22 Mar. 2008

SOUZA FILHO, Joanilo de. **Análise comparativa entre ambientes computacionais baseados nas tecnologias de processamento distribuído e de thin client**. 2003. 78 p. Trabalho de Conclusão de Curso - Universidade Regional de Blumenau, Blumenau.

TANENBAUM, Andrew S. **Distributed operating systems**. Upper Saddle River: Prentice-Hall, 1995. 614 p.

TANENBAUM, Andrew S. **Modern operating systems**. 2nd ed. Upper Saddle River: Prentice Hall, 2001. 951 p.

TANENBAUM, Andrew S. **Organização estruturada de computadores**. 5. ed. São Paulo: Pearson Prentice Hall, 2007. p.

TECNOWORLD. **Tecnoworld**. Desenvolvido por: GOMES, Geraldo F, 2008. Site instituicional da empresa Tecnoworld. Disponível em: <http://tecnoworld.tempsite.ws/>. Acesso em: 22 Mai. 2008.

TOMIYAMA, Michele N. **Desenvolvimento de um Sistema de Rastreabilidade para a Cachaça de Minas Gerais**. 2004. 79 p. Monografia de Conclusão de Curso - Universidade Federal de Lavras, Lavras.

TYKHOMYROV, Olexiy. TONKONOG, Denis. **Starting Share Files with NFS**. In: Linux Journal : Belltown Media, 2002. . Disponível em: <http://www.linuxjournal.com/article/4880>. Acesso:05 Dez. 2007.

UBUNTU DOCUMENTATION. **Canonical LTDA**. Desenvolvido por: CANONICAL, 2007. Documentação sobre LTSP na distribuição UBUNTU. Disponível em: <https://help.ubuntu.com/community/UbuntuLTSP>. Acesso em: 17 Mai. 2008.

XDM - X DISPLAY MANAGE WITH SUPPORT FOR XDMCP. **MIT X Consortium**. Desenvolvido por: PACHARD, Keith, 2008. Site com informações sobre o XDM. Disponível em: <http://www.xfree86.org/current/xdm.1.html#sect25>. Acesso em: 13 Mai. 2008.

YLONEN, T.; LONVICK C.; ED.; **The Secure Shell (SSH) Protocol Architecture**. Internet Engineering Task Force (IETF), 2006a. (Request for Comments: 4251). Disponível em: <http://www.ietf.org/rfc/rfc4251.txt>. Acesso em: 29 Mar. 2008

YLONEN, T.; LONVICK, C.; Ed.; **The Secure Shell (SSH) Authentication Protocol**. Internet Engineering Task Force (IETF), 2006b. (Request for Comments: 4253). Disponível em: <http://www.ietf.org/rfc/rfc4253.txt>. Acesso em: 29 Mar. 2008

YLONEN, T.; LONVICK, C.; Ed.; **The Secure Shell (SSH) Authentication Protocol**. Internet Engineering Task Force (IETF), 2006c. (Request for Comments: 4252). Disponível em: <http://www.ietf.org/rfc/rfc4252.txt>. Acesso em: 29 Marc. 2008

YLONEN, T.; LONVICK, C.; Ed.; **The Secure Shell (SSH) Connection Protocol**. Internet Engineering Task Force (IETF), 2006d. (Request for Comments: 4254). Disponível em: <http://www.ietf.org/rfc/rfc4254.txt>. Acesso em: 29 Mar. 2008

ZANONI, Paulo R.; VIGNATTI, Tiago; OLIVEIRA, Ander C. de; SILVA, Fabiano; BONA, Luis C. Erpen. **O Estado Atual dos Multiterminais.** In: IX Workshop sobre Software Livre - WSL 2008, Porto Alegre, p. 6, Abril 2008.