UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CURSO DE ESPECIALIZAÇÃO EM CONFIGURAÇÃO E GERENCIAMENTO DE SERVIDORES E EQUIPAMENTOS DE REDE

JOÃO ÉDER ANCELMO LEMES

MONITORAMENTO EM REDES IPv6 COM ZABBIX E RASPBERRY PI

MONOGRAFIA

CURITIBA
2013

JOÃO ÉDER ANCELMO LEMES

MONITORAMENTO EM REDES IPv6 COM ZABBIX E RASPBERRY PI

Monografia apresentada como requisito parcial para a obtenção do grau de Especialista em Configuração e Gerenciamento de Servidores e Equipamentos de Rede, do Departamento Acadêmico de Eletrônica da Universidade Tecnológica Federal do Paraná – UTFPR.
Orientador: Prof. MSc. Christian Mendes

CURITIBA
2013

DEDICATÓRIA

Dedico este trabalho a minha amada mãe, por seu amor, carinho e educação que levarei para toda a minha vida como um exemplo a ser passado adiante.

AGRADECIMENTOS

Agradeço primeiramente a Deus, pois sem Ele, nada é possível.

A minha família, que trouxe apoio e compreensão, tornando esta etapa em minha vida possível.

A todos os meus professores, desde sempre, que têm meu respeito e admiração, obrigado por sempre me indicarem o caminho do verdadeiro conhecimento, e por me mostrarem que é possível aprender mais com erros do que acertos.

RESUMO

LEMES, João Éder Ancelmo. **Monitoramento de Redes IPv6 com Zabbix e Raspberry Pi.** 2013. 64 páginas. Monografia (Especialização em Configuração e Gerenciamento de Servidores e Equipamentos de Redes). Universidade Tecnológica Federal do Paraná. Curitiba, 2013.

Esta monografia pretende demonstrar uma solução de baixo custo para monitoramento de redes IPv6 com o *software* Zabbix. A topologia da rede terá um Access Point com suporte a IPv6 conectado a Internet, um servidor de gerenciamento em cima da plataforma Raspberry onde será instalado o *software* de monitoramento Zabbix, um servidor com Windows 2008 Server rodando IIS e MSSQL Server onde será instalado o agente Zabbix para coleta de dados dos serviços. A premissa é demonstrar que é possível montar uma solução de monitoramento para pequenas redes com um investimento extremamente pequeno, e que o baixo custo não reflete na qualidade ou desempenho do sistema.

Palavras-chave: IPv6. Zabbix. Monitoramento de redes. Raspberry Pi.

ABSTRACT

LEMES, João Eder Ancelmo. **Network Monitoring with Zabbix IPv6 and Raspberry Pi**. In 2013. 64 pages. Monograph (Specialization in Configuring and Managing Servers and Networking Equipment). Federal Technological University of Paraná. Curitiba, 2013.

This monograph aims to demonstrate a low-cost solution for monitoring IPv6 networks with Zabbix software. The network topology will have an Access Point that supports IPv6 connected to the Internet, a management server on the platform Raspberry where you install your software monitoring Zabbix, a server with Windows 2008 Server running IIS and MSSQL Server where you install the agent Zabbix for data collection services. The premise is to demonstrate that it is possible to set up a monitoring solution for small networks with a very small investment, and that low cost does not reflect the quality or performance of the system.

Keywords: IPv6. Zabbix. Monitoring networks. Raspberry Pi.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

Neste capítulo serão apresentados os motivos que levaram o desenvolvimento do projeto e os elementos preliminares relacionados ao estudo e simulação do monitoramento de redes IPv6 com *software* de monitoramento *Open-Source* Zabbix.

1.1 TEMA

As redes de computadores tornaram-se essenciais para a vida moderna, seria impensável se o mundo moderno simplesmente parasse por falta de planejamento. Isso quase aconteceu. Em 2011 a IANA (*Internet Assigned Numbers Authority*), maior autoridade da Internet, distribuiu os últimos blocos IPv4 (*Internet Protocol version* 4) disponíveis. A falta de planejamento já causou muitos transtornos no mundo IPv4 devido a sua distribuição descontrolada e, em consequência o esgotamento de endereços, levou ao desenvolvimento de um novo protocolo, o IPv6 (*Internet Protocolo version* 6). Seu desenvolvimento foi feito procurando contornar os problemas enfrentados no IPv4, além de oferecer um número quase ilimitado de endereços, ele melhora a performance no roteamento dos pacotes através da Internet (BRITO, 2013). O IPv6 é um novo desafio que surge para os administradores de rede.

Com o advento do novo protocolo, diversas ferramentas do antigo IPv4 deverão ser atualizadas para suportar o IPv6, os sistemas operacionais e milhares de aplicações já estão prontas e adaptadas a nova realidade (BRITO, 2013). Nesse contexto encontram-se também os *softwares* de gerenciamento e monitoramento de redes, como Cacti, Nagios, OpenNMS, Zabbix entre outros.

1.2 PROBLEMAS E PREMISSAS

Quando as redes ainda eram projetos de pesquisa, não se tinha uma estrutura organizada e interligada e nem era considerada um serviço essencial como hoje, portanto não havia necessidade de gerência ou monitoramento, os problemas eram pontuais e geralmente fáceis de resolver com testes simples e configurações básicas no sistema, pois não havia milhões de dispositivos e usuários como atualmente. Quando foi percebido o potencial das redes, seu crescimento começou a ocorrer de forma exponencial, junto com sua expansão e adoção cresceu também os problemas e a preocupação em manter a disponibilidade. A necessidade de

controlar, monitorar e coordenar os dispositivos de *hardware* e *software* era evidente (KUROSE & ROSS, 2010).

A administração da rede pode ser um problema pois as redes hoje são em sua maioria heterogêneas e geralmente extensas com sítios remotos, e portanto de difícil administração, a detecção de falhas em redes remotas torna-se mais trabalhosa. Cabe ao administrador de rede prevenir, detectar e corrigir problemas de *hardware* e *software* que possam tornar a rede ineficiente ou indisponível, para isso deve monitorá-las (COMER, 2007).

1.3 OBJETIVOS

1.3.1 OBJETIVOS GERAIS

Avaliar o desempenho geral do computador Raspberry Pi em um ambiente corporativo pequeno simulado, criando um cenário propício onde:

- Será utilizado o computador Raspberry Pi como servidor de monitoramento, onde será instalado o Zabbix Server;
- Serão monitorados os recursos de *hardware* de um servidor com Windows 2008 Server, um Servidor Linux e dois computadores com Windows 7, todos com IPv6 e IPv4;
- Alertar que a transição para o IPv6 é necessária e que os *softwares* preparados para o novo protocolo já estão funcionais e não apresentam grandes mudanças de interface;
- Fornecer a documentação necessária para a instalação e configuração do gerente e agente Zabbix para monitorar uma rede IPv6 no Raspberry Pi.

1.3.2 OBJETIVOS ESPECÍFICOS

- Instalar a versão gerente (*Server*) e agente do *software* Zabbix;
- Configurar o *software* para monitorar os recursos de *hardware* nos servidores Windows e Linux;
- Avaliar o desempenho do Zabbix na plataforma Raspberry Pi;

- Não será aprofundado o entendimento dos protocolos utilizados pelo *software*;
- Não serão avaliados os dados e relatórios extraídos ou gerados pelo *software* dos servidores Windows e Linux;
- Não serão resolvidos possíveis problemas decorrentes da instalação do *software*;

1.4 JUSTIFICATIVA

A tecnologia invadiu nossas casas e trabalhos, é difícil imaginar uma pequena empresa hoje que não tenha recursos tecnológicos que envolvam Tecnologia da Informação (TI), a grande maioria precisa de um servidor com banco de dados e seu sistema de gestão de clientes, além dos dispositivos básicos para dar conectividade a seus funcionários como *switches*, roteadores, modens etc. Esses componentes acabam sendo essenciais em nosso dia-a-dia, porém tem-se o costume de levar os recursos tecnológicos como custo e não como investimento e acreditar que uma solução eficaz é muito dispendiosa. O monitoramento e gerenciamento de recursos de TI é crucial para alguns segmentos, e temos como justificativa:

- Demonstrar a facilidade em configurar o *software* de monitoramento Zabbix na plataforma de baixo custo do computador Raspberry Pi;
- Utilizar o protocolo IPv6 em toda a rede e serviços, demonstrando que já existem ferramentas preparadas para o novo protocolo;
- Fornecer uma solução para pequenas empresas monitorarem servidores com custo relativamente baixo;
- Fornecer documentação para implementar o Zabbix Server e Agent;
- Avaliar o desempenho do Raspberry Pi utilizando o próprio Zabbix Server.

1.5 METODOLOGIA

Realizar uma pesquisa bibliográfica para entender os conceitos de monitoramento e gerenciamento de redes de computadores e do novo protocolo IPv6. Após abordar e entender os conceitos, será realizada uma pesquisa de caráter exploratório experimental com o objetivo de reunir informações úteis para este

projeto tendo como referência livros, revistas, fóruns oficiais, internet e manuais do desenvolvedor.

Neste projeto é utilizado o modelo B do Raspberry Pi. A versão do *software* Zabbix testada é a 2.2.1, lançada em Dezembro de 2013. A instalação dos aplicativos ocorrerá no Raspbian, um sistema operacional customizado para o Raspberry Pi a partir do Linux Debian 7.3 Wheezy.

Os processos de instalação e configuração do Zabbix Server e Agent serão documentados de forma detalhada. Não serão abordados os processos de instalação do sistema operacional Raspbian e dependências.

A rede utilizada será projetada especificamente para o projeto e montada em laboratório, não sendo uma rede em produção.

Serão testadas as funcionalidades de:

- Desempenho do Raspberry Pi.
- Ferramentas de medição do Zabbix;
- Ferramentas de mostra de dados do Zabbix.

Após o cenário montado, será realizado análise dos dados coletados no monitoramento do Raspberry Pi para avaliar seu desempenho.

2 CONCEITOS DE GERENCIAMENTO DE REDES

2.1 GERENCIAMENTO DE REDES

Nos anos 80 foi introduzido na norma 10040 da *International Organization for Standardization* (ISO), que trata da Interconexão de Sistemas Abertos, os termos *Fault*, *Configuration*, *Accounting*, *Performance* e *Security* (FCAPS – Falha, Configuração, Contabilidade, Desempenho e Segurança) (ITU-T, 1999), este é um modelo criado pela ISO para detectar e corrigir problemas da infraestrutura de rede com maior agilidade, segundo KUROSE este é “um modelo de gerenciamento de rede que é útil para situar os cenários apresentados em um quadro mais estruturado” (KUROSE & ROSS, 2010, p. 573).

O termo FCAPS faz referência a 5 partes: (KUROSE & ROSS, 2010)

- Gerenciamento de Falhas (*Fault*): seu objetivo é detectar, registrar e reagir a falhas identificadas;
- Gerenciamento de Configuração (*Configuration*): permite a um administrador de rede saber quais os dispositivos na rede monitorada e suas configurações de *hardware* e *software*;
- Gerenciamento de Contabilização (*Accounting*): permite ao administrador de rede especificar, registrar e controlar o acesso a recursos da rede;
- Gerenciamento de Desempenho (*Performance*): sua meta é analisar e controlar o desempenho através de medições nos dispositivos;
- Gerenciamento de Segurança (*Security*): a meta da segurança é controlar o acesso de acordo com a política de segurança definida, engloba a distribuição de chaves e autoridades certificadoras.

De acordo com LOPES (2009) de modo amplo a arquitetura de gerência de redes apresenta quatro componentes, são eles:

- Componentes Gerenciados: possui um agente que monitora e controla o equipamento através da estação de gerência.
- Sistema de Gerência: controla e monitora os agentes através de um *software* de gerenciamento, também conhecido como gerente.
- Protocolo de Gerência: organiza a troca de informações entre gerente e agente para monitoramento e controle.

- Informações de Gerência: definem quais dados podem ser trocados entre gerente e agente.

No surgimento da preocupação da necessidade de monitoramento de redes foi percebido a necessidade de um protocolo para reger a comunicação entre gerente e agente. No final dos anos 80 surgiram dois padrões considerados os mais promissores, o primeiro criado pela ISO é o *Common Management Service Element/Common Management Information Protocol* (CMISE/CMIP – Elemento de Serviço de Gestão Comum/Protocolo de Informação de Gerenciamento Comum), o segundo criado pelo *Internet Engeneering Task Force* (IETF – Força Tarefa de Engenheiros da Internet) é o *Simple Network Management Protocol* (SNMP – Protocolo Simples de Gerenciamento de Rede). Este último foi projetado e oferecido mais rápido no momento da necessidade de um protocolo de gerenciamento, e acabou alcançando maior aceitação. Atualmente, este protocolo é amplamente utilizado na estrutura de gerenciamento de rede (KUROSE & ROSS, 2010).

A definição de gerenciamento é definida em uma única sentença por SAYDAM (SAYDAM & MAGEDANS, 1996, pp. 345-348):

> "Gerenciamento de rede inclui o desenvolvimento, a integração e coordenação de todo o *hardware*, *software* e elementos humanos para monitorar, testar, consultar, configurar, analisar, avaliar, e controlar os recursos da rede e seus elementos para atender em tempo real os requisitos de desempenho operacional e de qualidade de serviço a um custo razoável."

É fácil notar que o gerenciamento de rede não é uma tarefa simples, porém é fundamental para o bom funcionamento da rede, e envolve vários componentes de *hardware* como roteadores e servidores e componentes de *software* como protocolos e aplicações, portanto deve ser um processo bem planejado e contínuo que abrange toda a rede e acompanha o seu crescimento.

2.2 MONITORAMENTO

Atualmente o principal protocolo utilizado para monitoramento e gerenciamento de rede é o SNMP desenvolvido pela IETF e definido por diversas RFCs [1]. O SNMP é um protocolo utilizado para facilitar o monitoramento e gerenciamento de uma rede de computadores, ele fornece um conjunto simples de operações que permitem coletar informações e configurar parâmetros em dispositivos remotos (MAURO & SCHMIDT, 2005).

[1] RFC: *Request for Comments* – Documentos que descrevem os diversos padrões de protocolos da Internet.

O monitoramento de redes possui duas entidades (MAURO & SCHMIDT, 2005):

- Gerente: também conhecidos como estações de gerenciamento de rede (NMSs - *Network Management Stations*), são basicamente *softwares* capazes de gerenciar uma rede e são instalados em um servidor, eles executam operações de *polling* (sondagem), que consiste na tarefa de consultar informações dos agentes. Um administrador de rede pode acessar o gerente para visualizar os dados de um nó monitorado.

- Agente: um programa executado nos clientes, é responsável pela coleta das informações do dispositivo e se reporta ao gerente em duas situações: em resposta a uma solicitação feita pelo gerente, ou em resposta a um evento ocorrido no dispositivo, neste caso o agente avisa o gerente sobre o evento sem intervenção do gerente, esta ação é chamada de *trap*.

2.3 A ESTRUTURA DE GERENCIAMENTO

A estrutura de gerenciamento padrão da Internet é constituída por quatro partes (KUROSE & ROSS, 2010):

- SMI – *Structure of Management Information*: é a linguagem utilizada que define sintaxe e semântica das informações trocadas, é baseada na ASN.1 (*Abstract Syntax Notation One* – Notação de Sintaxe Abstrata 1).

- MIB – *Management Information Base*: é uma base virtual de informação formada por objetos monitorados (*hardware* – placa de rede/cpu ou *software* – serviço/aplicação) que é acessada pelos agentes.

- SNMP: protocolo padrão utilizado para troca de informações entre gerentes e agentes.

- Capacidade de segurança e administração: implementa autenticação e maior controle sobre os dados transmitidos, o protocolo que melhor suporta essa funcionalidade é o SNMPv3.

O SNMP não especifica uma MIB, ele define apenas um padrão no formato das mensagens e sua codificação. Os objetos da MIB são definidos através na SMI pela ASN.1, cada objeto deve possuir um nome único e seu prefixo é longo e hierárquico para garantir isso. Por exemplo, abaixo tem o nome e a representação via notação ASN.1 do objeto que contém o número de endereço IPv6 de uma interface:

iso.org.dod.internet.mgmt.mib-2.ip.ipv6interfacetable.ipv6interfaceentry

O nome do objeto, ao ser representado em uma mensagem SNMP será representado por um inteiro para ser mais compacto:

1.3.6.1.2.1.4.30.1

Segundo o IETF, o SNMP é parte de um grupo de protocolos para acompanhamento da Internet. Seu antecessor foi o *Simple Gateway Management* Protocol (SGMP) que foi desenvolvido para gerenciar roteadores na Internet. O SNMP possui a capacidade de monitorar praticamente qualquer tipo de dispositivo, a nível de *hardware* e *software*, desde roteadores, *switches*, impressoras etc., até sistemas operacionais e banco de dados, não se limitando a estes.

Utilizado para transporte de informações de gerenciamento de rede, o SNMP define a comunicação entre gerente e agente. O modelo de comunicação utilizado é simples, tem a função de carregar e armazenar valores e é chamado de *fetch-store*, esse modelo consiste em duas tarefas básicas, obter um valor e configurar um valor no dispositivo (DUARTE, 2010). Essas operações fazem parte da interação entre um gerente e um agente (COMER, 2007).

Uma rede gerenciada com SNMP possui três componentes básicos (COMER, 2007):

- Dispositivo gerenciado e seu *software* (contém um agente SNMP);
- Agentes (coleta informação da MIB e envia à NMS);
- Gerentes ou NMS (controlam, recebem e processam os dados dos agentes).

É possível notar que a simplicidade remetida ao S do protocolo SNMP não é tão trivial assim, existem diversos fatores que devem ser levados em consideração ao se implementar o monitoramento de redes, é necessário ao administrador conhecê-las.

3 TEORIA

Atendendo a proposta do projeto, foi necessário avaliar qual o *software* mais adequado para uma implementação simples, porém completa e escalável. Como citado, existem várias soluções para monitorar e gerenciar uma rede, uma muito comum é a união de dois dos *softwares* mais conhecidos, o Nagios e o Cacti, juntos eles oferecem monitoramento e gerenciamento da rede, porém sua configuração pode se tornar complexa pois necessita de plug-ins externos para funções adicionais.

A solução escolhida foi o Zabbix, pois sua autossuficiência diminui a dependência de *plug-ins*, oferecendo vários serviços nativos com uma interface *web* amigável e de simples instalação. No próximo tópico será abordado mais sobre o Zabbix e as características que o elegeram para este projeto.

3.1 ZABBIX

O Zabbix foi criado por Alexei Vladishev, é uma solução *open-source* de monitoração de ativos para empresas e atualmente é desenvolvido e suportado pela Zabbix SIA. Oferece mecanismos de notificação flexíveis como emails de alerta baseado em eventos, geração e acesso rápido a relatórios e estatísticas através de seu *frontend* que é a interface *web*, é capaz de atender grandes organizações com estruturas heterogêneas e robustas, como também pequenas organizações com estrutura simples e tem uma baixa curva de aprendizado. É distribuído de acordo com a *Gerenal Public Licence – version 2*, o suporte comercial está disponível e é fornecido pela Zabbix Company (VLADISHEV, 2013).

Segundo OLUPS (2010, p. 9), o Zabbix pode ser considerado um sistema de monitoramento semidistribuído com gerenciamento centralizado, ele provê diversas formas de monitorar os vários parâmetros da infraestrutura de TI, utilizando *proxies*, e agentes.

Segue algumas das funcionalidades do Zabbix (Zabbix SIA, 2013):

- Monitoramento de recursos;
- Gerenciamento de ativos;
- Notificação (email, sms, jabber);
- Agentes de monitoramento;
- Interface *Web* amigável e configurável;

- Relatórios e estatísticas.

O Zabbix possui quatro componentes básicos que interagem para dar suporte ao monitoramento de rede (VLADISHEV, 2013):

- Zabbix Server
- Zabbix Agent
- Interface Web Zabbix
- Zabbix Proxy

Na figura 1 é apresentado um diagrama simples de como é possível implementar os componentes do Zabbix.

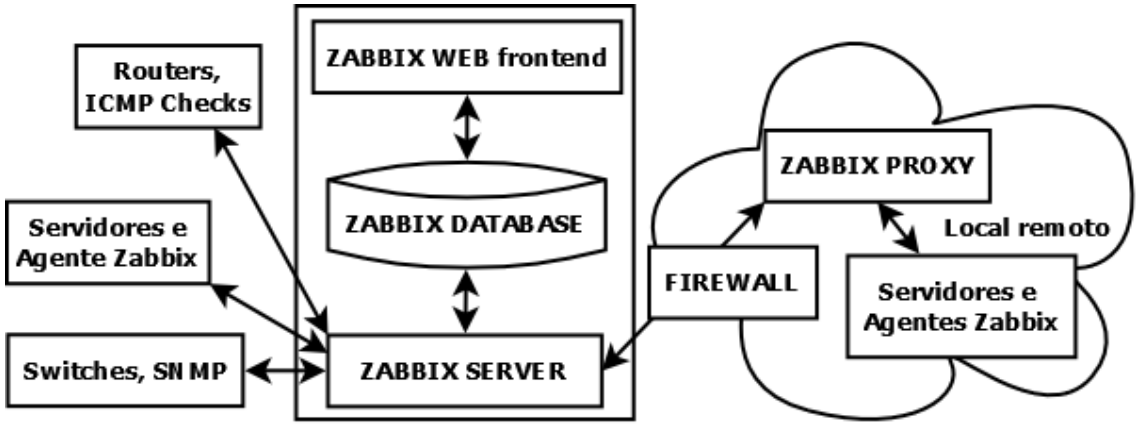

Figura 1: Diagrama de exemplo de implementação de componentes Zabbix
Fonte: Adaptado de OLUPS (2010, p. 9)

Neste projeto serão abordados apenas o Zabbix Server, Agent e a Interface Web, o Zabbix Proxy é utilizado para segmentar e/ou realizar o monitoramento de sítios remotos.

3.1.1 ZABBIX SERVER

O servidor Zabbix é o processo central do *software* que interage com agentes, *proxies* e banco de dados, administra alertas e é o repositório central de configurações e dados estatísticos e operacionais. Seu funcionamento básico possui três componentes distintos, o servidor Zabbix, o *frontend web* e o armazenamento de dados (VLADISHEV, 2013).

O servidor Zabbix é responsável pela coleta dos dados dos agentes, seu monitoramento pode ser através do *software* agente nativo do Zabbix ou agentes

SNMP, esta opção é para quando o agente Zabbix não é compatível com o sistema operacional. O Zabbix utiliza o protocolo de comunicação baseado em *JavaScript Object Notation* (JSON) para comunicação com o Zabbix *Agent* (VLADISHEV, 2013).

3.1.2 ZABBIX AGENT

O agente Zabbix é instalado no *host* a ser monitorado, ele monitora ativamente recursos e aplicações. Os dados coletados são enviados ao servidor Zabbix para processamento e em casos de falha ou um evento específico os administradores podem ser alertados imediatamente pelo servidor Zabbix (VLADISHEV, 2013).

Os agentes Zabbix podem realizar ações ativas ou passivas. Na verificação passiva o agente responde a um pedido do servidor Zabbix. As verificações ativas são mais complexa, pois obtém uma lista de itens do servidor Zabbix e periodicamente envia novos valores ao servidor (VLADISHEV, 2013).

O agente Zabbix é compatível com várias plataformas: Linux, Windows, Unix-like, MacOS etc.

3.1.3 INTERFACE WEB ZABBIX

A interface *web* do Zabbix, também conhecida como *frontend*, é parte integrante do Zabbix Server e permite acompanhar o estado da rede de qualquer lugar, foi escrita na linguagem PHP e através dela é realizado a inserção dos *hosts* a serem monitorados e configurado os parâmetros de alertas.

3.2 PROTOCOLO IPV6

A principal motivação para o desenvolvimento do novo protocolo foi sem dúvidas o esgotamento de endereços IPv4 (BRITO, 2013), porém com a oportunidade de desenvolver algo novo, tendo a experiência de conhecer as deficiências do antigo protocolo, ajudou a compreender as necessidades atuais e futuras da Internet, assim foram melhorados outros aspectos relacionados ao novo protocolo da Internet como segurança e desempenho, e não apenas o problema de endereçamento.

Para os usuários as mudanças são quase transparentes, grande parte se não todos os sistemas operacionais modernos já estão prontos para o novo protocolo, essa transição amigável é devido ao modelo em camadas do conjunto de protocolos *Transmission Control Protocol/Internet Protocol* (TCP/IP). Porém para os administradores de rede a mudança não será trivial. Serão vistas as principais mudanças do protocolo IPv6 com relação ao IPv4 a seguir.

3.2.1 CABEÇALHO

O cabeçalho do protocolo IP contém as informações de controle que são utilizadas para determinar sua origem e destino, é com base nas informações contidas no cabeçalho que os roteadores na Internet determinam a rota dos pacotes.

Os cabeçalhos de ambos os protocolos (IPv4 e IPv6) são diferentes, houve mudanças significativas pois foram eliminados, criados e renomeados alguns campos. Nas figuras 2 e 3 temos os cabeçalhos dos protocolos e uma breve comparação (BRITO, 2013).

Cabeçalho IPv4:

<table>
<tr><td>Versão</td><td>IHL[1]</td><td colspan="2">Tipo de Serviço (ToS)</td><td>Tamanho Total</td></tr>
<tr><td colspan="3">Identificação</td><td>Flags</td><td>Fragmentação</td></tr>
<tr><td colspan="2">Tempo de Vida (TTL[2])</td><td>Protocolo</td><td colspan="2">Verificação de erro no cabeçalho</td></tr>
<tr><td colspan="5">Endereço de Origem</td></tr>
<tr><td colspan="5">Endereço de Destino</td></tr>
<tr><td colspan="5">Opções</td></tr>
</table>

Figura 2: Cabeçalho do protocolo IPv4
Fonte: Adaptado de BRITO (2013, p. 41)

Os campos em cinza foram removidos do cabeçalho IPv6, os campos em branco foram mantidos ou renomeados. O cabeçalho IPv4 pode variar entre 20 e 60 *bytes*. Segue uma breve descrição dos campos (BRITO, 2013):

- Versão: Indica a versão do protocolo, preenchido com o valor 0100_2 que representa o IPv4, mantido;
- IHL (Tamanho do Cabeçalho): Indica o tamanho do cabeçalho IPv4, fundamental pois o cabeçalho IPv4 tem tamanho variável, no IPv6 o cabeçalho tem tamanho fixo em 40 *bytes*, tornando o campo desnecessário portanto foi removido;

[1] IHL: Internet *Header Length* – Tamanho do Cabeçalho de Internet.
[2] TTL: *Time to Live* – Tempo de Vida.

- Tipo de Serviço (*Type of Service* - ToS): É utilizado para fazer a marcação dos pacotes ao aplicar Qualidade de serviço (*Quality of Service* - QoS), renomeado;
- Tamanho Total: Definido em *bytes*, é o tamanho total do pacote IP, cabeçalho mais dados, renomeado;
- Identificação: Permite identificar fragmentos de um pacote original, removido;
- *Flags*: Usado para indicar, identificar e controlar fragmentos, removido;
- Fragmentação: Utilizado para determinar o local de um fragmento num pacote, removido;
- Tempo de Vida: Limite de saltos ou roteadores que o pacote pode passar antes de ser descartado, renomeado;
- Protocolo: Indica protocolo da camada superior (ex.: TCP[1], UDP[2], ICMP[3]), renomeado;
- Verificação de erro no cabeçalho: Valor resultante de um cálculo de Checagem de Redundância Cíclica (CRC ou *Checksum*), removido;
- Endereço de Origem: Endereço IP de origem de 32 *bits*, mantido;
- Endereço de Destino: Endereço IP de destino de 32 *bits*, mantido;
- Opções: cabeçalhos adicionais ou opções específicas, foi removido pois era pouco utilizado.

Devido ao seu tamanho variável e diversas opções, o cabeçalho IPv4 é mais custoso de processar, pois o roteador deve calcular o tamanho do pacote e identificar os campos utilizados e importantes antes de determinar a rota do pacote. Os campos Versão, Endereço de Origem e Endereço de Destino foram mantidos no novo cabeçalho IPv6. Os campos ToS, Tamanho Total, Tempo de Vida e Protocolo foram renomeados para descrever melhor sua função, os demais campos IHL, *Flags*, Fragmentação, Verificação de erro e Opções foram removidos. Essas mudanças se devem a nova forma que o protocolo IPv6 trabalha, será visto que essas mudanças tornam o cabeçalho mais simples e rápido (BRITO, 2013).

O cabeçalho do IPv6 foi planejado para oferecer grande flexibilidade e agilidade no processamento, além de excluir alguns campos e renomear outros, alguns dos campos foram realocados para aumentar a agilidade e melhorar sua performance. Com a convergência das redes, temos voz, vídeos e dados trafegando na mesma infraestrutura de rede, porém os serviços de voz e vídeo requerem uma atenção especial, pois são aplicações em tempo real que requerem o mínimo de atraso possível, no IPv4 isso era tratado através do campo ToS (Type of Service) que não oferecia muita flexibilidade, no IPv6 o campo ToS foi renomeado, e foi

[1] TCP: *Transmission Control Protocol* – Protocolo Orientado a Conexão.
[2] UDP: *User Datagrama Protocol* – Protocolo Sem Conexão.
[3] ICMP: *Internet Control Message Protocol* – Protocolo de Mensagens de Controle da Internet.

criado um novo campo para melhorar o desempenho dessas aplicações, o Identificador de Fluxo (BRITO, 2013).

Na figura 3 é ilustrado o cabeçalho do protocolo IPv6:

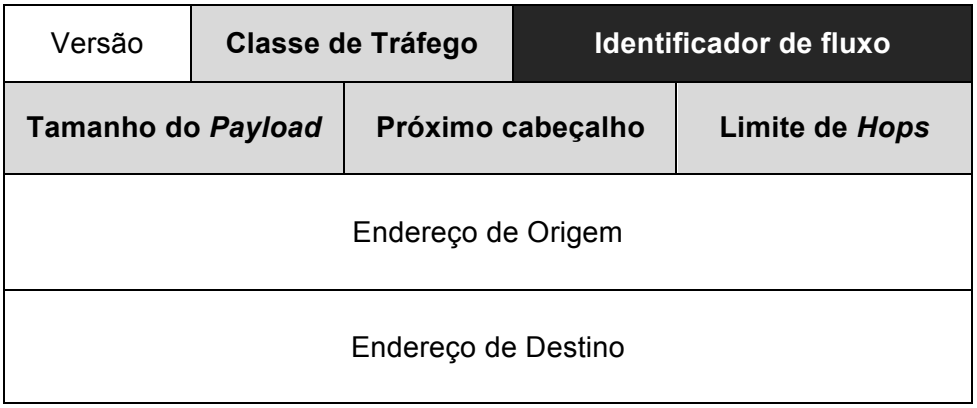

Figura 3: Cabeçalho do protocolo IPv6
Fonte: Adaptado de BRITO (2013, p. 42)

Os campos em branco foram mantidos do cabeçalho IPv4, os cinzas foram renomeados, e o campo em preto é novo, segue uma breve descrição (BRITO, 2013):

- Versão: preenchido com o valor 0110_2, indica a versão do protocolo IPv6;
- Classe de Tráfego: Campo ToS renomeado, provê a mesma funcionalidade de aplicar QoS;
- Identificador de fluxo: Campo novo, criado especialmente para complementar o campo Classe de Tráfego, permite identificar um fluxo de dados, melhorando assim o QoS;
- Tamanho do *Payload*: Campo Tamanho Total renomeado, indica o tamanho total do pacote IPv6;
- Próximo cabeçalho: Campo Protocolo renomeado. No IPv4 este campo definia o protocolo da camada superior, agora ele indica o próximo cabeçalho de extensão após o endereço de destino, isso torna o protocolo mais flexível e possibilita a eliminação do campo Opções e contempla informações complementares. Quando é necessário incluir mais informações no protocolo, basta adicionar um protocolo de extensão e identificá-lo. Os cabeçalhos de camada superior também são identificados como cabeçalhos de extensão como TCP (6), UDP (11), ICMP (1) ou ICMPv6 (58) e são por último.
- Limite de Hops: Número máximo de saltos ou roteadores que o pacote pode passar. Foi renomeado pois este campo não trata de tempo como dá a entender o termo Tempo de Vida, a unidade de medida é salto e não segundos ou minutos.
- Endereço de Origem: Endereço IPv6 de origem de 128 *bits*;

- Endereço de Destino: Endereço IPv6 de destino de 128 *bits*.

Conhecer e entender os campos do cabeçalho do protocolo IPv6 é importante para compreender seu funcionamento.

3.2.2 *INTERNET CONTROL MESSAGE PROTOCOL* - ICMPv6

O ICMP é utilizado no IPv4 e fornece funcionalidades como *Echo Request/Reply*, utilizadas em testes como *Ping* e *Traceroute*, além de funcionalidades de controle que retornam mensagens de erro ou falha. O ICMP é um protocolo relativamente simples, porém fundamental e muito utilizado nas redes IPv4, mas nas redes IPv6 ele ganhou muita importância, se tornando vital para o funcionamento do protocolo (BRITO, 2013).

O protocolo ICMPv6 é indicado no campo Próximo Cabeçalho com valor 58_{10} ou $0x3A_{16}$. O cabeçalho do ICMPv6 segue a linha de simplicidade do seu antecessor, tendo apenas 4 campos como visto na figura 4 (BRITO, 2013):

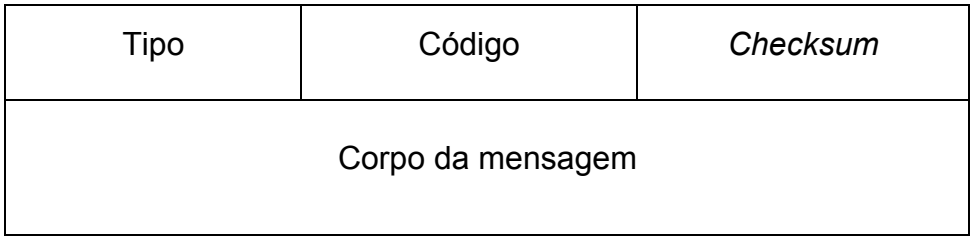

Figura 4: Cabeçalho do protocolo ICMPv6
Fonte: Adaptado de BRITO (2013, p. 78)

Campos:

- Tipo/Código: Identificam o formato da mensagem, Tipo identifica do que trata-se a mensagem e Código a descreve;
- *Checksum*: Verificação de integridade da mensagem;
- Corpo da mensagem: A mensagem real.

É possível dividir as mensagens ICMPv6 em duas categorias:

- Mensagem de erro: Valor de 0 a 127 preenchido no campo Tipo;
- Mensagem de informação: Valor de 128 a 255 preenchido no campo Código.

Além das funcionalidades já conhecidas do ICMPv4, o ICMPv6 assumiu funções essenciais ao funcionamento do IPv6, absorvendo as funcionalidades de outros protocolos como o *Address Resolution Protocol* (ARP), *Reverse* ARP (RARP) e *Internet Group Message Protocol* (IGMP) (BRITO, 2013).

3.2.3 ESTRUTURA DO ENDEREÇO

Como principal fator para necessidade de mudança o endereçamento do IPv6 passou a ser muito maior, alocando quatro vezes mais *bits* do que no IPv4. No IPv4 temos 32 *bits* para endereçamento, o que nos fornece aproximadamente 4,2 bilhões de endereços teóricos, sua representação é decimal e divida em quatro grupos de oito *bits*, chamados de octetos, os octetos são separados por um ponto (BRITO, 2013).

Exemplo:
200.143.134.56

No IPv6 são usados 128 *bits* para endereçamento, o que nos fornece um total de 2^{128} possibilidades, ou 340 undecilhões de endereços teóricos. Sua representação é expressa em hexadecimal e dividido em oito grupos de 16 *bits*, denomina-se cada grupo como decahexateto, duocteto ou ainda quarteto fazendo referência aos quatro algarismos hexadecimais (BRITO, 2013).

Exemplo:
2001:0db8:dad0:cafe:ca5a:bebe:cab0:b105

Algumas particularidades na notação do endereço IPv6 devem ser levadas em consideração, como o endereço é extenso e complexo foram desenvolvidas algumas regras para reduzir seu tamanho e facilitar sua leitura e escrita.

A primeira regra é omitir os *bits* a esquerda que sejam zero.

Exemplo:
2001:**0db8**:**0abc**:**0001**:ab43:ca5a:**0000**:baba

2001:~~**0db8**~~:~~**0abc**~~:~~**0001**~~:ab43:ca5a:~~**0000**~~:baba

2001:**db8**:**abc**:**1**:ab43:ca5a:**0**:baba

A segunda técnica é suprimir sequências de zeros contínuos, substituindo a sequência por duas vezes dois pontos (**::**).

Exemplo:
2001:0db8:**0000**:**0000**:**0000**:ab43:**0000**:**0000**

2001:0db8:**~~0000~~**:**~~0000~~**:**~~0000~~**:ab43:**0000**:**0000**

2001:db8**::**ab43:**0**:**0**

Essa técnica só pode ser utilizada uma única vez no endereço, isso se deve ao fato de um dispositivo pronto para IPv6 saber que o endereço possui 128 *bits*, sendo divididos em 8 grupos de duoctetos, quando o dispositivo encontra os **::** seguidos, sabe que tem que substituir por zeros até completar os 128 *bits* ou 8 grupos de octetos (BRITO, 2013).

No IPv4, ao configurar um equipamento com um endereço IP manual ou automaticamente, é necessário fornecer uma máscara de subrede que geralmente é expressa da mesma forma que o IP, por exemplo: 255.255.255.0. A máscara de subrede corresponde a quantidade de *bits* do endereço que contempla o endereço de rede, a outra forma de expressar uma máscara de subrede é através do prefixo, o prefixo é o número de *bits* reservados para o endereço de rede precedido por barra (*/*) (BRITO, 2013).

Exemplo:

Máscara de subrede /24:

2 5 5 . 2 5 5 . 2 5 5 . 0

8 bits 8 bits 8 bits 8 bits

1 1 1 1 1 1 1 1 . 1 1 1 1 1 1 1 1 . 1 1 1 1 1 1 1 1 . 0 0 0 0 0 0 0 0

Porção de rede /24 **Porção de *host***

O IPv6 eliminou a necessidade de máscaras extensas. Para representar a porção de rede no IPv6 basta escrever seu prefixo precedido do número IPv6, ao realizar a configuração de um equipamento a porção de rede também é definida pelo prefixo.

Exemplo:

2001:0db8:abcd:1234 **/64**

Devido a enorme quantidade de endereços, mudou um pouco a forma de trabalhar no planejamento de distribuição dos endereços IPv6. De acordo com a RFC 4291, todas as redes locais devem ter um prefixo /64. Temos no IPv6 um novo mecanismo de autoconfiguração chamado *Stateless Address Autoconfiguration* (SLAAC). Este recurso elimina em alguns casos a necessidade de um servidor DHCP[1] na rede, pois utiliza o próprio roteador para propagar o prefixo da rede. O SLAAC utiliza o prefixo propagado para calcular o endereço do *host* através do algoritmo *64-bit Extended Unique Identifier* (EUI-64), isso é feito no próprio *host* e utiliza o endereço físico da placa de rede (*Media Access Control* – MAC). Alguns sistemas operacionais como o Windows não utilizam esse mecanismos por questões de privacidade e segurança (BRITO, 2013).

[1] DHCP: *Dynamic Host Configuration Protocol* – Protocolo de configuração dinâmica de *host*.

3.2.4 TIPOS DE ENDEREÇOS

No contexto de redes de computadores, há diferentes tipos de endereços associado com o tipo de comunicação. No IPv4 o endereço pode ser (BRITO, 2013):

- *Unicast*: é o tipo mais utilizado, a comunicação ocorre *host* a *host*, ou seja, o pacote é destinado à apenas um *host* na rede.
- *Multicast*: sua utilização é comum em aplicações de VoIP[1] e *streaming* de vídeos. A comunicação ocorre de um *host* para dois ou mais *hosts*.
- *Broadcast*: a comunicação ocorre de um *host* para todos os outros *hosts* na rede. Fundamental para o funcionamento da rede, porém causa *overhead* e pode se tornar um problema.

No IPv6 houveram mudanças significativas, os endereços de *unicast* e *multicast* permanecem, porém o endereço de *broadcast* já não existe mais. Para atender as necessidades proporcionadas por ele foi criado o grupo *multi-cast-allnodes*, quando uma interface é inicializada ela já ingressa neste grupo que responde no endereço **ff02::1**, é similar a um *broadcast*, porém não é chamado assim pois o tipo de endereço é diferente e para evitar confusão.

Além desta mudança houve a criação de um novo grupo ou modelo de comunicação chamado (BRITO, 2013):

- *Anycast*, neste modelo a comunicação ocorre de um *host* para um *host* em muitos. Pode haver uma preferência para um *host* na escolha da rota por exemplo, ou apenas aquele que responder mais rápido.

A figura 5 contém uma ilustração que exemplifica como ocorre em cada um dos endereços a comunicação:

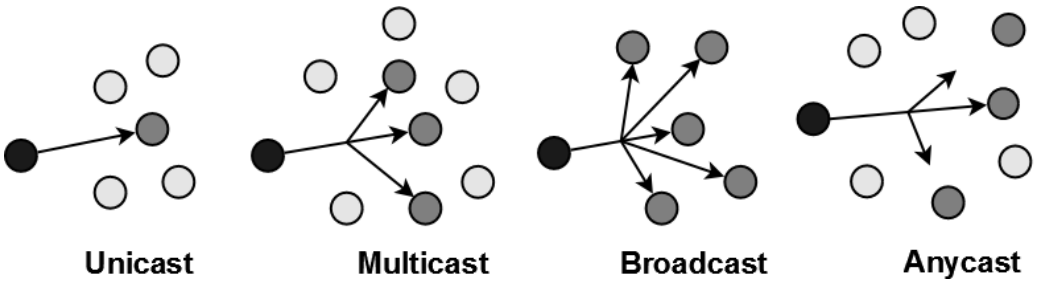

Figura 5: Tipos de comunicação em redes de computadores
Fonte: Adaptado de BRITO (2013, p. 58)

Serão vistos mais detalhes destes endereços a seguir.

[1] VoIP: Voz sobre IP, tecnologia onde é possível compactar e encapsular a comunicação humana em pacotes e encaminhá-la pela rede.

3.2.4.1 *UNICAST*

Os endereços *unicast* são responsáveis pelo endereçamento único e inequívoco dos *hosts*, possibilitando a comunicação fim-a-fim prevista no início da Internet. No IPv6 os endereços *unicast* podem ser *link-local*, *unique-local*, e *global unicast*.

- *Link-local*: esse endereço é atribuído automaticamente a interfaces que tenham suporte ao IPv6 a partir do prefixo **fe80::/10** (RFC 4291) conforme figura 6, os roteadores não tem permissão para encaminhar pacotes com endereços com esse prefixo, portanto são reservados para comunicação local no enlace, por isso o nome *link-local*. Compreendem o intervalo do prefixo fe80::/10 os intervalos: fe80::/10, fe90::/10, fea0::/10 e feb0::10, os próximos 54 *bits* são preenchidos com zero formando o prefixo fe80:0000:0000:0000/64, os últimos 64 *bits* são formados pelo algoritmo EUI-64 (BRITO, 2013).

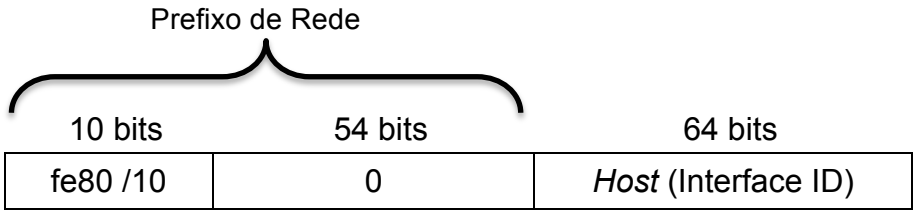

Figura 6: Endereço Link-local
Fonte: Adaptado de BRITO (2013, p. 59)

- *Unique-Local Address* (ULA): análogos aos endereços privados do IPv4 (10./8, 172.16./12 e 192.168./16 – RFC 1918), seus prefixos são fc00::/8 e fd00::/8 definido na RFC 4193, conforme figura 7 (BRITO, 2013).

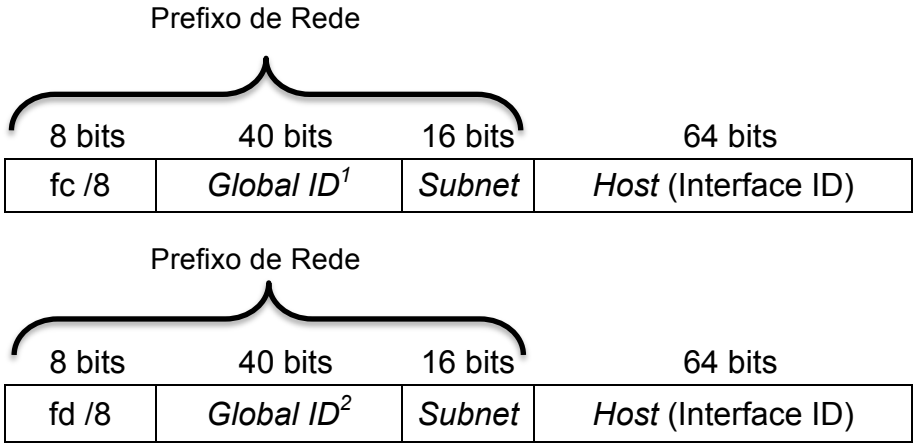

Figura 7: Endereço Unique-local (ULA)
Fonte: Adaptado de BRITO (2013, p. 61)

[1] *Global Unicas*t: Identificador global aleatório atribuído por uma autoridade.
[2] *Global Unicast*: Identificador global aleatório atribuído localmente.

- *Global Unicast*: endereços públicos e roteáveis, cabe a IANA e suas autoridades subordinadas controlar sua distribuição. Apenas uma pequena porção do total de endereço IPv6 foi separada para o *global unicast*, seu prefixo é **2000::/3**, conforme figura 8 (BRITO, 2013).

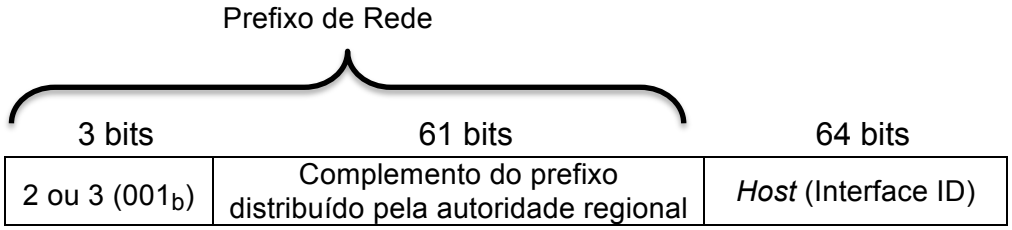

Figura 8: Endereço Global Unicast
Fonte: Adaptado de BRITO (2013, p. 62)

3.2.4.2 *MULTICAST*

Os endereços *multicast* são utilizados em aplicações que necessitam de uma comunicação de um para muitos, seu prefixo é **ff00::/8** definido na RFC 3306. Os endereços *multicast* são fundamentais para o funcionamento do IPv6, através deles foi possível eliminar o *broadcast* com criação do grupo *multicast-all-nodes* (endereço ff02::1), quando uma máquina ingressa em uma rede não há um endereço IPv6 definido para sua interface, sendo assim ela precisa se comunicar com outros dispositivos buscando mais detalhes sobre como aquele enlace funciona, fica claro a necessidade de um endereço que permita essa comunicação em qualquer enlace, mesmo sem *broadcast* o nó consegue se comunicar através do *multicast-all-nodes* (BRITO, 2013), veja a tabela 1.

Endereço	Escopo	Descrição
ff01::1	Enlace	Todas as interfaces
ff02::1	Enlace	Todos os *hosts* no enlace
ff02::2	Enlace	Todos os roteadores no enlace
ff02::5 e 6	Enlace	Protocolo de roteamento OSPFv3
ff02::9	Enlace	Protocolo de roteamento RIPng
ff02::A	Enlace	Protocolo de roteamento Cisco®/EIGRP
ff02::1:ffxx:xxxx	Enlace	*Solicited-Node*

Tabela 1: Principais endereços multicast
Fonte: Adaptado de BRITO (2013, p. 64)

3.2.4.3 *ANYCAST*

Já utilizado em IPv4 porém não nativamente, o *anycast* é destinado a comunicação um para um de muitos. Isso é possível atribuindo um endereço *unicast* a uma interface e denominando-o como *anycast*, nos roteadores é possível determinar qual o melhor caminho para um nó *anycast* dos vários existentes para

enviar o pacote. Os servidores DNS[1] públicos do Google utilizam endereços *anycast* (BRITO, 2013).

3.2.4.4 ENDEREÇOS ESPECIAIS

Assim como no IPv4, o IPv6 também possui alguns endereços especiais, seguem:

- *Loopback*: Endereço utilizado para direcionar o tráfego do nó para si mesmo. No IPv4 é reservada uma rede inteira para este propósito, a 127.0.0.0/8. No IPv6 foi criado apenas um endereço para essa função, evitando assim o desperdício, o endereço é **::1/128**.
- Endereço para documentação: No IPv6 foi separado o bloco **2001:db8::/32** para ser utilizado em documentações e textos.
- Não especificado: Endereço utilizado para indicar a ausência de endereço IP, corresponde a **::/128**.

[1] DNS: *Domain Name System* – Sistema de nome de domínio, resolve nomes de domínios para endereços IP, por exemplo: google.com para 200.195.190.87.

4 SIMULAÇÃO

Nesta seção serão listados os *softwares* e *hardwares* utilizados para o projeto e os procedimentos realizados para instalação e configuração do servidor Zabbix no Raspberry Pi, os servidores que serão monitorados já estão prontos.

Componentes de *Software*:

- Sistema operacional Raspbian para Raspberry Pi (Base Debian)
- Sistema operacional Windows Server 2008
- Sistema operacional Windows 7
- VirtualBox versão 4.3.6[1]
- Zabbix Server versão 2.2.1
- Zabbix Agent versão 2.2.1
- Zabbix *Frontend Web* versão 2.2.1
- Dependências (serão listadas na instalação)

Componentes de *Hardware*:

- Roteador sem fio Cisco Linksys EA2700
- Computador Desktop (Core2Quad Q9550, 4GB RAM)
- Raspberry Pi Modelo B (com cartão SD de 8GB classe 10)

4.1 CENÁRIO

Criamos um cenário simples para o projeto, serão utilizados máquinas virtuais para virtualizar os servidores no VirtualBox e neles serão instalados os agentes, o Raspberry Pi irá executar o Zabbix Server, Agent e o *Frontend Web*, a topologia de rede é apresentada na figura 9.

[1] VirtualBox: *Software* para emular computadores virtuais.

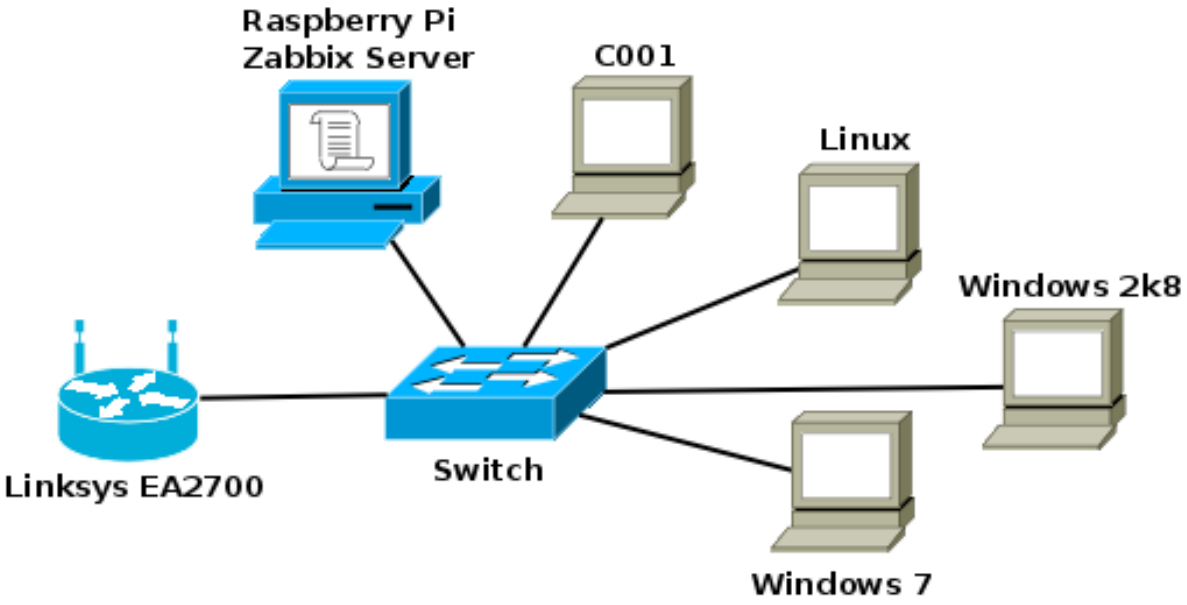

Figura 9: Topologia da Rede
Fonte: Própria (*Software* DIA)

4.2 REQUISITOS MÍNIMOS DE *HARDWARE* E *SOFTWARE*

Segundo OLUPS (Zabbix 1.8 Network Monitoring, 2010) e VLADISHEV (Manual Zabbix versão 2.2, 2013), os requisitos de *software* e *hardware* podem variar dependendo do sistema operacional que suportará o Zabbix, o tamanho da rede e necessidades específicas de cada implementação. Como não há precedentes na configuração no cenário proposto por este projeto do sistema Zabbix e tendo em vista que a arquitetura de *hardware* é diferente, foram realizados testes e determinado que as dependências citadas no tópico Preparando o Raspbian são as necessárias para a abordagem do projeto.

Para a instalação e configuração completa do Zabbix da forma proposta neste projeto, estima-se que é necessário 350MB de espaço em disco disponível, porém no Raspberry Pi é sugerido a utilização de um cartão de memória de no mínimo 4GB.

4.3 RASPBERRY PI

O Raspberry Pi é um computador de baixo custo desenvolvido pela Raspberry Foundation com a premissa de criar um computador barato para incentivar jovens e crianças a aprenderem a programar. Com dimensões pequenas de 85,60 mm x 56 mm x 21 mm e pesando apenas 45 gramas, o pequeno notável apresentado na

figura 10 se tornou muito popular e há diversos projetos com as mais variadas aplicações onde ele é utilizado (Raspberry Foundation, 2006).

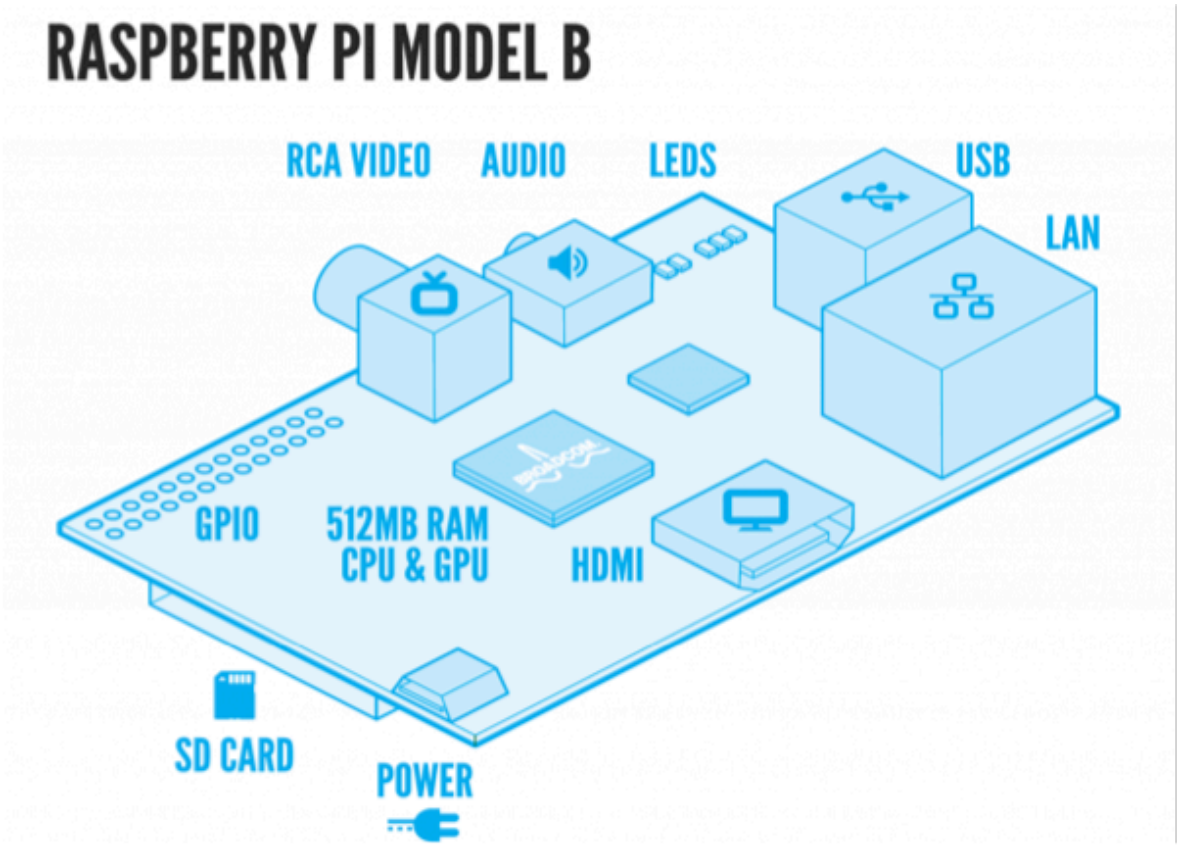

Figura 10: Raspberry Pi – Modelo B
Fonte: http://www.raspberrypi.org/wp-content/uploads/2011/07/RaspiModelB.png

Hardware do Raspberry Pi Modelo B:

- SoC[1] Broadcom BCM2835 (ARM1176JZFS/ARMv6) - 700MHz[2]
- GPU[3] 300MHz - (1Gpixel/s, 1.5Gtexel/s ou 24 GFlops) - Saída RCA[4] e HDMI[5] 1080p
- RAM 512MB
- 2 Portas USB
- 1 *Ethernet* 10/100
- SD *Card* 8GB

[1] SoC: *System-on-a-Chip* – Tipo de chip com vários chips de diferentes funções integrados.
[2] MHz: *Mega Hertz* – Unidade de medida de frequência ou clock do processador.
[3] GPU: *Graphic Processor Unit* – Unidade gráfica de processamento.
[4] RCA: Conector de vídeo para TV analógica.
[5] HDMI: *High-Definiton Multimidia Interface* – Conector de vídeo para TV digital.

4.3.1 SISTEMA OPERACIONAL DO RASPBERRY PI

O sistema operacional do Raspberry Pi é geralmente baseado em GNU/Linux e é compilado para processadores ARM. Existem várias distribuições disponíveis: Raspbian, Arch, Pidora, RISC OS entre outras. Neste projeto será utilizada a distribuição Raspbian.

O Raspbian é uma versão otimizada do Debian para o Raspberry Pi, a versão utilizada neste projeto é a lançada em 20 de Dezembro de 2013.

Não serão abordados aqui os processos de configuração inicial do Raspberry Pi ou do Sistema Raspbian para seu funcionamento básico necessário no projeto. No site www.raspberrypi.org podem ser encontrados tutoriais que ensinam a instalar e configurar o Raspberry Pi.

Os procedimentos necessários para preparar o Raspbian para receber o Zabbix Server são descritos no Anexo I.

4.4 ZABBIX SERVER E *FRONTEND WEB*

Para garantir que todos os recursos necessários estejam disponíveis, a instalação do Zabbix Server será feita a partir do código fonte.

O *frontend web* do Zabbix foi desenvolvido em PHP e oferece uma interface gráfica amigável com geração de gráficos e relatórios (VLADISHEV, 2013).

Os procedimentos para compilar, instalar e configurar o Zabbix e seu *frontend* são abordados no Anexo II.

4.5 ZABBIX AGENT

As checagens realizadas pelos agentes Zabbix podem ser de dois tipos (VLADISHEV, 2013):

- Passivas: É uma solicitação de dados simples, onde o servidor Zabbix pede um dado do *host* monitorado e o agente Zabbix no *host* envia o resultado de volta ao servidor.
- Ativas: É uma solicitação de dados complexa, onde o agente Zabbix deve recuperar uma lista de itens para o processamento independente, esse valores podem ser enviados periodicamente ao servidor Zabbix.

O agente Zabbix utiliza a porta 10050 TCP e UDP para se comunicar com o servidor Zabbix na porta 10051 TCP e UDP.

Os passos para instalação do agente Zabbix nos sistemas operacionais Linux e Windows são encontrados no Anexo III, depois de instalados será necessário adicionar os *hosts* na interface *web* do Zabbix para o monitoramento.

4.5.1 INTERFACE *WEB*

Acesse o *frontend web* do Zabbix através do seu navegador, na URL[1] **http://<IP_do_Servidor_Zabbix>/zabbix**, informe o usuário **admin** e senha **zabbix**.

A interface web do Zabbix é dividida em categorias, são elas (OLUPS, 2010):

- *Monitoring*: Contém a maioria das páginas de vigilância, exibe dados, problemas e gráficos;
- *Inventory*: Inventário do sistema monitorado, pode conter número de série, localização, responsável etc.;
- *Reports*: Exibe relatórios, é possível criar relatórios personalizados;
- *Configuration*: Configuração de tudo relacionado ao monitoramento, parâmetros, notificações, *hosts* etc.;
- *Administration*: Configuração dos itens internos do Zabbix, usuários, permissões, métodos de autenticação etc.;
- *Profile*: Administração do perfil do usuário, tema, linguagem, senha, *auto-login* etc.

4.5.1.1 ADICIONANDO *HOSTS*

Na aba ***Configuration***, clique em ***Hosts***, esta aba fornece acesso fácil para diversas configurações do *host*, inclusive Itens e *Triggers* antes de adicionar *hosts*, é necessário habilitar o monitoramento do próprio servidor Zabbix. O servidor será listado, conforme figura 12, no campo ***Status*** clique em ***Not monitored*** e confirme.

Para adicionar *hosts*, clique em ***Create host***, conforme figura 11.

[1] Uniform Resource Locator: Endereço de um recurso disponível em uma rede.

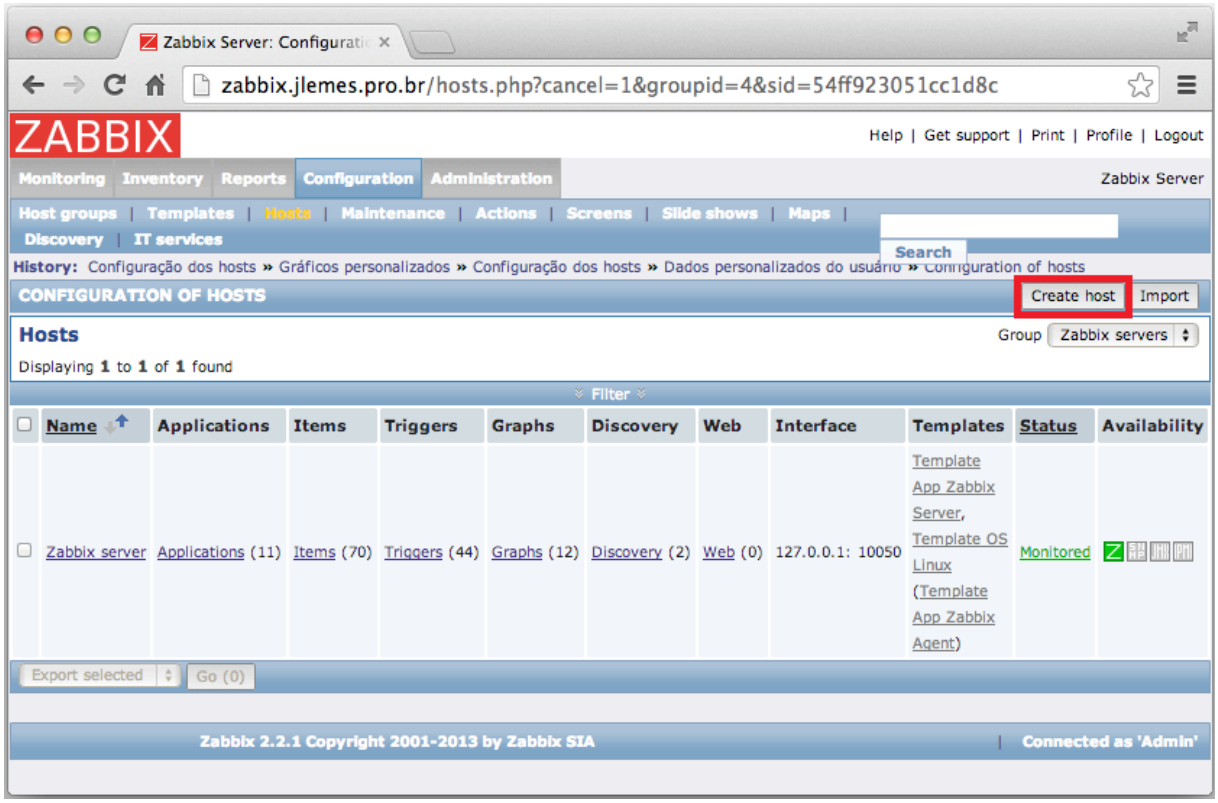

Figura 11: Adicionando *hosts* no Frontend web – Interface Zabbix
Fonte: Interface Zabbix (Própria)

Preencha os campos ***Host name***, ***Visible name*** e ***IP address*** com os dados do *host* a ser monitorado. Em ***Groups*** é possível criar um novo grupo preenchendo o campo ***New group*** ou selecionar um grupo existente, ao final clique em ***Save***, o *host* foi adicionado, porém ainda não há monitoramento nele, isto será visto na próxima seção.

4.5.1.2 MONITORAMENTO

Os *hosts* cadastrados são monitorados através de **itens**, em geral no Zabbix, um item é um conjunto de dados coletados de um *host*, sem itens a coleta de dados não é possível. Os itens devem obrigatoriamente estar atrelados a um *host* (OLUPS, 2010).

Segundo OLUPS (2010), um *host* é uma entidade lógica que agrupa itens. A definição do *host* pode ser livremente adaptada ao meio ambiente ou situação específica. Um host pode ser um *switch*, um servidor físico, uma máquina virtual ou um *web site*.

Os ***Templates*** do Zabbix fornecem um conjunto pronto de itens para monitorar o *host*, isso facilita o monitoramento quando não há necessidades específicas. Para adicionar um *template* acesse ***Configuration > Hosts*** e clique no *host* adicionado na seção anterior, clique na aba *Templates* e no campo ***Link new***

templates digite um termo para buscar um *template*, por exemplo, Windows ou Linux, clique em ***Add***, em seguida em ***Save***.

O Zabbix permite monitorar *hosts* de várias formas, o monitoramento abordado neste projeto é realizado através do agente do Zabbix e *templates* nativos do Zabbix, porém é possível monitorar via SNMP e IPMI[1], além de criar *scripts* e *templates* personalizados para coleta de dados específicos. Para mais detalhes consulte o capítulo 4 – *Quickstart* do manual do Zabbix versão 2.2 em http://www.zabbix.com/documentation.

A coleta de dados ocorre periodicamente, basta esperar alguns minutos para começar a visualizar as informações dos *hosts*.

Para visualizar os dados coletados acesse a aba ***Monitoring*** e submenu ***Latest data***, clique no "+" ao lado do nome do *host*, serão exibidos os itens monitorados pelo *template* e os últimos dados recebidos do agente Zabbix, é possível acessar os gráficos clicando em ***Graph***, ou clicar no submenu ***Graphs*** conforme figura 12.

Acessando o submenu *Graphs* será necessário selecionar o Grupo, o *Host* e o Gráfico que deseja exibir.

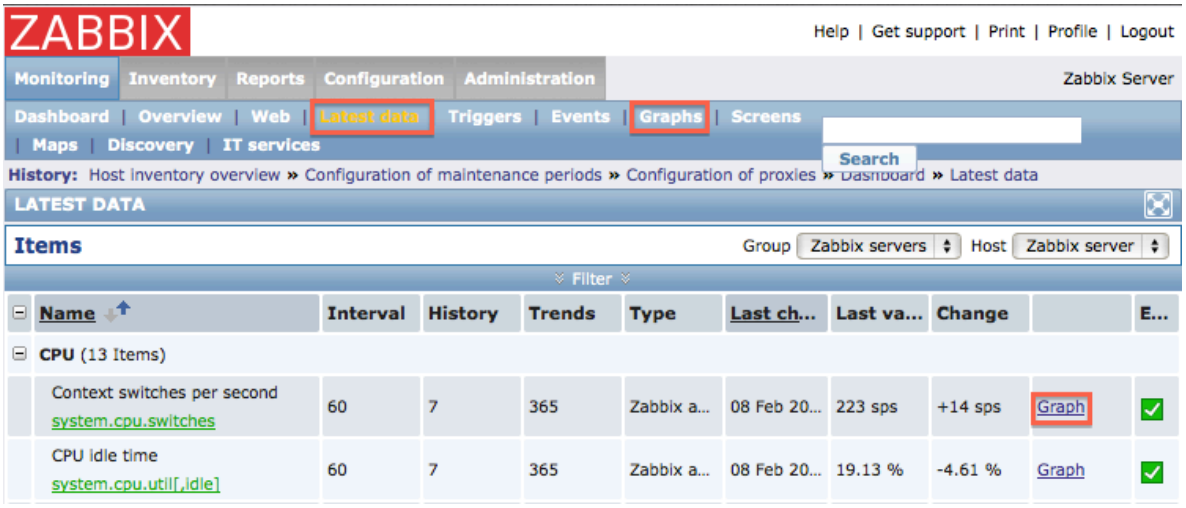

Figura 12: Submenu Latest data no Frontend web – Interface Zabbix
Fonte: Interface Zabbix (Própria)

4.5.1.3 *TRIGGERS*

Um item apenas realiza a coleta dos dados e isto é importante para acompanhar o desempenho do *host*, porém seria impensável ficar acompanhando cada item monitorado para identificar um problema. Desta forma é necessário que o Zabbix alerte sobre determinadas circunstâncias dos itens consideradas de

[1] IPMI: *Intelligent Platform Management Interface*, plataforma de monitoramento promovido pela Intel, Dell, HP e NEC.

importância para o administrador, e para isso é necessário fazer com que o Zabbix reconheça o que é um problema (OLUPS, 2010).

Segundo OLUPS (2010), uma *trigger* ou gatilho, é uma entrada que contém uma expressão que reconhece um problema nos itens monitorados.

Um gatilho é uma expressão que fará uma comparação com os dados coletados de um item e emitirá um alerta caso essa condição seja atendida. É através de gatilhos que são definidos os limiares aceitáveis para determinado item, os gatilhos disparam alertas que são exibidos na *Dashboard*, *Latest data*, os alertas também são utilizados para disparos de notificações via emails, sms, *chat* jabber etc., para isso é necessário configurar uma Ação ou ***Action*** no Zabbix.

As *triggers* não serão abordadas em detalhes neste projeto para não fugir a proposta inicial, porém para criar uma *trigger*, basta acessar o menu ***Configuration > Hosts*** escolher um *host* e clicar em *trigger*, em seguida em ***Create trigger***. Para mais detalhes consulte o capítulo 4 – *Quickstart* do manual do Zabbix versão 2.2 em http://www.zabbix.com/documentation.

4.5.1.4 *ACTIONS*

As ações dizem ao Zabbix o que fazer em determinadas condições. Uma ação tem três componentes principais (OLUPS, 2010):

- Configuração principal: opções gerais de configuração, como assunto e mensagem do email.
- Operações de ação: especifica exatamente o que será feito, como quem enviará a mensagem, qual mensagem será enviada ou até mesmo executar um comando.
- Condições de ação: especifica quando esta ação é utilizada e quando as operações são realizadas. É possível configurar várias condições específicas, como *hosts*, grupos, tempo, *triggers*, severidade etc.

As ações são responsáveis por disparar mensagens ou executar comandos remotos com base nos alertas gerados pelos gatilhos. As ações usam as Mídias configuradas no Zabbix para disparar mensagens via email, sms, jabber, *script* etc. Para acessar o menu *Actions* vá no menu ***Configuration*** e submenu ***Actions***. Para mais detalhes consulte o capítulo 7 *Notifications upon events* do manual do Zabbix versão 2.2 em http://www.zabbix.com/documentation.

4.5.1.5 *MEDIAS TYPES*

Os tipos de mídias ou *media types* são os meios ou canais usados pelo Zabbix para enviar notificações e alertas (VLADISHEV, 2013).

O Zabbix suporta as mídias:

- Email: é possível configurar uma conta de email para enviar as notificações e alertas via SMTP [1] porém o Zabbix não suporta autenticação;
- SMS: mensagens de texto via modem GSM conectado ao servidor Zabbix;
- Jabber: serviço de mensagem instantânea, similar ao Google Talk;
- *Scripts* personalizados: o Zabbix consegue chamar *scripts* desenvolvidos em shell, PHP, Perl entre outros, criados para envio em mídias diferentes.

Para mais detalhes consulte o capítulo 7 *Notifications upon events* do manual do Zabbix versão 2.2 em http://www.zabbix.com/documentation.

[1] SMTP: *Simple Mail Transfer Protocol* – Protocolo Simples de Transferência de Email.

5 RESULTADOS

Nesta seção serão apresentados os resultados alcançados no projeto e as dificuldades enfrentadas.

5.1 APRESENTAÇÃO DOS DADOS COLETADOS

Foram monitorados um total de 9 *hosts*, incluindo o Zabbix Server, o cronograma é apresentado na tabela 2. O acompanhamento ocorreu por 5 dias, num período de doze horas, começando com a coleta apenas no Zabbix Server e aumentando em 2 *hosts* a cada dia.

Data	Período	*Hosts*								
27/jan	9h as 21h	Zabbix Server	-	-	-	-	-	-	-	-
28/jan		Zabbix Server	C001 IPv6	Linux IPv6	-	-	-	-	-	-
29/jan		Zabbix Server	C001 IPv6	C001 IPv4	Linux IPv6	Linux IPv4	-	-	-	-
30/jan		Zabbix Server	C001 IPv6	C001 IPv4	Linux IPv6	Linux IPv4	Win2k8 IPv6	Win2k8 IPv4	-	-
31/jan		Zabbix Server	C001 IPv6	C001 IPv4	Linux IPv6	Linux IPv4	Win2k8 IPv6	Win2k8 IPv4	Win7 IPv6	Win7 IPv4

Tabela 2: Cronograma de monitoramento de ativos
Fonte: Própria

O monitoramento foi realizado em utilizando IPv6 preferencialmente, porém devido a limitação de recursos e para aumentar o número de *hosts* monitorados, também será realizado o monitoramento em IPv4. Como estamos em um momento de transição para o IPv6, ainda necessitamos manter a rede IPv4 funcionando, portanto o monitoramento em ambos os protocolos é essencial e pode ser realizado sem dificuldades.

O desempenho do Raspberry Pi será mensurado através da métrica ***load average***. Segundo ALVES (2009, p. 67), essa métrica é:

> "a média da soma dos processo que estão executando (ou na fila de execução) mais as tarefas que estão aguardando pela finalização de operações de I/O."

O *load average* representa um valor resumido da **carga** do sistema. Quando há gargalos nos subsistemas de entrada e saída é possível perceber um aumento no valor do *load average* (ALVES, 2009).

Não será aprofundado o estudo sobre desempenho de sistemas Linux ou mesmo a métrica do *load average* pois é um assunto complexo e extenso,

facilmente se tornaria um novo projeto de pesquisa, porém sabemos que essa métrica denota a carga real do sistema como um todo, e isso a torna adequada como indicador para acompanhar o desempenho do Raspberry Pi.

De acordo com a tabela 2, foram montados cinco cenários para os testes de performance do sistema, para cada cenário é considerado o período de doze horas de monitoramento. Foi aumentado o número de *hosts* e consequentemente o número de itens monitorados em cada cenário, desta maneira procuramos encontrar uma relação entre números de itens monitorados e o consumo de recursos pelo Zabbix Server através da métrica *cpu load* ou *load average*. Os dados apresentados no gráfico conterão a média do *load average* de 1 minuto.

Durante todo o período de monitoramento foi deixado uma janela do navegador aberta na aba *Dashboard*, esta aba recarrega algumas informações a cada 30 segundos, isto foi feito para simular a utilização da interface do Zabbix, porém em um ambiente real a carga causada pelo uso da interface *web* pode ser alta devido a sessões abertas do Apache e geração de gráficos.

5.1.1 CENÁRIO 1

O cenário 1 foi realizado de acordo com a tabela 3, foram monitorados apenas os recursos do Zabbix Server e um total de 65 itens.

Host	Período	Número de Itens
Zabbix Server	9h as 21h	65

Tabela 3: Número de itens monitorados – Cenário 1
Fonte: Própria

A figura 13 apresenta o gráfico do monitoramento do dia 27 de Janeiro.

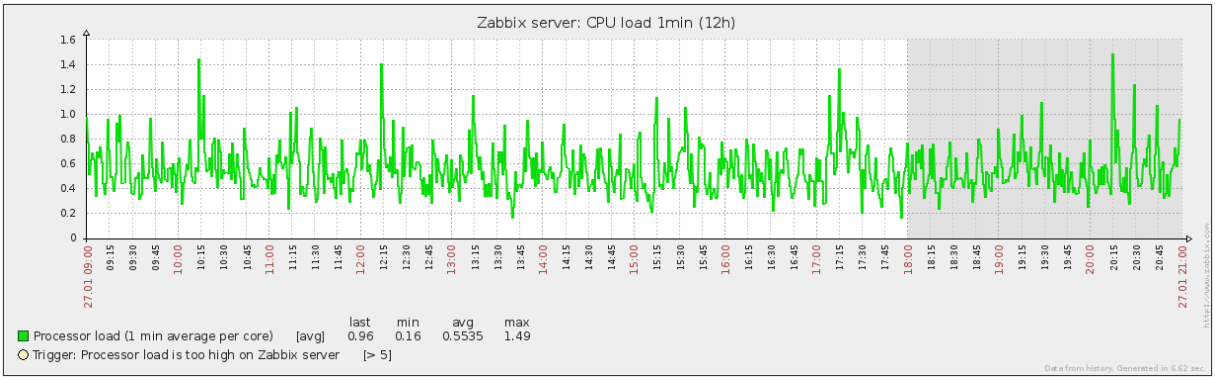

Figura 13: Gráfico de load average 1 minuto, 27 de Janeiro – Interface Zabbix
Fonte: Interface Zabbix (Própria)

O *load average* deste cenário ficou abaixo de um na maior parte do tempo, a média é de 0.55 tendo picos que alcançaram até 1.49.

5.1.2 CENÁRIO 2

O cenário 2 foi realizado de acordo com a tabela 4, foram monitorados os recursos o Zabbix Server e dos *hosts* C001_IPv6 e Linux_IPv6 e um total de 224 itens.

Hosts	Período	Número de Itens	
Zabbix Server	9h as 21h	65	
C001_IPv6		120	
Linux_IPv6		39	
		224	Total de itens

Tabela 4: Número de itens monitorados – Cenário 2
Fonte: Própria

A figura 14 apresenta o gráfico do monitoramento do dia 28 de Janeiro.

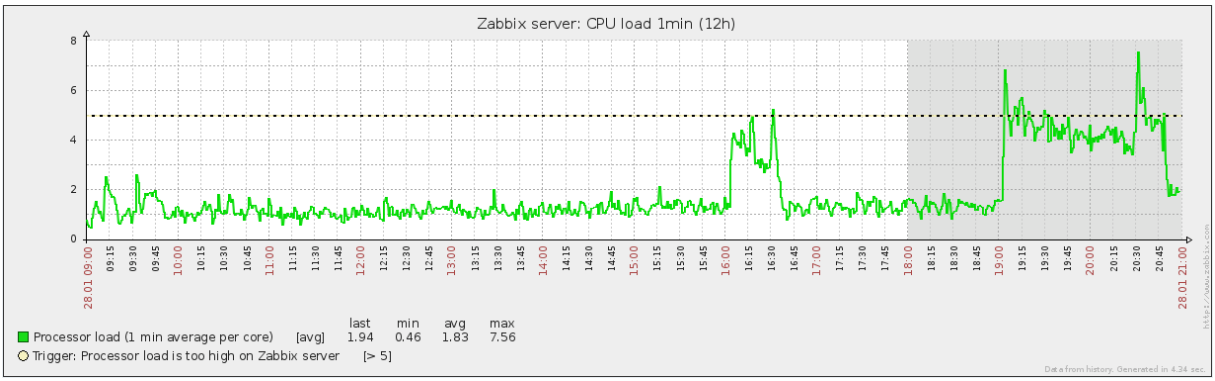

Figura 14: Gráfico de load average 1 minuto, 28 de Janeiro – Interface Zabbix
Fonte: Interface Zabbix (Própria)

Aumentando para três o número de *hosts* monitorados é possível notar um salto da média do *load average* de 0.55 do cenário 1 para 1.83 no cenário 2, os picos aqui chegaram a 7.56.

5.1.3 CENÁRIO 3

O cenário 3 foi realizado de acordo com a tabela 5, foram monitorados os recursos o Zabbix Server e dos *hosts* C001 IPv6 e IPv4 e Linux IPv6 e IPv4 e um total de 383 itens.

Hosts	Período	Número de Itens	
Zabbix Server	9h as 21h	65	
C001_IPv6		120	
C001_IPv6		120	
Linux_IPv6		39	
Linux_IPv4		39	
		383	Total de itens

Tabela 5: Número de itens monitorados – Cenário 3
Fonte: Própria

A figura 15 apresenta o gráfico do monitoramento do dia 29 de Janeiro.

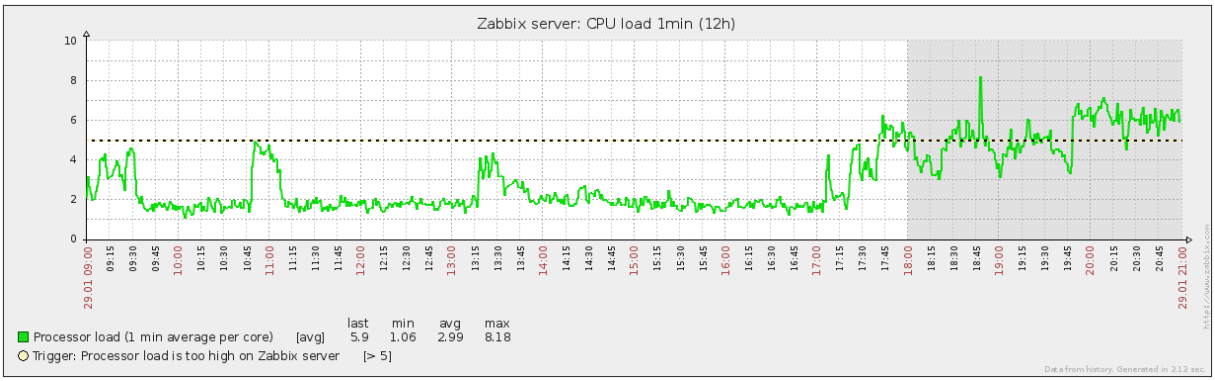

Figura 15: Gráfico de load average 1 minuto, 29 de Janeiro – Interface Zabbix
Fonte: Interface Zabbix (Própria)

Aumentando para 5 o número de *hosts* há novamente um leve salto da métrica para média de 2.99, com picos de 8.18.

5.1.4 CENÁRIO 4

O cenário 4 foi realizado de acordo com a tabela 6, foram monitorados os recursos o Zabbix Server e dos *hosts* C001 IPv6 e IPv4, Linux IPv6 e IPv4, e Win2k8 IPv6 e IPv4, com um total de 495 itens.

Hosts	Período	Número de Itens
Zabbix Server	9h as 21h	65
C001_IPv6		120
C001_IPv6		120

Linux_IPv6		39	
Linux_IPv4		39	
Win2k8_IPv6		56	
Win2k8_IPv4		56	
		495	Total de itens

Tabela 6: Número de itens monitorados – Cenário 4
Fonte: Própria

A figura 16 apresenta o gráfico do monitoramento do dia 30 de Janeiro.

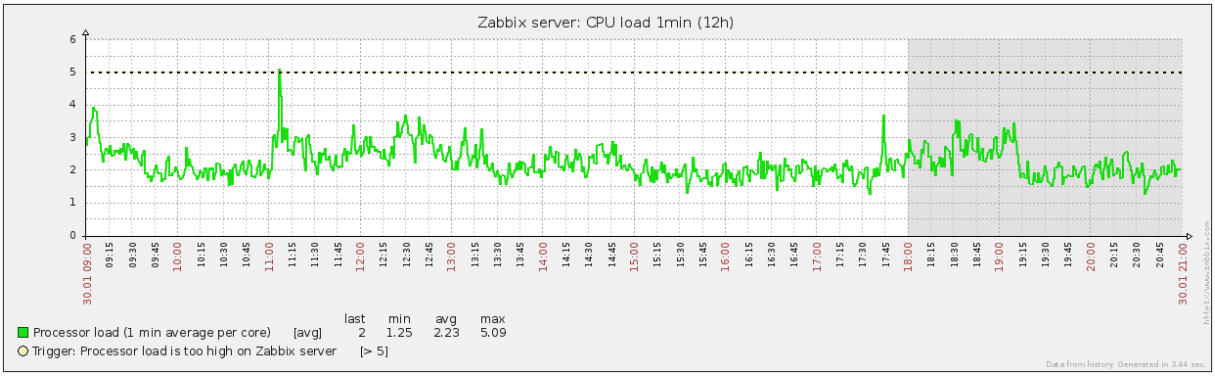

Figura 16: Gráfico de load average 1 minuto, 30 de Janeiro – Interface Zabbix
Fonte: Interface Zabbix (Própria)

No cenário 4 houve uma queda inesperada no *load average*, a média foi de 2.23 com pico de 5.09, além da adição de dois *hosts*, totalizando 7 *hosts* monitorados, não houve mudanças na topologia, configuração ou estrutura do servidor, e o Servidor Zabbix não foi reiniciado. Não foi possível levantar dados para análise ou identificar em mais detalhes o motivo da diminuição da carga.

5.1.5 CENÁRIO 5

O cenário 5 foi realizado de acordo com a tabela 7, foram monitorados os recursos o Zabbix Server e dos *hosts* C001 IPv6 e IPv4, Linux IPv6 e IPv4, Win2k8 IPv6 e IPv4, e Win7 IPv6 e IPv4, totalizando 607 itens.

Hosts	Período	Número de Itens
Zabbix Server	9h as 21h	65
C001_IPv6		120
C001_IPv6		120
Linux_IPv6		39

Linux_IPv4		39	
Win2k8_IPv6		56	
Win2k8_IPv4		56	
Win7_IPv6		56	
Win7_IPv4		56	
		607	Total de itens

Tabela 7: Número de itens monitorados – Cenário 5
Fonte: Própria

A figura 17 apresenta o gráfico do monitoramento do dia 31 de Janeiro.

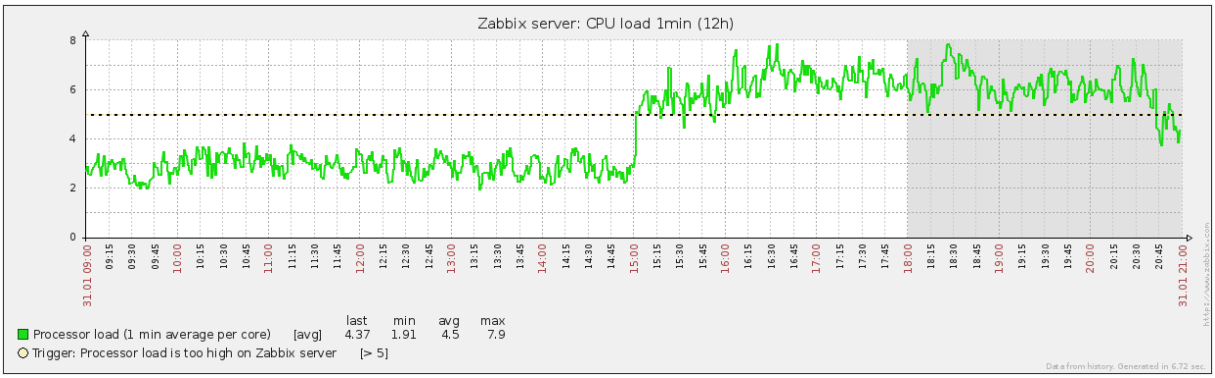

Figura 17: Gráfico de load average 1 minuto, 31 de Janeiro – Interface Zabbix
Fonte: Interface Zabbix (Própria)

No último cenário o *load average* voltou a subir com média de 4.5 e pico de 7.9, porém na primeira metade do monitoramento (das 9h as 15h) a carga se manteve abaixo de 4, e na segunda metade (das 15h as 21h) a carga se manteve acima de 5. O motivo deste comportamento também é desconhecido.

5.2 ANÁLISE DE DESEMPENHO DO RASPBERRY PI

A disposição dos gráficos de monitoramento foi planejada para ser fácil notar um aumento do *load average* conforme aumenta-se o número de itens monitorados de forma simétrica e relacional, porém isso não ocorreu da forma esperada em todos os cenários.

A figura 18 apresenta os gráficos dos cinco dias de forma mais concisa, desta forma fica ainda mais fácil visualizar os gráficos e a mudança decorrente da adição de *hosts*.

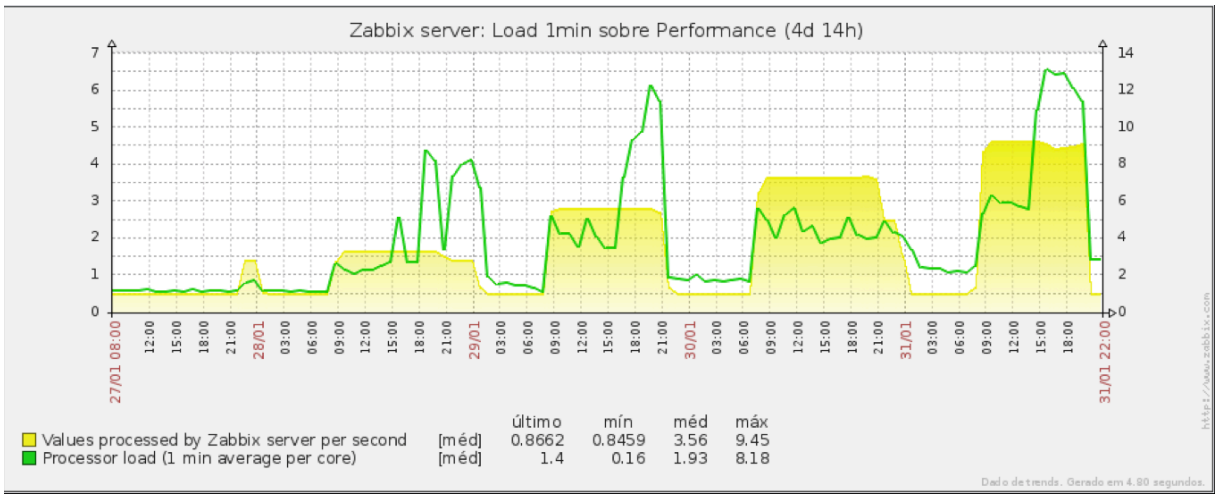

Figura 18: Gráfico dos 5 dias de monitoramento – Interface Zabbix
Fonte: Interface Zabbix (Própria)

Analisando o gráfico é possível notar um aumento da carga nos três primeiros cenários, mas a simetria não ocorre a partir do cenário 4, houve no início uma desconfiança de que essa redução da carga do sistema seja devido ao *cache* gerenciado pelo próprio sistema Linux, porém é necessário uma análise mais detalhada que foge ao foco deste projeto.

Segundo ALVES (2009, p. 89), *"o valor de cached é especialmente usado para armazenar arquivos que foram lidos do disco para a memória..."*. O *cache* é usado para agilizar a interação entre processos e os dados do arquivo, pois é custoso ao sistema acessar o disco rígido. É possível visualizar esta métrica utilizando os programas *top* ou *free*, é necessário interpretar os valores apresentados.

Estabelecer uma relação entre o número de *hosts* monitorados e a carga gerada no servidor não é tão simples quanto a proposta inicial. Ao analisar os gráficos fica visível que a relação entre o número de *hosts* monitorados e o *load average* pode depender de outros fatores.

O acesso a interface *web* do Zabbix é um pouco lento para navegação e geração de gráficos. Ao navegar pela interface houve uma lentidão constante, algumas páginas demoravam mais de cinco segundos para carregar. Na geração dos gráficos a lentidão é relativa, pois dependendo da quantidade de dados extraídos e complexidade dos cálculos, os gráficos podem demorar mais para serem processados, porém mesmo para gráficos simples envolvendo apenas um item e algumas horas de intervalo faziam a imagem demorar até 10 segundos para ser carregada.

O monitoramento com o Raspberry Pi ocorreu por mais de 20 dias no período das 9h as 21h de forma ininterrupta, fora deste intervalo o Raspberry Pi continuava ligado e monitorando o próprio Zabbix Server, em nenhum momento houve

travamento ou necessidade de reiniciar o sistema. No dia 4 de Fevereiro houve uma interrupção no fornecimento de energia, o que acarretou no desligamento do Raspberry Pi, após normalização do serviço o computador voltou a ser ligado para continuar o monitoramento.

De modo geral, considerando as limitações do Raspberry Pi, seu desempenho ao monitorar esta pequena rede foi satisfatório, tendo em vista que é uma solução barata e acessível.

No auge do monitoramento o Zabbix estava acompanhando cerca de 607 itens, tendo em vista que não ocorreram travamentos ou falha na coleta dos dados, é possível aumentar este número em pelo menos cinquenta por cento, chegando próximo de mil itens monitorados. Isso é possível porém será necessário uma análise detalhada e estudo sobre o caso, pois ao utilizar os *templates* prontos de monitoramento, vários itens são adicionados para coleta de dados que podem não ser relevantes para um determinado propósito. Desta forma ainda é possível diminuir o número de itens monitorados se realizado o monitoramento apenas nos itens que são importantes para acompanhar o *host* monitorado.

5.3 DESAFIOS ENFRENTADOS

Ao implementar o Zabbix Server na plataforma Raspberry Pi houveram diversos desafios que foram superados com pesquisa e perseverança, porém apenas um desafio acabou se tornando uma barreira, abaixo há um breve detalhamento.

5.3.1 BANCO DE DADOS SQLITE

Devido as características do Raspberry Pi, é necessário utilizar um cartão de memória SD como disco rígido, porém cartões de memória tem baixa taxa de leitura/escrita e entrada/saída e isso afeta diretamente o desempenho do computador. O Zabbix fornece suporte a vários bancos de dados, inclusive MySQL e SQLite, este último é um banco de dados simples e extremamente leve utilizado em aplicações para dispositivos embarcados como celulares e tablets. No início do projeto foi pensado em utilizar os recursos mais leves disponíveis para suportar o Zabbix, foi realizada a instalação do Zabbix com SQLite, porém não funcionou adequadamente. Eram exibidos erros na interface *web* similares aos exibidos no quadro 1, para mais detalhes do erro veja o Apêndice II.

sem_get(): failed for key 0x7a020458:Permission denied [hosts.php:22 → require_once () → ZBase -> run () → ZBase -> initdb () → DBConnect () → init_sqlite3_access () → sem_get () in /var/www/zabbix/include/db.inc.php:1222]
sem_acquire() expects parameter 1 to be resource, boolean Given [hosts.php:22 → require_once() → ZBase -> run() → ZBase - > initdb() → DBConnect() → lock_sqlite3_access() → sem_acquire() in/var/www/zabbix/include/db.inc.php:1233]
sem_release()expects parameter 1 to be resource , boolean Given [hosts.php:22 → require_once() → ZBase -> run() → ZBase -> initdb() → DBConnect() → unlock_sqlite3_access() → sem_release() in /var/www/zabbix/include/db.inc.php:1244]

Quadro 1: Erros apresentados na interface web ao usar SQLite.

Como a base de dados criada no SQLite armazena o conteúdo do Zabbix Server e também do *frontend web*, após uma breve pesquisa foi levantada a hipótese de que o erro ocorria quando as aplicações, interface *web* e do servidor Zabbix, tentavam acessar a base de dados simultaneamente. Foi também identificado que esse erro pode estar relacionado com os semáforos do PHP, que regulam o acesso a base de dados, porém não foram realizados mais testes.

Devido a esse desafio não superado, foi decidido implementar o MySQL como banco de dados para suportar o Zabbix, mesmo consumindo mais recursos que o SQLite, o Raspberry Pi se saiu bem em sua tarefa. A instalação do MySQL foi padrão sem ajustes de parâmetros para melhoria de performance, portanto esperasse um resultado ainda mais satisfatório se houver um otimização do aplicação de banco de dados para o Raspberry Pi.

5.3.2 NOTIFICAÇÕES

As notificações de alertas não foram abordadas neste projeto, porém foram configuradas, como o Zabbix não suporta o SMTP com autenticação, foi recorrido a um *script* externo para a função de enviar emails, quando surgiam alertas os emails eram disparados, porém não eram enviados. O problema pode ser proveniente apenas de uma configuração incorreta, seria interessante uma análise detalhada.

Também foi testado a notificação pela plataforma Jabber, que funcionou normalmente com o serviço Jabber do site jwchat.org notificando uma conta do Google Talk. O serviço oficial disponível no site jabber.org está temporariamente desabilitado para criação de conta, desta forma não foi possível testá-lo.

5.4 SUGESTÕES DE PROJETOS FUTUROS

A plataforma Raspberry Pi fornece inúmeras possibilidades em diversas áreas ou segmentos, não é difícil imaginar casos onde o pequeno computador seria muito bem aproveitado.

Na área de Redes de Computadores a gama de possibilidades é igualmente ampla. Seria de grande valor um estudo mais detalhado sobre o servidor Zabbix em um cenário completo, com agentes Zabbix, agentes SNMP, *proxies* e diversos itens sendo monitorados, assim como teste de outras funcionalidades como alertas, execução de comandos etc. A funcionalidade de alertas foi testada e funciona, porém não foi abordada por questões de foco e falta de recursos para um cenário propício de teste da ferramenta.

Ao visualizar os gráficos e notar que não há uma simetria no aumento de *hosts* monitorados e carga do sistema pode despertar a curiosidade de alguns. Será possível mensurar o número ideal de *hosts* ou itens monitorados pelo Raspberry Pi? Quantos *hosts* é possível monitorar com o Raspberry sem perda de dados? Quais otimizações podem ser feitas para melhorar o desempenho e vida útil do Raspberry Pi e Servidor Zabbix?

Essas e outras questões como a do SQLite apresentadas na sessão de desafios enfrentados podem ser abordadas em futuros projetos, e quem sabe serem respondidas.

6 CONCLUSÃO

Os tópicos apresentados neste trabalho abordam os conceitos de gerenciamento de redes de computadores, introduzindo a importância do monitoramento de ativos na rede e quais as preocupações e os pontos a serem levados em consideração ao iniciar um projeto para monitorar uma rede. O novo protocolo IPv6 é contrastado com seu antecessor, o protocolo IPv4 para que seja possível fazer analogias entre ambos, auxiliando assim no entendimento e desmistificando o novo protocolo da Internet.

Os conceitos e funcionalidades do Zabbix são apresentados de forma breve, porém são abordados os tópicos principais para o funcionamento básico do servidor. É visto como instalar e configurar o *software* servidor na plataforma Raspberry Pi e seus agentes nos sistemas Windows e Linux.

Foi necessário mudar o banco de dados utilizado para suportar o Zabbix para MySQL devido a erros apresentados na base de dados SQLite, porém o Raspberry Pi se mostrou competente em executar o servidor Zabbix mesmo rodando *softwares* destinados a computadores mais robustos. O desempenho do Raspberry Pi, considerando suas características e limitações, foi satisfatório. Monitorando um total de nove *hosts* e 607 itens, o acesso a interface *web* é um pouco demorado, chegando a demorar vários segundos para carregar abas e processar gráficos, no entanto essa demora não é por causa do Zabbix, e sim pela limitação de recursos do Raspberry Pi.

Os pontos fortes do Zabbix é sua facilidade de instalação e configuração básica, com *templates* prontos e interface intuitiva tem uma curva de aprendizado baixa. Para o Raspberry Pi seu ponto forte é a robustez para executar o monitoramento de tantos *hosts* e permanecer estável, infelizmente não foi possível estender o monitoramento do Zabbix por vários dias para acompanhar seu desempenho.

Os pontos fracos do Raspberry Pi são justamente suas limitações, porém elas estão atreladas ao seu custo, o que os torna uma característica. Os pontos fracos percebidos no Zabbix são relacionados a geração de gráficos, em alguns momentos os gráficos saem com poucos detalhes, arredondando valores, porém isso ocorre para dados coletados a mais de sete dias, portanto pode ter relação com alguma configuração, será necessário uma pesquisa para identificar precisamente.

Apesar das limitações de recursos computacionais que refletem na lentidão da interface *web*, o Raspberry Pi se saiu bem para a tarefa proposta de monitorar uma pequena rede, porém com alguns ajustes e refinamento das configurações do Zabbix, o desempenho do Raspberry Pi pode ser ainda melhor.

REFERÊNCIAS

Zabbix SIA. (2013). *The Enterprise-class Monitoring Solution for Everyone*. Acesso em 30 de Dezembro de 2013, disponível em Zabbix: http://www.zabbix.com / http://www.zabbix.org

VLADISHEV, A. (2013). *Manual Zabbix versão 2.2.* Acesso em 30 de Dez de 2013, disponível em Zabbix: http://www.zabbix.com/documentation/2.2/

ALVES, M. M. (2009). *Linux Performance & Monitoramento.* (S. M. Oliveira, Ed.) Rio de Janeiro, RJ, Brasil: BRASPORT.

BRITO, S. B. (2013). *IPv6 - O novo protocolo da Internet* (Primeira Edição ed.). (R. Prates, Ed.) São Paulo, São Paulo, Brasil: Novatec.

COMER, D. E. (2007). *Redes de Computadores e Internet - Abrange Tansmissão de Dados, Ligação Inter-Redes, Web e Aplicação* (4 Edição ed.). Rio de Janeiro, Rio de Janeiro, Brasil: Bookman.

DUARTE, O. C. (18 de Junho de 2010). *SIMPLE MANAGEMENT NETWORK PROTOCOL*. Acesso em 18 de Novembro de 2013, disponível em Grupo de Teleinformática e Automação - UFRJ: http://www.gta.ufrj.br/grad/10_1/snmp/index.htm

ITU-T. (Aug de 1999). *ITU-T RECOMMENDATION X.701, INTERNATIONAL STANDARD 10040*. Acesso em 16 de Nov de 2013, disponível em www.itu.int: http://www.itu.int/rec/T-REC-X.701-199708-I

KUROSE, J. F., & ROSS, K. W. (2010). *Redes de Computadores e a Internet - Uma abordagem Top-Down* (5 ed.). (A. Wesley, Ed., & O. Translations, Trad.) São Paulo, São Paulo, Brasil: Pearson.

LOPES, R. V. (2009). *Melhores Práticas para a Gerência de Redes de Computadores.* São Paulo, São Paulo, Brasil: Campus.

MAURO, D. R., & SCHMIDT, K. J. (2005). *SNMP ESSENTIAL.* SEBASTOPOL, CA, EUA: O'REILLY.

OLUPS, R. (2010). *Zabbix 1.8 Network Monitoring* (Vol. 1). Birmingham, UK: Packt Publishing Ltd.

SAYDAM, T., & MAGEDANS, T. (Dezembro de 1996). From Networks and Networks Management Into Service and Service Management. *Journal of Networks and System Management* , 345-348.

Raspberry Foundation. (2006). *Raspberry Pi*. Acesso em 24 de Nov de 2013, disponível em Raspberry Pi: www.raspberrypi.org

ANEXO I

PREPARANDO O RASPBIAN

Para o funcionamento do Zabbix e para atender as necessidades deste projeto, é necessário algumas dependências instaladas, são elas:

- Apache2 e bibliotecas de extensão;
    - apache2, libapache2-mod-php5
- PHP5 e bibliotecas de extensão;
    - php5, php5-mysql, php5-gd, php-net-socket
- MySQL e bibliotecas de extensão;
    - mysql-server, mysql-client, libmysqld-dev
- libcurl4-gnutls-dev, snmp e libsnmp-dev, libopenipmi-dev, fping, vim, flex, gpp, libpq5, libpq-dev, libiksemel-dev, libssh2-1-dev, curl

Instale as bibliotecas com o comando abaixo com privilégios de usuário root:

 # apt-get install vim flex gpp fping curl apache2 libapache2-mod-php5 php5 php5-mysql php5-gd php-net-socket mysql-server mysql-client libmysqld-dev openipmi libopenipmi-dev libssh2-1-dev libcurl4-gnutls-dev snmp libsnmp-dev libiksemel-dev libpq5 libpq-dev

Na instalação do MySQL será necessário informar a senha de usuário root para o banco de dados.

No Rasbpbian o protocolo IPv6 não é habilitado por padrão, ou seja, não está carregado este módulo no *Kernel*, para isso é necessário editar o arquivo **/etc/modules** adicionando ao final do arquivo a linha abaixo:

ipv6

O endereço fixo IPv6 é inserido no arquivo **/etc/network/interfaces** com a sintaxe abaixo:

iface eth0 inet6 static
address <endereço IPv6>
netmask 64
gateway <gateway IPv6>
dns <dns IPv6>

Reinicie o Raspberry Pi para que o módulo seja carregado junto com a inicialização do sistema operacional e o endereço IPv6 seja atribuído a interface:

 # reboot

Os processos do *daemon* Zabbix são executados com um usuário sem privilégios, para isso é necessário criar um usuário zabbix no sistema, use o comando abaixo (VLADISHEV, 2013):

 # adduser zabbix

A base de dados é necessária para o Zabbix Server, *Proxy* e *Frontend Web*. Crie as bases de dados MySQL necessárias para utilização pelo Zabbix e garanta o privilégio de acesso para o usuário zabbix com a respectiva senha com o comando abaixo (VLADISHEV, 2013):

 # mysql –u root –p
mysql> create database zabbix character set utf8;
mysql> GRANT ALL PRIVILEGES ON *.* TO zabbix@localhost IDENTIFIED BY 'senha' WITH GRANT OPTION;
mysql> quit

São necessários alguns ajustes no PHP para que o Zabbix consiga utilizá-lo, mude os seguintes parâmetros no arquivo **/etc/php5/apache2/php.ini**:

max_execution_time = 300
max_input_time = 300
post_max_size = 16M
date.timezone = "America/Brasília"

A linha **date.timezone** estará comentada iniciando em " **;** ", apague o caractere.

Reinicie o serviço Apache para que as modificações no arquivo php.ini sejam reconhecidas e aplicadas:

 # service apache2 restart

Para manter organizado os arquivos do Zabbix, crie pastas para armazenar os arquivos de log e o *frontend web*, seguem comandos:

 # mkdir /var/log/zabbix
 # mkdir /var/www/zabbix

Torne o usuário zabbix dono desta pasta com o comando abaixo:

 # chown zabbix.zabbix /var/log/zabbix

ANEXO II

COMPILANDO E INSTALANDO O ZABBIX

No diretório **/home/zabbix** baixe o código fonte do Zabbix, descompacte o arquivo e adicione permissão de execução conforme abaixo:

 # wget http://downloads.sourceforge.net/project/zabbix/ZABBIX%20Latest%20Stable/2.2.1/zabbix-2.2.1.tar.gz

 # tar -zxvf zabbix-2.2.1.tar.gz

 # chmod -R +x zabbix-2.2.1

Para verificar a última versão disponível e encontrar outras versões para download acesse o endereço http://www.zabbix.com/download.php.

Para popular o banco de dados, acesso o diretório **/home/zabbix/zabbix-2.2.1/database/mysql** e execute:

 # cat schema.sql | mysql -u zabbix -p<senha> zabbix
 # cat images.sql | mysql -u zabbix -p<senha> zabbix
 # cat data.sql | mysql -u zabbix -p<senha> zabbix

A pasta **zabbix-2.2.1** contém os arquivos compactados necessários para compilar o código fonte, o comando abaixo compila o código com as configurações descritas:

 # cd zabbix-2.2.1
 # ./configure --prefix=/etc/zabbix --exec-prefix=/bin/zabbix --enable-server --enable-agent --enable-ipv6 --with-mysql --with-net-snmp --with-libcurl --with-ssh2 --with-openipmi --with-jabber

Os parâmetros passados habilitam componentes e funcionalidades no Zabbix, segue:

- prefix=/etc/zabbix: indica o local onde o zabbix deverá instalar seus arquivos de configuração;
- exec-prefix=/bin/zabbix: indica o local onde os binários ficarão;
- enable-server: habilita o servidor Zabbix;
- enable-agent: habilita o agente Zabbix localmente;
- enable-ipv6: habilita o protocolo IPv6;
- with-mysql: habilita o uso do banco de dados MySQL;

- with-net-snmp: habilita o protocolo SNMP;
- with-libcurl: habilita o monitoramento de páginas *web*;
- with-ssh2: habilita o monitoramento via ssh;
- with-openipmi: habilita monitoramento via ipmi;
- with-jabber: habilita a notificação via jabber.

Para mais detalhes do que pode ser configurado para ser compilado e instalado junto ao Zabbix execute o comando abaixo:

 # ./configure --help

Ao final da configuração é exibido uma tela similar a do quadro 2, indicando o que foi configurado para ser instalado.

O comando ./configure verifica se todas as dependências necessárias para suportar os componentes habilitados estão presentes do sistema operacional, caso contrário ele irá parar e indicar na última linha qual dependência está faltando, basta instalar a executar novamente.

Após configurar o código fonte com os componentes desejados para serem executados junto ao Zabbix Server, basta compilar e instalar o *software* com o comando abaixo:

 # make install

```
Configuration:
  Detected OS:                linux-gnueabihf
  Install path:               /etc/zabbix
  Compilation arch:           linux
  Compiler:                   gcc
  Compiler flags:      -g -O2 -I/usr/include/mysql -DBIG_JOINS=1 -fno-strict-aliasing -
g -I/usr/local/include -I/usr/lib/perl/5.14/CORE -I. -I/usr/include -I/usr/include -
I/usr/include
  Enable server:              yes
  Server details:
    With database:            MySQL
    WEB Monitoring:           yes
    Native Jabber:            yes
    SNMP:                     yes
    IPMI:                     yes
    SSH:                      yes
    ODBC:                     no
```

```
  Linker flags:       -rdynamic    -L/usr/lib/arm-linux-gnueabihf    -L/usr/lib/arm-linux-
gnueabihf  -L/usr/lib  -L/usr/lib  -L/usr/lib
  Libraries:          -lm -ldl -lrt  -lresolv    -lmysqlclient    -liksemel   -lcurl   -lnetsnmp
-lcrypto -lssh2 -lOpenIPMI -lOpenIPMIposix
  Enable proxy:               no
  Enable agent:               yes
  Agent details:
    Linker flags:             -rdynamic    -L/usr/lib/arm-linux-gnueabihf
    Libraries:                -lm -ldl -lrt  -lresolv    -lcurl
  Enable Java gateway:        no
  LDAP support:               no
  IPv6 support:               yes
**********************************************************
*          Now run 'make install'                        *
*                                                        *
*          Thank you for using Zabbix!                   *
*            <http://www.zabbix.com>                     *
**********************************************************
```

Quadro 2: Saída do comando ./configure ao finalizar a configuração.

CONFIGURANDO O ZABBIX

Alguns parâmetros devem ser ajustados nos arquivos de configuração do Zabbix, estes arquivos encontram-se no diretório **/etc/zabbix/etc**. Procure e edite os parâmetros dos respectivos arquivos:

zabbix_agentd.conf

LogFile=/var/log/zabbix/zabbix_agentd.log
Server=127.0.0.1, ::1
ListenPort=10050
ListenIP=0.0.0.0, ::
#ServerActive=127.0.0.1
Hostname=informe aqui o nome exato do host (nome do servidor)

zabbix_server.conf

ListenPort=10051
LogFile=/var/log/zabbix/zabbix_server.log
DBHost=localhost
DBName=zabbix
DBUser=zabbix
DBPassword=senha da base de dados
ListenIP=::, 0.0.0.0
FpingLocation=/usr/bin/fping

Fping6Location=/usr/bin/fping

Os *scripts* de inicialização do Zabbix Server e Agent estão juntos no código fonte. Copie os *scripts* para o diretório **/etc/init.d/** e adicione permissão de execução:

 # cp /home/zabbix/zabbix-2.2.1/misc/init.d/debian/* /etc/init.d/
 # chmod +x /etc/init.d/zabbix-server /etc/init.d/zabbix-agent

Como o diretório onde os arquivos binários foram alterados na instalação, é necessário também alterar a localização nos arquivos de inicialização:

zabbix-server e zabbix-agent

DAEMON=/bin/zabbix/sbin/${NAME}

Coloque os serviços para iniciar junto com o sistema operacional:

 # update-rc.d zabbix-agent defaults
 # update-rc.d zabbix-server defaults

Um alerta pode ser exibido ao colocar o *script* de inicialização do Zabbix para iniciar junto ao sistema operacional. Para solucionar o problema consulte o Apêndice I.

FRONTEND WEB

O conteúdo do *frontend* está na pasta descompactada do código fonte, acesse o diretório **/home/zabbix/zabbix-2.2.1/frontends/php** e copie os arquivos com o comando:

#cp -R *.* /var/www/zabbix/

É necessário corrigir as permissões dos arquivos copiados, torne o usuário do Apache *owner* dos arquivos com o comando abaixo:

 # chown -R www-data.www-data /var/www/zabbix

Para finalizar a instalação é necessário acessar o *frontend web* e seguir os seis passos finais, no navegador *web* digite o IP do servidor Zabbix e a pasta no qual ele está instalado e clicar em *Next*, veja na figura 18.

Figura 19: Instalação do Frontend web – Interface Zabbix
Fonte: Interface Zabbix (Própria)

Os próximos passos serão para checar os requisitos e configurar alguns últimos detalhes.

- Passo 2: Checar os pré-requisitos do PHP, todos os parâmetros devem estar OK, caso contrario reveja a configuração do PHP, clique em *Next*;
- Passo 3: Configuração do banco de dados MySQL, basta informar o nome da base, o usuário e a senha, clique em *"Test connection"* em seguida em *Next*;
- Passo 4: Identificar o servidor Zabbix e nomeá-lo, não há necessidade de alterar os campos *Host* e *Port*, clique em *Next*;
- Passo 5: Checar os parâmetros de configuração, se estiver correto clique em *Next*;
- Passo 6: A configuração é finalizada ao clicar em *Finish*.

ANEXO III

INSTALANDO AGENTE ZABBIX NO LINUX

A instalação do Zabbix *Agent* no Debian 7.3 aqui será feita através da compilação do código fonte, porém também é possível instalar o agente através do *apt-get* (gerenciador de pacotes Linux).

Crie o usuário zabbix no sistema:

 # adduser zabbix

Baixe e descompacte o código fonte do Zabbix na pasta do usuário zabbix:

 # wget http://downloads.sourceforge.net/project/zabbix/ZABBIX%20Latest%20Stable/2.2.1/zabbix-2.2.1.tar.gz

 # tar -zxvf zabbix-2.2.1.tar.gz

 # chmod -R +x zabbix-2.2.1

Acesse a pasta do Zabbix e configure a instalação apenas do Agente Zabbix, em seguida faça a instalação:

 # cd zabbix-2.2.1
 # ./configure --prefix=/etc/zabbix --enable-agent --enable-ipv6
 # make install

É necessário alterar o IP do servidor Zabbix no arquivo de configuração do agente Zabbix, localizado em **/etc/zabbix/etc/zabbix_agentd.conf**. Altere os seguintes parâmetros:

Server=<IPv4_do_Servidor_Zabbix>,<IPv6_do_Servidor_Zabbix>

Copie o *script* de inicialização do agente Zabbix do código fonte:

 # cp /home/zabbix/zabbix-2.2.1/misc/init.d/debian/zabbix-agent /etc/init.d/
 # chmod +x /etc/init.d/zabbix-agent

Coloque o agente Zabbix para iniciar junto com o sistema operacional:

 # update-rc.d zabbix-agent defaults

Um alerta pode ser exibido ao colocar o *script* de inicialização do Zabbix para iniciar junto ao sistema operacional. Para solucionar o problema consulte o Apêndice I.

INSTALANDO AGENTE ZABBIX NO WINDOWS

O executável do agente Zabbix do Windows pode ser encontrado também no código fonte, no diretório **zabbix-2.2.1/bin**, ou baixá-lo direto do site no *link* http://www.zabbix.com/download.php.

Após baixar e descompactar o arquivo, acesse o diretório \bin e copie a pasta win32 ou win64 de acordo com a arquitetura do sistema monitorado, crie uma nova pasta Zabbix na raiz do C:\ e cole a pasta copiada. Dentro do diretório \conf copie o arquivo zabbix_agentd.win.conf para dentro da pasta C:\Zabbix.

Abra o arquivo zabbix_agentd.win.conf com o WordPad para que seja mantido a formatação e edite os parâmetros conforme abaixo:

LogFile=c:\Zabbix\zabbix_agentd.log
Server=<IPv4_do_Servidor_Zabbix>,<IPv6_do_Servidor_Zabbix>
Hostname=<Hostname_do_computador>

Para instalar o agente abra um *prompt* de comando como Administrador e execute:

C:\Zabbix\win32\zabbix_agentd.exe -i -c C:\Zabbix\zabbix_agentd.win.conf

O agente Zabbix já está instalado porém não está sendo executado. Acesse Menu Iniciar > Painel de Controle > Ferramentas administrativas > Serviços, procure por Zabbix Agent e inicie o serviço. A próxima vez que o sistema for reiniciado o agente será executado automaticamente.

APÊNDICE I

Linux Standard Base (LSB)

Ao tentar adicionar os serviços do Zabbix para iniciar automaticamente com o sistema, a seguinte mensagem de alerta pode surgir:

```
insserv: warning: script 'zabbix-agent' missing LSB tags and overrides
insserv: warning: script 'zabbix-server' missing LSB tags and overrides
insserv: warning: script 'mathkernel' missing LSB tags and overrides
```

O alerta é apresentado pois os scripts citados não possuem o cabeçalho LSB. Para corrigir basta adicionar o cabeçalho apresentado abaixo logo após a linha **!/bin/sh** nos scripts.

```
### BEGIN INIT INFO
# Provides:          (nome do serviço)
# Required-Start:    $remote_fs $syslog
# Required-Stop:     $remote_fs $syslog
# Should-Start:      $portmap
# Should-Stop:       $portmap
# X-Start-Before:    nis
# X-Stop-After:      nis
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# X-Interactive:     true
# Short-Description: (Descrição breve)
# Description:       This file should be used to construct scripts to be
#                    placed in /etc/init.d.
### END INIT INFO
```

O *Linux Standard Base* (LSB) é um projeto baseado nas especificações POSIX, UNIX e outros padrões abertos, sob a estrutura da *The Free Standars Group* para padronizar a estrutura interna de sistemas operacionais Linux.

Para mais detalhes, leia o arquivo README localizado na pasta **/etc/init.d/**, ou acesse o *link* http://www.linuxfoundation.org/collaborate/workgroups/lsb.

APÊNDICE II

Erros apresentados na interface *web* do Zabbix Server ao utilizar o banco de dados SQLite na plataforma Raspberry Pi.

sem_get(): failed for key 0x7a020458:Permission denied [hosts.php:22 → require_once () → ZBase -> run () → ZBase -> initdb () → DBConnect () → init_sqlite3_access () → sem_get () in /var/www/zabbix/include/db.inc.php:1222]

sem_acquire() expects parameter 1 to be resource, boolean Given [hosts.php:22 → require_once() → ZBase -> run() → ZBase - > initdb() → DBConnect() → lock_sqlite3_access() → sem_acquire() in/var/www/zabbix/include/db.inc.php:1233]

sem_release() expects parameter 1 to be resource , boolean Given [hosts.php:22 → require_once() → ZBase -> run() → ZBase -> initdb() → DBConnect() → unlock_sqlite3_access() → sem_release() in /var/www/zabbix/include/db.inc.php:1244]

SQLite3:: exec():database is locked [dashboard.php:21 → require_once() → ZBase - > run() → ZBase - > AuthenticateUser() → CWebUser :: checkAuthentication() → CAPIObject - > checkAuthentication() → CAPIObject - > __call() → czbxrpc::call() → czbxrpc callAPI::():: → czbxrpc transactionEnd() → DBend() → DBcommit() → DBexecute() → SQLite3 - > exec() in/var/www/zabbix/include/db.inc.php:570]

Error in query [COMMIT] Error code [5] Message [database is locked]

SQLite3::exec():can not start the transaction Within a transaction [dashboard.php:282 make_sysmap_menu → () → make_sysmap_submenu() → CAPIObject -> get() → CAPIObject - > __call() → czbxrpc::call() → czbxrpc callAPI::():: → czbxrpc transactionBegin() → dbstart() → DBexecute() → SQLite3 - > exec() in/var/www/zabbix/include/db.inc.php:570]

Error in query [BEGIN] Error code [1] Message [can not start a transaction Within a transaction]

SQLite3::exec():database is locked [dashboard.php:491 → require_once() → add_user_history() → DBexecute() → SQLite3 - > exec() in/var/www/zabbix/include/db.inc.php:570]

Error in query [UPDATE SET user_history title1=title2, url1=url2, title2=TITLE3, url2=url3, TITLE3=TITLE4, url3=url4, TITLE4=TITLE5, url4=url5, TITLE5='Dashboard', url5='dashboard.php' WHERE userid=1] Error code [5] Message [database is locked]

SQLite3::exec():database is locked [dashboard.php:491 → require_once() → Cprofile::flush() → DBend() → DBcommit() → DBexecute() → SQLite3 - > exec() in/var/www/zabbix/include/db.inc.php:570]

Error in query [COMMIT] Error code [5] Message [database is locked]