Universidade Federal do Piauí
Campus Senador Helvídio Nunes de Barros
Curso de Bacharelado em Sistemas de Informação

Varton Jose Bezerra Junior

Monitoramento de Servidores *Linux* na Plataforma Android

PICOS
2014

Varton Jose Bezerra Junior

Monitoramento de Servidores *Linux* na Plataforma Android

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Sistemas de Informação, Campus Senador Helvídio Nunes de Barros da Universidade Federal do Piauí, como parte dos requisitos para obtenção do Grau de Bacharel em Sistemas de Informação, sob orientação de Rayner Gomes Sousa.

PICOS

2014

Eu, **Varton José Bezerra Júnior**, abaixo identificado(a) como autor(a), autorizo a biblioteca da Universidade Federal do Piauí a divulgar, gratuitamente, sem ressarcimento de direitos autorais, o texto integral da publicação abaixo discriminada, de minha autoria, em seu site, em formato PDF, para fins de leitura e/ou impressão, a partir da data de hoje.

Picos-PI 13 de março de 2014.

Varton José Bezerra Júnior
Assinatura

FICHA CATALOGRÁFICA
Serviço de Processamento Técnico da Universidade Federal do Piauí
Biblioteca José Albano de Macêdo

B574m Bezerra Júnior, Varton José.
Monitoramento de servidores linux na plataforma android / Varton José Bezerra Júnior. – 2013.
CD-ROM : il. ; 4 ¾ pol. (51 p.)

Monografia(Bacharelado em Sistemas de Informação) – Universidade Federal do Piauí. Picos-PI, 2013.
Orientador(A): Prof. MSc. Rayner Gomes Sousa

1. Comunicação. 2. Android. 3. Monitoramento. 4. Servidores Linux. I. Título.

CDD 005.43

Varton Jose Bezerra Junior

Monitoramento de Servidores *Linux* na Plataforma Android

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Sistemas de Informação, Campus Senador Helvídio Nunes de Barros da Universidade Federal do Piauí, como parte dos requisitos para obtenção do Grau de Bacharel em Sistemas de Informação, sob orientação de Rayner Gomes Sousa.

Data de Aprovação:

Prof. MSc. Juliana Oliveira de Carvalho _____ UFPI

Prof. MSc. Patricia Medyna Lauritzen de Lucena Drumond _____ UFPI

Prof. ESp. Leonardo Pereira de Sousa _____ UFPI

PICOS

2014

Dedico esta monografia a toda minha família em especial aos meus pais Varton Jose Bezerra e Maria Vilma de A. Bezerra aos meus irmãos Lidiane Alencar Bezerra e Gladstone de Alencar Bezerra por estarem sempre à meu lado em todas as minhas conquistas e decisões.

Agradeço primeiramente a Deus, pelo dom da vida, pela minha capacidade de pensar, ter me proporcionado mais uma vitória e por me guiar nessa jornada de conquista.

Agradeço a toda minha família pela paciência e por acreditar em meu trabalho, mesmo com as dificuldades encontradas, a meu cunhado Luciano e a minha cunhada Josefa Regineide aos meus tios em especial Gardênia e Jose Davi e primos entre eles Fabrício Bezerra pelos seus conselhos agradeço também a toda turma: Ana Samila, Dedilsa, Eliene Alves, Jussara Isabel, Valeria Hipolito e Lina Mara por todo apoio.

Aos professores que compõem o curso de Sistemas de Informação, que contribuíram através de seus conhecimentos repassados: Algeir, Arlino, Dennis Sávio, Frank César, Fredison, Ismael, Ivenilton, Juliana, Leonardo, Patricia Medyna, Patricia Vieira, Romuere.

Aos meus colegas, amigos do curso, da UFPI, em especial a equipe do LIPPO, Allan, Marco Antonio, Wilson, Bruno Rafael, Cliciano, Mezzomo, Klisanderson, Auricélio e Paula Michelle obrigado por estarem presente durante toda essa jornada.

Nossa maior fraqueza está em desistir. O caminho mais certo de vencer é tentar mais uma vez.

Thomas Edison

É melhor atirar-se à luta em busca de dias melhores, mesmo correndo o risco de perder tudo, do que permanecer estático, como os pobres de espírito, que não lutam, mas também não vencem, que não conhecem a dor da derrota, nem a glória de ressurgir dos escombros. Esses pobres de espírito, ao final de sua jornada na Terra não agradecem a Deus por terem vivido, mas desculpam-se perante Ele, por terem apenas passado pela vida.

Bob Marley

Pouco conhecimento faz com que as pessoas se sintam orgulhosas. Muito conhecimento, que se sintam humildes. É assim que as espigas sem grãos erguem desdenhosamente a cabeça para o Céu, enquanto que as cheias as baixam para a terra, sua mãe.

Leonardo da Vinci

RESUMO

Este trabalho apresenta o desenvolvimento de um aplicativo que auxilia o gerente de redes no monitoramento de sistemas *Linux* através de dispositivo móvel utilizando a plataforma Android. Para o funcionamento do projeto proposto, foram desenvolvidas duas aplicações: gerente e agente. A aplicação gerente é instalada e executada em um dispositivo móvel com sistema operacional Android 4.1 e a aplicação agente é instalado em uma máquina servidor conectada a uma rede local ou/a *Internet* e responde as requisições efetuadas pela aplicação gerente. As aplicações foram testadas e os resultados experimentais indicaram eficiência, sendo capazes de auxiliar o gerente de redes a partir da identificação dos resultados efetuando a melhor decisão sobre o funcionamento da máquina servidor.

Palavras-chave: Comunicação, Android, Monitoramento, Servidor *Linux*.

ABSTRACT

This paper presents the development of an application that assists the network manager in the monitoring of Linux systems via mobile using the Android platform. For the functioning of the proposed project, two applications were developed: manager and. The manager application is installed and run on a mobile device with operating system Android 4.1 and application agent is installed on a server machine connected to a local or network / Internet and answers the requests made by the application manager. The applications were tested and experimental results indicated efficiency, being able to assist the network manager from the identification of outcomes making the best decision on the functioning of the server machine.

Keywords: Communications, Android, Monitoring, Linux Server.

Lista de Figuras

Lista de Tabelas

Lista de abreviaturas e siglas

CPU	unidade central de processamento
HD	Disco Rigido
IP	Protocolo de Internet
JDK	*Java* Development Kit
JDT	Ferramentas de desenvolvimento Java
JRE	Java Runtime Environment
JVM	Máquina virtual *Java*
OHA	Open Handset Alliance
SDK	kit de desenvolvimento
SO	Sistema Operacional
TCP	Protocolo de controle de transmissão
UDP	User Datagram Protocol
VM	Máquina Virtual
XML	eXtensible Markup Language

Sumário

1 Introdução

Com a popularização da *Internet*, surgiu nas empresas, geralmente de grande ou médio porte, a necessidade de uma rede interna de computadores (rede de computadores) com o propósito de executarem os seus próprios serviços de e-mail, distribuição de sinal de *Internet*, armazenamento de arquivos, entre outros. Para realização desses serviços e necessário a utilização de máquinas (servidores) que disponham de uma potente configuração de *hardware* visando obter um maior desempenho na realização dessas tarefas. Para que a execução desses serviços ocorra de forma adequada é preciso que um administrador de redes monitore o funcionamento deste servidor periodicamente, para que, caso algum problema esteja ocorrendo, este possa ser detectado o quanto antes e resolvido sem causar grandes transtornos aos usuários desses serviços, garantindo assim a qualidade do serviço que está sendo disponibilizado.

O monitoramento de servidores é geralmente realizado em rede local, onde são utilizados *softwares* específicos como DSTAT[1] , CACTI[2] e NAGIOS[3], utilizados para captar informações acerca dos serviços disponibilizados, que são mostradas na tela com intuito de auxiliar o administrador da rede a identificar e prevenir possíveis falhas. Este monitoramento também pode ser realizado de maneira remota, onde, em qualquer lugar que haja uma conexão com a *Internet* é possível acessar o servidor a ser monitorado, inclusive através de dispositivos móveis, como celular conectado à *Internet* WiFi e 3G. O uso de celular para realizar tarefas, que até então só eram desempenhadas por computadores se torna cada vez mais comum na medida em que as tecnologias aplicadas a estes aparelhos crescem, seus *hardwares* evoluem, *softwares* mais complexos e flexíveis são desenvolvidos e esses utilizam um Sistema Operacional (SO), de forma similar aos computadores.

A Google desenvolveu um SO, chamada Android que vem destacando-se devido a sua eficiência e simples usabilidade, além de possuir uma grande quantidade de aplicativos disponíveis.

> "Android é a plataforma e\ou Sistema Operacional para Dispositivos Móveis da empresa Google. A base do Sistema Operacional conta um *Kernel Linux* na versão 2.6, é neste lugar onde esta presente os serviços de segurança, gerenciamento de memória, e outros processos sistémicos, de uma forma clara, esse *Kernel* é que realiza a comunicação entre o *hardware* e o *software* Android SDK (SOARES, 2011)".

Este trabalho de conclusão de curso propõe o desenvolvimento de um sistema composto por um aplicativo na plataforma Android, com intuito de auxiliar o gerente de redes no gerenciamento de recursos de servidores *Linux*. O mesmo terá acesso remoto ao servidor que se deseja

[1] DSTAT está acessível a partir de: http://dag.wieers.com/home-made/dstat/
[2] CACTI está acessível a partir de: http://www.cacti.net/
[3] NAGIOS esta acessível a partir de: http://www.nagios.org/

monitorar através de um dispositivo móvel, onde será possível obter as informações necessárias para a tomada de decisões.

1.1 Motivação

A motivação para o desenvolvimento desse trabalho surgiu da análise das atividades desempenhadas pelo gerente da rede (gerenciamento de memória, processamento e disco), onde percebe-se que para ter acesso a essas informações é necessário estar na frente de um computador conectado a *Internet*. Diante dessa análise o desenvolvimento desse trabalho tem como intuito proporcionar ao gerente de redes a mobilidade e acessibilidade às informações necessárias para o gerenciamento de seu servidor tornando possível a análise de possíveis problemas que venham a ocorrer e consequentemente facilitar a tomada de decisões, sendo possível acessá-la de qualquer local e a qualquer hora, sendo preciso apenas ter um dispositivo móvel com sistema Android e acesso à *Internet*.

1.2 Objetivos

1.2.1 Objetivos Gerais

O objetivo desse trabalho é desenvolver um aplicativo que auxiliará o gerente de redes no gerenciamento e monitoramento de sistemas *Linux* através de celular utilizando a plataforma Android.

1.2.2 Objetivos Específicos

Como resultado proveniente do desenvolvimento dessa aplicação, pretende-se alcançar os seguintes objetivos específicos:

- Monitorar o uso da CPU para determinar as taxas de uso;
- Monitorar o HD (*HardDisc*) para leitura e escrita do disco;
- Monitorar a quantidade de memória física e virtual(swap) do sistema.

1.3 Estrutura da Monografia

Capítulo 2 - É apresentado o referencial teórico, contendo os conceitos no qual o presente trabalho esta fundamentado. Onde se descreve sobre Gerenciamento de Redes, a impor-

tância das redes e trabalhos correlacionados.

Capítulo 3 - É apresentado a modelagem do sistema onde se descreve os requisitos funcionais e específicos da plataforma para o monitoramento da rede através de diagramas de: classe, caso-de-uso e de sequência, apresenta ainda o projeto e sua fase de desenvolvimento.

Capítulo 4 - É descrito o ambiente ao qual foi utilizado para a realização dos testes e a forma como foram realizados, em seguida apresenta-se os resultados obtidos.

Capítulo 5 - Por fim, o capítulo de considerações finais mostra os objetivos que foram alcançados após o desenvolvimento do trabalho, com sugestões para trabalhos futuros.

2 Referencial Teórico

Este capítulo destina-se ao estudo das tecnologias envolvidas no desenvolvimento deste trabalho, contendo conceitos no qual o presente trabalho está fundamentado, onde se descreve sobre o Gerenciamento de Redes, Protocolos de Gerenciamento e a Plataforma Android. As subseções a seguir mostram detalhes de cada tópico citado.

2.1 Monitoramento de Rede

As redes de computadores tem se tornado cada vez mais importantes para as empresas e instituições, sendo indispensáveis para gerir o funcionamento da empresas, encurtando assim a distância e diminuindo o tempo de resposta entre as transações de informações das organizações de todo o mundo. Ao longo do tempo e com a diminuição dos preços dos equipamentos, as redes tornaram-se parte do cotidiano dos usuários, o que ocasionou o seu rápido crescimento. Tais redes têm se tornado cada vez maior englobando diferentes tipos de dispositivos, além dos computadores de mesa podem fazer parte da rede *notebooks*, impressoras, celulares, roteadores, entre outros.

Por trás de todo o funcionamento das redes de computadores estão os servidores, que são máquinas mais potentes com função de fornecer serviços à rede de computador, seja ele serviço de e-mail ou mesmo de distribuição de sinal de Internet. Para o bom funcionamento de uma rede, não é preciso que estes servidores estejam apenas funcionando, eles também devem estar executando suas tarefas de forma perfeita, sendo assim é necessário ter alguém com conhecimento para monitorar e dar suporte a essa rede (SZTAJNBERG, 1996).

Esse monitoramento é feito pelo gerente de rede que tem como função projetar e manter uma rede de computadores funcionando corretamente. Quando o responsável por esta tarefa está presente na empresa os dados desejados são acessados de maneira local geralmente através de *softwares* como: DSTAT, CACTI e NAGIOS, porém nem sempre o gerente de rede pode estar presente no local de trabalho e então para executar esta função ele pode fazer um monitoramento a distância através da Internet onde acessa de maneira remota a sua central de trabalho obtendo as informações que ele precisa para manter a rede funcionando.

2.2 Gerenciamento de Redes

O gerenciamento de uma rede permite a detecção e correção de problemas que tornam a comunicação ineficiente ou impossível. Com o crescimento do número e da heterogeneidade dos equipamentos envolvidos nas redes, o número de problemas potenciais e a complexidade envolvida nestes problemas tornam-se críticos, e exigem que os gerentes de rede possuam uma vasta quantidade de informação sobre as redes administradas e seus problemas. Assim, o gerenciamento de redes destina-se a auxiliar os gerentes a trabalhar com a complexidade dos dados envolvidos, de modo a garantir a máxima eficiência e transparência da rede para os seus usuários (MELCHIORS; CRISTINA, 2010).

> Gerenciamento de redes é o controle de qualquer objeto passível de ser monitorado numa estrutura de recursos de redes físicos ou lógicos distribuídos em diversos ambientes. O gerenciamento de uma rede de computadores torna-se uma atividade essencial para garantir o seu funcionamento continuo assim como para assegurar um elevado grau de qualidade dos serviços oferecido (TANENBAUM, 2003).

.

O gerenciamento de uma rede inclui a coordenação de elementos de *hardware*, esse controle é obtido através de *software* e humanos, com intuito de monitorar, testar, consultar, analisar, avaliar os recursos de *hardware* da máquina servidora da rede, obtendo as informações necessárias para a tomada de decisões (KUROSE, 2010).

Para uma melhor compreensão sobre o conceito de gerenciamento de redes, vejamos a especificação de alguns termos, são eles: Mecanismo Gerenciado, Ambiente Gerenciado, Sistema de Gerenciamento, Gerente, Agente e Protocolos de Gerenciamento.

2.2.1 Mecanismo Gerenciado

Mecanismo gerenciado pode ser entendido como um elemento que faz parte da rede e que tenha necessidade de gerência ou monitoramento. Segundo Werner (2010) mecanismo compreende tanto *hardware* como *software* que apresente necessidade e condições de ser gerenciado. Como exemplo de *hardware* pode-se citar: interfaces de rede, discos e impressoras. O *software* compreende aspectos relacionados à implementação da pilha TCP/IP, como por exemplo, estatísticas sobre o processamento de datagramas IP.

2.2.2 Ambiente Gerenciado

Com o contiínuo crescimento em número e diversidade dos componentes das redes de computadores tornou a atividade de seu gerenciamento muito mais complexa. Em vista disto, impõe-se a necessidade de um ambiente que contenha mecanismos que apoiem este processo

de diagnóstico de falhas, erros e etc. O ambiente deverá ser constituído de dispositivo (roteadores, servidores de acesso, impressoras, computadores, servidores de rede) com aspectos de comunicação através de protocolos.

> "Compreende mecanismos com suporte a funcionalidades de gerenciamento, juntamente com os aspectos de comunicação que permitem suas interconexões. Desta forma, o ambiente gerenciado pode ser constituído de um ou mais mecanismos. Como exemplos, pode-se citar: roteador, dispositivos gerenciáveis de uma mesma sub-rede, dispositivos gerenciáveis de um conjunto de LANs interligadas (WERNER, 2010)".

Entretanto é o ambiente que será gerenciado, pois este pode ser de qualquer dimensão e conter diversos objetos, no entanto nem todos precisam ser necessariamente gerenciados.

2.2.3 Sistema de Gerenciamento

O sistema de gerenciamento é um conjunto composto por (seres humanos, objeto gerenciado, gerente e agente) que são utilizados no monitoramento e controle da rede.

> "Um sistema de gerência de rede pode ser definido como um conjunto de ferramentas integradas para o monitoramento e controle, que oferece uma interface única e que traz informações sobre o status da rede podendo oferecer ainda um conjunto de comandos que visam executar praticamente todas as atividades de gerenciamento sobre o sistema em questão (PINHEIRO, 2006)".

Segundo Werner (2010) o sistema de gerenciamento de rede é formado por uma coleção de ferramentas utilizadas no monitoramento e controle da rede organizada em camadas, poderá ser usada para representar as categorias de *software* presentes em um sistema de gerenciamento.

2.2.4 Gerente

O gerente é um *software* que permite receber ou enviar informações de gerenciamento as máquinas gerenciadas mediante comunicação com um ou mais agentes.

> "Um processo gerente é parte de uma aplicação distribuída que tem a responsabilidade de uma ou mais atividades de gerenciamento, ou seja, ele transmite operações de gerenciamento (*actions*) aos agentes e recebe notificações (*events*) destes (PINHEIRO, 2002)".

De forma resumida, o *software* gerente é o responsável pela comunicação direta com o servidor e a comunicação com o(s) agente(s).

2.2.5 Agentes

Ainda de acordo com Werner (2010), agente é um tipo de *software* presente nos dispositivos gerenciados com funções de atender a requisições enviadas pelo *software* gerente, retornando automaticamente informações de gerenciamento ao gerente, quando previamente programado.

> "Um processo é dito agente quando parte de uma aplicação distribuída irá executar as diretivas enviadas pelo processo gerente. Assim, ele passará para o Gerente uma visão dos objetos sendo gerenciados e refletirá o comportamento desses objetos, emitindo notificações sobre os mesmos (PINHEIRO, 2002)".

O *software* agente é o responsável por captar informações do ambiente gerenciado, através da comunicação com o *software* gerente, essas informações são utilizadas pelo Gerente de rede para monitorar o funcionamento do seu servidor.

2.2.6 Protocolos de Gerenciamento

O protocolo de gerenciamento compreende o conjunto de regras e formatos de mensagens. Os mecanismos de comunicação entre gerentes e agentes são implementados com base nas especificações de um determinado protocolo de gerenciamento.

> "Os protocolos de monitoramento de redes descrevem um formato para o envio de informações entre os ativos de redes monitorados e as máquinas responsáveis pelo armazenamento de tais informações (DIAS, 2008)".

No entanto a função do protocolo é garantir a comunicação entre os recursos de redes homogêneas ou não.

Protocolo SNMP

Lançado em 1988, o protocolo SNMP atende a necessidade cada vez maior de um padrão para gerenciar dispositivos de IP (*Internet Protocol*), ou seja, trata-se de um protocolo definido como padrão da *Internet* para gerenciar tais dispositivos em redes IP (MAURA; SCHMIDT, 2001).

> "O SNMP é um protocolo de gerência definido a nível de aplicação, e é utilizado para obter informações de servidores SNMP - agentes espalhados numa rede baseada na pilha de protocolos TCP/IP. As informações são obtidas através de requisições de um gerente para um ou vários agentes que utilizam os serviços do protocolo de transporte UDP para o envio e recepção da mensagens (DIAS, 2008)".

Segundo Case et al. (1990), foi desenvolvido para facilitar a troca de informações de monitoramento entre ativos de redes, o protocolo SNMP pertence à camada de aplicação e está especificado na *Request for Comments* (RFC 11571).

O SNMP funciona baseado no conceito de agente e gerente. O agente é um programa executado na máquina monitorada e tem a função de coletar informações da respectiva máquina. O agente deve responder às requisições do gerente enviando, quando programado, as informações coletadas de forma automática. Um dos motivos pela escolha de um protocolo que não oferece garantia na comunicação é o fato do UDP ter menor sobrecarga na rede, reduzindo assim o impacto do sistema de gerência na performance da rede.

Na Figura 1, é apresentado o funcionamento do protocolo SNMP, onde o gerente é um programa que é executado em um servidor e, mediante a comunicação com um ou mais

agentes, com intuito de obter informações de monitoramento referentes a cada um dos ativos que hospedam o agente. Para obter essas informações é utilizada uma técnica chamada *pooling*, que é uma interação do tipo pergunta-resposta entre gerente e agente na qual apenas um dispositivo pode transmitir informação por vez.

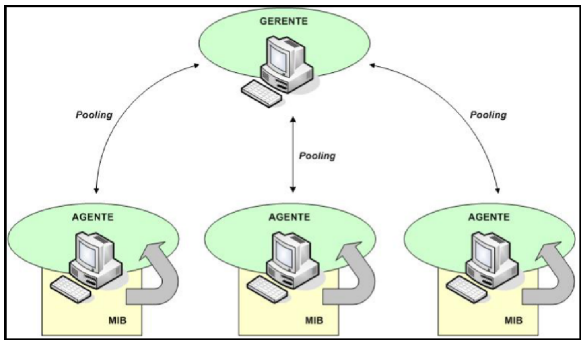

Figura 1 *– Funcionamento do protocolo SNMP*

[DIAS, 2008]

Protocolo TCP/IP

O modelo de referência TCP/IP foi definido pela primeira vez por **Vinton Gray Cerf** e **Robert Kahn**. Esse protocolo é utilizado entre computadores em rede e para validar essa comunicação, é preciso que os computadores utilizem o mesmo protocolo. Existem diversos tipos de protocolos. As aplicações de *Sockets* são feitas baseadas no protocolo TCP/IP.

Devido a popularização da *Internet*, atualmente este é o protocolo mais utilizado pois uma das grandes vantagens do TCP/IP em relação aos outros protocolos existentes, é que ele é roteável, isto é, foi criado pensando em redes de longa distância, onde pode haver vários caminhos para o dado atingir seu destino.

Segundo Scrimger et al. (2002) o protocolo TCP/IP não é apenas um único protocolo e sim um conjunto de protocolos, sendo baseado no modelo de quatro camada. Os aplicativos baseados em *sockets* utilizam o TCP/IP sendo necessario três elementos: o endereço de IP, uma porta e um tipo de serviço para a comunicação, é orientado a conexão e garante a entrega dos pacotes. Um problema do protocolo TCP/IP é o grande fluxo na rede causando sobre-carga, já ao contrário do protocolo TCP/IP é o protocolo UDP que não é orientado a conexão.

Protocolo UDP

O *User Datagram Protocol* (UDP) é um padrão TCP/IP e está definido pela **RFC 768** (POSTEL, 1980) . O UDP é usado por alguns programas para o transporte rápido de dados entre *host* TCP/IP. Esse protocolo não fornece garantia de entrega e nem verificação de dados.

De uma maneira simples, dizemos que o protocolo UDP manda os dados para o destino, porém não garantindo que os dados vão chegar de forma integra ou não. Contudo, em determinadas situações, o fato de o UDP é muito mais rápido (por não fazer verificações e por não estabelecer sessões), o uso do UDP é recomendado.

> "UDP é definido para disponibilizar um modo datagrama de comutação de pacotes de comunicação de computador no ambiente de um conjunto interligado de redes de computadores. Este protocolo prevê um procedimento de programas aplicativos para enviar mensagens para outros programas com um mínimo de mecanismo de protocolo (POSTEL, 1980)".

Uma área na qual o protocolo UDP é especialmente útil, é a de situações cliente/servidor. Com frequência, o cliente envia uma pequena solicitação ao servidor e espera uma pequena resposta de volta. Se a solicitação ou a resposta se perderem, o cliente simplesmente chegará a um tempo limite de espera (timeout) e tentará novamente (TANENBAUM, 2003).

2.3 *Sockets*

A ideia de um *socket* faz parte do TCP/IP, o conjunto de protocolos usado pela *Internet*. Um *socket* essencialmente é uma conexão de dados transparente entre dois computadores numa rede. Ele é identificado pelo endereço de rede dos computadores e por seus pontos finais e uma porta de comunicação. Por existir 65.536 portas TCP numeradas de 1 a 65.536. Cada porta pode ser usada por um programa ou serviço diferente, mas as portas de 0 a 1023 são **"portas reconhecidas"** ou **"reservadas"** para os processos do sistema ou aos programas executados por utilizadores privilegiados.

De acordo com Santiago (2009) *socket* é uma porta de comunicação que permite a um processo executando em um computador enviar/receber mensagens para/de outro processo que pode estar sendo executado no mesmo computador ou num computador remoto. O *socket* permite que uma aplicação cliente se conecte e se comunique com outras aplicações que estão plugados na mesma rede.

Os *sockets* têm dois modos principais de operação: o modo baseado em conexões e o modo sem conexão. O modo a ser utilizado é determinado pelas necessidades de um aplicativo. Se a conformidade é importante, então a operação baseada em conexões é a melhor opção. Os servidores de arquivos precisam fazer todos os seus dados chegarem corretamente.

Segundo Tanenbaum (2003) operações baseada em conexão usam o TCP (Transport Control Protocol). Um *socket* nesse modo de operação precisa se conectar ao destino antes de transmitir os dados. Uma vez conectados, os *sockets* são acessados pelo uso de uma interface de fluxos: abertura-leitura-escrita-fechamento. Tudo que é enviado por um *socket* é recebido pela outra extremidade da conexão, exatamente na mesma ordem em que foi transmitido. A operação

baseada em conexões é menos eficiente do que a operação sem conexão, mas é garantida.

Ainda de acordo com Tanenbaum (2003) operações sem conexão utiliza o UDP *(User Datagrama Protocol)*. Um datagrama é uma unidade autônoma que tem todas as informações necessárias para tentar fazer sua entrega. Similar a um envelope, o datagrama tem um endereço do destinatário e do remetente e contém em seu interior os dados a serem enviados. Um *socket* nesse modo de operação não precisa se conectar a um *socket* de destino, ele simplesmente envia o datagrama. O protocolo UDP só promete fazer o melhor esforço possível para tentar entregar o datagrama. A operação sem conexão é rápida e eficiente, mas não é garantida.

Quando se deseja rapidez e eficiência à operação sem conexão é mais adequada, mas não garantida. Datagramas têm menos *overhead* do que *sockets* TCP, pois nenhum controle de fluxo é necessário quando se envia ou recebe um datagrama UDP. Os datagramas UDP também podem ser usados num servidor *Java* que envia atualizações periódicas para um conjunto de clientes. O servidor teria de fazer menos trabalho e consequentemente seria mais rápido, se ele pudesse simplesmente enviar datagramas. No presente projeto foi utilizada a operação sem conexão por desejar uma rapidez e eficiência entre as aplicações.

2.4 Linguagem *Java*

As redes de computadores estão cada vez mais abrangentes, se tornando mais heterogêneas, sendo compostas por diferentes *hardwares* e *softwares*. Contudo, uma solução de monitoramento de redes necessita de uma linguagem que possa ser compreendida em qualquer tipo de plataforma, atendendo às necessidades. A linguagem *Java*, atende a estes requisitos.

Segundo Deitel e Deitel (2006) a linguagem *Java* foi anunciada pela *Sun*, em conferência que aconteceu em maio de 1995, chamando a atenção da comunidade de negócios por conta do enorme interesse na *World Wide Web*. Não é uma linguagem muito antiga, mas já conhecida no mundo todo.

Java é a base para praticamente todos os tipos de aplicações em rede e é o padrão global para o desenvolvimento de aplicações móveis, jogos, conteúdo baseado na *web*. Com mais de 9 milhões de desenvolvedores espalhados pelo mundo, *Java* permite desenvolver e implementar de forma eficiente aplicativos interessantes. Com ferramentas abrangentes, um ecossistema maduro e desempenho robusto, *Java* oferece portabilidade as aplicações em ambiente de computação (ORACLE, 2013).

De acordo com PAMPLONA e Fernando (2009), *Java* é uma linguagem multiplataforma na qual quando um programa é compilado um código intermediário é gerado, chamado de *bytecode*. Este *bytecode* é interpretado pelas máquinas virtuais *Java* (JVMs) para a maioria dos sistemas operacionais. A máquina virtual é responsável por criar um ambiente multipla-

taforma, ou seja, se alguém construir um sistema operacional novo, basta criar uma máquina virtual que traduza os *bytecodes* para código nativo e pronto. Todas as aplicações *Java* estarão rodando sem problemas.

Para o desenvolvimento de aplicações na linguagem *Java*, é necessário apenas o kit de desenvolvimento JDK(*Java development*) e uma plataforma IDE de desenvolvimento. Sendo que para as aplicações desenvolvidas na linguagem *Java* serem executadas é necessário à instalação do JDK no computador.

2.5 Plataforma Eclipse

Eclipse é uma comunidade de código aberto cujos projetos são concentrados na criação de uma plataforma de desenvolvimento aberta composta de estruturas e ferramentas. Foi criada pela IBM em novembro de 2001, e suportada por um consórcio de fornecedores de *software* em 2004 (ANISZCYK; GALLARDO, 2012).

Segundo Aniszcyk e Gallardo (2012), o Eclipse é uma plataforma extensível de desenvolvimento baseado em Java com estrutura e conjunto para a construção de ambiente de desenvolvimento, possui um conjunto padrão de *plug-ins* incluindo *Java Development Tools*(JDT) e o *Plug-in Development Environment* (PDE), que é utilizado pelos desenvolvedores que desejam estender o Eclipse.

Ainda de acordo com Aniszcyk e Gallardo (2012), embora o Eclipse seja escrito em Java o seu uso não se limita a linguagem Java, existem plug-ins disponíveis ou planejados que incluem suporte para linguagens de programação como C / C++ e COBOL. A estrutura do Eclipse também pode ser usada como base para outros tipos de aplicações não relacionadas com o desenvolvimento de software, por exemplo, adiciona suporte para JSPs, *servlets*, EJBs, XML, *Web services* e banco de dados de acesso.

O Eclipse têm um poderoso conjunto de serviços que suportam *plug-ins* como JDT, PDE e o *ADT Plug-in* que é utilizado neste projeto. Apresenta como área de trabalho o *plug-in* responsável pela gestão dos recursos do usuário. O Eclipse possui uma plataforma de execução que é o *Kernel* que é responsável por descobrir na inicialização que plug-ins estão instalados criando informações sobre os mesmos. É uma plataforma que funciona tanto em sistemas operacionais *Windows* como *Linux*, facilitando assim para os desenvolvedores, que podem optar em qual sistema operacional será desenvolvido suas aplicações.

2.6 Plataforma Android

De acordo Martins (2009) os dispositivos móveis estão surgindo com várias flexibilidade e funcionalidade como: tocadores de áudio, câmeras fotográficas e placas de comunicação sem fio multiprotocolos. Os *smartphones* são um desses dispositivos que se caracteriza por possuir funcionalidades avançadas, estas que podem ser estendidas por meio de programas executados por seu próprio sistema operacional. As empresas, entre elas a Google, atentas ao crescimento do mercado, se juntaram para formar em novembro de 2007 a (OHA) *Open Handset Alliance* para lançar a plataforma nomeada Android.

A plataforma Android é formada de um pacote de *software* para dispositivos móveis que inclui um sistema operacional, um *middleware* (que intermedeia a comunicação entre diferentes aplicativos e distintos protocolos) e aplicativos de características móvel completa, livre e aberta apresentando características incorporadas de outros sistemas, até de inúmeros conceitos inovadores para o segmento móvel. Essa plataforma caracteriza-se por usar como base o *Kernel* do *Linux 2.6*, o qual sofreu várias alterações para que ficassem adaptados para dispositivos portáteis, sendo este o responsável por várias tarefas do sistema, como gerenciamento de memória, segurança e vários outros (SANTOS et al., 2002).

Criado pela Open Handset Alliance, resultado da união de 47 empresas de tecnologia e mobilidade sobre a liderança da Google Inc., o Android tem como principais objetivos (LECHETA, 2010):

- A oportunidade de personalização das aplicações e componentes presentes em um sistema, por ser de código aberto e gratuito;
- A possibilidade de desenvolvimento rápido e moderno de aplicações corporativas, uma vez em que sua plataforma é moderna e flexível.

De acordo com IDGNews (2013) o iOS pode ser o sistema móbile de crescimento mais rápido, mas o Android ainda domina o mercado, com uma participação de 57%. A Global WebIndex estima que os usuários da plataforma do Google devem alcançar 708 milhões neste ano. A empresa de pesquisas apontam que o sistema operacional Android será o número do mundo.

O Google disponibiliza um SDK (kit de desenvolvimento de *software*) gratuitamente para quem deseja desenvolver aplicativos para a plataforma Android, as aplicações desenvolvidas utilizam a linguagem *Java*.

2.6.1 Arquitetura do Android

Basicamente, a arquitetura do sistema operacional Android é dividido em quatro camadas, como demonstrados na Figura 2, são elas: Aplicações, *Framework*, Bibliotecas, Android *Runtime* e *Kernel Linux* . Sendo cada uma dessas camadas composta por módulos internos especializados na resolução de alguma tarefa específica.

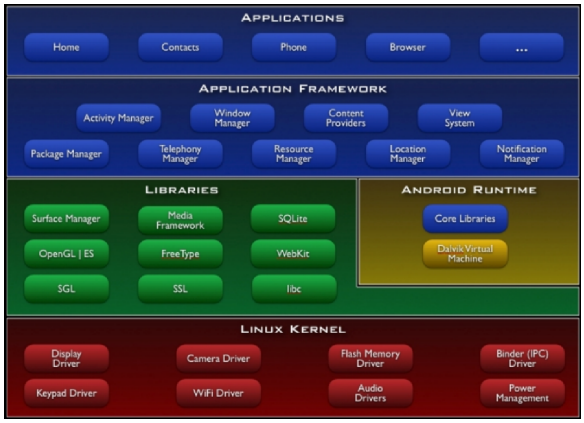

Figura 2 *– Arquitetura do Android*

Fonte: <http://developer.android.com/>

Conforme Santos et al. (2002) existe um conjunto de definições que são úteis para o entendimento das camadas da plataforma Android, as quais são demonstradas na Figura 2 e descritas a seguir:

- Aplicações: é a camada que está no topo da camada do sistema operacional Android composta por um conjunto de aplicativos desenvolvidos na linguagem de programação *Java*, são executados em uma máquina virtual (VM). É importante relatar que a VM não é uma JVM (*Java Virtual Machine*), como você pode esperar, mas é uma *Dalvik Virtual Machine*, uma tecnologia de *software* livre. Cada aplicativo Android é executado em uma instância da *Dalvik VM*, que por sua vez, reside em um processo gerenciado por *Kernel Linux*.

- *Framework*: esta camada disponibiliza aos desenvolvedores as Interface de Programação de Aplicativos (APIs) utilizadas para a criação de aplicações originais do sistema operacional Android. Este *framework* permite que o programador tenha o mesmo acesso ao sistema que os aplicativos da camada de aplicativos possuem. Este *framework* foi criado para abstrair a complexidade, simplifica e a reutilização de procedimentos. Essa camada funciona como um *link* com a camada de bibliotecas do sistema operacional que serão acessadas através de APIs contidas no *framework*.

- Bibliotecas: Essas bibliotecas são responsáveis por fornecer funcionalidades para manipular diversos componentes do sistema Android (áudio, vídeo, gráficos, banco de dados e browser). Algumas bibliotecas são: OpenGL/ES (para trabalhar com interface gráfica), SQLite (para trabalhar com banco de dados), etc. Aqui também estão os serviços usados em camadas superiores, como máquina virtual *Java Dalvik*.

- Android *Runtime:* a camada de execução onde cada aplicação Android roda em seu próprio processo, com sua própria instância da máquina virtual *Dalvik*. Essa máquina virtual foi otimizada especialmente para dispositivos móveis e foi escrita de forma que um dispositivo pode executar vários VMS *(Virtual Memory System)* de forma eficiente. A máquina virtual *Dalvik* executa os arquivos com extensão formato (.DEX) executável. Essa máquina virtual foi construída pelos engenheiros da Google, para obter um consumo mínimo de memória e isolamento de processos, onde ela invoca o *Kernel do Linux* para a funcionalidade subjacente como encadeamento e de baixo nível de gerenciamento de memória permitindo que as aplicações escritas em *Java* sejam executadas normalmente.

- *Kernel Linux*: A camada do *Kernel* é baseada em um sistema do sistema operacional *Linux* versão 2.6. Esta camada atua também como responsável pela abstração entre o *hardware* e os aplicativos e é responsável pelos serviços principais do sistema operacional Android.

2.7 Trabalhos Relacionados

Dentre os trabalhos encontrados, foi selecionado dois, O primeiro relata sobre a "A Importância do Monitoramento de Ativos de Redes: Um Estudo de Caso com o Sistema CACIC" de autoria de Dias (2008). Neste trabalho, Dias descreve sobre os recursos de redes numa organização e o quando é importante a utilização de ferramentas de monitoramento para o gerenciamento destes recursos. O trabalho do referente autor foi utilizado para o bom entendimento sobre o gerenciamento dos dispositivos de redes.

O segundo trabalho selecionado diz respeita a utilização dos dispositivos móveis no gerenciamento de redes de computadores. Intitulado "Gerência e Monitoramento de Redes Através de Dispositivos Móveis", é de autoria de Balsemão (2008). Neste trabalho Balsemão apresenta o avanço nas tecnologias dos dispositivos móveis e funcionalidades que poderão ser agregadas a estes dispositivos através do desenvolvimento de softwares.

Este capítulo apresentou o referencial teórico no qual fornece embasamento para este trabalho. Começando pelo monitoramento que é o foco deste trabalho e em seguida alguns termos principais de gerência de redes na qual faz parte da tarefa de monitoramento de servidores.

Outro ponto importante para o esclarecimento deste trabalho é o significado de protocolo na qual é apresentado sua definição, exemplo e funções que um protocolo exerce, assim como a linguagem de progração utilizada.

E para finalizar este capítulo, o Android é posto em evidência sendo listadas algumas de suas principais características e arquitetura.

3 Modelagem do Sistema

Este capítulo apresenta as análises dos requisitos funcionais e específicos da plataforma para o monitoramento de rede através de diagramas de: classe, caso-de-uso e sequência e a comfigução do ambiente de desenvolvimento da aplicação gerente.

3.1 Análises: Requisitos Funcionais

Para o desenvolvimento das tarefas propostas nesse trabalho foram detectados os requisitos a seguir:

1. O agente depende da plataforma *Linux* para ser executado.
2. As duas aplicações (agente/gerente) devem estar conectadas a uma rede local ou a *Internet*;
3. Deve existir uma comunicação entre as duas aplicações (agente e gerente) para que possam se comunicar.
4. A aplicação agente retorna informações sobre espaço utilizado e livre do tamanho da memória, disco rígido e informações da CPU à aplicação gerente de acordo com a requisição efetuada.

3.2 Análises: Especificação dos Requisitos

Neste trabalho, foram desenvolvidas duas aplicações na qual uma serve como agente que executa suas funções sobre um servidor, coletando os dados definidos como parâmetro (disco, CPU e memória), e a outra aplicação serve para acessar remotamente os dados coletados pela primeira aplicação, possibilitando assim um monitoramento remoto através de um dispositivo móvel com a plataforma Android.

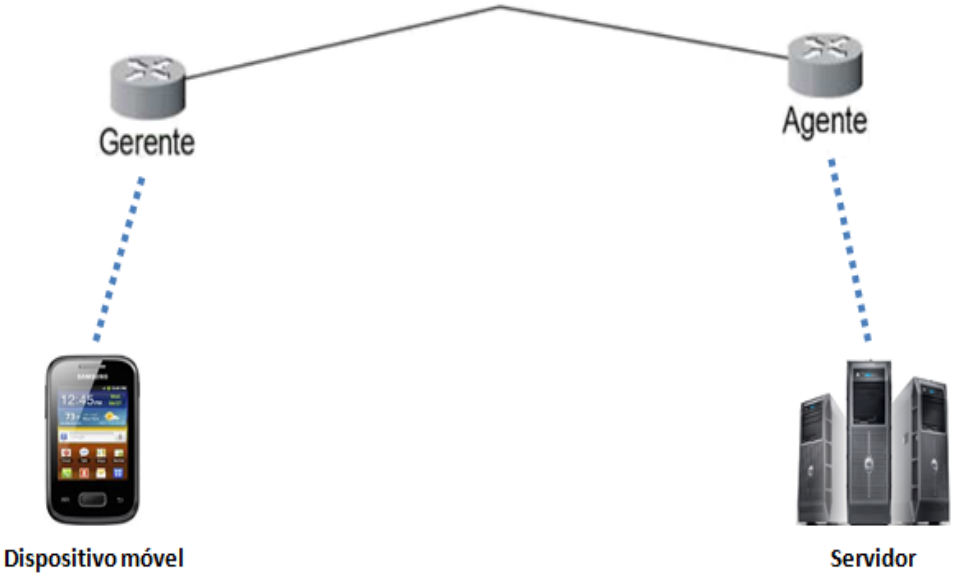

Figura 3 *– Arquitetura do Projeto*

A Figura 3 ilustra de forma resumida os principais componentes deste ambiente de monitoramento. O ambiente é composto por um dispositivo móvel com o sistema operacional Android e o *software* gerente instalado e uma maquina servidor que possui o *software* agente instalado ambos se comunicam através de uma rede local ou a *Internet*.

3.2.1 Diagrama de Caso de Uso

O diagrama de caso de uso descreve um cenário que ilustra as funcionalidades do sistema do ponto de vista do usuário. Segundo Guedes (2011), diagrama de caso de uso apresenta uma linguagem simples e de fácil compreensão para que o usuário possa ter uma ideia geral de como o sistema irá se comportar.

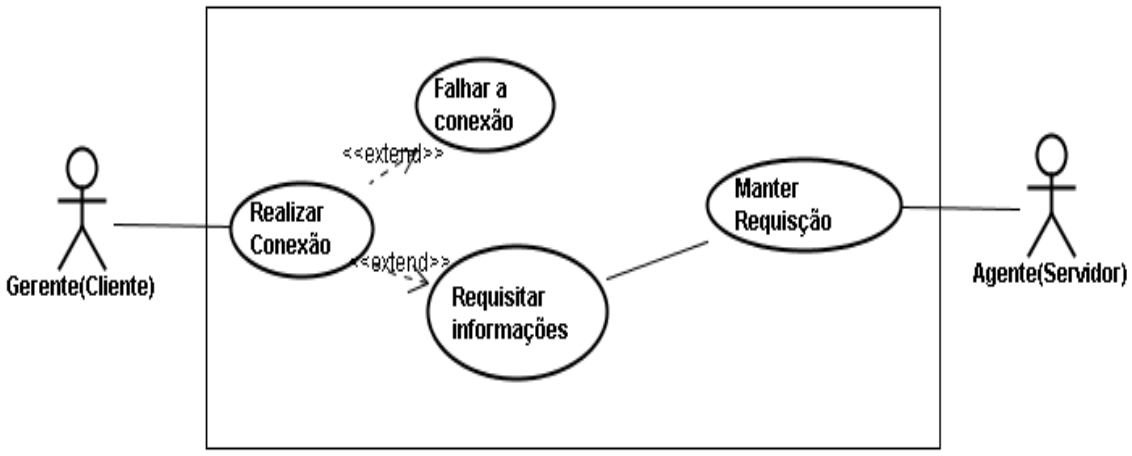

Figura 4 *– Descrição textual do caso de uso*

A Figura 4 descreve o caso de uso utilizado para representar o processo de comunicação entre as duas aplicações (agente/gerente). O Gerente de redes utiliza um dispositivo móvel com sistema operacional Android, e através da aplicação instalada no seu dispositivo móvel faz requisições sobre o funcionamento do servidor, sendo que as duas aplicações estão conectadas numa rede local ou a *Internet* e utilizam *socket* como meio de comunicação.

O processo de utilização do gerente é realizada através do uso da aplicação no dispositivo móvel para obter as informações necessárias sobre o funcionamento do servidor, que de acordo com as informações recebidas poderá tomar possíveis decisões sobre o seu funcionamento, como apresentado na descrição a seguir:

1. O Gerente de Redes conecta seu dispositivo na *Internet* e executa o aplicativo em seu dispositivo móvel;

2. Na tela inicial do aplicativo é requisitado o endereço IP da máquina Servidor (agente) utilizado para comunicação;

3. Conexão entre os dois aplicativos é estabelecida;

4. Há mudanças de tela no aparelho, onde tem botões que acionados enviarão requisições para outra aplicação.

   Além da sequência mencionada anteriormente, podem ocorrer eventos alternativos, tais como:

5. Caso não estabeleça a conexão, por exemplo: Um dos aplicativos não está conectado a *Internet* ou alguma rede que possibilite a comunicação entre as duas aplicações, o endereço IP de comunicação foi informado incorretamente.

6. A mudança de tela não acontece caso a conexão venha a falhar.

3.2.2 Diagrama de Classes

O diagrama de classe serve de apoio para os demais diagramas, demonstrando a estrutura estática das classes de um sistema e seus relacionamentos. De acordo com Guedes (2011) o diagrama de classe é onde determina os atributos e métodos que cada classe tem, além de estabelecer como as classe se relacionam e trocam informações entre si.

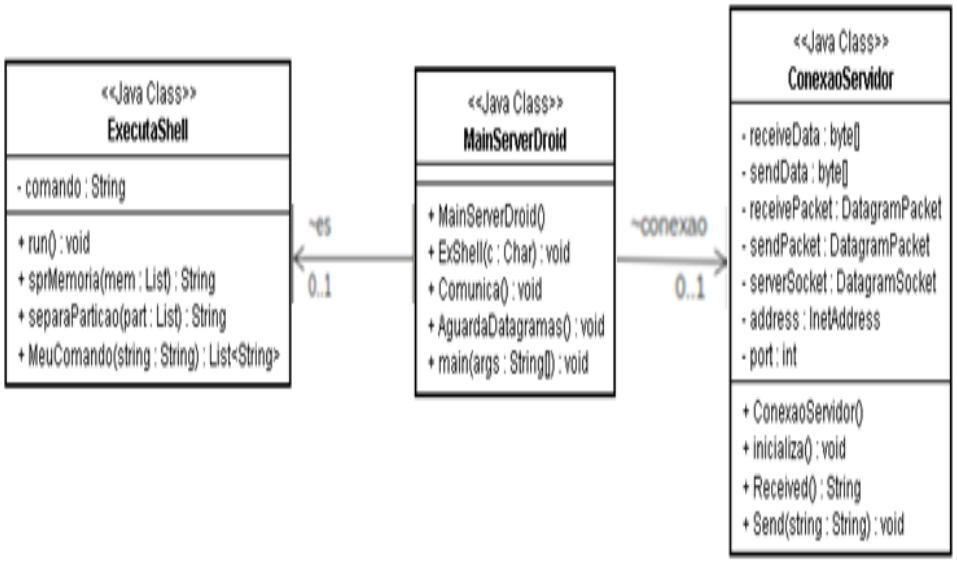

Figura 5 *– Diagrama de classe software agente.*

A Figura 5 apresenta o diagrama de classe do *software* agente contendo as classes ConexaoServidor, ExecutaShell e MainServerDroid que é a classe principal do *software* agente.

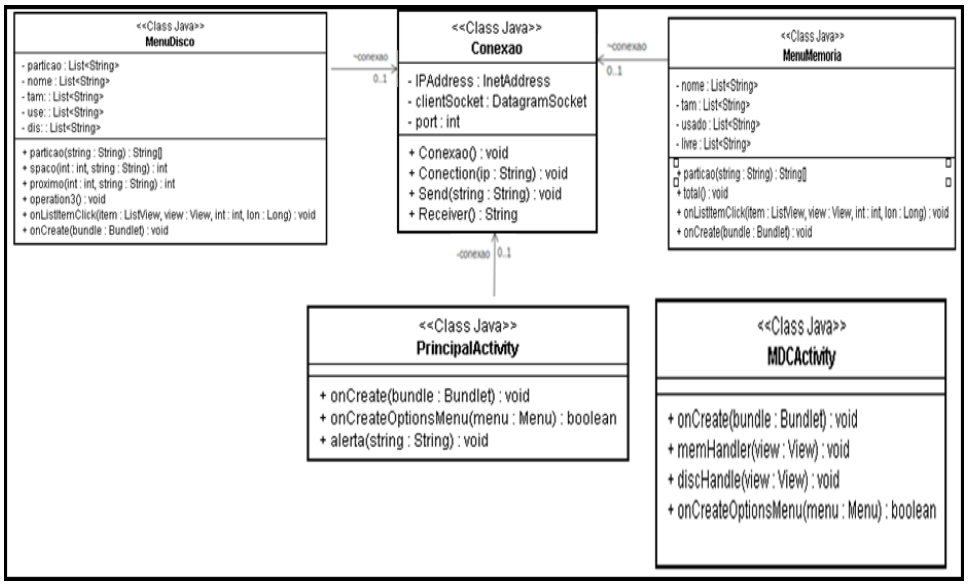

Figura 6 *– Diagrama de classe software gerente*

Na Figura 6, é apresentado diagrama de classes do *software* gerente. No diagrama é apresentado o relacionamento estrutural que descreve o conjunto de ligações que são conexões entre os objetos, bem como as classes responsáveis pelo funcionamento da aplicação.

3.2.3 Diagrama de Sequência

O diagrama de sequência apresenta uma interações de objetos ordenados numa sequência de tempo. Segundo Guedes (2011) um diagrama de sequência costuma identificar o evento gerador do processo modelado, bem como o ator responsável por esse evento, e determina como o processo deve se desenrolar e ser concluído por meio da chamada de métodos disparados por mensagens enviadas entre os objetos.

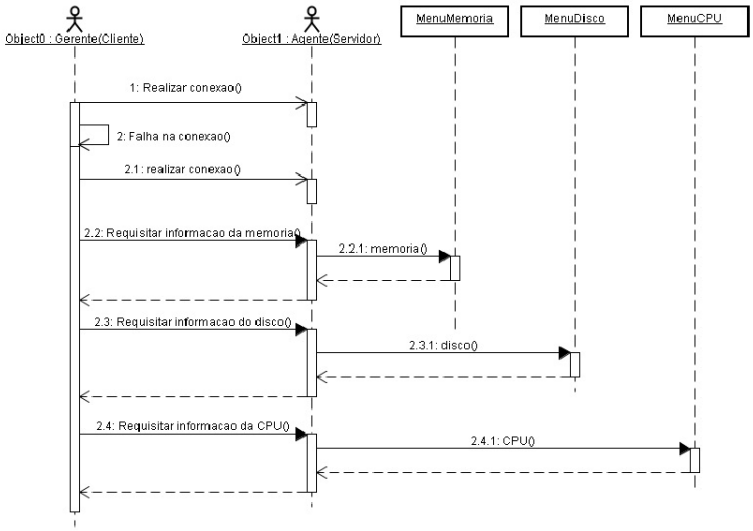

Figura 7 *– Diagrama de Sequência*

Na Figura 7 é ilustrado de forma específica a troca de mensagens entre as duas aplicações(agente/ gerente).

1. Na etapa 1:realiza conexao() o gerente envia uma mensagem contendo endereço e a porta de conexão buscando assim efetuar uma conexão com o agente, caso ocorra uma falha de comunicação,

2. Na etapa 2:falha na conexão() pelo o servidor(agente) ou dispositivo móvel(gerente) não está conectado a uma rede local oua *Internet* é estourado o tempo de conexão, o gerente receberá uma mensagem de erro na tela.

3. Na etapa 2.1: realiza a conexão() e assim efetuar as requisições sendo que a cada requisição efetuada é criada uma nova conexão.

4. Na etapa 2.2: requisitar informação da memoria() e o agente efetua o processo de requisição retornando informações sobre o tamanho, espaço usado e livre da memória principal e *swap* do servidor.

5. Na etapa 2.3: requisitar informação do disco() ao agente, este retorna informações sobre o tamanho, partições, espaço usado e livre do disco.

6. Na etapa 2.4: requisitar informação da CPU() onde o agente retorna a mensagem com informações sobre a CPU. Após todo o processo o gerente deve sair ou repetir o processo de requisição novamente caso deseje.

3.3 Projeto

As aplicações foram desenvolvidas na Linguagem *Java*, utilizando como plataforma Eclipse 3.6.2, juntamente com a JDK 6, SDK 4.1 e ADT *plug-in* instalado no Eclipse para o desenvolvimento de aplicações do sistema operacional Android.

3.3.1 Ambiente de Desenvolvimento

As aplicações para dispositivos móveis com sistema operacional Android é importante apresentar a preparação do ambiente de desenvolvimento, levando em consideração alguns requisitos como sistemas suportados pela plataforma que de acordo com Lecheta (2010) são: *Windows XP(32-bits)*, *Vista (32 ou 64-bits)* ou *Windows 7(32 ou 64-bits)*, *Mac OS X 10.5.8* ou *superior (somente x86)* e *Linux*, assim como o Eclipse 3.6.2(Hellios) ou superior.

3.3.2 Preparando o Ambiente

De início é efetuado o *download* dos pacotes SDK(kit de desenvolvimento) que estão disponível no site “http://developer.android.com/sdk/index.html”. O Android SDK proporciona ferramentas chamadas via API na linguagem *Java* para o desenvolvimento de aplicações.

Plataforma	Pacote	Tamanho	MD5 checksum
Windows 32-bit	adt-pacote-windows-x86-20131030.zip	503599460 bytes	cd490a531ec24667354f6473e999b988
Windows de 64 bits	adt-pacote-windows-x86_64-20131030.zip	503735416 bytes	ddddbb1b9028015779d68dde01f96b14
Mac OS X 64-bit	adt-bundle-mac-x86_64-20131030.zip	470386961 bytes	3e80e7a92b549029d91bdcf2ae82657f
Linux de 32 bits	adt-bundle-linux-x86-20131030.zip	496876498 bytes	d389139ad9f59a43bdd34c94bc850509
Linux de 64 bits	adt-bundle-linux-x86_64-20131030.zip	497171697 bytes	99b51a4f0526434b083701a896550b72

Figura 8 *– Arquivos para downloads*

A Figura 8, mostra as plataformas e suas versões que suportam o SDK. Ao carregar a página é apresentada uma tabela de opções de *download* do arquivo SDK(kit de desenvolvimento)

que será escolhido de acordo com o sistema operacional da máquina que vai ser utilizada para o desenvolvimento da aplicação. Após o término do *download* o arquivo deverá ser executado.

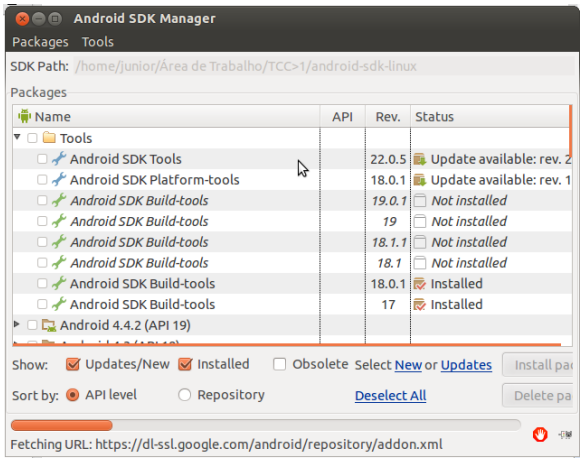

Figura 9 *– API's do Android SDK*

Após a execução do SDK será exibida uma tela para instalação das versões disponíveis das API's do Android e extensões, como mostra na Figura9 Sendo que no projeto foi utilizado a API Android 4.1, é importante a instalação dos pacotes extras para minimizar possíveis problemas de execução.

Terminando a parte de instalação do SDK(kit de desenvolvimento), inicia-se a preparação do Eclipse que de acordo com Lecheta (2010) é uma das ferramentas mais indicadas para o desenvolvimento de aplicações para a plataforma Android, pois, o Eclipse inclui um ambiente de desenvolvimento integrado e é extensível a *plug-ins* de sistema.

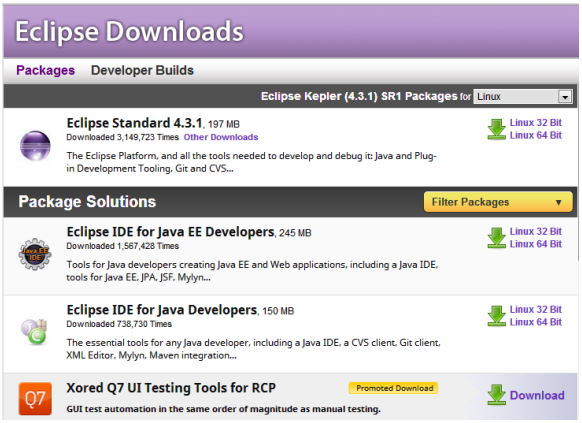

Figura 10 *– Arquivos para downloads da IDE Eclipse*

Fonte: <http://www.eclipse.org/downloads/>

A Figura 10 apresenta o Eclipse e suas versões disponíveis para cada sistema operacional. Após a instalação do Eclipse é importante verificar se a máquina utilizada para o desenvolvimento da aplicação dispõem do JDK(*Java Development kit*), kit de desenvolvimento em *Java*,

lembrando que o Android é compatível com o JDK 6. No JDK já contêm a JRE(*Java Runtime Environment*) que possui o necessário para executar aplicativos *Java* no sistema, sendo que se tratando do Android é necessário a configuração da variável de ambiente PATH e JAVA_HOME do sistema operacional. Depois de todo o procedimento de instalação do Eclipse será iniciado o processo de instalação do ADT-*Plug-in* do Android.

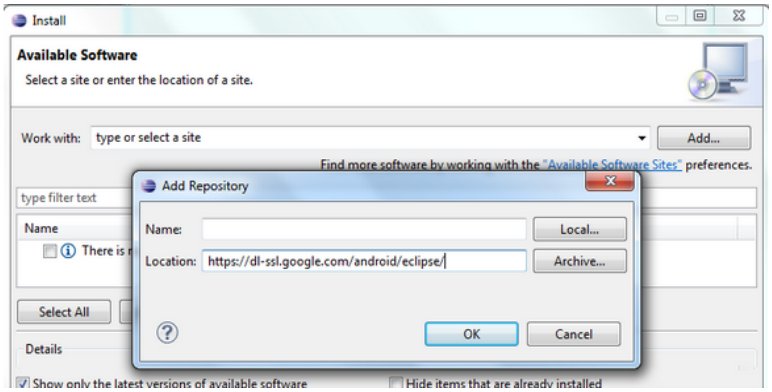

Figura 11 *– Processo de instalação do ADT-Plug-in*

Na Figura 11, é apresentado o processo de instalação do ADT-*Plug-in* dentro do Eclipse, selecione *Help>Install New Software...*, clique em *Add* no canto superior direito e adicione a URL(https://dl-ssl.google.com/android/eclipse), para que o Eclipse faça a busca do ADT-*Plug-in*. Ao término da instalação do Eclipse reiniciará automaticamente concluindo o processo de instalação, depois de reiniciar estará pronto para o desenvolvimento de aplicações para a plataforma Android.

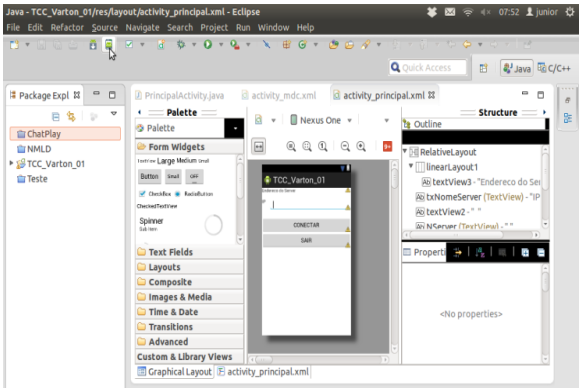

Figura 12 *– Ambiente de desenvolvimento*

A Figura 12 ilustra o ambiente de desenvolvimento com o ADT_Plug-in instalado e pronto para a criação de aplicações para os dispositivos móveis com o sistema operacional Android.

3.3.3 Desenvolvimento

Após o levantamento dos requisitos e a definição da linguagem de programação foram desenvolvidas as duas aplicações (agente/gerente), onde é exibida as principais fases do desenvolvimento.

***Software* Agente**

Como relatado nos requisitos funcionais o *software* agente será executado na plataforma *Linux*, possibilitando a comunicação entre as duas aplicações e respondendo de forma imediata as requisições feitas pelo *software* gerente. O *software* agente não necessita de interface gráfica, pois, ao inicia a máquina servidor a aplicação é executada automaticamente onde fica aguardando as requisições, até que a máquina seja finalizada ou desligada.

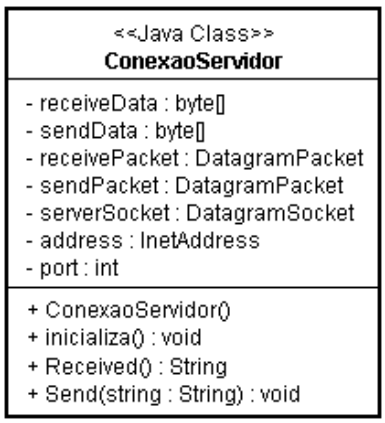

***Figura 13** – Classe ConexaoServidor*

A classe ConexaoServidor apresentada no diagrama de classe do *software* agente é a classe responsável pela comunicação entre as duas aplicações (agente/gerente), ela possui os métodos inicializa(), *Received*() e *Send*(*String*) que são métodos responsáveis pelo funcionamento e troca de mensagem entre as aplicações. No método inicializa() foi criado o *socket* de comunicação que tem como padrão a porta "5000". Após a definição da porta de comunicação, foi definido os vetores de *bytes* sendData e *receiveData* que definem o tamanho das mensagens em 1024 *bytes*, é criado um pacote de dados a ser recebido que tem como parâmetro o vetor *receiveData* que recebe a mensagem e o *receiveData*.lengt que obtêm o tamanho da mensagem. O *Received*() é o método que fica aguardando a mensagem enviada pela aplicação gerente, já o método Send (*String*) é o responsável pelo retorno da mensagem.

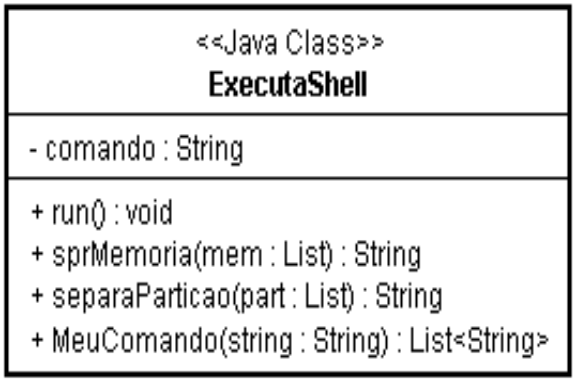

Figura 14 *– Classe ExecutaShell*

Na Figura 14 a classe ExecutaShell possui o método MeuComando(String). Este método tem como função fazer a consulta do comando que foi solicitado pela aplicação gerente. A execução desse método é apresentado na Figura 15.

```
63  try {
64      System.out.println("Executando o comando:" + comando);
65      Process proc = Runtime.getRuntime().exec(comando);
66      BufferedInputStream buffer = new BufferedInputStream(proc.getInputStream())
67      BufferedReader command = new BufferedReader(new InputStreamReader(buffer));
68      while ((Saida = command.readLine())!= null){
69              result.add(Saida);
70      }
```

Figura 15 *– Código de execução da requisição*

A Figura 15, encontra-se o código de execução da requisição efetuada. Observa-se que na linha 65, enquanto ocorre a execução do programa a variável "proc" que é um processo vai receber instruções de código em *Linux*. Na linha 66, é criado um *buffer* de entrada de fluxo, e a seguir na linha 67, é declarada a variável *command* que irá armazenar toda informação do processo requisitado, em seguida é criado um laço de repetição para que toda informação armazenada se transforme numa *string*. Na linha 69 é adicionada em uma lista. A execução da aplicação é efetuada na classe MainServerDroid.

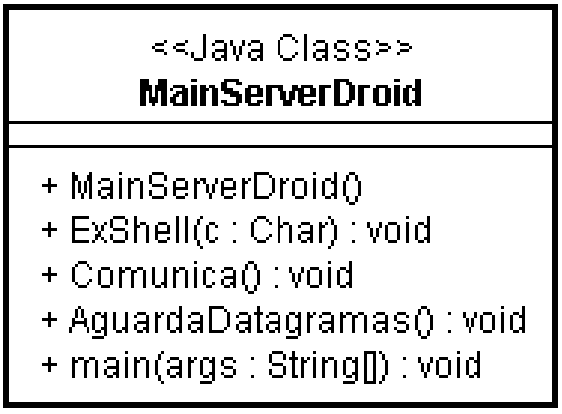

Figura 16 *– Classe MainServerDroid*

A classe MainServerDroid é a classe de execução da aplicação agente (Servidor), apresenta o método ExShell(char) que recebe por parâmetro um caractere que foi definido como protocolo padrão entre as duas aplicações (agente/gerente). Este método é responsável por tratar qual a requisição é realizada. A tabela 1 a seguir mostra as requisições com seus respectivos comandos e execuções.

Requisição Recebida	Comando	Execução
'1'	free -m –t	A execução deste comando busca informação da utilização da memória em megabytes.
'2'	dt –Th	A execução deste comando busca informação da utilização do(s) disco(s) rígido(s).
'3'	top -b -n 1	A execução deste comando retorna informação da Unidade Central de Processamento (CPU).

Tabela 1 *– tabela das requisições, comandos e execução*

Na tabela de requisições a primeira linha se refere ao comando que o *software* agente irá executar quando o gerente solicitar as informações da memória, a segunda linha será executado as informações do disco se a requisição for referente ao caractere '2' e a terceira linha o caractere '3' refere-se a execução o comando **top -b -n 1** que busca informações da CPU. A seguir é apresentado as classes e suas principais funções ilustadas anteriormente no diagrama de classe do *software* gerente.

Software Gerente

No *software* gerente é apresentado as classes e suas refentes telas de interação com o usuário.

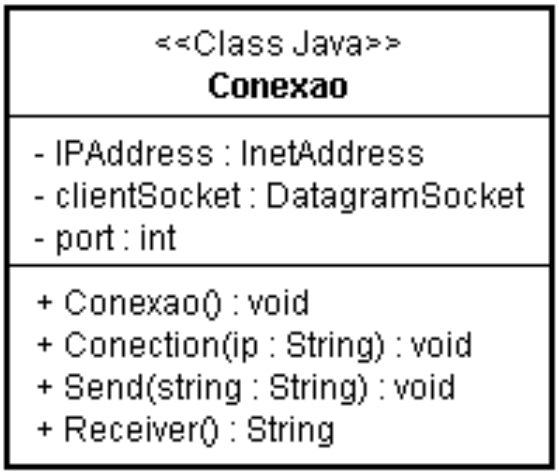

Figura 17 *– Classe Conexao*

A classe Conexao é a classe que apresenta os principais métodos da aplicação gerente, entre eles estão Conection(String) recebe por parâmetro o endereço da máquina Servidor sendo este método responsável pela comunicação das aplicações (gerente/agente), e o *Send(String)* e *Receiver()* que são utilizados para troca de informações entre as duas aplicações.

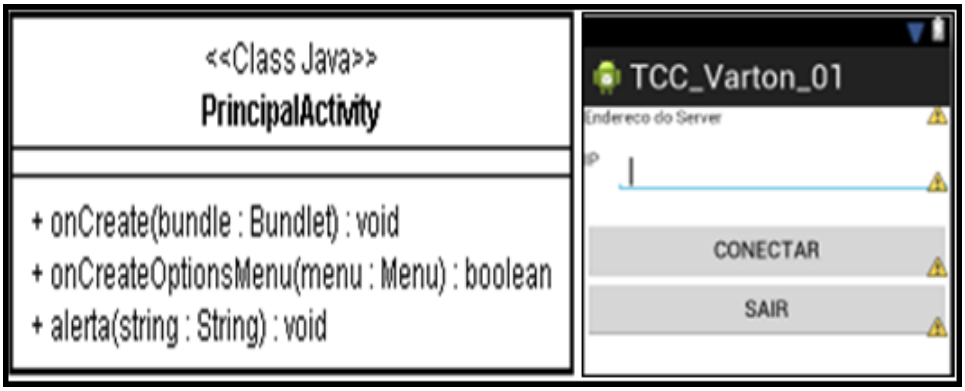

Figura 18 *– Classe PrincipalActivity e sua interface gráfica(layout)*

A Figura 18 ilustra a classe PrincipalActivity estende *Activity* que possui o setContent-View(Ver) que é chamado como recurso de definição do layout com o usuário. As subclasses da Activity apresentam métodos que são implementados, entre eles o *onCreate (Bundle)* que é responsável por iniciar as atividades da aplicação, o *findViewById(int)* usado para recuperar os *widgets(EditText,Button)* e outros que você necessita para interagir com a programação. A classe PrincipalActivity utiliza o método Conection(String) da classe Conexao.

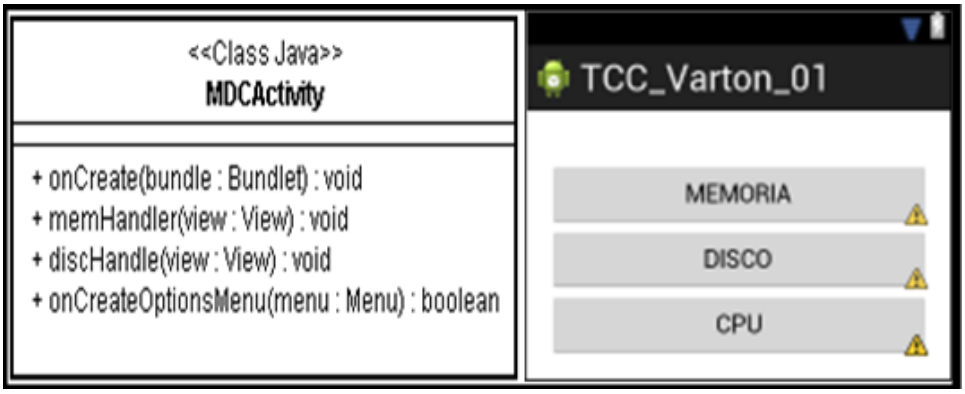

Figura 19 *– Classe MDCActivity e sua interface gráfica(layout)*

A classe MDCActivity mostra o menu de opções para a troca de informações, nessa classe foi definido um protocolo padrão entre as duas aplicações (agente/gerente). O protocolo apresenta um caractere do tipo *"char"* que é enviado a aplicação agente, onde esta aplicação vai tratar a informação referente a requisição solicitada. As requisições são apresentadas na tabela 2.

Requisição enviada	Informações
'1'	Informações sobre tamanho, espaço usado e disponível da memória
'2'	Informações das partições do disco do Servidor como: tamanho, espaço usado e disponível
'3'	Informações de funcionamento da CPU.

Tabela 2 *– tabela das requisições enviadas*

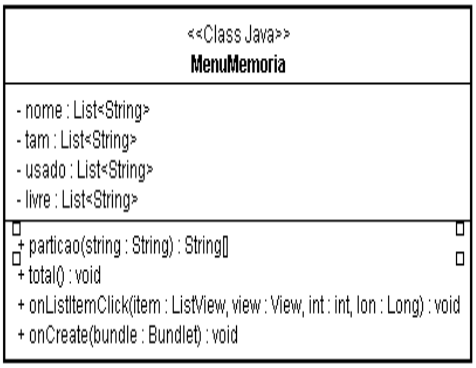

Figura 20 *– Classe MenuMemoria*

A Classe MenuMemoria, Figura 20, possui o método partição(*String*), método este que recebe uma *string* que foi enviada pela aplicação agente como resposta da requisição efetuada ao clicar no botão memória da classe MDCActivity. A interface gráfica dessa classe sobrepõe a interface gráfica da classe MDCActivity se tornando assim uma única interface.

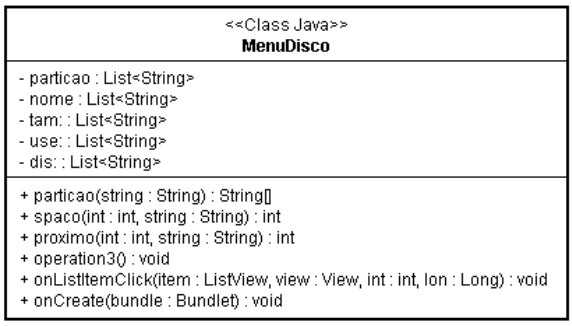

Figura 21 *– Classe MenuDisco*

A Figura 21 apresenta a classe MenuDisco que possui o método onListItemClick(ListView, View,in,long) que apresenta uma lista de partições existentes na aplicação agente (servidor) que quando clicar em uma das opções da lista é exibido o tamanho, espaço usado e disponível. A lista é resultado da requisição enviada pela aplicação gerente à aplicação agente. A interface gráfica dessa classe sobrepõe a interface gráfica da classe MDCActivity se tornando assim uma única interface.

4 Interface de Uso e Testes Funcionais

Este capítulo apresenta o ambiente gráfico da aplicação gerente e suas funcionalidades bem como os testes realizados e seus resultados.

Os testes foram realizados utilizando um celular com sistema operacional android 4.1, um servidor com a aplicação agente instalada. O servidor foi instalado numa máquina virtual executada no notebook com as seguintes configurações: 320GB de disco, 4GB de memória RAM.

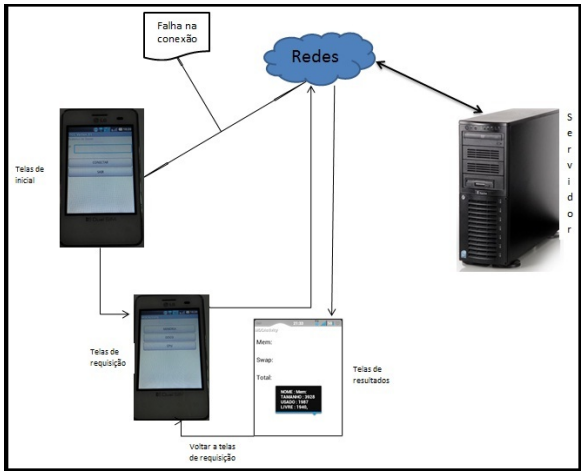

Figura 22 *– InfraEstrutura do Projeto*

A Figura 22 apresenta a aplicação em execução no dispositivo móvel:

1. O dispositivo móvel deve estar conectado na rede local/*Internet*;
2. O gerente de redes deve clicar no botão Conectar para realizar a comunicação;
3. Ao ocorrer falha na comunicação é apresentada uma mensagem de erro;
4. O gerente de redes deve realizar a comunicação novamente;
5. Não ocorrendo falha de comunicação é exibida a tela de requisições;
6. Na tela de requisições o gerente de redes deve clicar no botão que deseja obter as informações;
7. Ao clicar no botão é enviada a requisição e o servidor ao tratar a informação, retorna as informação que é a presentada na tela de resultados;

8. Apresentada a tela de resultados no caso da memóra e disco o gerente de redes deve clicar na opção para vizualizar o resultado final;

9. Após a visualização do resultado o gerente de redes deve clicar no botão voltar do próprio dispositivo móvel para retornar a tela de requicição;

Figura 23 *– Tela inicial da aplicação gerente*

A Figura 23 mostra a tela inicial da aplicação gerente, onde no campo de texto é informado o endereço IP da máquina servidora onde foi instalada a aplicação agente. A tela inicial apresenta dois botões que são: botão Conectar e o botão Sair. O botão Conectar é responsável por enviar a mensagem contendo o endereço e a porta de conexão do servidor e o botão Sair é utilizado para finalizar a aplicação.

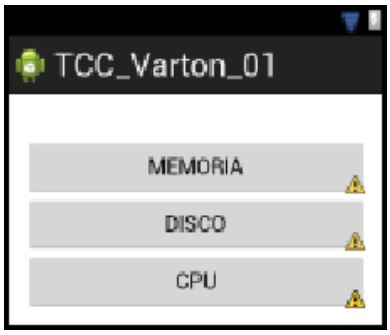

Figura 24 *– Tela de requisições botões de memória, disco e CPU*

A Figura 24 apresenta a tela contendo os botões de requisição onde o Gerente de redes vai clicar para obter a informação de acordo com o botão clicado seja informações da memória, disco ou CPU (central única de processamento).

4.1 Testes Funcionais

O primeiro teste foi realizado na rede local da UFPI. Para realização dos testes a aplicação gerente foi instalada e executada no celular com sistema operacional Android 2.3. como

mostra a Figura 25.

Figura 25 *– Tela inicial com endereço da rede local*

Na tela inicial foi inserido o endereço IP do servidor. Ao clicar no botão CONECTAR foi enviada uma mensagem contendo o endereço e a porta de comunicação do servidor, ao conectar é exibida à tela de requisições contendo os botões de memória, disco e CPU.

Figura 26 *– Tela de Requisições*

Na Tela de requisições da Figura 26 foi solicitado as informações da memória onde obteve o menu de memórias existentes no sistema como ilustra a Figura 27.

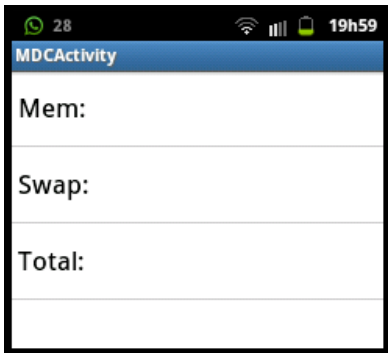

Figura 27 *– Tela de resultado da memóra*

Ao clicar na opção do menu é apresentado os resultados em que apresenta: memóra, espaço usado e livre, em *Megabytes*.

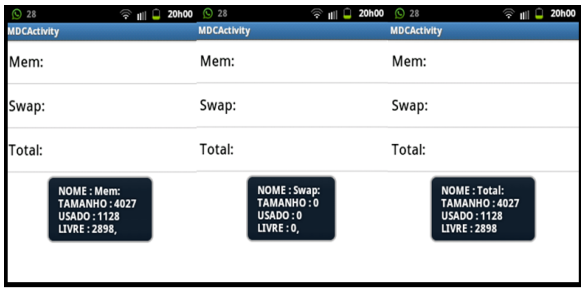

Figura 28 *– Tela de resultados da memória*

Ao retornar a tela de requisições utilizando o botão de retorno do próprio celular foi feita a requisição do disco, na primeira requisição obteve a informação do disco em que o sistema está instalado Figura 29.

Figura 29 *– Tela de resultado do disco rigido*

A efetuar uma nova requisição do disco é visualizada a presença de outros discos, ao análisar a extensão percebe que se trata de uma unidade secundária e outra lógica descrita na tabela 3. O *Linux* possui um arquivo no diretório /dev que identifica o disco através de letras como apresenta a Figura 30.

Figura 30 *– Tela de resultado do disco*

Tipo de disco	Configurações de disco	Número de partições
sd:HD SCSI	a: Master Primária b: Escrava Primária	5: Primeira Partição 1: Segunda Partição 6: Partição lógica

Tabela 3 *– Tabela tipo de disco, configurações do disco e número de partições*

A tabela 3 descreve os discos existentes na máquina servidor na Figura 29 apresenta a partição em que o sistema operacional *Linux* está instalado. A Figura 30 mostra a unidade primária, secundária e lógica, pois na instalação do sistema não foi feita a montagem das partições. Ao montar a partição e adicionar um pendrive no servidor. Ao efetuar uma nova requisição do disco o *software* agente retornou a partição do sistema e as demais adicionadas.

A fazer um novo teste na rede foi apresentado uma mensagem de erro, detectando que o endereço IP foi digitado errado pelo usuário. Essa mensagem foi um tratamento de excessão criado, ja que o protocolo UDP não garante a entrega é necessario que trate a possivel falha de comunicação.

Figura 31 *– Falha na Comunicação*

O segundo teste foi realizado fora de uma rede local, onde a comunicação se deu através da *Internet*, ou seja, tanto o servidor como a aplicação do celular estavam conectados diretamente à *Internet*. Ao efetuar a comunicação entre as aplicações foi utilizado um endereço de IP válido.

Figura 32 *– Tela Comunicação usando à Internet*

Figura 33 *– Falha na comunicação*

A Figura 32 apresenta a tela da comunicação no celular entre as duas aplicações utilizando um endereço IP válido. Durante a realização dos testes, foi encontrado um problema mostrado na Figura 33, pelo fato do IP ser dinâmico e sofrer constantes mudanças, então ao efetuar uma nova comunicação seria necessário saber o novo endereço IP.

A solução encontrada para este problema foi à utilização de um serviço DNS Dinâmico (*Domain Name System*) onde você atribui o seu endereço IP a um nome fixo como o endereço **"serverdroid.no-ip.org"** para a conexão entre as aplicações. Em seguida foi feita um redirecionamento de porta nas configurações do roteador para que a conexão fosse redirecionada para a máquina servidor dentro da rede local, obtendos os resultados desejados.

Figura 34 *– Tela de comunicação com dominio*

A Figura 34 refere a comunicação da aplicação gerente usando um domínio criado para a comunicação entre as aplicações, ao clicar no botão Conectar foi realizada a comunicação e os resultados foram adquirido.

5 Considerações Finais

Este trabalho objetivou o desenvolvimento de duas aplicações (gerente/agente) que auxiliasse o gerente de redes na tomada de decisões no funcionamento da máquina servidor viabilizando maior eficiência, levando em consideração que na ausência do gerente de redes em seu ambiente de trabalho sua eficiência é notável, pois o permitirá obter as informações necessárias para posteriormente resolver os problemas referentes ao funcionamento da máquina servidora. A metodologia utilizada foi experimerimental e dividida em três fases: definição do ambiente de desenvolvimento, definição da linguagem de programação e utilização do meio de comunicação.

O principal motivo de não utilizar o protocolo TCP foi por este ser orientado a conexão e dependendo da carga de trabalho de sua utilização poderia ocorrer uma sobrecarga na rede. Desta forma foi escolhido o protocolo UDP, que não é orientado a conexão, não sendo necessário manter uma conexão entre a aplicação agente e a gerente. Cada vez que o gerente de rede solicitar uma informação do servidor a mesma será enviada e retornada de maneira rápida, prática e eficiente, sem a necessidade de manter a conexão após a solicitação.

Os resultados obtidos nos experimentos foram satisfatórios, comprovando que o tempo de obtenção das informações foi mantido dentro dos níveis esperados na fase de projeto das aplicações, considerando a qualidade e carga de utilização da rede no momento da transmissão dos dados.

Para trabalhos futuros propõe-se o aperfeiçoamento das aplicações para que assim seja possível efetuar o gerenciamento da máquina servidor, executando ações através de comandos enviados diretamente da aplicação gerente, havendo ainda a possibilidade de encerrar processos e até mesmo verificar o tráfego da rede.

Referências

ANISZCYK, Chris; GALLARDO, David. ***Introdução a Plataforma Eclipse***. 2012. Disponível em: <https://www.ibm.com/developerworks/br/library/os-eclipse-platform/>. Acesso em: 23 de Jun. 2013.

BALSEMãO, Fabio Torres. ***Gerência e monitoramento de redes através de dispositivos móveis***. 2008. Disponível em: <http://hdl.handle.net/10183/15969>. Acesso em: 04 de Nov. de 2013.

CASE, Jeffrey Dhon et al. ***A Simple Network Management Protcol (SNMP)***. 1990. Disponível em: <http://www.rfc-base.org/txt/rfc-1157.txt>. Acesso em: 10 de Ago. 2013.

DEITEL, P. J.; DEITEL, H. M. *Java: como programar*. 6. ed. São Paulo: Pearson Education do Brasil, 2006.

DIAS, Henrique de Lima. ***A Importancia do monitoramento de ativos de redes:Um estudo de caso com o sistema CACIC***. 2008. Monografia.

GUEDES, Gillianes T. A. *UML 2 Uma Abordagem Prática*. [S.l.]: Novatec, 2011.

IDGNEWS, Service / EUA. ***iOS registra o crescimento mais rápido em 2013; Android ainda lidera***. 2013. Disponível em: <http://idgnow.uol.com.br/internet/2013/10/25/ios-registra-o-crescimento-mais-rapido-em-2013-android-ainda-lidera/>. Acesso em: 25 de Out. de 2013.

KUROSE, James F. *Redes de Computadores e a Internet*. 2010. 614 p. Livro.

LECHETA, Ricardo R. ***Google Android Aprenda a criar aplicações para dispositivos móveis com o Android SDK***. 2010.

MARTINS, Rafael J. W. de A. *Desenvolvimento de Aplicativo para Smartphone com a Plataforma Android.* 2009. Disponível em: <http://www.icad.puc-rio.br/~projetos/android/files/monografia.pdf>. Acesso em: 15 de Ago. 2013.

MAURA, Douglas R.; SCHMIDT, Kevin J. ***SNMP Essencial***. 2001. 336 p. Livro.

MELCHIORS; CRISTINA. ***Raciocínio baseado em casos aplicado ao gerenciamento de falhas em redes de computadores***. 2010. Dissertação. Disponível em: <http://www.lume.ufrgs.br/handle/10183/26310>. Acesso em: 29 de Out. 2013.

ORACLE. ***ORACLE Java***. 2013. Disponível em: <http://www.oracle.com/us/technologies/java/overview/index.html>. Acesso em: 26 de Mar. 2013.

PAMPLONA; FERNANDO, Vitor. *Tutorial.[Internet: http://javafree. uol. com. br/artigo/871498/Tutorial-Java-O-que-e-Java. html*. [S.l.]: Campus, 2009.

PINHEIRO, Jose Mauricio Santos. ***Gerenciamento de Redes de Computadores***. 2002. Dissertação. Acesso em: 01 de Nov. 2013.

PINHEIRO, Jose Mauricio Santos. ***Gerenciamento de Redes de Computadores: Uma Breve Introdução***. 2006. Artigo. Disponível em: <http://www.projetoderedes.com.br/artigos/artigo_gerenciamento_de_redes_de_computadores.php>. Acesso em: 01 de Nov. 2013.

POSTEL, Jonathan Bruce. ***RFC 768 - User Datagram Protocol***. 1980. Disponível em: <http://tools.ietf.org/rfc/rfc768.txt>.

SANTIAGO, Rodrigo Marques de Melo. ***Aplicação cliente/servidor utilizando sockets***. 2009. Disponível em: <http://www.gta.ufrj.br/~santiago/trabalhos.html>. Acesso em: 26 de Ago. 2013.

SANTOS, Vanessa de Moura; SOUSA, Aislan Rafael Rodrigues de; NETO, Otilio Paulo da Silva. ***Introdução a Plataforma Android***. 2002. Disponível em: <http://pt.scribd.com/doc/15981338/Introducao-a-Plataforma-Android>. Acesso em: 02 de Out. 2013.

SCRIMGER, Rob et al. Rua Sete de Setembro 111 16o andar Centro Rio de Janeiro: Campus, 2002.

SOARES, Diego. ***O que é Android?*** 2011. Disponível em: <http://www.mestreandroid.com.br/google-android/>. Acesso em: 29 de Out. 2013.

SZTAJNBERG, Alexandre. ***Conceitos Básicos sobre os Protocolos SNMP e CMIP***. 1996. COPPE-Coordenação dos Programas de Pós-Graduação. Disponível em: <http://cassio.orgfree.com/disciplinas/gredes/GerenciamentoRedes.pdf>. Acesso em: 29 de Abril 2013.

TANENBAUM, Andrew S. ***Redes de Computadores***. 2003. 614 p. Livro.

WERNER, Rodrigo. ***Gerência de Redes***. 2010. Disponível em: <http://rodrigowerner.tripod.com/>. Acesso em: 30 de Out. 2013.