Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Implementação da integração do Barramento de Serviços da UnB com ferramentas de monitoramento

Felipe Evangelista dos Santos

Dissertação apresentada como requisito parcial para conclusão do
Mestrado Profissional em Computação Aplicada

Orientador
Prof. Dr. Sérgio Antônio Andrade de Freitas

Brasília
2020

Ficha catalográfica elaborada automaticamente,
com os dados fornecidos pelo(a) autor(a)

ESA237i

Evangelista dos Santos, Felipe
Implementação da integração do Barramento de Serviços da UnB com ferramentas de monitoramento. / Felipe Evangelista dos Santos; orientador Sérgio Antônio Andrade de Freitas. - Brasília, 2020.
116 p.

Dissertação (Mestrado - Mestrado Profissional em Computação Aplicada) -- Universidade de Brasília, 2020.

1. Monitoramento de Sistemas Distribuídos. 2. Service Oriented Architecture. 3. Web services. 4. Enterprise Service Bus. I. Andrade de Freitas, Sérgio Antônio, orient. II. Título.

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Implementação da integração do Barramento de Serviços da UnB com ferramentas de monitoramento

Felipe Evangelista dos Santos

Dissertação apresentada como requisito parcial para conclusão do
Mestrado Profissional em Computação Aplicada

Prof. Dr. Sérgio Antônio Andrade de Freitas (Orientador)
CIC/UnB

Prof. Dr.a Edna Dias Canedo
CIC/UnB

Prof. Dr. Gibeon Soares De Aquino Junior
DIM/UFRN

Prof.a Dr.a Aletéia Patrícia Favacho de Araújo
Coordenadora do Programa de Pós-graduação em Computação Aplicada

Brasília, 11 de fevereiro de 2020

Dedicatória

Dedico este trabalho aos meus pais, Rosana e Japir que me ensinaram os valores que guiam minha vida. Aos meus irmãos, Viviane e Thiago que me incentivaram nessa jornada. A minha esposa Telma que sempre ofereceu força, apoio e motivação. Aos meus filhos Matheus, Teylor e Luana por me alegrarem nos momentos difíceis, sem essas pessoas não iria conseguir chegar até aqui.

Agradecimentos

Ao meu orientador, Prof. Dr. Sérgio Antônio Andrade de Freitas, agradeço o apoio e dedicação.

À Prof.ª Dr.ª Edna Dias Canedo pelo auxílio no ingresso ao programa de mestrado e ao Prof. Dr. Gibeon Aquino pelas considerações no trabalho.

Aos meus amigos e colegas, da vida, do trabalho e do mestrado, especialmente Mateus Manuel, Pedro Henrque, Renan Filgueiras, Eduardo Henrique, Everton Agilar, Consuelo Galo, Luiz Martins, Carlos Vinícius, Riane Torres, Rodrigo Fonseca e Alysson Ribeiro que auxiliaram diretamente na execução deste trabalho.

Resumo

A implementação de serviços e microsserviços para aplicações de sistemas distribuídos com a utilização de uma Arquitetura Orientada a Serviços (SOA) permite utilizar padrões de desenvolvimento, facilitar a manutenção, flexibilizar o desenvolvimento de serviços e permitir a interoperabilidade de serviços e sistemas. O Centro de Informática (CPD) da Universidade de Brasília (UnB) trabalha com vários processos de automação de *softwares*, desde a manutenção de sistemas legados, passando pelo desenvolvimento de novas aplicações até a implantação de *softwares* adquiridos, com várias frentes tecnológicas relacionadas à sistemas. Acompanhar e monitorar o funcionamento de serviços, microsserviços e sistemas é imprescindível. Este trabalho tem caráter exploratório e busca investigar sobre soluções e ferramentas para implementação e implantação de monitoramento de serviços e sistemas distribuídos da Universidade de Brasília (UnB), por meio de um mapeamento sistemático. Com embasamento teórico obteve-se um modelo que foi implementado como módulo de monitoramento do barramento de serviços da Universidade de Brasília (UnB). Neste trabalho foram executadas simulações na solução que permitiu analisar a integração do barramento de serviços com a ferramenta de monitoramento através da solução proposta.

Palavras-chave: Monitoramento de Sistemas Distribuídos, *Service-Oriented Architecture*, *Web services*, *Enterprise Service Bus*.

Abstract

The Implementation services and microservices for distributed system applications using a Service Oriented Architecture (SOA) allows to use development standards facilitate maintenance flexibly develop services and enable interoperability of services and systems. Computer Center (CPD) of the University of Brasilia (UnB) works with several softwares automation processes, from the maintenance of legacy systems, through the development of new applications to the deployment of purchased softwares, with several systems related technological fronts. Mark and monitor the functioning of services, microservices and systems is essential. This work is exploratory and seeks to investigate solutions and tools for the implementation of monitoring of distributed services and systems of the University of Brasilia (UnB), through systematic mapping. With a theoretical basis, a model was obtained, which was implemented as a service bus monitoring module at the University of Brasilia (UnB). In this work, simulations were performed on the solution that allowed to analyze the integration of the service bus with the monitoring tool through the proposed solution.

Keywords: Distributed Systems Monitoring, Service-Oriented Architecture, Web services, Enterprise Service Bus.

Sumário

1 Introdução **1**

1.1 Definição do Problema 1
1.2 Motivação 2
1.3 Objetivos 2
1.4 Metodologia 3
1.5 Resultados Esperados 3
1.6 Estrutura do Trabalho 3

2 Fundamentação Teórica **5**

2.1 Monitoramento 5
2.1.1 Monitoramento de Sistemas Distribuídos 5
2.1.2 Protocolos de Monitoramento 6
2.1.3 Ferramentas de Monitoramento 9
2.1.4 Agente de Monitoramento 14
2.1.5 Instrumentação 16
2.1.6 Métricas de Monitoramento 16
2.2 Arquitetura Orientada a Serviços - SOA 17
2.2.1 *Enterprise Service Bus* - ESB 19
2.2.2 Erlangms 21
2.2.3 *Representational State Transfer* - REST 23
2.2.4 *JavaScript Object Notation* - JSON 23
2.3 Trabalhos Relacionados 24
2.4 Síntese do Capítulo 25

3 Mapeamento Sistemático **26**

3.1 Questões de Pesquisa 26
3.2 Estratégia de busca 26
3.3 Critérios de Inclusão e Exclusão 28
3.4 Extração dos Dados 29

3.5 Resultados 41
3.6 Síntese do Capítulo 45

4 Monitoramento do Barramento de Serviços pelo Ems-Monitor 46
4.1 Arquitetura do Ems-Monitor 46
4.2 Monitoramento dos Recursos do Barramento de Serviços 49
4.2.1 Instrumentação de Código 55
4.2.2 Execução do Ems-Monitor 59
4.2.3 Execução do Agente de Monitoramento 60
4.3 Funcionamento do Protocolo SNMP 62
4.4 Definição das Métricas de Monitoramento 64
4.5 Síntese do Capítulo 70

5 Análise do Ems-Monitor 71
5.1 Análise das Aplicações 71
5.2 Resultados 73
5.3 Síntese do Capítulo 77

6 Conclusão 78
6.1 Contribuições 79
6.2 Trabalhos Futuros 79

Referências 80

Apêndice 88

A Código do Módulo do Serviço de Monitoramento 89

B Aplicação do Código para a realização da coleta das informações para o Monitoramento 93

Anexo 94

I Arquivo MIB sem os OIDs 95

II Arquivo MIB com os OIDs 96

III Arquivo MIB convertido pelo plugin SNMPTTCONVERTMIB 100

IV Registro dos hosts e serviços no arquivo de configuração do Nagios® 101

Lista de Figuras

2.1 Gerente e Agente comunicando por meio do protocolo SNMP[1] 8
2.2 Representação de propagação epidêmica [2] 9
2.3 Monitoramento de componentes[3] 10
2.4 Monitoramento de *clusters* [4] 11
2.5 Monitoramento de rede [5] 12
2.6 Gráfico de monitoramento do RRDTool [6] 13
2.7 *Dashboard* de monitoramento [7]. 14
2.8 Esquema de comunicação entre os agentes e as estações de gerenciamento [8]. 15
2.9 Arquitetura do Exometer [9]. 15
2.10 Arquitetura do trabalho Erlangms descrita por Agilar [10] 22

3.1 Etapas do mapeamento sistemático [11] 27
3.2 Resultado da Fase 3 classificação (CI e CE) em porcentagem. 31

4.1 Arquitetura conceitual do Ems-Monitor. 48
4.2 Serviço de consulta da utilização dos recursos de memória do Erlangms. . . 51
4.3 Interface gráfica do *Observer*. 52
4.4 Fluxo de monitoramento pelo Ems-monitor. 54
4.5 Exemplo de identificadores das métricas no Erlangms. 55
4.6 Serviço que recupera informações históricas dos contadores do Erlangms. . 56
4.7 Funções dos contadores para instrumentação no Erlangms. 57
4.8 Os métodos do Exometer utilizados no Erlangms. 58
4.9 *Design* da integração das ferramentas para monitoramento. 59
4.10 Estrutura base, arquivo MIB do Exometer. 61
4.11 Configuração do servidor de destino, para envio de *TRAPs*. 62
4.12 Arquivo de configuração do agente SNMP. 63
4.13 Arquivo de configuração do gerente SNMP. 63
4.14 Informações do *TRAP*. 64
4.15 Mapeamento dos recursos do barramento de serviços. 66

4.16 Identificação dos recursos do barramento de serviços monitorados. 69

5.1 *Dashboard* do Nagios® com as métricas do registro de *Log*. 75
5.2 Análise executada em tempo real pela ferramenta Observer. 76
5.3 Análise executada em tempo real pela ferramenta Observer com o Ems-Monitor. 77

Lista de Tabelas

3.1 Trabalhos retornados por base de dados digitais 29
3.2 Trabalhos excluídos, incluídos, rejeitados e selecionados. 33
3.3 Trabalhos selecionados. 33
3.4 Trabalhos selecionados para leitura completa 38

Lista de Abreviaturas e Siglas

ACM ACM Digital Library.

API Application Programming Interface.

CPD Centro de Informática.

CPU Unidade central de processamento.

CSV Comma-separated values.

DOI Digital Object Identifier.

ESB Enterprise Service Bus.

GOSIP Government OSI Profiler.

HTTP Hypertext Transfer Protocol.

IEEE IEEE Xplore.

IP Internet Protocol address.

JSON JavaScript Object Notation.

MIBs Management Information Bases.

MPCA Mestrado Profissional em Computação Aplicada.

OID Object Identifier.

OSI Open System Interconnection.

REST Representational State Transferl.

RFC Request for Comments.

RRDTool Round Robin Database Tool.

Scps Scopus.

SD ScienceDirect.

SMI Structure of Management Information.

SMS Short Message Service.

SNMP Simple Network Management Protocol.

SNMPTT SNMP Trap Translator.

SOA Service-Oriented Architecture.

SOAP Simple Object Access Protocol.

StArt State of the Art through Systematic Review.

TAG The W3C Technical Architecture Group.

TI Tecnologia da informação.

UnB Universidade de Brasília.

URL Uniform Resource Locator.

W3C World Wide Web Consortium.

WoS Web of Science.

WSDL Web Services Description Language.

WWW World Wide Web.

XML Extensible Markup Language.

Capítulo 1

Introdução

A implementação de sistemas, seja ela pela criação, migração ou modernização tecnológica é um plano utilizado pelas organizações para gerenciar os *softwares* que automatizam seus processos [10]. Quando se trata de um processo de modernização, espera-se reduzir os custos financeiros de mão de obra, com a manutenção dos sistemas em execução que possuem complexidade alta, defasagem tecnológica e a falta de documentação de apoio, com a criação de novos sistemas e serviços. Algumas organizações procuram manter ativos os *softwares* com essas características, com o funcionamento normal e a inclusão de módulos e serviços que possibilitam a integração e comunicação entre serviços e sistemas [10].

O Centro de Informática da Universidade de Brasília, órgão responsável por desenvolver, gerenciar e manter os sistemas novos e legados, trabalha na implantação de uma arquitetura *Service-Oriented Architecture*, e a utiliza para implementação dos serviços que realizam a comunicação por meio de um barramento de serviços. No entanto, a flexibilidade que a arquitetura SOA proporciona para modernização dos sistemas novos e legados da UnB, dificulta o gerenciamento e o monitoramento dos serviços que são implementados [10]. A razão é a quantidade de serviços criados e o modelo negocial de cada sistema, sendo que em alguns casos há situações em que vários serviços e microsserviços são necessários e determinantes para um único fluxo de negócio do sistema.

1.1 Definição do Problema

Com o aumento da implementação e disponibilização de serviços na UnB por meio do barramento de serviços Erlangms[10], foi identificada a necessidade de um efetivo monitoramento dos serviços, por meio da coleta de dados ou informações extraídas das requisições durante a execução dos sistemas e serviços. Para gerenciar o monitoramento são necessárias ferramentas que auxiliem essa atividade, o CPD utiliza ferramentas de mercado com

esse propósito, essas ferramentas ou plataformas são utilizadas para acompanhamento e monitoramento da infraestrutura de redes da UnB.

Os sistemas e serviços mantidos pelo CPD, não são monitoradas nem fornecem informações relevantes como por exemplo: a situação, a execução, o funcionamento e disponibilidade dos sistemas e serviços, pois não há integração entre sistemas e serviços de forma apropriada com as ferramentas de monitoramento mantidas pelo CPD, o que implica em um hiato no acompanhamento e monitoramento nos sistemas e serviços, também não dispõem de acompanhamento específico voltado para o monitoramento do ambiente em que as aplicações e serviços estão hospedados. Percebe-se que o gerenciamento de importantes funcionalidades são falhos ou ausentes e precisam ser melhorados.

1.2 Motivação

A necessidade de modernizar os *softwares* da UnB é eminente. A implementação de serviços e microsserviços para contemplar os sistemas novos, legados e adquiridos recebem prioridades para criação. A integração dessas aplicações é construída com diversos tipos de serviços, alguns desses serviços funcionam a partir das informações de outros serviços, a indisponibilidade de um serviço pode impactar em outros serviços, ou até mesmo no sistema como um todo, dessa maneira acompanhar seu funcionamento é essencial.

É necessária, a criação de meios tecnológicos para realizar o monitoramento via ferramentas de monitoramento utilizadas pelo CPD, com o barramento de serviços por meio de integração que possibilite a comunicação entre as ferramentas, contribuindo com o acompanhamento abrangente dos serviços e *softwares* da UnB. Pretende-se com a integração o monitorar serviços. A partir da implantação ou implementação, especificar processos e métodos que possibilite a extração das informações coletadas.

1.3 Objetivos

O objetivo deste trabalho é propor um modelo para o monitoramento dos sistemas e serviços da UnB com uma solução tecnológica que permita realizar a comunicação por meio de integração com ferramentas de monitoramento utilizadas, gerenciadas e mantidas pelo Centro de Informática (CPD).

Para alcançar este objetivo, foram determinados os seguintes objetivos específicos:

- Realizar uma revisão da literatura para identificar trabalhos sobre monitoramento de sistemas distribuídos;

- Propor uma arquitetura de integração para monitoramento dos sistemas e serviços da UnB;
- Implantar o monitoramento dos sistemas e serviços da UnB;
- Avaliar e analisar os resultados da solução que foi proposta para a realização da integração das ferramentas de monitoramento dos serviços da UnB.

1.4 Metodologia

A metodologia de pesquisa utilizada neste trabalho será a revisão da literatura em conjunto com um estudo de caso exploratório. A revisão da literatura objetiva reconhecer a unidade e a diversidade interpretativa existente no eixo temático em que se insere o problema em estudo, para ampliar, ramificar a análise interpretativa, bem como para compor as abstrações e sínteses que qualquer pesquisa requer colaborando para a coerência nas argumentações do pesquisador [12]. Ao final da revisão da literatura, espera-se obter um conjunto de indicadores que serão utilizados na aplicação do modelo proposto.

O estudo de caso exploratório tem como finalidade proporcionar mais informações sobre o assunto a ser investigado, possibilitando sua definição e seu delineamento, isto é, facilitar a delimitação do tema da pesquisa; orientar a fixação dos objetivos e a formulação das hipóteses ou descobrir um novo tipo de enfoque para o assunto [13].

1.5 Resultados Esperados

Implantar a integração entre as ferramentas de monitoramento utilizada pelo CPD e o barramento de serviços denominado Erlangms[10]. Espera-se também que este trabalho possa auxiliar na realização do monitoramento de sistemas e serviços, e possibilite monitorar a execução e disponibilidade de sistemas e serviços da UnB.

1.6 Estrutura do Trabalho

O presente trabalho de pesquisa apresenta uma solução para o monitoramento dos serviços implementados, com a implementação e implantação de um módulo de monitoramento de serviços. Com essa solução espera-se proporcionar o monitoramento dos serviços por meio de ferramentas de monitoramento com a utilização de um protocolo de comunicação capaz de permitir a integração entre o barramento de serviços e ferramentas de monitoramento, para coletar informações e transmiti-las, esses dados sobre monitora-

mento podem indicar o funcionamento normal, condições de alertas ou situações de falhas nos serviços em execução monitorados.

Este trabalho está assim estruturado: no capítulo 2 é descrita a fundamentação teórica, no capítulo 3 é apresentado o mapeamento sistemático, utilizado na realização de pesquisa para identificação dos principais trabalhos sobre o tema, no capítulo 4 são apresentadas a implementação e a implantação do monitoramento de serviços, no capítulo 5 são apresentados a avaliação e os resultados da implementação do monitoramento do barramento de serviços e, finalmente no capítulo 6 são apresentadas as conclusões, as contribuições e a possibilidade da realização de trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Este capítulo apresenta uma breve revisão conceitual da essência do que venha ser, monitoramento e arquitetura orientada à serviços. Neste seguimento, temos as descrições dos conceitos de monitoramento de sistemas distribuídos, protocolos, ferramentas de monitoramento, agente de monitoramento, instrumentação e métricas, também sobre conceitos de ESB, Erlangms, REST e JSON. As definições de conceitos e tecnologias estudadas são de grande ajuda tanto para o entendimento do problema quanto para execução e implementação do trabalho. Por fim são apresentados trabalhos relacionados a este tema que colaboram e orientam a pesquisa e fundamentação deste trabalho.

2.1 Monitoramento

O monitoramento é forma utilizada para acompanhar ou observar o andamento de um fluxo ou processo, segundo Martino Jannuzzi [14], na área de tecnologia da informação é responsável por verificar e coletar informações sobre funcionamento de ativos de rede ou serviços, com propósito de apresentar informações mais resumidas e favoráveis em painéis ou sistemas para a realização de monitoramento. De acordo com Hollingsworth [15] o monitoramento é necessário para uma série de fins, incluindo a verificação de status, solução de problemas, ajustes de desempenho e depuração.

2.1.1 Monitoramento de Sistemas Distribuídos

Em tecnologia da informação de monitorar ativos de rede e serviços, é acompanhar, seja pelo mau funcionamento de um ativo de rede, serviço ou por uma indisponibilidade(queda de energia ou internet). Como mencionado anteriormente o ato de acompanhar o funcionamento desses recursos propõe aos gestores desses processos encontrar uma forma

mais visível e gerenciável para solucionar problemas, alertar sobre possíveis falhas ou até determinar uma situação para o retorno automático desses serviços.

O monitoramento implantado no CPD para o acompanhamento dos sistemas e serviços(*web services*), desenvolvidos pela unidade de desenvolvimento *software* que utiliza o barramentos de serviços Erlangms, não possui um acompanhamento específico no funcionamento de seus serviços, visto que há casos e relatos de usuários que acessam os sistemas, realizam a autenticação, clicam nos menus, mas nada aparece como resultado das pesquisas e funcionalidades dos sistemas. Por dispor de uma arquitetura orientada à serviços, a camada de apresentação funciona normalmente, mas os recursos(*web services*) encontram-se indisponíveis.

Sistemas distribuídos podem ser interpretados como aplicações distintas que possuem um *middleware* que realize algum tipo de conexão entre as aplicações como descrevem os autores Penteado e Treveleim [16] em seu trabalho. Esses sistemas ou aplicações podem funcionar com seus serviços de forma independente, ou dependente de outros serviços de aplicações distintas, o funcionamento desses serviços são essenciais, principalmente os que são dependentes, por esse motivo a implantação do monitoramento de sistemas distribuídos é utilizada para um melhor acompanhamento e gerenciamento das aplicações e serviços para verificar o funcionamento e disponibilidade.

2.1.2 Protocolos de Monitoramento

SNMP

O Protocolo Simples de Gerenciamento de Rede (*Simple Network Management Protocol*) é um protocolo da camada de aplicação responsável pela transmissão de dados e informações de gerenciamento e monitoramento entre dispositivos e ativos de rede. De acordo com Roohi *et al.* [17], o SNMP é um dos protocolos mais utilizados atualmente, devido a sua arquitetura que dispõe de uma estrutura simples capaz de realizar o monitoramento em tempo real sem comprometer a eficiência que está sendo monitorado. Nesse sentido os autores Presuhn e Mankin [17, 18] descrevem em seu trabalho, como o SNMP gerencia os ativos de rede baseando-se no esquema cliente-servidor, onde os dispositivo e ativos de rede recebem um agente para que essas informações sejam coletadas e transmitidas, por conta desse paradigma e fácil implantação a popularidade do protocolo aumentou, permitindo as empresas fornecerem diversos tipos de estruturas denominadas MIBs. Os MIBs são objetos criados e definidos por meio de uma estrutura virtual capaz de armazenar de informações de gerenciamento.

O autor Nadeau[1] cita três componentes que formam o SNMP:

- SMI

Alinguagem de modelagem dos dados usados para definir *syntax* dos dados de gerenciamento, como por exemplo, o tipo de objeto "INTEGER".

- MIBs

  Estrutura usada para formar um modelo de dados do sistema(objetos), responsável por representar o tipo, definir o comportamento e a política de acesso aos objetos.

- SNMP

  Que consiste em sua arquitetura, o gerente, um dispositivo a ser gerenciado e um agente, que dispõe das seguintes operações:

  - GET

    Operação utilizada para recuperar o valor de uma instância específica de um objeto gerenciado.

  - GET-NEXT

    Operação utilizada para percorrer interativamente o OID.

  - GET-BULK

    Operação utilizada para recuperar informações de um grupo de objetos.

  - SET

    Operação utilizada para modificar o valor de uma instância de objeto.

  - TRAP

    Operação utilizada para que um agente possa reportar uma notificação de forma assíncrona aos gerentes.

Após a realização de uma análise, estudos e definições, obteve-se de forma empírica o respaldo para utilização do protocolo na implementação do projeto, vale também observar que o CPD utiliza o protocolo para o monitoramento de seus ativos de rede, conforme a representação na figura 2.1 apresentada a seguir.

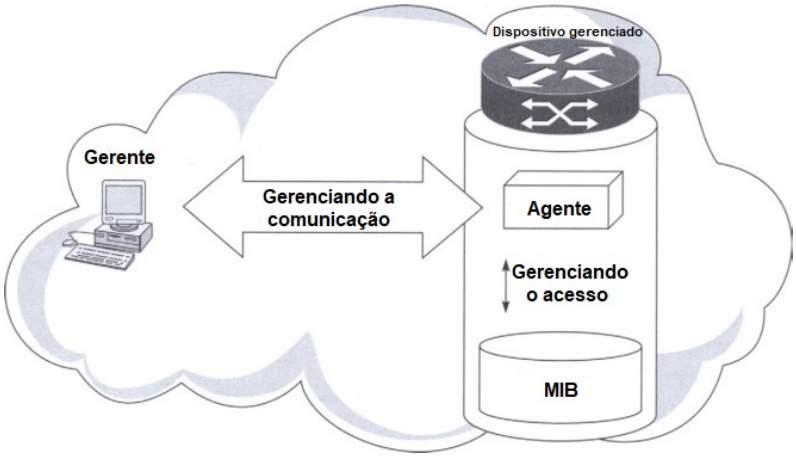

Figura 2.1: Gerente e Agente comunicando por meio do protocolo SNMP[1]

GOSIP

O GOSIP é um protocolo pertencente ao modelo OSI que funciona por meio de software na camada de aplicação em dispositivos *peer-to-peer*, teve seu lançamento anunciado no fim da década de 80 e inicio da década de 90, para facilitar a comunicação e transferência de dados em um modelo governamental prometendo ser a comunicação integrada de sistemas militares do futuro, houve até a formalização de um documento nos padrões RFC o RFC 1169, como descreve o autor Mankin[19], porém Harris[20] descreve em seu trabalho que o protocolo proposto foi implementado com seus objetivos bem definidos, mas que não obtiveram o sucesso almejado, pois esperava-se que pela realização fim-a-fim conseguiriam obter melhores resultados em relação a economia da largura de banda e melhor desempenho com a redundância dos dados.

Apesar dos relatos citados no parágrafo anterior sobre as expectativas não alcançadas com o protocolo, há uma técnica bastante importante que foi extraída, e que é utilizada por meio dos dispositivos de modo a garantir que a comunicação e transmissão da informação sejam garantidas. Diante desse cenário o protocolo trabalha de forma epidêmica, disseminando facilmente as informações nos sistemas distribuídos em larga escala, cujo principal objetivo é propagar rapidamente essas informações, o que permite a realização de um monitoramento *peer-to-peer*, onde os dispositivos se comunicam diretamente como afirma Tanenbaum [21], a representação dessa propagação poderá ser visualizada na figura 2.2.

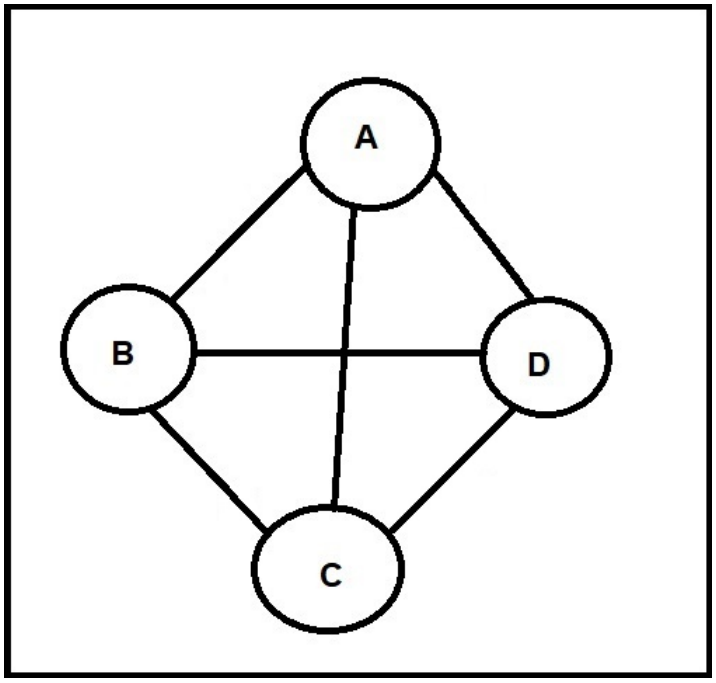

Figura 2.2: Representação de propagação epidêmica [2]

Escolha do Protocolo

Neste trabalho, o GOSIP não foi utilizado no experimento, foi utilizado como fonte de pesquisa para a realização de um comparativo conceitual e tecnológico entre os protocolos, que permitem avaliar e auxiliar na escolha do SNMP como protocolo a ser utilizado no projeto.

2.1.3 Ferramentas de Monitoramento

Na monitoração de ativos de rede ou *softwares* alcançar um nível satisfatório de cobertura das aplicações é um desafio, no mercado existem diversas ferramentas relacionadas ao monitoramento. Com o aumento da disponibilidade de serviços tanto das redes como de *softwares*, é quase inviável acompanhar o funcionamento sem uma ferramenta com especificidade para o monitoramento, o que subsidia necessariamente a escolha de uma solução tecnológica. Nessa seção serão apresentadas as ferramentas de monitoramento de mercado que serão examinados seus prós e contras.

Nagios®

O Nagios® é uma aplicação com interface *web* para o monitoramento de rede baseada na web idealizada e desenvolvida por Ethan Galstad [22]. O Nagios® é projetado para

a realizar o acompanhamento de ativos de rede, sistemas e serviços com a finalidade de notificar os usuários e responsáveis pelos ativos de rede, sistema e serviços que estão registrados na ferramenta como contatos emergenciais, caso aconteça alguma anomalia ou problema durante o funcionamento da rede, sistemas e serviços.

O Nagios® é uma ferramenta que possui uma alta complexidade em sua configuração, pois possui uma gama de recursos e funcionalidades disponíveis para sua utilização, devido a grande quantidade de recursos, o Nagios® é bastante utilizado, pois também possui um grande número de *plug-ins* disponíveis para realização do monitoramento, assim podendo personalizar o monitoramento de serviços como SMTP,SNMP, POP3, HTTP, PING [3]. Nesse projeto o Nagios® será utilizado como ferramenta de monitoramento dos sistemas e serviços fornecidos pela UnB com utilização de serviço SNMP para o monitoramento, haja visto que o CPD já utiliza a ferramenta para o monitoramento dos ativos de rede em toda a Universidade.

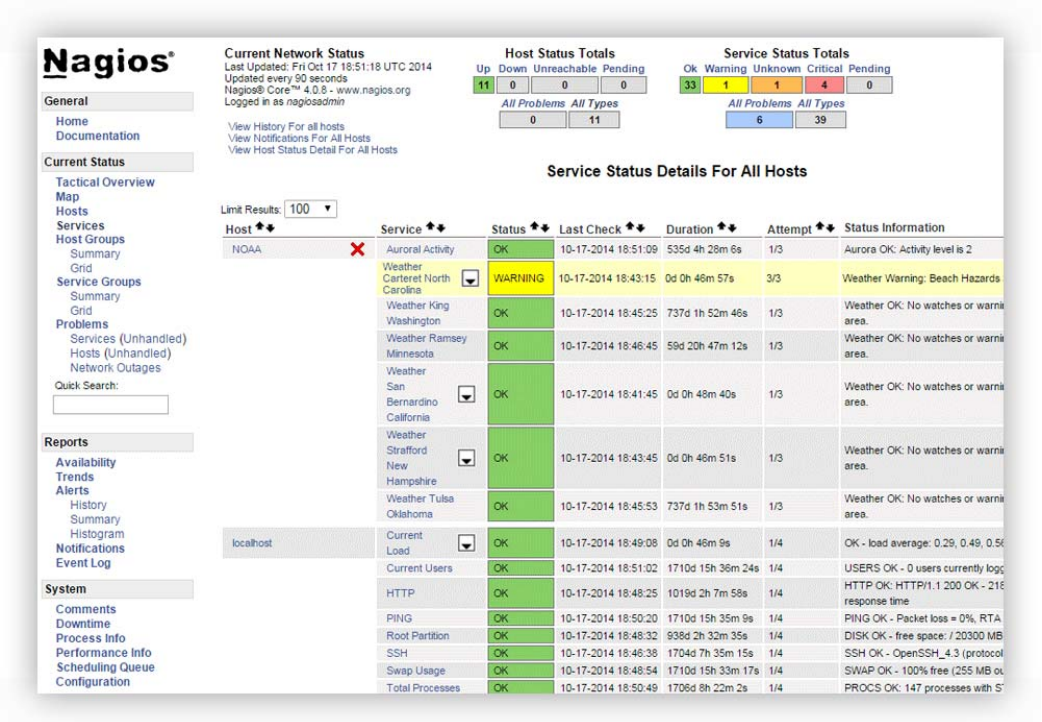

Figura 2.3: Monitoramento de componentes[3]

Ganglia

O Ganglia é um *software* de monitoramento escalonável, com a finalidade de coletar dados para acompanhar o funcionamento de sistemas distribuídos e sistemas de alto desempenho, como *clusters*. Por conta da estrutura hierárquica implementada, o *software* utiliza a linguagem de marcação de texto XML para a representação dos dados.

O os autores Vyas *et al.* [23] relatam em seu trabalho que o Ganglia é baseado em um protocolo *multicast* de escuta e anúncio, e necessariamente precisa ter em cada *host* que será monitorado a instalação de um *Gmond*. O *Gmond* é um serviço em execução em cada *host* que coleta as informações de estado de um *host* de forma individual e insere essas informações no canal de comunicação *multicast* para enviar esse dados ao Ganglia.

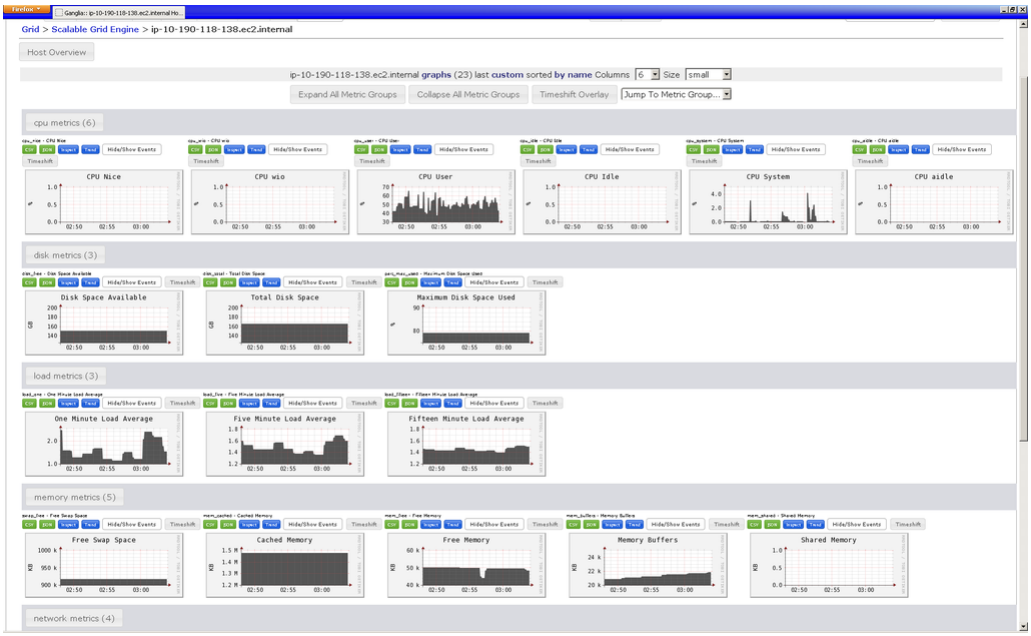

Figura 2.4: Monitoramento de *clusters* [4]

Em uma breve análise após a utilização das ferramentas de monitoramento, Benincosa [24] cita em seu artigo os seguintes itens:

- O Ganglia não possui um sistema de notificação integrado enquanto que o Nagios® se destaca nisso.
- Já o Nagios® não possui agentes integrados escaláveis nos *hosts* de destino (um motivo de reclamação das pessoas), enquanto que isso já faz parte do projeto original e intencional do Ganglia.

Diante desse cenário o autor Benincosa [24] sugere a utilização das ferramentas em conjunto de modo que sejam utilizados em sua potencialidade, porém como o CPD já utiliza o Nagios® e a intenção da Gestão CPD e do projeto é trabalhar com o acompanhamento a notificação dos serviços caso haja alguma falha, ou mau funcionamento, e como o Ganglia não executa uma das principais funcionalidades do monitoramento, a ferramenta não entrará como *software* a ser utilizado no projeto.

Cacti

O Cacti é uma ferramenta que monitora, coleta e analisa informações por meio de gráficos em tempo de execução, com um excelente desempenho, o monitoramento pode ser feito em redes complexas e ativos de rede. O Cacti armazena suas informações coletadas em bancos de dados, para coleta das informações que são necessárias *scripts*/comandos que são implementados em cada ativo e são denominados cactos, e para apresentação dos gráficos é necessário a utilização do *software* Round Robin Database Tool (RRDTool), que possui apresentação gráfica bastante intuitiva e de fácil utilização [5].

Segundo Oetiker[6] o RRDTool é uma ferramenta que realiza o registro de dados e gráficos de alto desempenho para dados de séries temporais, e como os cactos instalados nos *hosts* são *scripts*, o RRDTool pode ser facilmente integrado com *scripts* de *shell*.

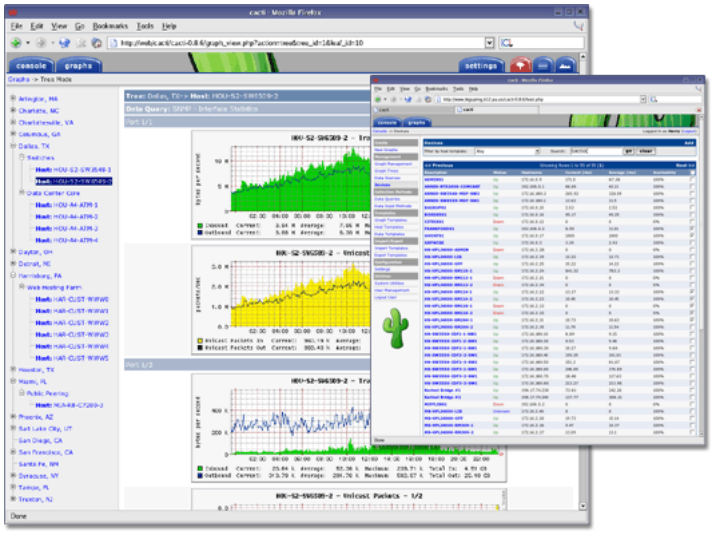

Figura 2.5: Monitoramento de rede [5]

O Cacti possui *plugins* para integração com outras funcionalidades e ferramentas, incluído o protocolo SNMP, porém não é o seu foco devido a preocupação com o desempenho na troca de informações e no monitoramento. Como o trabalho que visa uniformizar o meio de comunicação entre as aplicações e definir especificamente um protocolo de comunicação, para transmitir informações para realização do monitoramento, esse *software* não será utilizado no projeto.

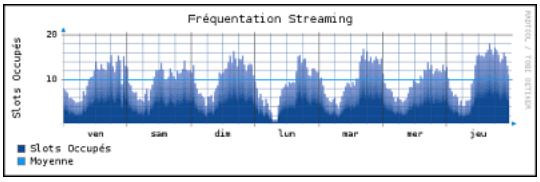

Figura 2.6: Gráfico de monitoramento do RRDTool [6]

Zabbix

O Zabbix é uma solução tecnológica conceituada e desenvolvida para realização de monitoramento de rede, servidores e serviços. A monitoração poderá ser realizada com o agrupamento destes recursos objetivando a potencialização das vantagens de utilização dos ativos de TI, ou a diminuição de riscos da utilização desses ativos aos usuários, esse procedimento é denominado *pooling* ou por mensagens de notificações e alertas por meio de *trapping*, como afirmam os autores Contessa e Fraga[25]. O Zabbix tem como idealizador Alexei Vladishev, e atualmente é mantido pela Zabbix SIA [7].

Como as demais aplicações e *softwares* citados anteriormente, o Zabbix possui funcionalidades para monitorar e coletar de dados por servidores ou agentes, a ferramenta tem suporte ao protocolo SNMP e também interface *web* para visualização de gráficos gerados por meio dos dados coletados dos *hosts* ou agentes. Por definição arquitetural o Zabbix é composto por componentes com funcionalidades primordiais para o seu funcionamento que são eles [7]:

- Servidor Zabbix;
  - É o componente central do Zabbix; que realiza o monitoramento ecom isso interage com os proxies e agentes, calcula as mudanças de estado nas *triggers*, envia notificações, e controla o repositório central de dados.
- Agente Zabbix;
  - É o componente instalado nos servidores que monitora ativamente seus recursos e aplicações.
- Proxy Zabbix;
  - É o componente com a capacidade de realizar a coleta de dados, no lugar do Servidor Zabbix, distribuindo a carga de processamento.

Em comparação com ferramentas mais utilizadas atualmente, os autores Marik e Ondrej [26] descrevem em seu artigo que o Zabbix está entre as melhores aplicações, citadas

neste trabalho, por possuir pré-requisitos básicos na instalação e configuração, excelente tempo de resposta em caso da falha de serviços e recursos de TI, e notificações por e-mail e SMS, porém o *dashboard* para realização do monitoramento é complexo, e não intuitivo ao usuário, como pode ser visualizado a seguir na figura 2.9, podendo não auxiliar em um gerenciamento adequado aos serviços e aplicações que são monitoradas, apesar de uma boa aplicação de monitorização, o Zabbix não será utilizado como ferramenta para este trabalho.

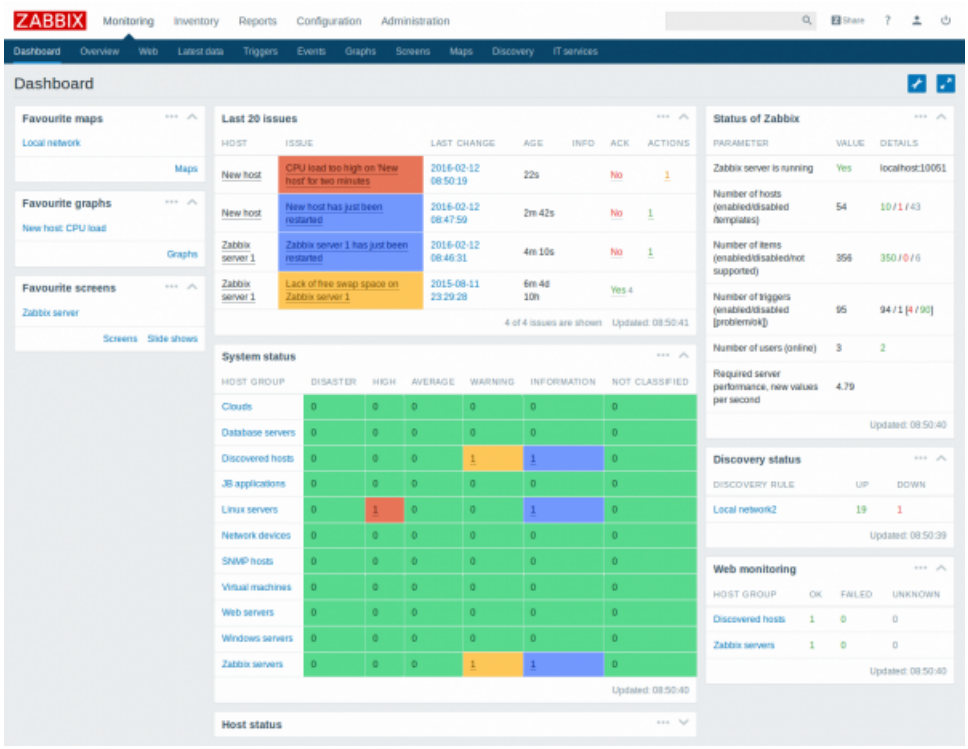

Figura 2.7: *Dashboard* de monitoramento [7].

2.1.4 Agente de Monitoramento

Em uma arquitetura de monitoramento de serviços ou ativos de rede, que utilize protocolos para comunicação e transferência de dados, é necessário a implementação de agentes de monitoramento para instalação em *hosts* e dispositivos, que serão monitorados, estes agentes consistem em coletar dados e enviar mensagens de alerta aos gerentes de monitoramento, como apresentado na figura 2.8, como por exemplo, o uso de CPU e a quantidade de memória de utilização, de um processo ou um alerta de falha, de um dispositivo ou serviço, como descrevem em seu trabalho os autores Grover e Naik [27].

Figura 2.8: Esquema de comunicação entre os agentes e as estações de gerenciamento [8].

Segundo Kazaz *et al.* [28], um agente de monitoramento pode ser definido como um componente ou aplicação de gerenciamento de rede ou serviço no dispositivo a ser gerenciado e possuem duas operações básicas *GET* e *SET*, o agente em execução fica responsável por exportar os dados de gerenciamento dos sistemas, serviços ou ativos de rede gerenciados com variáveis organizadas de forma hierárquica.

Essa hierarquia é contemplada por metadados que são representados por MIBs, que são um conjunto de objetos gerenciados, que procuram abranger todas as informações necessárias para gerência de rede.

Neste trabalho o agente de monitoramento utilizado é o Exometer que funciona como agente de monitoramento responsável pela coleta das informações geradas pelo Barramento de serviços Erlangms, e a transmissão dessas informações ao Nagios® por meio do protocolo SNMP.

Exometer

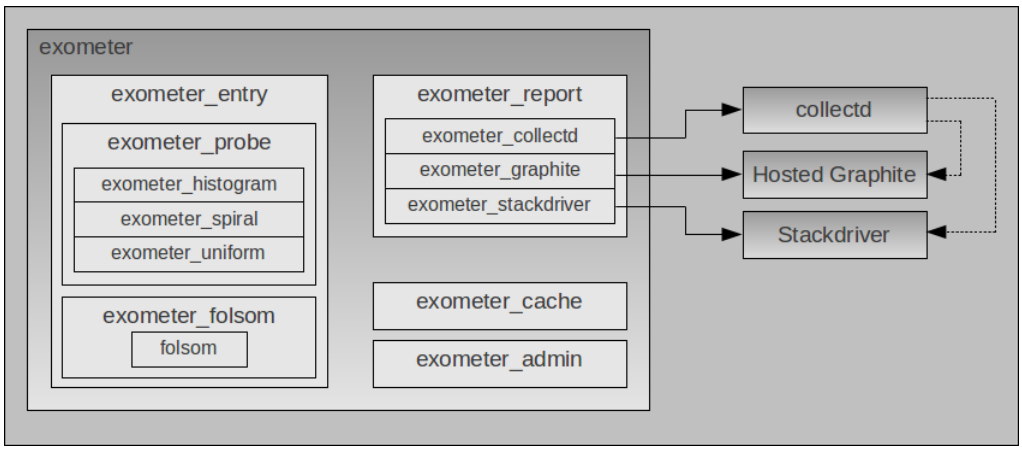

Figura 2.9: Arquitetura do Exometer [9].

O Exometer é um *framework* capaz de executar a instrumentação, e permitir a exportação de dados de maneira fácil e eficiente utilizando bibliotecas implementadas em código Erlang, normalmente os dados exportados são relacionados ao desempenho dos sistemas e serviços. Com a utilização de módulos, o Exometer pode realizar a integração com uma grande variedade de sistemas de monitoramento, neste trabalho a integração é com aplicação de monitoramento Nagios®. De acordo com Wiger [9], o potencial do Exometer é fornecer métricas de várias formas e parâmetros, para isso dispõe de um conjunto de componentes pré-definidos, que podem ser estendidos de forma personalizada para lidar com diversos tipos de métricas e alguns protocolos.

2.1.5 Instrumentação

Segundo Ribeiro [29], a instrumentação é uma das técnicas que utiliza instrumentos para a medição e monitoração. A instrumentação é muito utilizada no ambiente industrial, principalmente no que tange a medição dos equipamentos de produção que necessitam de acompanhamento para verificação de pressão, temperatura, vazão e nível. Os instrumentos podem estar associados e aplicados à alguns equipamentos, como por exemplo, forno, refrigerador, aquecedor, condicionador de ar e compressor. A operação da instrumentação funciona para que possam ser observados os instrumentos para a medição, alarme e monitoração. De acordo com Robinson [30], a instrumentação pode permitir que as informações criadas por meio da execução de instrumentos (código-fonte de *software*) possam ser interferidas no funcionamento das aplicações.

O barramento de serviços Erlangms dispõe em sua arquitetura uma estrutura facilitada para sua instrumentação, e ainda possui um modelo de dados em que são registrados contadores. No modelo de dados são registradas algumas informações como; a data e hora do registro, o identificador do contador e a descrição do contador. Segundo os autores Frota e Finkelstein [31], a instrumentação pode ser utilizada como um componente de processamento da informação de sistemas de medição. Por isso a instrumentação pode ser considerada não apenas como ferramenta de contagem de valores para medição, e poderá ser utilizada também como uma forma de aprendizagem conceitual e científica.

2.1.6 Métricas de Monitoramento

A métrica de *software* é uma das opções que possibilita acompanhar o funcionamento das aplicações por meio de dados e registros, como afirmam os autores Castro e Hernandes [32]. À medida que os sistemas e serviços crescem em quantidade, tamanho, complexidade e criticidade, a necessidade de monitorá-los é cada vez mais evidente.

Nesse seguimento os autores Munawar *et al.* [33], descrevem que o monitoramento representa o acompanhamento dessas aplicações de forma detalhada e clara aos responsáveis pelas aplicações. A ISO/IEC 9126-1 [34] descreve e normatiza a forma para medir a qualidade de *software*, de acordo com a norma [34], a métrica é o método e escala de medição definidos, a métrica pode ser direta ou indireta, a medição de uma métrica direta se dá quando não depende de medidas de outros atributos e a métrica indireta se dá quando há ou existe a necessidade de derivação de um ou mais atributos de medição.

No que tange um ambiente computacional, as métricas ajudam a medir o *software* de uma forma eficaz, para isso é preciso que as métricas sejam bem definidas, justificadas e formalizadas, além disso, é desejável que as métricas apresentem de forma clara o que está sendo medido, e descrevam rapidamente a medição sem custeio computacional alto e que o resultado do que está sendo medido não seja mudado ou sofra interferência em seu resultado caso haja a mudança de *software*, plataforma ou linguagem de programação, conforme Meirelles [35] relata em sua tese.

Por fim, escolher e utilizar o Barramento de serviços da UnB, para o monitoramento permitiu realizar a especificação de algumas métricas. Neste trabalho serão utilizadas as seguintes métricas:

- Registros do *LOG* do barramento de serviços
- Registros de informações de Acesso
- Registros do *Dispatcher*
- Registros do HTTP*(REST Server)*
- Registros de informações do LDAP

2.2 Arquitetura Orientada a Serviços - SOA

A Arquitetura Orientada a Serviços é um estilo arquitetural que permite realizar a comunicação entre aplicações heterogêneas por meio de mensagens, como afirmam os autores Fraser *et al.* [36]. Na implementação e disponibilização de *softwares*, comum que essas aplicações possuam estruturas e linguagens completamente distintas, pois essas aplicações independem umas das outras, mas precisam que a comunicação entre elas aconteça.

Para que haja a comunicação interoperável entre as aplicações é previsto na arquitetura SOA que serviços (*web services*) possam ser utilizados como componentes distribuídos, que ao mesmo tempo que possam fornecer informações, mas possam também consumi-las, conforme descrição dos trabalho dos autores Sward, Boleng e Bianco *et al.*: [37, 38].

O Manifesto SOA apresenta uma proposta de definição da conceituação para área. A partir do manifesto as discussões não cravam apenas conceitos, criavam orientações para guiar as organizações de maneira consistente e sustentável, de modo a agregar valor ao negócio, com maior agilidade e efetividade de custos, em alinhamento com a dinâmica das necessidades de negócio, priorizando os seguintes itens, como afirmam os autores Erl *et al.* [39].

- Valor do negócio em relação a estratégia técnica;
- Objetivos estratégicos em relação a benefícios específicos de projetos;
- Interoperabilidade intrínseca em relação a integração personalizada;
- Serviços compartilhados em relação a implementações de propósito específico;
- Flexibilidade em relação a otimização; e
- Refinamento evolutivo em relação a busca da perfeição inicial.

Para realização da comunicação entre as aplicações ou serviços em componentes distribuídos no estilo arquitetural SOA, os autores Clements *et al.* [40] descrevem em seu trabalho que a comunicação poderá ser executada por meio de um intermediador ou na falta dele poderá ser realizada ponto-a-ponto. Entre os intermediadores do estilo arquitetural SOA, os autores Bass *et al.* [41] citam que a arquitetura dispõe dos seguintes tipos de componentes que compõem uma arquitetura orientada a serviços, são eles :

- Prestador de serviços;
- Consumidores de serviço;
- Enterprise Service Bus (ESB);
- Registro de serviços;
- *Server Orchestration*;
- Simple Object Access Protocol (SOAP).
- Representational State Transferl (REST)
- Conector de mensagens assíncronas

Com a utilização desses componentes, os autores Bass *et al.* [41] tratam como principal benefício de uma arquitetura SOA a interoperabilidade, pois permite que aplicações, dispositivos, sistemas e serviços heterogêneos possam realizar a comunicação entre si, fornecendo e consumindo informações de forma integrada.

O CPD para implementação e implantação da arquitetura SOA na modernização dos seus sistemas, escolheu como componente o ESB e como plataforma o barramento Erlangms[10],que atua como intermediador prestando serviço de mensageria.

2.2.1 *Enterprise Service Bus* - ESB

Um componente ESB é fornece o serviço de infraestrutura e intermedeia a comunicação por meio de serviços entre aplicações consumidoras e fornecedoras de serviços, fazendo de forma uniforme a transmissão das mensagens por meio de um protocolo ou tecnologia, utilizando conectores do tipo *Call-return*, onde os mais comuns e também mais utilizados no mercado são o SOAP e o REST, como descrevem os autores Clements *et al.* [40].

Para a implementação de um ESB no estilo SOA a organização ou instituição deverá avaliar as tecnologias utilizadas em seus sistemas ou serviços, visto que, a complexidade de uma arquitetura SOA é alta, segundo os autores Bianco *et al.* [38], dependendo da situação não é necessário a implementação ou implantação de um ESB quando por exemplo, uma organização que possui em sua arquitetura de sistemas a mesma linguagem, plataforma e tecnologia, podendo fazer a comunicação ou troca de mensagens ponto-a-ponto. Nesse seguimento os autores Bianco *et al.* [38] descrevem sobre o caso em que existe a necessidade da implementação ou implantação de um ESB, a padronização e táticas descritas no trabalhos dos autores Bianco *et al.* [38] e que devem ser seguidas, e de acordo com manifesto SOA[39] e possuem as seguintes características:

- Roteamento Intermediário - Um serviço de roteamento genérico intercepta mensagens e com base no encaminhamento a lógica que determina para onde as mensagens devem ser enviadas.
  - Balanceamento de carga - Possuir um *Failover* para um *backup* no caso de o serviço de destino primário não estar disponível.
  - Seleção da versão do Serviço - Verificar se os pedidos são enviados para versões compatíveis de um serviço para suportar compatibilidade com versões anteriores durante os períodos de transição.
  - Serviço de seleção com base em dados da mensagem - Verificar se pedidos de clientes são enviados para um componente de processamento mais rápido.
  - Regras de controle de acesso - Analisar um pedido de um usuário não autenticado é encaminhado para um início de sessão página.
  - Tratamento de exceções - Apresentar uma mensagem de erro de resposta que é redirecionada para um serviço responsável para manipulação de exceção centralizada.

- O *Service Broker* - Integrar componentes que foram desenvolvidos em diferentes linguagem por diferentes organizações.
  - Modelo de Transformação de Dados - Os dados enviados do consumidor de serviços numa dada estrutura transforma-se em uma estrutura diferente, que está prevista para o prestador de serviços.
  - Conversão de Formato de dados - Usar sempre consumidor de serviço e o fornecedor precisa de trocar dados representados em diferentes formatos como por exemplo, XML, CSV, JSON.
  - Protocolo *Bridging* - O consumidor de serviço envia uma solicitação usando um protocolo e o *Service Broker* intercepta o pedido e o converte para um pedido ao fornecedor do serviço usando um protocolo diferente.
- Mensagens *Asynchronous* - Alguns ESBs fornecem capacidade suficiente, para que o sistema de mensagens permita que os pedidos e respostas de um serviço possam ser trocados através de canais de mensagens.
- Interceptor - Alguns ESBs oferecem a capacidade de configurar interceptores, que são elementos de software que são ativados para todas as solicitações e respostas.

Segundo os autores Bianco *et al.* [38] o ESB deve ter as vantagens e benefícios que uma arquitetura SOA pode fornecer, e para alcançá-los cita os seguintes itens(Atributos de Qualidade), como representação de um ESB padrão e operável. O atendimento à esses itens são de suma importância:

- Interoperabilidade - O ESB permite que sistemas diferentes possam interoperar, por meio de protocolos de comunicação, e tecnologias de implementação.
- Modificabilidade - A capacidade de executar a transformação do modelo de dados que permite a implantação de novas versões de um serviço, sem interromper consumidores de serviços existentes.
- Confiabilidade - Quando o receptor de uma solicitação de serviço ou resposta falhou, e o ESB cria ou gera uma fila, para que a mensagem seja executada para quando o serviço estiver disponível novamente.
- Segurança - O ESB pode incluir a funcionalidade do controle de acesso. Pode aplicar a autenticação e regras de autorização em trocas de mensagens de serviço.

O ESB deve responder a altura seus *trade-offs*, principalmente em uma arquitetura complexa como esta, entre os desafios que deverão ser suportados, para um bom funcionamento da plataforma, os autores Bianco *et al.* [38] citam:

- Manutenibilidade - Ter cuidado e atenção ao especificar demais a codificação do ESB, ao ponto de restringir comunicação com outras tecnologias.
- Performance - Não permitir a perda comprometedora no desempenho do ESB, por conta das lógicas de roteamento ou interpretação dos dados durante a comunicação.
- Segurança - verificar a configuração a fim de evitar acessos não autorizados ao ESB.
- Disponibilidade - O ESB pode ser um ponto único de falha no sistema.

2.2.2 Erlangms

O Barramento de serviços da UnB denominado Erlangms foi conceituado em uma disciplina do MPCA, e proposto como trabalho para solucionar alguns problemas relacionados a quantidade de sistemas heterogêneos, defasagem tecnológica das aplicações e a falta de um padrão de comunicação entre eles no CPD, como Agilar [10] descreve em seu trabalho.

O Erlangms dispõe de uma arquitetura SOA, com o estilo arquitetural REST e sua implementação foi feita na linguagem funcional Erlang. O barramento atualmente está implantado no CPD, e funcionando com um serviço de mensageria. A realização da comunicação e troca de mensagens dos sistemas e clientes, é feita por meio de serviços(*web services*) utilizando o formato JSON. O modelo arquitetural do Erlangms é apresentado na figura 2.10 .

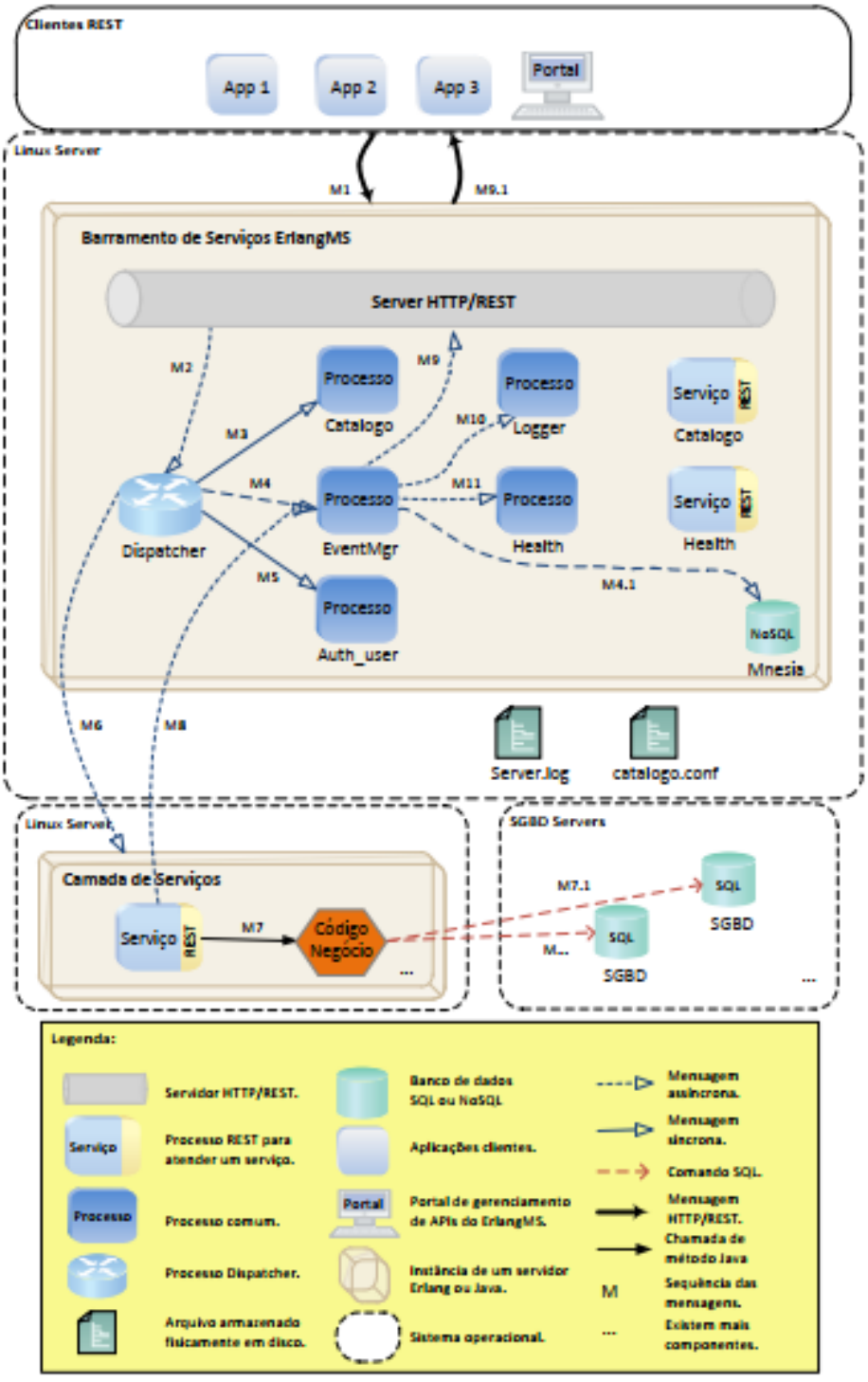

Figura 2.10: Arquitetura do trabalho Erlangms descrita por Agilar [10]

2.2.3 *Representational State Transfer* - REST

A Transferência de Estado Representacional - REST pode ser definida como um estilo arquitetural híbrido derivado de vários estilos arquiteturais, que são baseados em redes e que definem um conjunto de restrições e propriedades baseados no protocolo de comunicação HTTP, com a intenção de fornecer a interoperabilidade entre sistemas, serviços e aplicações distribuídas e conectadas à rede mundial de computadores - WWW para a realização de troca de informações, conforme descrevem os autores Fielding e Taylor [42]. Os *web services* compatíveis com REST permitem que os sistemas solicitantes acessem e manipulem representações textuais de recursos da *Web* utilizando um conjunto uniforme e predefinido de operações sem estado. De acordo com os autores Booth *et al.* [43] o estilo arquitetural proposto por Fielding e Taylor, que inspirou a criação do documento de arquitetura definido pelo TAG - W3C, quanto muitos outros arquitetos de *software*, que o veem como um modelo norteador para construir e padronizar serviços da Web. O REST fornece semântica para uniformizar sua interface, principalmente no que tange à operações para criar, recuperar, atualizar e excluir dados, e também a possibilidade da utilização de padrões de tecnologia como XML e o JSON.

Em REST as principais operações por meio do protocolo HTTP são:

- POST - Cria uma nova entidade ou recurso.
- GET - Recupera a informação de um recurso.
- PUT - Atualiza os dados de um recurso informado.
- DELETE - Exclui os dados de um recurso informado.

2.2.4 *JavaScript Object Notation* - JSON

Segundo Bray [44], JavaScript Object Notation (JSON) é um formato baseado em texto, de fácil leitura e interpretação para humanos, para a leitura de máquinas o JSON é baseado em um subconjunto da linguagem de programação *JavaScript*, e por possuir um um padrão, seu funcionamento independe de uma tecnologia ou linguagem de programação, focando princialmente na troca de dados e mensagens.

Por decisão técnica especificamente relacionada a interoperabilidade e o baixo acoplamento o barramento utilizado pelo CPD o Erlangms, utiliza o JSON por conta do baixo *overhead* gerado na troca de mensagens, em relação às trocas de mensagens feitas com a utilização protocolo SOAP com a linguagem WSDL, e apontando também o nível baixo da curva de aprendizagem para utilização do JSON.

2.3 Trabalhos Relacionados

O Abdu *et al.* [45] descrevem em seu trabalho sobre a opção por investigar a necessidade para realizar o monitoramento de sistemas distribuídos e a necessidade de minimizar a degradação do desempenho de sistemas monitorados. E relatam que a configuração adotada depende da informação a ser monitorada através de diretivas de monitoramento.

Seguindo a preocupação com a escalabilidade do monitoramento durante o funcionamento das aplicações Repantis *et al.* [46] cita sobre a rede Akamai, que permite o processamento dos dados de mais de 60.000 servidores de borda, em quase tempo real.

A disponibilidade de uma poderosa interface SQL-like para que os dados tem sido um elemento importante na forma como gerimos a nossa rede. Neste trabalho, têm-se centrado sobre as escolhas do design que permite a consulta para escalar conforme o tamanho da rede, o volume de dados, e o número de consultas a crescer. E mostram como a consulta aborda esses desafios de escalabilidade, ao fornecer uma latência de dados na ordem de minutos e um tempo médio de resposta de consulta na ordem de décimos de segundo.

O trabalho de Subramanyan *et al.* [47] apresenta estudos sobre o atraso na execução na tarefa de monitoramento devido ao gargalo gerado durante as operações dos agentes, e também descrevem que a solução proposta e implementada utiliza o protocolo do monitoramento rede SNMP, que garante uma facilidade de execução, interoperabilidade e aceitabilidade, e baseia-se nos mais recentes padrões propostas pela RFC 2592.

Phan [48] relata em seu trabalho sobre o nível de segurança do protocolo e questiona sobre a falta de confidencialidade e privacidade, itens faltantes no SNMPv1 e, e descreve em seu trabalho, e dispõe de um experimento, e recomenda a utilização do SNMPv3.

Com a realização da análise qualitativa, os autores Lee *et al.* [49] buscaram apresentar o funcionamento do protocolo SNMP por meio de experimentos valendo-se de suas vantagens, entre elas seus principais componentes a utilização entre eles: 1) Da estação de gestão; 2) Agente de gestão; 3) Base de informações de gestão; e 4) protocolo de gestão. Eles descrevem o trabalho científico de forma conectada, concisa e objetiva, sobre as técnicas utilizadas para realização do estudo. O trabalho apresenta uma abordagem orientada à objetos, utilizada para alcançar o projeto sistemático para implementação de agentes SNMP em sistemas de monitoramento, para o gerenciamento remoto.

Em seu texto Pătruţ *et al.* [50] expõe os resultados obtidos no desenvolvimento e manutenção de aplicações distribuídas, e sistemas de monitoramento multiagente, com exemplos práticos, a fim de implementar e testar as abordagens teóricas. Os sistemas de monitoramento de agentes múltiplos trazem qualidade e eficiência em quase todas as áreas de atividade, na qual inclui pesquisa científica, educação, saúde ou defesa. Utilizando os conceitos sólidos da inteligência artificial e reunindo-os para os recursos infinitos dos

sistemas distribuídos e da Internet, dessa forma buscando reduzir a duração do alcance dos objetivos.

Com a leitura e entendimento dos trabalhos pode-se perceber uma preocupação com a utilização do monitoramento, em relação ao desempenho, mesmo quando utilizadas técnicas e tecnologias distintas, mas observa-se a importância da realização do monitoramento, visto a grande demanda de *softwares* disseminados na rede mundial de computadores.

2.4 Síntese do Capítulo

Neste capítulo foram apresentados conceitos sobre o monitoramento de sistemas distribuídos, desde os conceitos básicos sobre monitoramento, descrevendo e conceituando sobre os protocolos disponíveis para execução do monitoramento, bem como as ferramentas que utilizam desses protocolos para realização da comunicação. Foram abordados também conceitos sobre agentes de monitoramento, técnica de instrumentação e métricas de monitoramento e como elas funcionam, assim como o conceito arquitetural SOA, e seus componentes, entre eles o barramentos de serviços utilizados pela UnB. O Capítulo 3 seguinte apresenta a realização do mapeamento sistemático.

Capítulo 3

Mapeamento Sistemático

Este capítulo descreve o mapeamento sistemático, método utilizado para estruturar a pesquisa da literatura. A pesquisa possibilita identificar, avaliar e interpretar outros trabalhos científicos, além de contribuir no estudo e na resposta da questão desta pesquisa, como afirma os autores Kitchenham *et al.* [51, 12, 13]. O objetivo desta revisão é verificar na literatura textos correspondentes às questões definidas neste trabalho, e que possam colaborar nas respostas sobre monitoramento de sistemas distribuídos.

3.1 Questões de Pesquisa

Para alcançar o objetivo, foram levantadas algumas questões de pesquisa **(QP)**, neste trabalho serão utilizadas 2 questões, com o intuito de fornecer auxílio na fundamentação da pesquisa relacionada ao monitoramento de sistemas distribuídos, as questões são descritas a seguir. As questões têm a intensão de fornecer subsídio durante a busca de informações sobre o tema possibilitando identificar trabalhos, e experiências técnicas que já foram executadas e analisadas, como Feltrim[52] descreve em seu trabalho.

QP1) Quais os estudos primários existentes na literatura que discutem os mecanismos de monitoramento que são aplicados à sistemas distribuídos?

QP2) Quais são as principais características relativas ao monitoramento de sistemas distribuídos mencionadas na literatura?

3.2 Estratégia de busca

Para identificação e busca dos trabalhos, com maior relevância e aderência ao tema abordado, e definido nas questões de pesquisa, foi criada uma *string* de busca. De acordo com Keele *et al.* [53], a forma para criação de uma *string* de busca, é feita a partir da

identificação de sinônimos, abreviações. E também por meio de operadores lógicos, como por exemplo, *AND* e *OR* que auxiliam e permitem concatenar os termos identificados, o que possibilita a elaboração de uma *string*.

Para realização da pesquisa levou-se em conta os padrões e tecnologias mais comuns do mercado utilizados para o monitoramento de sistemas distribuídos, juntamente com algumas palavras-chave: *"Monitoring Protocol"*, *"Distributed Systems"*, *"Monitoring"*, "SOA", "*web services*"e "ESB". As palavras chave foram escritas em inglês, para uma maior abrangência de trabalhos publicados em jornais e revistas internacionais. Diante da situação obteve-se a seguinte *string* de busca: *(("Protocol Monitoring"OR "Monitoring Systems") AND ("Distributed Systems"OR "SOA")) OR (("ESB"AND "Web Services"AND "REST"))*.

De acordo com Kitchenham [51], a obtenção da *string* de busca, foi iniciada a pesquisa dos trabalhos e artigos científicos, as pesquisas foram realizadas nas bases de dados digitais que indexam os principais trabalhos científicos do ramo da Tecnologia da informação, para essa atividade foi utilizado um protocolo de pesquisa para execução do mapeamento sistemático, o protocolo foi separado por etapas, uma representação protocolo de pesquisa poderá ser visualizada na figura 3.1, a execução das etapas proporcionaram uma maior abrangência no acesso à literatura do tema pesquisado.

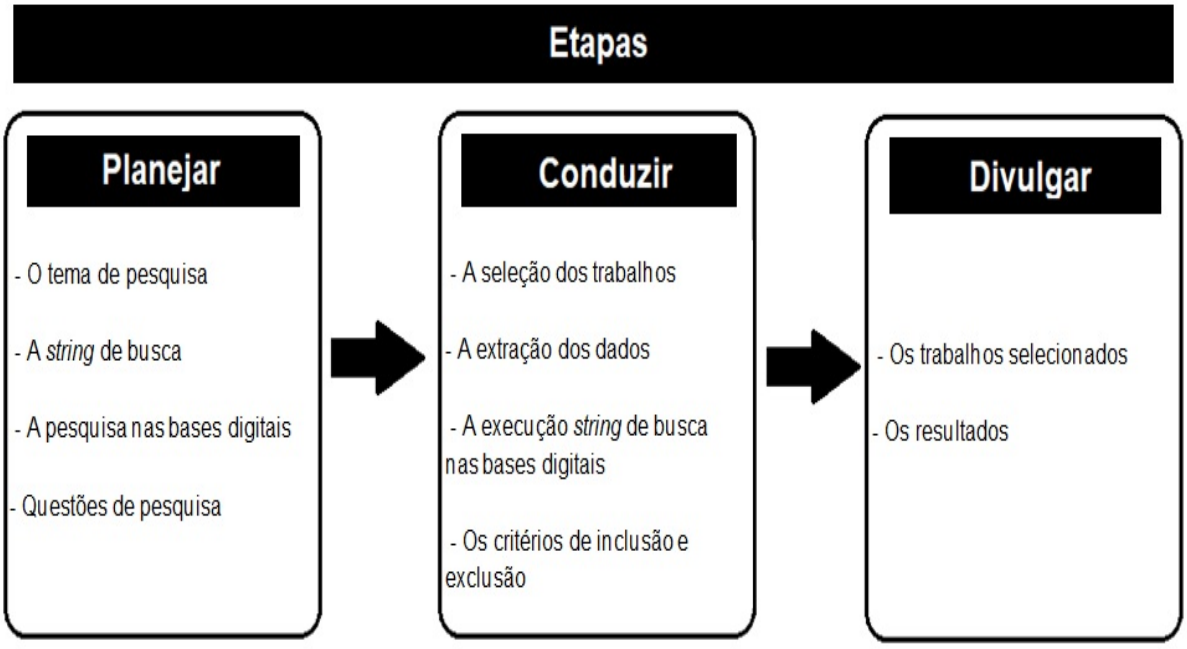

Figura 3.1: Etapas do mapeamento sistemático [11]

Para continuidade das atividades definidas na estratégia de busca seguindo o protocolo de pesquisa definido, a realização das consultas nas bases foi feita por meio da execução

da *string* de busca definida. As bases selecionadas e consultadas para a realização do mapeamento sistemático foram:

- ACM Digital Library - `https://dl.acm.org/`
- IEEE Xplore - `https://ieeexplore.ieee.org/`
- ScienceDirect - `https://www.sciencedirect.com/`
- Scopus - `https://www.scopus.com/`
- Web of Science - `https://www.webofknowledge.com/`

3.3 Critérios de Inclusão e Exclusão

Os critérios de inclusão e exclusão foram definidos para auxiliar na seleção dos trabalhos, estes critérios auxiliam na buscar por trabalhos científicos de maior relevância, e mais aderentes às questões de pesquisas definidas.

Estes critérios foram utilizados em uma ferramenta de apoio que permitiu analisar os trabalhos parcialmente. E possibilitando incluir ou excluir da seleção os trabalhos a partir da leitura do titulo, resumo, introdução e conclusão dos trabalhos disponíveis, que foram obtidos durante a pesquisa nas bases de dados digitais.

Os critérios de inclusão **(CI)** utilizados na seleção dos trabalhos foram:

CI1) Estudo sobre monitoramento de serviços distribuídos;

CI2) Estudo sobre monitoramento de serviços em barramentos SOA;

CI3) Estudo sobre monitoramento de *web services* pelo Protocolo SNMP.

Os critérios de exclusão **(CE)** utilizados na seleção dos trabalhos foram:

CE1) Artigos publicados com baixa relevância definidos pela pontuação de *Score*[1];

CE2) Artigos publicados como *Short Paper*;

CE3) Artigos publicados sem referencia com a área de Tecnologia da Informação.

CE4) Artigos cujo foco não seja monitoramento de serviços distribuídos, ou o monitoramento por Protocolo SNMP, monitoramento em barramentos SOA, ou monitoramento de *web services*;

[1] *Pontuação fornecida pela ferramenta StArt®.Baixa relevância definida por Score < 5*

3.4 Extração dos Dados

A atividade de extração e seleção dos trabalhos foi dividida e realizada em quatro fases, a primeira fase foi a busca automática nas bases de dados digitais com a execução da *string* de busca descrita na seção 3.2, conforme o protocolo definido para busca de trabalhos científicos aderentes ao tema abordado. A segunda fase foi a realização de uma pesquisa com intenção de encontrar uma ferramenta que possibilite o registro e o gerenciamento dos trabalhos buscados, a terceira fase foi a seleção de forma manual para refinar a escolha dos trabalhos relacionados ao tema, e a quarta foi a leitura dos trabalhos seguindo os itens definidos no protocolo para extração das informações dos trabalhos selecionados.

Durante a realização da extração dos dados por meio da execução da *string* de busca em cada base obteve-se vários tipos de informações sobre os dados buscados, como por exemplo, o ano da publicação do trabalho, os locais de realização dos eventos, o DOI dos trabalhos publicados, além do expressivo número de trabalhos que uma base retornou e o número significativo que outra retornou, após a consulta realizada nas bases digitais.

Fase 1

Nessa fase foi realizada a busca dos trabalhos científicos nas bases digitais ACM Digital Library, IEEE Xplore, ScienceDirect, Scopus e Web of Science. Os trabalhos retornados foram obtidos por meio da utilização da *string* de busca descrita na seção 3.2, que após a execução da *string* como meio de filtragem para obtenção dos trabalhos em cada base, obteve-se o retorno dos trabalhos relacionados a partir das palavras ou palavras-chave definidas na *string* de busca.

A busca nas bases de dados foi possível por causa da definição das palavras e textos registrados na *string* de busca, ocasionando o refinamento por meio de filtragem de busca dos trabalhos, totalizando o número de 3.227 trabalhos publicados, que foram retornados após a pesquisa, conforme o registro na tabela 3.1.

Tabela 3.1: Trabalhos retornados por base de dados digitais

Fase 1	
Base de dados	**Total**
ACM Digital Library	86
IEEE Xplore	68
ScienceDirect	2492
Scopus	515
Web of Science	66

Fase 2

Após a conclusão da fase 1, que foi a realização da busca dos trabalhos nas bases científicas foi detectada a necessidade de utilização de uma ferramenta que possibilitasse um melhor controle e gerenciamento dos trabalhos científicos que foram retornados das bases.

A necessidade surgiu por causa do grande volume de trabalhos retornados, e que durante a realização do processo de extração poderia ocasionar a perda do conhecimento ou confusão no entendimento do problema, devido a não organização dos dados obtidos pela leitura dos trabalhos. A ideia foi identificar uma ferramenta capaz de auxiliar no registro de informações obtidas na extração de trabalhos buscados. Nesse seguimento, a escolha de uma ferramenta para auxiliar no processo metodológico para na extração de dados, foi necessária e a ferramenta selecionada para execução do trabalho foi o StArt®.

Fase 3

Após a filtragem e extração dos trabalhos das bases de dados científicas, e a seleção da ferramenta de apoio para o auxílio no processo de extração dos dados, foi feita a exportação dos metadados dos trabalhos retornados de cada base digital em formato BibTex®, como afirma o autor Chomsky [54], após a exportação desses dados foi realizada a importação desses dados na ferramenta StArt®.

A ferramenta possui duas etapa de funcionamento, uma para planejar e outra para executar. Na etapa de planejamento é possível a criar de um protocolo de revisão. Este protocolo é composto por um formulário onde é possível preencher: o objetivo, questões de pesquisa (principais e secundárias), palavras-chave, assim como os critérios de inclusão e exclusão. Neste mapeamento sistemático, o objetivo é a verificação de mecanismos de monitoramento em barramento de serviços distribuídos. Na etapa de execução é possível a organização dos trabalhos por base cientifica, a seleção dos trabalhos e a extração dos dados dos trabalhos científicos. Neste trabalho, a importação foi organizada por base científica digital.

Após a importação e organização dos trabalhos por base, ferramenta sugere um *score* de maior valor para os trabalhos com maior relevância aos itens definidos no protocolo de revisão, e um *score* de menor valor para os trabalhos com menor relevância em referência ao protocolo definido. O *score* gerado pela ferramenta, e relacionado aos trabalhos inicia em zero, que é o valor mínimo de pontuação até quarenta e um, que foi a pontuação máxima gerada após a associação do protocolo de pesquisa com os trabalhos importados.

A partir da indicação do *score* de menor valor e a verificação dos critérios de exclusão, 3.176 trabalhos foram excluídos do mapeamento sistemático, para esta exclusão

foram retirados os trabalhos com *score* menor que cinco. A exclusão dos trabalhos com a verificação dos critérios de exclusão, durou aproximadamente cinco semanas.

Após a realizar a exclusão dos trabalhos de menor relevância para este trabalho, o próximo passo foi realizar a execução dos critérios de inclusão e exclusão, a fim de identificar os trabalhos com maior aderência os tema de pesquisa.

Durante a leitura dos títulos e resumos, ou a leitura dos títulos e introduções, ou a leitura dos títulos, resumos e conclusões dos trabalhos importados, foram verificados os critérios de inclusão e exclusão para seleção dos trabalhos, com os tópicos de inclusão e exclusão descritos na seção 3.3.

De acordo com Petersen *et al.* [55], a execução desse método, visa minimizar a leitura completa de trabalhos que necessariamente não tratam do assunto. Com a utilização e verificação dos critérios de inclusão e exclusão nos trabalhos, por meio da leitura dos títulos e resumos, ou a leitura dos títulos e introduções, ou a leitura dos títulos, resumos e conclusões dos trabalhos, foi possível selecionar os trabalhos mais aderente ao tema do trabalho para leitura totalizando 51 trabalhos selecionados. A figura 3.2 representa a execução dos critérios para identificar os trabalhos.

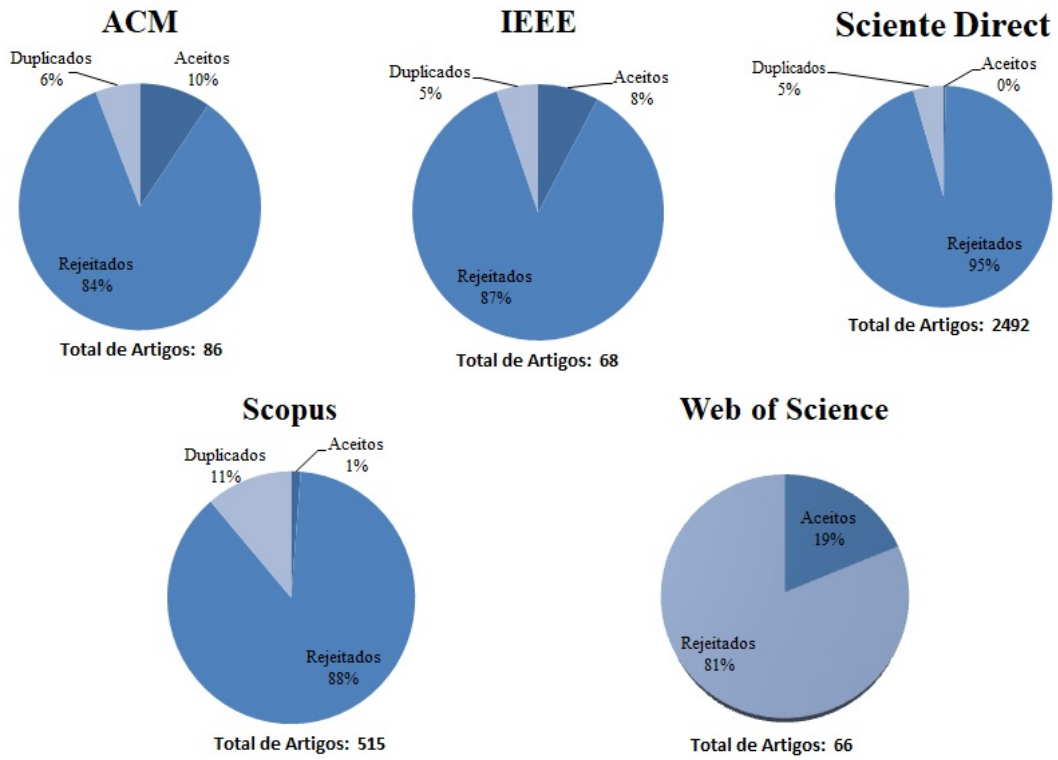

Figura 3.2: Resultado da Fase 3 classificação (CI e CE) em porcentagem.

Fase 4

Após a realização da classificação dos trabalhos científicos metodologicamente permitiu-se a utilização dos critérios de inclusão e exclusão obtendo o total de 51 trabalhos selecionados. Na etapa de execução, é possível além de avaliar os *scores* dos trabalhos com a possibilidade para classificar, na ferramenta, o status de seleção dos trabalhos (Aceito, Rejeitado e Duplicado) e a prioridade de leitura (Muito alta, alta, baixa e muito baixa).

Neste trabalho, após a leitura dos títulos e resumos, ou a leitura dos títulos e introduções, ou a leitura dos títulos, resumos e conclusões dos 51 trabalhos selecionados, foi realizada a classificação de aceitação e prioridade de leitura. Nesse seguimento, foram obtidos 29 trabalhos aceitos, mas com a indicação para prioridade de leitura baixa, por apresentarem pouca aderência e relevância ao tema proposto, e 22 trabalhos aceitos com indicação de prioridade de leitura alta, por possuírem, maior relevância e aderência ao tema, para a leitura completa dos trabalhos.

Após a classificação foi iniciada a 4º fase, com 22 trabalhos aceitos e com prioridade de leitura alta, que possibilitou a observação mais aprofundada nos trabalhos com maior aderência ao tema.

A partir da leitura completa dos trabalhos, a ferramenta StArt® foi utilizada para continuidade do mapeamento sistemático, onde nessa fase foram incluídos nos formulários da ferramenta, atributos definidos no protocolo de pesquisa, e especificamente registrados como "campos para extração dos dados", que possibilitou o registro de informações, com a leitura completa dos trabalhos, o que permitiu a posteriori à avaliação dos trabalhos por meio de uma a análise empírica.

As devidas anotações e classificações de cada trabalho foi registrada na ferramenta, o que permitiu catalogar o trabalho para facilitar a recuperação dos pontos mais importantes, de uma forma mais rápida e precisa, assim como identificar os trabalhos mais relevantes e aderentes às questões de pesquisa. Após a leitura, e ainda a realização de registros e análise dos trabalhos obteve-se uma leitura e entendimento dos 22 trabalhos extraídos e selecionados dos 51 trabalhos que foram selecionados nessa fase.

Tabela 3.2: Trabalhos excluídos, incluídos, rejeitados e selecionados.

	Base de Dados					
	Critério	ACM	IEEE Xplore	ScienceDirect	Scopus	Web of Science
Fase 2	Busca	86	68	2492	515	66
Fase 3	CE1	37	18	1529	259	21
	CE2	18	15	460	156	15
	CE3	14	9	299	61	14
	CE4	9	7	198	33	4
	CI1, CI2, CI3	8	19	6	6	12
Fase 4	Rejeição	0	18	6	4	1
	Seleção	8	1	0	2	11

Os 51 trabalhos selecionados estão na tabela 3.3, e os 22 trabalhos selecionados para realização da extração contribuíram para o mapeamento sistemático do trabalho, estão da tabela 3.4. A revisão é fundamental para auxiliar e fundamentar o conhecimento a fim subsidiar as respostas das questões de pesquisa.

Tabela 3.3: Trabalhos selecionados.

ID	Título	Referęncia
1	Service Oriented Architecture-based Framework for WBAN-enabled Patient Monitoring System	Abousharkh, Maha and Mouftah, Hussein [56].
2	An Investigation of Monitoring Configurations	Abdu, Hasina and Lutfiyya, Hanan L. and Bauer, Michael A [57].
3	Monitoring, Accounting and Automated Decision Support for the Alice Experiment Based on the MonALISA Framework	Cirstoiu, Catalin C. and Grigoras, Costin C. and Betev, Latchezar L. and Costan, Alexandru A. and Legrand, Iosif Charles [58].
4	Scaling a Monitoring Infrastructure for the Akamai Network	Repantis, Thomas and Cohen, Jeff and Smith, Scott and Wein, Joel [46].

5	Implementing and Operating an Internet Scale Distributed Application Using Service Oriented Architecture Principles and Cloud Computing Infrastructure	Sedayao, Jeff [59].
6	A Scalable SNMP-based Distibuted Monitoring System for Heterogeneous Network Computing	Subramanyan, Rajesh and Miguel-Alonso, José and Fortes, José A. B. [47].
7	Monitoring Distributed Systems	Joyce, Jeffrey and Lomow, Greg and Slind, Konrad and Unger, Brian [60].
8	Monitoring Overhead in Distributed Systems: Visualization and Estimation Techniques	Abdu, Hasina and Lutfiyya, Hanan and Bauer, Michael A.[45].
9	Improvements to online distributed monitoring systems	Wang, Bo and Song, Ying and Sun, Yuzhong and Liu, Jun [61].
10	A Formal Engineering Approach for Control and Monitoring Systems in a Service-Oriented Environment	Nagorny, Kevin and Harrison, Robert and Colombo, Armando Walter andKreutz, Gerhard [62].
11	Personal Health Service Framework	Ghorbani, Shirin and Du, Weichang [63].
12	JMonitor: A monitoring tool for distributed systems	Penteado, Mauricio G. and Trevelin, Luis Carlos [16].
13	Agent Technology in Monitoring Systems	Patrut, Bogdan and Tomozei, Cosmin [50].
14	Cryptanalysis of the application secure alternative to SNMP (APSSNMP)	Phan, Raphael C. -W. [48].

15	Profiling Node Conditions of Distributed System with Sequential PatternMining	Hirate, Yu and Yamana, Hayato[64].
16	SeNsIM-Web: a Service Based Architecture for Sensor Networks Integration	Casola, Valentina and Gaglione, Andrea and Mazzeo, Antonino [65].
17	A SOA-based information management model for Next-Generation Network	Kotsopoulos, Konstantinos and Lei, Pouwan and Hu, Yim Fun [66].
18	A Flexible Monitoring and Notification System for Distributed Resources	Smith, Garry and Baker, Mark [67].
19	Design and implementation of the SNMP agents for remote monitoring andcontrol via UML and Petri nets	Lee, JS and Hsu, PL [49].
20	Distributed mobile communication base station diagnosis and monitoringusing multi-agents	Feng, JQ and Buse, DP and Wu, QH and Fitch, J. [68].
21	An authentication and authorization solution supporting SOA-based distributed systems	P. Qi-rui and W. Cheng and W. Jing and L. Jun and L. Qing and S. Bei-en [69].
22	Scalable Agentless Cloud Network Monitoring	M. Brattstrom and P. Morreale [70].
23	Implementation of secure customized monitoring tool for adapative distributed systems	M. Kotari and N. N. Chiplunkar and H. R. Nagesh [71].

24	Multi-layer SOA implementation pattern with service and data proxies for distributed data-intensive application system	Takdir and A. I. Kistijantoro [72].
25	ABMOM for cross-platform communication in SOA systems	N. M. Ibrahim and M. F. Hassan and M. H. Abdullah [73].
26	Reliable ESB and Distributed Transactional Memory for SOA	S. Yong and R. Yi-Zhi [74].
27	Technologies for SOA-based distributed large scale process monitoring and control systems	F. Jammes and B. Bony and P. Nappey and A. W. Colombo and J. Delsing and J. Eliasson and R. Kyusakov and S. Karnouskos and P. Stluka and M. Till [75].
28	Agent-based intelligent monitoring in large-scale continuous material transport	Y. Pang and G. Lodewijks [76].
29	Web Services Open Test Suites	N. El Ioini [77].
30	Researching and Designing the Architecture of E-government Based on SOA	P. Yan and J. Guo [78].
31	Towards Role-Based Self-healing in Autonomous Monitoring Systems	W. Funika and P. Pegiel and M. Bubak and J. Kitowski [79].
32	Modeling Service Oriented Architectures of Mobile Applications by Extending SoaML with Ambients	N. Ali and M. A. Babar [80].

33	Research on Multi-tier Distributed Systems Based on AOP and Web Services	J. Zhang and F. Meng and G. Liu [81].
34	Research on SOA-Based Applications Based on AOP and Web Services	J. Zhang and F. Meng and G. Liu [82].
35	Monitoring distributed embedded systems	R. Ford [83].
36	Monitoring distributed systems	M. Mansouri-Samani and M. Sloman [84].
37	The dark side of SOA testing: Towards testing contemporary SOAs based on criticality metrics	P. Leitner and S. Schulte and S. Dustdar and I. Pill and M. Schulz and F. Wotawa [85].
38	Leveraging Service Oriented Architecture: A case study for ocean energy information management	A. Bosnjak and S. Huang and J. J. Mulcahy [86].
39	A managerial community of Web Services for management of communities of Web Services	A. Benharref and M. A. Serhani and S. Bouktif and J. Bentahar [87].
40	Decision guidance models for microservice monitoring	Haselbock, S. and Weinreich, R. [88].
41	Building the monitoring systems for complex distributed systems: Problems & solutions	Korableva, O. and Kalimullina, O. and Kurbanova, E. [89].
42	Pattern detection model for monitoring distributed systems	Dinu, C.-M. and Pop, F. and Cristea, V. [90].
43	Dynamic agent based monitoring mechanism for web services	Wen, J. and Zhao, L. and He, P. [91].
44	A performance study of monitoring and information services for distributed systems	Zhang, X. and Freschl, J.L. and Schopf, J.M. [92].

45	HiFi: a new monitoring architecture for distributed systems management	Al-Shaer, Ehab and Abdel-Wahab, Hussein and Maly, Kurt [93].
46	Goal-driven adaptive monitoring of SOA systems	Marek Psiuk and Krzysztof Zielinski [94].
47	On heuristics for optimal configuration of hierarchical distributed monitoring systems	Jiannong Cao and Kang Zhang and Olivier de Vel [95].
48	What Can Web Services Bring to Integrated Management?	Aiko, Pras and Jean-Philippe Martin-Flatin [96].
49	Transformer Fleet Monitoring	Jaković, Tihomir, Ivan Murat, Filip Klarić and Samir Keitoue [97].
50	Model-based monitoring and policy enforcement of services	Xiaoying Bai and Yongli Liu and Lijun Wang and Peide Zhong [98].
51	Web Service Diagnoser Model for managing faults in web services	K. Jayashree and Sheila Anand [99].

Tabela 3.4: Trabalhos selecionados para leitura completa

ID	Título	Referência
1	Service Oriented Architecture-based Framework for WBAN-enabled Patient Monitoring System	Abousharkh, Maha and Mouftah, Hussein. [56].
2	An Investigation of Monitoring Configurations	Abdu, Hasina and Lutfiyya, Hanan L. [57].

3	Monitoring, Accounting and Automated Decision Support for the Alice Experiment Based on the MonALISA Framework	Cirstoiu, Catalin C. and Grigoras, Costin C. and Betev, Latchezar L. and Costan, Alexandru A. and Legrand, Iosif Charles. [58].
4	Scaling a Monitoring Infrastructure for the Akamai Network	Repantis, Thomas and Cohen, Jeff and Smith, Scott and Wein, Joel. [46].
5	Implementing and Operating an Internet Scale Distributed Application Using Service Oriented Architecture Principles and Cloud Computing Infrastructure	Sedayao, Jeff. [59].
6	Scalable SNMP-based Distibuted Monitoring System for Heterogeneous Network Computing	Subramanyan, Rajesh and Miguel-Alonso, José and Fortes, José A. [47].
7	Monitoring Distributed Systems	Joyce, Jeffrey and Lomow, Greg and Slind, Konrad and Unger, Brian. [60].
8	Dynamic agent based monitoring mechanism for web services	Wen, J. and Zhao, L. and He, P. [100]
9	Monitoring Overhead in Distributed Systems: Visualization and Estimation Techniques	Abdu, Hasina and Lutfiyya, Hanan and Bauer, Michael A. [45].

10	Improvements to online distributed monitoring systems	Wang, Bo and Song, Ying and Sun, Yuzhong and Liu, Jun. [61]
11	Scalable Agentless Cloud Network Monitoring	M. Brattstrom and P. Morreale [101].
12	Personal Health Service Framework	Ghorbani, Shirin and Du, Weichang. [63].
13	JMonitor: A monitoring tool for distributed systems	Penteado, Mauricio G. and Trevelin, Luis Carlos. [16].
14	Agent Technology in Monitoring Systems	Patrut, Bogdan and Tomozei, Cosmin. [50].
15	Cryptanalysis of the application secure alternative to SNMP (APSSNMP)	Phan, Raphael C. -W. [48].
16	Profiling Node Conditions of Distributed System with Sequential PatternMining	Hirate, Yu and Yamana, Hayato. [64].
17	SeNsIM-Web: a Service Based Architecture for Sensor Networks Integration	Casola, Valentina and Gaglione, Andrea and Mazzeo, Antonino. [65].
18	A SOA-based information management model for Next-Generation Network	Kotsopoulos, Konstantinos and Lei, Pouwan and Hu, Yim Fun. [66].

19	A Flexible Monitoring and Notification System for Distributed Resources	Smith, Garry and Baker, Mark. [67].
20	Design and implementation of the SNMP agents for remote monitoring andcontrol via UML and Petri nets	Lee, JS and Hsu, PL. [49].
21	Distributed mobile communication base station diagnosis and monitoringusing multi-agents	Feng, JQ and Buse, DP and Wu, QH and Fitch, J. [68].
22	Building the monitoring systems for complex distributed systems: Problems & solutions	Korableva, O. and Kalimullina, O. and Kurbanova, E.. [89].

3.5 Resultados

Após a realização da pesquisa dos trabalhos, com a maior relevância e maior aderência ao tema de pesquisa nas bases digitais, a seleção e extração dos trabalhos para leitura, nesta seção por meio da utilização do mapeamento sistemático.

Será possível responder às questões de pesquisa descritas na seção 3.1 de forma fundamentada pelos trabalhos científicos obtidos por meio de resumos referentes às questões.

As leituras dos trabalhos ampliaram o conhecimento sobre o tema, e permitiram a identificação e apresentação dos seguintes itens, para utilizá-los como base para execução do projeto, são eles:

- Monitoramento de sistemas distribuídos
- *Web services*
- Agentes de monitoramento
- Protocolo SNMP

- Armazenamento das Informações monitoradas
- Desempenho

QP1) Quais os estudos primários existentes na literatura que discutem os mecanismos de monitoramento que são aplicados a sistemas distribuídos?

Monitoramento de Sistemas Distribuídos

Na seleção dos trabalhos realizou-se uma pesquisa que foi identificada a necessidade do monitoramento de sistemas distribuídos, devido à grande quantidade de aplicações, dispositivos e *web services* em funcionamento, e que normalmente não possuem acompanhamento durante o seu funcionamento.

O autor Penteado [16], descreve sobre a importância das ferramentas de monitoramento possuírem padrões, para a transferência de dados aos Agentes de monitoramento.

No trabalho descrito por Cirstoiu *et al.* [58], sobre o monitoramento de sistemas distribuídos e da utilização das API's para comunicação de aplicativos e a preocupação com o baixo acoplamento dessas ferramentas.

Em outro trabalho, este descrito por Joyce *et al.*[60], sobre a importância do monitoramento de sistemas distribuídos, e a utilização de testes de programas para verificação dos sistemas monitorados.

No entanto o autor Abdu *et al.* [57], descreve o quanto é complexo o gerenciamento do monitoramento de sistemas distribuídos, sendo necessário a utilização de técnicas e métricas para obtenção dos resultados coletados, por conta da grande quantidade de informações geradas pelo monitoramento.

Web Services

Os autores Abousharkh *et al.*, Casola *et al.* e Ghorbani *et al.* [56, 65, 63], apresentam em seus trabalhos, como a flexibilidade e agilidade no desenvolvimento de *web services*, podem trazer resultados imediatos paras aplicações, suas características que facilitam a integração dos serviços e sistemas, e como é fácil a integração de ambientes com os padrões da Indústria e protocolos como SOAP e HTTP, porém a facilidade abre uma preocupação referente à segurança das aplicações, principalmente no que tange a integridade e confidencialidade dos dados. Também relatam sobre desafios como implementar uma solução que seja flexível, escalável e interoperável.

Agentes de Monitoramento

Agentes de monitoramento podem ser um dispositivo ou um software, esses podem ser utilizados para realizar a comunicação ou notificação do monitoramento de outros dispositivos ou sistemas distribuídos.

O autor Smith *et al.* [67], explica sobre a utilização e comparação de ferramentas (*plugins*) que funcionam como agentes de monitoramento de sistemas distribuídos e também da arquitetura definida para o gerenciamento e acompanhamento.

O autor Pătruţ *et al.* [50], relata sobre o experimento com informações metacognitivas que proporcionaram 12 definições para identificação de agentes inteligentes, propostas para definição de um agente ou Super agente, e como eles podem monitorar sistemas para realização de comunicação homem-máquina.

O autor Feng *et al.* [68] discorre em seu trabalho sobre a importância da realização de monitoramento por agentes, devido o grande numero de serviços disponíveis, e que por vários motivos que podem sofrer alguma interferir no funcionamento, e projeta a implementação de serviços com multiagentes para fornecer desempenho na transmissão de informações do monitoramento pelos agentes.

Protocolo SNMP

O autor Lee *et al.* [49], descreve sobre a qualidade do Protocolo SNMP, seu principais componentes e utilização da estação de gestão, agente de gestão, base de informações de gestão e o protocolo de gestão e suas vantagens.

Apesar de seu nome, Simple Network Management Protocol, o SNMP é um protocolo relativamente complexo para implementar. Também, o SNMP não é um protocolo muito eficiente. Os autores Phan e Subramanyan *et al.* [48, 47] também relatam sobre alguns pontos fracos, como a vulnerabilidade presente no SNMPv1, desempenho e falta de escalabilidade.

QP2) Quais são as principais características relativas ao monitoramento de sistemas distribuídos mencionadas na literatura?

Armazenamento das Informações Monitoradas

O Monitoramento dos sistemas distribuídos quando implementados geram informações durante a execução, essas informações são importantes para acompanhar o funcionamento de sistemas distribuídos e *web services*. O autor Repantis *et al.* [46], explica sobre a importância da coleta dessas informações e a coleta em larga escala utilizando instruções e comandos *SQL*. Porém devido ao grande número de sistemas distribuídos monitorados e a grande quantidade de informações geradas por meio do monitoramento, os autores

Abdu *et al.* e Hirate *et al.* [45, 64] apresentam as técnicas para mensurar a sobrecarga gerada pela quantidade de informações e padrões de mineração dos dados armazenados.

Desempenho

O autor Wang [61], apresenta algoritmos de compactação de dados, para realização de transferência dos dados de modo otimizado, devido à grande quantidade de informações geradas durante o monitoramento dos sistemas distribuídos. Nesse trabalho foram realizados vários testes, incluído conversão de formatos, como por exemplo, arquivos XML.

O autor Sedayao [59], descreve como a falta de experiência para implementar *web services* para realização de monitoramento, impactaram diretamente no desempenho da aplicação, e como adquiriram experiência para a implementação de forma correta dos serviços, tornando a aplicação robusta, e altamente distribuída.

Já o autor Kotsopoulos *et al.* [66], explica o motivo da não utilização do protocolo SNMP, por conta de algumas limitações, como por exemplo, a escalabilidade e eficiência.

Escalabilidade

A escalabilidade de sistemas distribuídos está entre as principais características técnicas de funcionamento, ela busca disponibilizar o acesso às aplicações mesmo com o aumento gradativo de usuários. O autor Brattstrom *et al.* [101], descreve que o sistema de computação distribuída deve ser dimensionado para o uso de um grande número de recursos de computação, a tendência é aumentar devido a grande evolução cibernética. No entanto, os sistemas devem escalar de modo que não atrapalhe o desempenho dos sistema, a escalabilidade é encarada com um desafio.

Nesse seguimento a autora Korableva *et al.* [89], relata a preocupação com a utilização de sistemas de monitoramento, que possam ser implementados de forma sistema integrada, escalável e fácil de usar, em seu trabalho descreve um método para alcançar esse objetivo em seu experimento. O autor Wen *et al.* [100], descreve a medida que o número de serviços à ser monitorado aumenta, o sistema tem impacto negativo na flexibilidade e escalabilidade, principalmente no tempo de resposta. A capacidade de um sistema pode ser aprimorada de maneira direta, e utiliza uma solução implementada que trabalha dinamicamente, para oferecer suporte a uma maior carga de informações durante a realização do monitoramento, sem sofrer degradação perceptível no desempenho do tempo de resposta.

Com o mapeamento sistemático, foi possível identificar os trabalhos com maior relevância e aderência ao tema deste trabalho. Pode-se perceber pontos, positivos e negativos

sobre os itens definidos para pesquisa, o monitoramento de sistemas distribuídos, *Web services*, agentes de monitoramento, o Protocolo SNMP, sobre o armazenamento das informações monitoradas e a preocupação como desempenho Verificar os trabalhos, também contribuíram para identificar os principais desafios relacionados à este trabalho.

3.6 Síntese do Capítulo

Este capítulo apresentou a execução do mapeamento sistemático para identificar nos trabalhos científicos, dados e informações relacionadas ao monitoramento de sistemas, ao protocolo de monitoramento SNMP e ferramentas de monitoramento, dessa maneira conseguir ir possibilitando a fundamentação para implementação do projeto de monitoramento dos serviços do barramento Erlangms, com a utilização do protocolo SNMP para integração com ferramentas de monitoramento. O protocolo SNMP foi identificado como solução para implementação do projeto, visto que o CPD atualmente já utiliza, pois possui ferramentas de monitoramento que realizam o monitoramento de *softwares* e ativos de rede, por meio do protocolo.

Capítulo 4

Monitoramento do Barramento de Serviços pelo Ems-Monitor

Este capítulo apresenta o modelo de monitoramento para o CPD denominado Ems-Monitor. Ems-Monitor realiza o monitoramento do barramento de serviços por meio do protocolo SNMP. Na Seção 4.1 é apresentado o conceito da arquitetura do Ems-Monitor. Na Seção 4.2 é descrito o monitoramento dos recursos do barramento de serviços. Na Seção 4.3 são discutidos detalhes do funcionamento do protocolo SNMP. Na Seção 4.4 são expostas as definições e escolhas das métricas adotadas, para utilização Ems-Monitor. Na Seção 4.2.1 são descritos argumentos para instrumentação para coleta de dados para realização do monitoramento.

4.1 Arquitetura do Ems-Monitor

A arquitetura de *software* é um conceito ou modelo de organização de uma estrutura, esta estrutura pode ser um módulo ou componentes do *software*. De acordo Shaw e Garlan [102], um estilo arquitetônico, pode definir um grupo de sistemas com organização estrutural. Estilo arquitetônico determina componentes e conectores que são utilizados, com conjuntos de restrições, essas restrições podem ser topológicas. Nesse seguimento Clements *et al.* [40] descreve, a arquitetura de *software* de um sistema é o conjunto de estruturas necessárias para entender o *software*, que incluem elementos, relações entre eles e suas propriedades.

A arquitetura do Ems-Monitor tem como objetivo estabelecer a integração entre o barramento de serviços e ferramentas de monitoramento por meio de um agente de monitoramento. Para isso foi selecionada uma visão arquitetural da categoria *Runtine*, que apresentam os componentes e suas interações em tempo de execução. Segundo Clements *et al.* [40], uma visão é uma representação de uma estrutura, e afirma uma visualização é

uma representação de um conjunto de elementos do sistema, e a relação entre eles. Nesse seguimento foram definidos os seguintes componentes:

- **Barramento de serviços** - Provedor de informações e dados, a partir do seu funcionamento, contempla o componente ***ESB***;
- **Ferramenta de monitoramento** - Coletor de informações e dados para realizar o monitoramento, e provedor de informações sobre o funcionamento de serviços e aplicações;
- **Agente de monitoramento** - Coletor de informações e dados de monitoramento, ele armazena informações e dados de monitoramento coletados pelo agente de monitoramento. Transmite informações e dados de monitoramento para ferramenta de monitoramento, e assim abrangendo o componente do **Banco de Dados**

Com a definição da arquitetura do Ems-Monitor e a organização de seus componentes, espera-se obter níveis baixos de dependência entre as soluções. Os componentes implementados podem ser conectados em outras aplicações.

Na arquitetura definida deste trabalho o meio de comunicação entre os componentes, é feito por meio de integração, que possibilita manter um nível baixo de dependência das aplicações e a mudança de barramento de serviços ou da ferramenta de monitoramento, sem prejuízo à arquitetura e ao funcionamento do Ems-Monitor. O modelo conceitual da arquitetura pode ser visualizado na figura 4.1

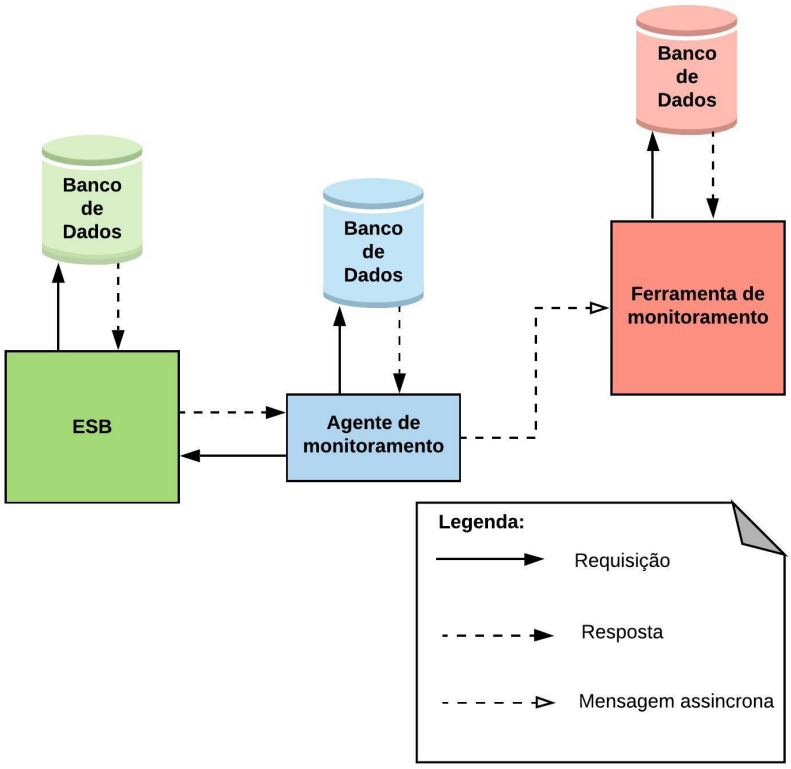

Figura 4.1: Arquitetura conceitual do Ems-Monitor.

Como parte da arquitetura, o barramento de serviços é o componente responsável por gerar dados e informações baseado no seu funcionamento. Com suporte dessas informações que o monitoramento é realizado, outro componente definido na arquitetura é a ferramenta de monitoramento, a ferramenta de monitoramento é responsável por apresentar informações do funcionamento dos serviços monitorados a partir das informações geradas pelo barramento de serviços.

Por fim, o componente agente de monitoramento é encarregado de coletar e transmitir as informações geradas pelo barramento de serviços para ferramenta de monitoramento.

Neste trabalho os componentes da arquitetura na visão de *runtime*[40] possuem as seguintes descrições:

Barramento de Serviços

O barramento de serviços é representado pelo Erlangms que dispõem de uma arquitetura SOA com o estilo arquitetural REST, sua implementação foi feita na linguagem funcional Erlang. O Erlangms é o ESB utilizado pelo CPD da UnB para implementação e modernização de *softwares* na UnB. O barramento de serviços é utilizado para integração de várias aplicações heterogêneas por meio de serviços.

Ferramenta de Monitoramento

A ferramenta de monitoramento é representada pelo Nagios®, e foi projetado para monitorar ativos de rede, sistemas e serviços, notifica os responsáveis pelos ativos de rede, além dos sistemas e serviços que estão registrados como contatos emergenciais. A ferramenta é utilizada pelo CPD da UnB para acompanhamento da disponibilidade de seus serviços.

Agente de Monitoramento

O agente de monitoramento é representado pelo Exometer que é um *framework*, que executa instrumentação e permite a exportação de dados, foi implementado em Erlang. Neste trabalho foi utilizado o módulo SNMP da ferramenta para realizar a integração do barramento de serviços e a ferramenta de monitoramento. Com o agente de monitoramento é possível criar métricas, e grupos de dados para monitoramento.

Comunicação

As mensagens trocadas entre os componentes do barramento de serviços, e a ferramenta de monitoramento, com intermediação do Agente de monitoramento é feita por meio do protocolo SNMP.

O protocolo é utilizado na camada de Aplicação do modelo OSI [103], é responsável pela transmissão de dados e informações de gerenciamento e monitoramento entre dispositivos e ativos de rede.

Cabe ressaltar a importância da definição de um protocolo para comunicação, a escolha de um modelo de comunicação possibilita a manutenção adequada, a seleção de outras ferramentas que trabalhem com o protocolo e a baixa dependência de aplicações.

Com o conceito arquitetural do Ems-Monitor definido, está fortemente alinhado com o ambiente computacional da UnB, este trabalho emprega ênfase na integração dos componentes para realização do monitoramento.

4.2 Monitoramento dos Recursos do Barramento de Serviços

O CPD da UnB adotou como solução para modernização, e migração dos sistemas e tecnologia a implementação de uma arquitetura orientada a serviços, que veem sendo bastante utilizadas principalmente para criação de serviços (*web services*).

A implementação dos serviços tem aumentado devido a flexibilidade definida na arquitetura, e o aumento da demanda para integração de sistemas, além da curva de aprendizagem ser alta.

A partir da implementação e disponibilização de vários serviços, monitorar e acompanhar o funcionamento, tem se tornado uma tarefa que tem demandado um grande esforço das equipes de desenvolvimento do CPD.

No CPD, o monitoramento do barramento de serviços da UnB pode ser realizado: uma por meio do *Log*, desenvolvido por Filgueiras [104], com um módulo de monitoramento, que utiliza componentes do *Elastic Stack* para a leitura de arquivos gerado pelo barramento, outra por meio de serviços (*web services*) providos pelo barramento, além da nativa que é disponibilizada pela empresa que mantém e gerencia a linguagem Erlang o denominado Observer [105].

Monitoramento do Barramento de Serviços por *Log* de Arquivos

As informações de registros do *Log* geradas pelo barramentos de serviços, são coletadas e armazenadas. A abordagem utilizada para acompanhamento do funcionamento do barramento de serviços possibilitou o monitoramento, sem a necessidade de modificação no código-fonte do barramento de serviços, observado que um dos motivos pelo acompanhamento por meio do *Log*, é realizar o monitoramento em tempo real sem a perda de desempenho do barramento de serviços durante o funcionamento.

De acordo com Filgueiras[104] as informações armazenadas originárias do arquivo de *Log* do Erlangms, são armazenadas pelo conjunto de ferramentas ElasticStack, composto pelo Beats, Logstash e Elasticsearch.

Esse conjunto possibilita pesquisar, transmitir e visualizar os dados em tempo real[104]. Suas métricas são apresentadas em um *dashboard*, esse *dashboard* foi desenvolvido para acompanhamento do funcionamento do barramento de serviços graficamente, como por exemplo o quantitativo de erros por serviço.

Monitoramento do Barramento de Serviços por Serviços (*Web Services*)

O barramento de serviços possui em seu código-fonte a implementação de serviços, que utilizam módulos Erlang que possibilitam o acompanhamento dos recursos computacionais em que o barramento de serviços está funcionando.

Esses serviços não dispõem de elementos gráficos, eles fornecem informações a partir da requisição de um serviços, para execução desse serviço basta apenas utilizar a URL do serviço específico e uma ferramenta de testes de API, que possibilita a obtenção das informações requisitadas. As informações retornadas a partir do serviço por meio de uma requisição, podem ser visualizada na figura 4.2.

Figura 4.2: Serviço de consulta da utilização dos recursos de memória do Erlangms.

Monitoramento do Barramento de Serviços por *Software* Nativo (Erlang)

O barramento de serviços foi desenvolvido na linguagem Erlang, essa linguagem dispõe de componentes de monitoramento que permite o acompanhamento do sistema operacional.

O monitoramento pode apresentar informações sobre a utilização de memória durante o funcionamento e o número de processos em execução.

A execução é realizada pelo denominado *Observer*, que dispõe de elementos gráficos que possibilita observar as características dos sistemas desenvolvidos em Erlang [105] em execução, assim como o funcionamento do sistemas operacionais em que as aplicações Erlang estão instaladas.

O *Observer* tem como empecilho a perda de desempenho em seu funcionamento, pois disponibiliza muitos dados referentes aos seus processos de execução. A interface gráfica do *Observer* pode ser visualizada na figura 4.3.

Figura 4.3: Interface gráfica do *Observer*.

Monitoramento do Barramento de Serviços pelo Ems-Monitor)

Este trabalho inclui o Ems-Monitor como uma nova forma para monitorar o barramento de serviços, esse monitoramento difere dos demais pelo modo de atuação, onde o foco é monitorar o funcionamento dos recursos que o barramento de serviços provém, desde o recurso responsável pelo registro das informações de *LOG* passando pelos recursos que executam os processos de autenticação e autorização até o recurso responsável pelo recebimento e transmissão de requisições.

Para realizar o Ems-Monitor foi necessário a definição de uma abordagem que permitisse o acompanhamento do funcionamento dos recursos do barramento de serviços.

Nessa abordagem foi definido que as informações seriam geradas por meio do funcionamento do Erlangms com os seguintes passos:

- Gerar dados por meio de instrumentação;
- Coletar e armazenar os dados oriundos da instrumentação; e
- Transmitir os dados para realização do monitoramento.

Gerar Dados por meio de Instrumentação

O primeiro passo para monitoração foi gerar dados, para isso foi utilizada uma abordagem que possibilitou a implementação no código-fonte do barramento de serviços. A abordagem inclui a criação de uma estrutura de dados e a definição de contadores numéricos, esses contadores possuem implementação simples e buscam não impactar de forma onerosa o funcionamento do Erlangms.

Os contadores funcionam a partir da inclusão dos métodos no código-fonte e contabilizam as informações de forma incremental, a partir do momento que os recursos do barramento de serviços são executados. A abordagem utilizada pelo Ems-Monitor possibilitou o entendimento do funcionamento do barramento de serviços, e permitiu um acompanhamento completo, visto que soluções utilizadas pelo CPD acompanham o funcionamento do sistema operacional, em que o Erlangms está instalado ou o acompanhamento das informações de erros e regras de negócio de serviços e aplicações.

Coletar e Armazenar os Dados oriundos da Instrumentação

O segundo passo foi implantar uma funcionalidade, que possibilitasse coletar e armazenar as informações geradas durante a execução do barramento de serviços. Para realização da coleta foi necessário a inclusão de funcionalidades no código-fonte do Erlangms, essas funcionalidades trabalham de forma incremental a partir da execução do barramento de serviços, a cada instrução executada um valor numérico é adicionado à informação de monitoramento.

As informações coletadas são armazenadas e gerenciadas pelo Ems-Monitor, que é responsável por gerenciar a informação referente ao monitoramento. O Ems-Monitor possui algumas funções exclusivas, uma dessas funções é preparar a informação para transmissão, vale lembrar que armazenar esses dados garante integridade da informação do monitoramento.

Transmitir os Dados para realização do Monitoramento

Por fim, outro passo é a transmissão dos dados de monitoramento, a transmissão dos dados é realizada após a coleta e armazenamento, o Ems-Monitor tem como responsabilidade transmitir as informações do monitoramento. Nesse trabalho a transmissão dos dados é encaminhada à uma ferramenta de monitoramento onde é possível visualizar graficamente as informações, ou a emissão de notificação caso haja alguma definição para realização do monitoramento.

Para transmitir os dados de monitoramento, foi utilizada uma abordagem que permitiu a realização da transmissão de forma padronizada por meio de um protocolo de

comunicação. A utilização de um protocolo possibilitou a realização da transmissão dos dados de monitoramento, assim como a integração com ferramentas de monitoramento que utilizam o protocolo SNMP como meio de comunicação.

A interação dos componentes do Ems-Monitor, via troca de mensagens, pode ser visualizada no diagrama de sequência, representado na figura 4.4, que demonstra o fluxo de criação dos dados, coleta e armazenamento dos dados e a transmissão dessas informações para a ferramenta de monitoramento.

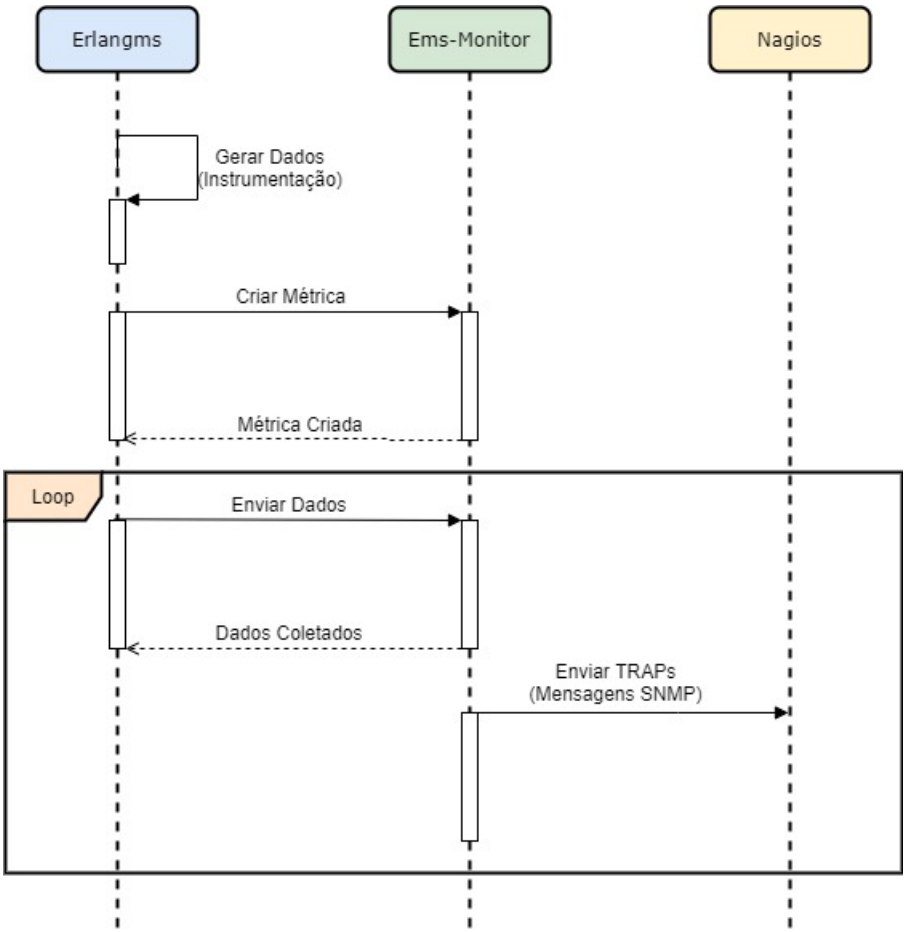

Figura 4.4: Fluxo de monitoramento pelo Ems-monitor.

A partir da definição da estrutura de monitoramento foi realizada a geração de dados, a coleta, o armazenamento e a transmissão dos dados para monitorar os recursos do barramento. O CPD possui e utiliza abordagens de monitoramento para sistemas e ativos de rede da UnB. As aplicações e ativos de rede já são monitorados em uma plataforma de monitoramento mantida e gerenciada pelo CPD.

Neste trabalho Ems-Monitor foi utilizado como solução para realização da integração por meio de um protocolo de comunicação para monitoramento dos serviços, sistemas e

ativos de rede na mesma plataforma mantida pelo CPD, e que permitisse apresentar o funcionamento dessas aplicações ou em casos eventuais notificar sobre possíveis falhas ou problemas.

Diante desse cenário optou-se por definir um meio de comunicação onde a ferramenta de monitoramento, que é mantida e gerenciada pelo CPD da UnB pudesse receber as informações, para realização do monitoramento do barramento de serviços, por meio de um protocolo, neste trabalho o protocolo utilizado para realização da comunicação, e permitir a integração das ferramentas foi o protocolo SNMP.

A escolha do protocolo para realização do monitoramento busca um meio para tentar garantir a implantação e integração das aplicações com baixo acoplamento, onde a importância é preservar a uniformidade na comunicação das aplicações por meio de um protocolo, principalmente quando houver alguma mudança de ferramenta de monitoramento.

No mercado as principais ferramentas de monitoramento se comunicam e permitem a transmissão de dados e informações por meio do protocolo SNMP, independentemente das ferramentas o serviços de monitoramento seria facilmente plugado, e configurado em ferramentas de monitoramento para utilização do protocolo SNMP como meio de comunicação.

4.2.1 Instrumentação de Código

Como descrito na seção 2.1.5 o barramento de serviços dispõe de recursos que possibilitam a implementação da instrumentação em seu código. O barramento de serviços possui um modelo de dados implementado, esse modelo é representado por uma tabela dispostas de duas colunas, uma que identifica, ou seja, a chave e a outra com o valor do tipo inteiro referente à chave da mesma linha, alguns exemplos de chaves (identificadores) para criação das métricas, poderão ser visualizados na figura 4.5.

```
ems_auth_user_public_success
ems_auth_user_oauth2_denied
ems_auth_user_basic_success
ems_auth_user_oauth2_success
ems_auth_user_public_success
ems_oauth2_grant_type_password
ems_oauth2_client_credentials_denied
ems_oauth2_grant_type_token
```

Figura 4.5: Exemplo de identificadores das métricas no Erlangms.

Para armazenar esses registros o barramento de serviços possui uma outra tabela que possibilita o registro dos contadores, com identificação da hora e data, serviço e URL, para fins de histórico, o barramento de serviços possui um serviço que retorna o histórico das estatísticas armazenadas, o resultado deste serviço poderá ser visualizado na figura 4.6.

Figura 4.6: Serviço que recupera informações históricas dos contadores do Erlangms.

Para realização dos registros dos contadores, o barramento de serviços dispõe de funções que podem ser facilmente distribuídas pelo código fonte, especificamente em funções dos módulos do barramento de serviços, onde por meio da chave criada é possível incrementar o decrementar valores dos contadores cridos no barramento, o código da função pode ser visualizado na figura 4.7.

```
init_counter(Name, Value) ->
    {atomic, ok}=mnesia:transaction(fun()->
                mnesia:write(#counter{key=Name, value=Value})end),
    ok.
inc_counter(Name)->mnesia:dirty_update_counter(counter, Name, 1).
dec_counter(Name)->mnesia:dirty_update_counter(counter, Name, -1).
current_counter(Name)->mnesia:dirty_update_counter(counter, Name, 0).
counter(Name, Inc)->mnesia:dirty_update_counter(counter, Name, Inc).
```

Figura 4.7: Funções dos contadores para instrumentação no Erlangms.

A partir da criação dos contadores, o incremento e o decremento de seus valores é possível realizar a criação das métricas. A criação das métricas é feita com utilização da aplicação Exometer, onde é possível criar métricas, e definir o tipo de valor da métrica, atualizar o valor da métrica, excluir as métricas criadas, recuperar as métricas criadas e reportar métricas, no caso do *report* este será feito por meio do módulo SNMP.

As métricas criadas na aplicação podem ser facilmente manipuladas e reportadas, graças aos métodos criados com objetivos específicos, os principais métodos utilizados nesse projeto são:

- Criar métricas.
- Excluir métricas.
- Definir os valores de métricas.
- Recuperar os valores de métricas.
- Criar *report* de uma métrica em SNMP.

A implementação desses métodos poderão ser visualizados na figura 4.8, a utilização desses métodos é bem simples e permitem a realização da instrumentação de maneira fácil, o que proporcionam exequibilidade na personalização das informações a serem enviadas para monitoramento, sem comprometer a eficiência do funcionamento das aplicações.

```
% Cria uma Métrica.
exometer:new( Name , Type ).

% Deleta uma Métrica.
exometer:delete( Name ).

% Atualiza o valor de uma Métrica.
exometer:update( Name , Value ).

% Recupera o valor de uma Métrica.
exometer:get_value( Name ).

% Cria um report de uma métrica em SNMP.
exometer_report:add_reporter( exometer_report_snmp, [] ).
```

Figura 4.8: Os métodos do Exometer utilizados no Erlangms.

Desempenho

Durante a realização da fundamentação teórica e do mapeamento sistemático, percebeu-se um grande número de trabalhos comentando, descrevendo e identificando o desempenho como um item preocupante, e que deveria ser mais analisado em relação às informações geradas para monitoramento, a transmissão dos dados de monitoramento e o funcionamento do monitoramento.

Esse item também foi analisado no projeto de modo a garantir que as informações criadas e transmitidas não comprometessem e nem onerassem a integração entre o barramento de serviços, e a ferramenta de monitoramento, e observando a grande quantidade de informações que podem ser geradas durante o funcionamento do barramento de serviços.

A preocupação com o desempenho foi minimizada após a conclusão da implementação da instrumentação dos dados estatísticos para monitoramento, que utilizaram um modelo de dados simples, com poucos atributos que permitem após a criação da variável do contador executar facilmente a modificação dos valores, com isso as métricas são definidas, criadas e utilizadas.

Para que não tenha perda no desempenho, neste projeto foram implementadas funções que permitem a atualização dos contadores, e a atualização das métricas ao mesmo tempo, onde primeiro executa a atualização do contador e em seguida a atualização da métrica.

4.2.2 Execução do Ems-Monitor

A execução do Ems-Monitor é inciada com a integração das aplicações, ou seja quando o barramento de serviços está em funcionamento, o agente de monitoramento está ativo, assim como a ferramenta de monitoramento está ligada e configurada em modo passivo para receber as informações que são transmitidas de modo que permiti a realização do monitoramento.

Para representar a execução do monitoramento foi criado um *design* que pode ser visualizado na figura 4.9.

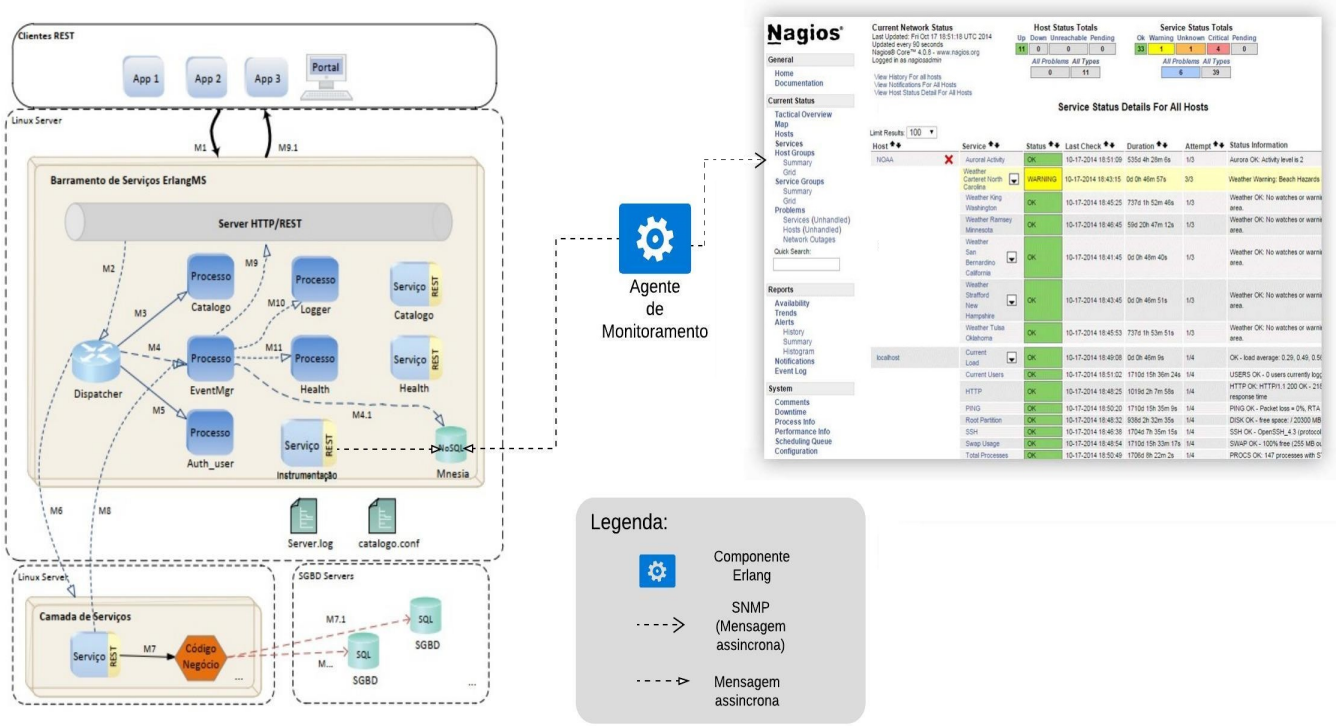

Figura 4.9: *Design* da integração das ferramentas para monitoramento.

O monitoramento é realizado por meio de dados gerados por serviços provenientes do barramento de serviços, esses dados possuem identificações e valores incrementais de contagem.

Em seu funcionamento o Erlangms pode incrementar ou decrementar informações de contagem dependendo do serviço utilizado, normalmente o valor é incrementado, quando o valor à ser monitorado é coletado, o mesmo é enviado ao agente de monitoramento, o agente de monitoramento tem a responsabilidade de realizar o envio dessa informação

realizando a comunicação com a ferramenta de monitoramento por meio do protocolo SNMP.

A informação transmitida poderá ser visualizada de modo personalizado e configurado na ferramenta de monitoramento, que dependendo da configuração poderá apresentar itens de serviços cadastrados com os status, *"Ok"* para serviços que estejam funcionando normalmente, *"Warning"* para serviços que estejam apresentando algo em incomum durante seu funcionamento e *"Critical"* para serviços que estejam apresentando anormalidade ou realmente estejam indisponíveis.

4.2.3 Execução do Agente de Monitoramento

Para realização da integração do barramento de serviços com a ferramenta de monitoramento por meio do protocolo SNMP, é utilizado um agente de monitoramento que gerencia a comunicação e transmissão dos dados que são monitorados.

O Ems-Monitor dispõe da utilização de um agente para coleta e transmissão desses dados, o agente é originário da aplicação Exometer que é um pacote que permite a instrumentação de aplicações desenvolvidas na linguagem de codificação Erlang, possibilitando que alguns dados do sistema sejam exportados, para um grande e variado número de ferramentas de monitoramento disponíveis no mercado[9], nesse projeto diante de um dos itens dos objetivos específicos descritos na seção 1.3 optou-se por utilizar o módulo que permite realizar a transmissão dos dados por meio do protocolo SNMP.

O Exometer cumpre os requisitos definidos na RFC 1157, que descreve sobre os procedimentos arquiteturais de implementação e implantação do protocolo SNMP.

Os pontos fortes do Exometer que pautaram a escolha e utilização deste *software* são; a criação das MIBs que são geradas dinamicamente em tempo de execução, vários tipos de criações e exportações de *reports*, vários tipos de *data points* que permitem selecionar o tipo de valor de uma determinada métrica, a fácil criação e utilização de métricas e o modo acessível para configuração do Gerente e Agente SNMP, onde simplesmente você pode registrar em um dos arquivos de configuração o servidor(aplicação de monitoramento), que receberá as informações do monitoramento por meio de *TRAPs* enviados pelo agente SNMP.

O Exometer foi a aplicação que permitiu a integração das aplicações de um modo dinâmico, visto que a estrutura de um arquivo MIB normalmente já vem configurada com informações específicas e estáticas o que impossibilitaria ou reduziria bastante o escopo de serviços, que poderiam ser utilizados para realizar o monitoramento de serviços do barramento.

A estrutura base do arquivo MIB utilizado pelo Exometer poderá ser visualizada na figura 4.10.

```
    EXOMETER-METRICS-MIB DEFINITIONS ::= BEGIN
IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE, NOTIFICATION-TYPE, Counter32, Counter64,
    Gauge32, Integer32, snmpModules, experimental FROM SNMPv2-SMI
    MODULE-COMPLIANCE, OBJECT-GROUP, NOTIFICATION-GROUP FROM SNMPv2-CONF;
exometerMetricsMIB MODULE-IDENTITY
        LAST-UPDATED "201401190525Z"
        ORGANIZATION "Feuerlabs"
        CONTACT-INFO "TODO"
        DESCRIPTION
                "This MIB module is used for exposing dynamic exometer metrics."
        REVISION  "201401190525Z"
        DESCRIPTION
                "The initial version"
        ::= { snmpModules 1 }
exometerMetrics OBJECT IDENTIFIER ::= { experimental 7 }

-- CONTENT START

-- CONTENT END

END
```

Figura 4.10: Estrutura base, arquivo MIB do Exometer.

As configurações do arquivo que gerencia o servidor de destino para receber as informações de monitoramento é apresentada na figura 4.11.

```
{"exometerManager",
transportDomainUdpIpv4,
[127,0,0,1],
5000,
1500,
3,
"std_inform",
"exometerManager",
"exometer engine",
[],
2048}.
```

Figura 4.11: Configuração do servidor de destino, para envio de *TRAPs*.

4.3 Funcionamento do Protocolo SNMP

O SNMP é o protocolo utilizado para monitoramento de dispositivos conectados em rede. O CPD da UnB para monitoramento dos ativos de rede utiliza-o por meio de ferramentas que recebem informações desse protocolo, no entanto as ferramentas e informações fornecidas para monitoramento servem apenas para controle da infraestrutura de rede e servidores de aplicação, não gerenciando, por exemplo, o funcionamento dos sistemas, serviços e também pode se dizer a ausência de verificação da saúde das aplicações.

Este trabalho implementa e implanta soluções capazes de fornecer e subsidiar recursos que possibilitem a monitoração e a integração com a ferramenta de monitoramento do CPD da UnB que vem durante alguns anos adotando o Nagios®, como a ferramenta de monitoramento dos seus ativos de rede, o SNMP tem como característica a singularidade na comunicação o que possibilita a escolha de outras ferramentas, serviços ou aplicações que conversem em SNMP.

O funcionamento do monitoramento por meio do protocolo SNMP neste trabalho é realizado pelo envio de *TRAPS*, a partir da coleta de informações geradas pelo barramento de serviços e enviadas pelo agente, para que o agente funcione é necessária a configuração de um gerente, que funciona como representação do cliente monitorado e o agente funciona como responsável por enviar as informações desse cliente, na figura 4.12 é apresentada a configuração do agente na aplicação e na figura 4.13 é exposta a configuração do gerente.

```
{intAgentUDPPort, 4000}.
{intAgentIpAddress, [127,0,0,1]}.
{snmpEngineID, "exometer_engine"}.
{snmpEngineMaxMessageSize, 484}.
```

Figura 4.12: Arquivo de configuração do agente SNMP.

```
{port, 5000}.
{address, [127,0,0,1]}.
{engine_id, "manager's_engine"}.
{max_message_size, 484}.
```

Figura 4.13: Arquivo de configuração do gerente SNMP.

O *TRAP* é um tipo de alerta ou evento, que pode ser transmitido pelo protocolo. Na figura 4.14 é apresentada a informações de um *TRAP* SNMP com a verbosidade *debug*.

Durante a coleta das informações em tempo de execução, e o envio de *TRAPS* pelo agente para ferramenta do monitoramento é necessário à instalação e configuração para de alguns *plugins*, visto que o Nagios® trabalha de forma em que é possível a realização do monitoramento ativo e passivo, como informado anteriormente este trabalho aplica o monitoramento passivo, diante desse cenário os *plugins* instalados e configurados são eles:

- SNMP Trap Translator (SNMPTT) que é um *plugin* que funciona com um tradutor de *TRAP* SNMP com diversas opções de saída para utilização no Nagios Core® e o Net-SNMP[106].
- SNMPTRAPD que é uma aplicação que funciona com o recebimento dos *TRAP* que foram traduzidos pelo SNMPTT [107].

Com a instalação e configuração realizada do SNMPTT e SNMPTRAPD foi possível realizar a interceptação das informações enviadas pelo agente SNMP, cada item com sua especificidade como por exemplo; no SNMPTT é necessário a configuração das variáveis dos OIDs criadas e definidas nas MIBs do agente, já na configuração SNMPTRAPD é necessário definir quais os OIDs serão enviados à ferramenta de monitoramento, e quais o tipo de informações serão enviadas, poderão ser enviadas informações dos tipos *"Ok"*,

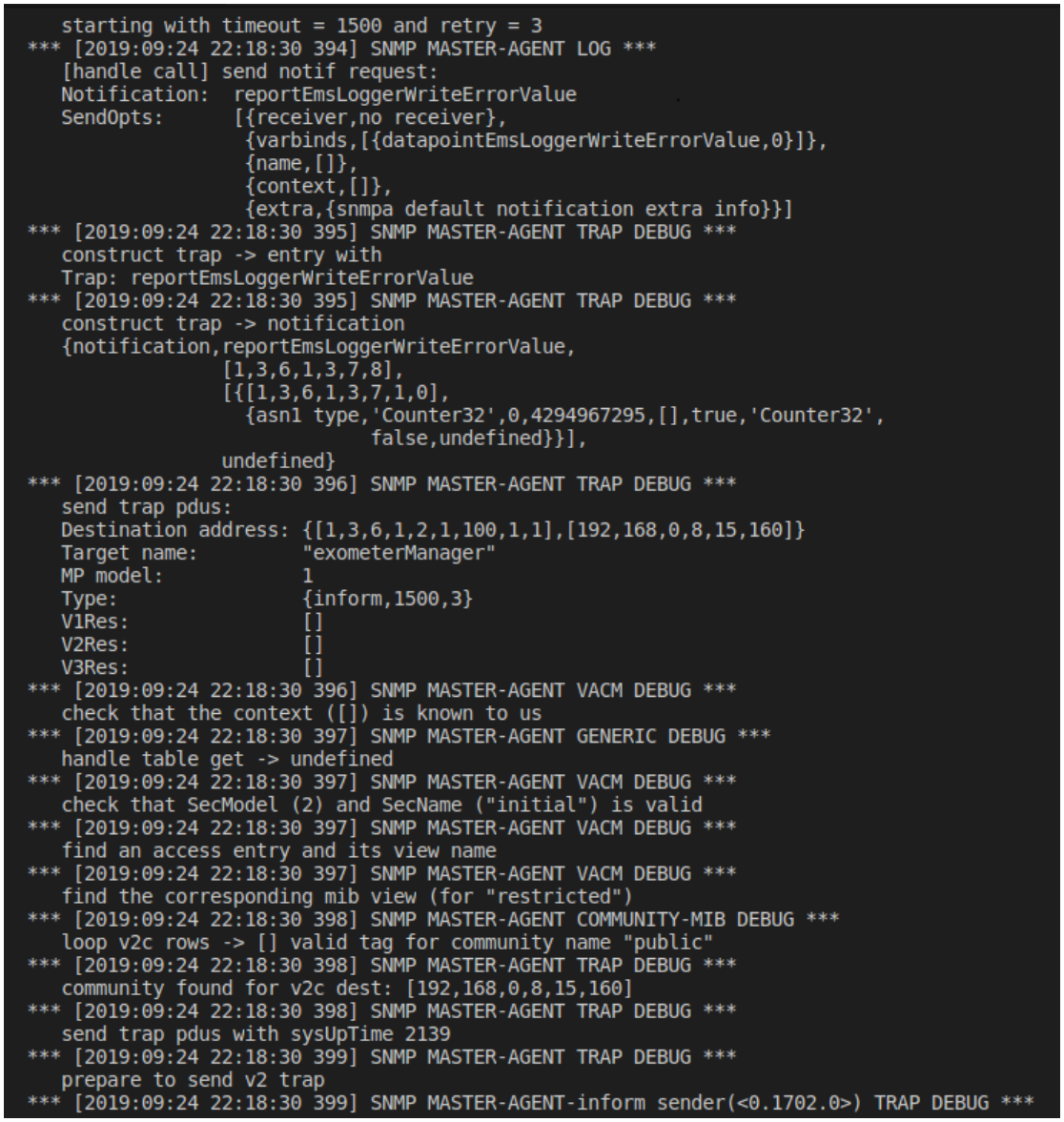

Figura 4.14: Informações do *TRAP*.

"Warning" ,*"Critical"* dependendo do OID e da definição que pode ser facilmente configurada. Desse modo é realizada a integração das ferramentas para execução do monitoramento, por meio do protocolo SNMP a comunicação entre o barramento de serviços e o Nagios® é praticada.

4.4 Definição das Métricas de Monitoramento

Com a elaboração da integração entre o barramento de serviços e o Nagios®, foi feita a pesquisa na literatura específica relacionada ao tema métricas de monitoramento para

auxiliar a definição do que deve ser monitorado.

Após realização da pesquisa, e durante a execução da configuração da ferramenta de monitoramento percebeu-se a necessidade de seguir a estrutura de configuração determinada na ferramenta de monitoramento, ao assumir essa estrutura foi possível detectar que o monitoramento deveria ser executado de um novo modo, implicando na mudança da estratégia de definição para execução do monitoramento dos recursos do barramento de serviços.

Como o monitoramento executado pelo Ems-Monitor é realizado em modo passivo é preciso informar na configuração da ferramenta de monitoramento quais as informações serão recebidas, ou seja, mesmo o agente SNMP do barramento de serviços criando os OIDs dinamicamente, é necessário que estes OIDs estejam também registrados na ferramenta de monitoramento.

Nesse momento não foi possível a realização da implementação de uma solução para automatizar a configuração dinâmica dos OIDs na ferramenta de monitoramento, mas em um outro momento oportuno essa automatização poderá ser implementada.

Com a restrição imposta pela ferramenta de monitoramento, foi necessário a realização de um estudo no código fonte do barramento de serviços Erlangms, afim de obter informações sobre o que monitorar, a análise do barramento de serviços, foi feita com passos que permitiram analisar o código-fonte, para o entendimento do funcionamento dos recursos do barramento de serviços, assim como a realização do mapeamento dos principais recursos, módulos e funções.

O primeiro passo realizado foi a elaboração do *design* dos recursos do barramento de serviços, este *design* pode ser visualizado na figura 4.15. No segundo passo foi possível identificar, e escrever sobre cada recurso do barramento de serviços, a descrição dos recursos poderão ser visualizadas a seguir, são eles:

- ***Loaders:*** Responsáveis por carregar as configurações para funcionamento do barramento, desde os catálogos de serviços, e passando pela identificação dos clientes até as permissões de aceso dos usuários.
- ***HTTP(Rest Server):*** Responsável por realizar o funcionamento do barramento como um servidor REST.
- ***LDAP:*** Responsável por prover, gerenciar e autorizar acessos aos serviços por meio do barramento e funcionando como um servidor REST.
- ***ODBC:*** Responsável pela realização da comunicação entre os serviços do barramento com SGBDs de mercado, atualmente o barramento por meio de seus serviços realiza consultas no Postgres® e no SQL Server®.

- ***Oauth2:*** Responsável pela autenticação e autorização de acesso aos serviços fornecidos pelo barramento.
- ***Dispatcher:*** Responsável pelo encaminhamento e organização das chamadas/requisições aos serviços que são executados pelo barramento.
- ***Daemons:*** Responsável por gerenciar os arquivos(pid) do barramento para execução dos serviços de forma única e confiável, sem a interrupção de seu funcionamento.
- ***Catalog:*** Responsável pela configuração, definição e funcionamento dos serviços providos pelo barramento.

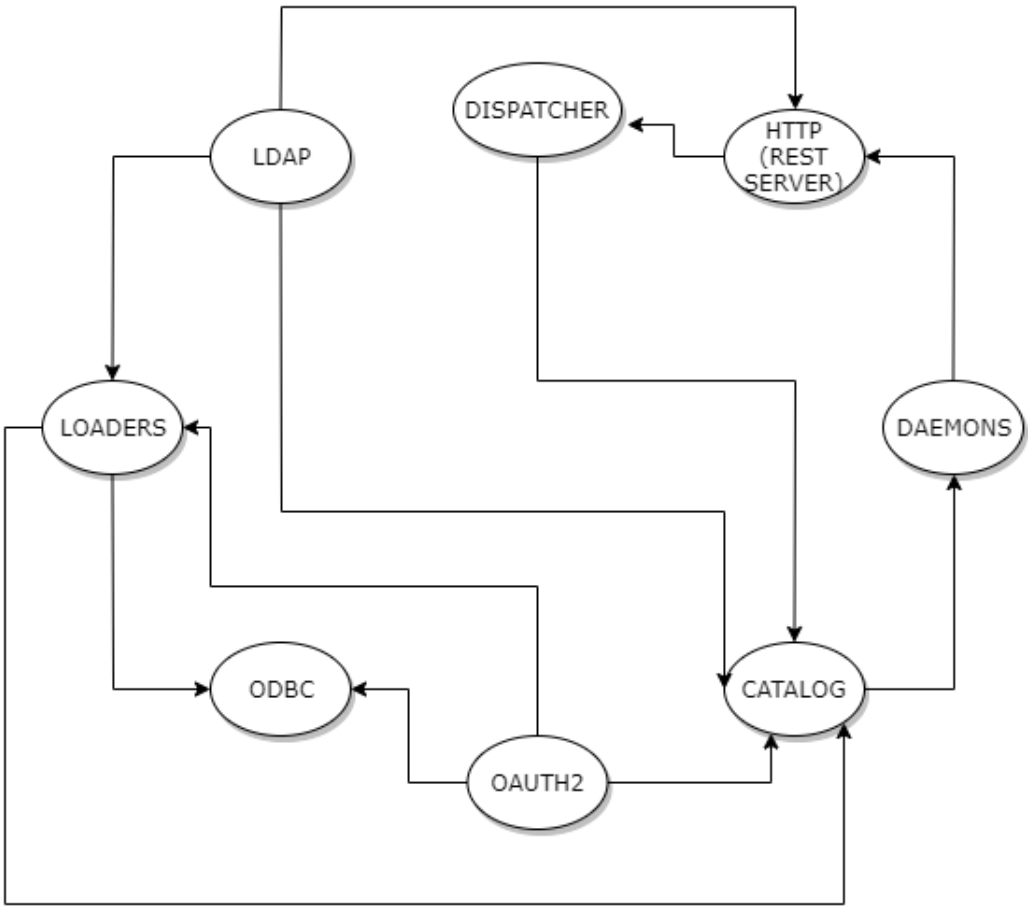

Figura 4.15: Mapeamento dos recursos do barramento de serviços.

Um outro passo importante foi a análise e identificação de contadores disponíveis no barramento de serviços, a partir da estrutura de arquivos e diretórios, que possibilitou mapear informações de *paths*, módulos e funções.

Foram encontrados cento e dez contadores registrados atualmente no barramento de serviços, o que não limita a implementação de mais contadores, eles estão organizados por módulos, como por exemplo, o de autenticação.

Após a conclusão da investigação, foi feita a definição das métricas por meio das informações que foram registradas durante a realização da identificação dos contadores, vale salientar que as métricas foram definidas de uma forma empírica por conta da natureza da configuração e funcionamento da ferramenta de monitoramento escolhida para execução deste trabalho, o que não significa um acoplamento alto entre as aplicações, mas sim uma forma melhor para analisar os dados enviados pelo agente SNMP e recebidos pela ferramenta de monitoramento.

Diante do cenário as cinco métricas foram levantadas e discutidas com colaboradores que atuam diretamente com desenvolvimento de *softwares* e colaboradores da equipe que atuam diretamente com o monitoramento dos serviços do CPD da UnB, as métricas levantadas são:

- Registros do *LOG* do barramento de serviços
- Registros de informações de Acesso (OAuth2)
- Registros do *Dispatcher*
- Registros do HTTP *(REST Server)*
- Registros de informações do LDAP

As métricas levantadas e identificadas foram definidas por apresentarem situações que poderiam contribuir diretamente na realização do monitoramento do barramento de serviços.

Ela é útil para transmissão dos dados de execução do recurso de *LOG*, para que todo funcionamento do barramento de serviços possa ser registrado, e se por algum motivo o recursos esteja registrando uma grande quantidade de falhas ou até mesmo uma exceção que o ocasione o não registro do *LOG* do barramento de serviços, e a ferramenta de monitoramento possa emitir notificações de alerta.

Esse monitoramento pode auxiliar no funcionamento do trabalho realizado por [104] podendo respaldar alguma eventual falha no registro de *LOG* do barramento de serviços, visto que o monitoramento do trabalho citado, é realizado por meio de informações registradas e coletadas do arquivo de *LOG* do barramento de serviços.

Outra métrica levantada para realização do monitoramento, é a que trata sobre os registros de informações de acesso por meio do protocolo OAuth2.

Com realização da transmissão dos dados dessa métrica busca-se alcançar o acompanhamento dos recursos de autenticação do barramento de serviços, que provém a autenticação e autorização de vários serviços e aplicações com informações provenientes de diversas bases de dados, centralizando a autenticação, o que sinaliza a necessidade de um acompanhamento em especial, visto que, uma possível falha do recurso, indisponibilidade ou um aumento expressivo de utilização do recurso possa impedir o funcionamento, impossibilitando a autenticação nas aplicações.

O registro do *Dispatcher* também foi incluído como uma das cinco métricas levantadas para realização do monitoramento do barramento de serviços.

A definição como métrica para realização do monitoramento é de suma importância para acompanhamento do barramento de serviços, pois *Dispatcher* tem como função enviar e receber as requisições de todos módulos do barramento de serviços, isso demonstra que o acompanhamento do funcionamento do *Dispatcher* é indispensável.

Seguindo o levantamento das métricas, o registros do HTTP *(REST Server)* também foi definido como métrica devido a execução dos operadores (GET, POST, PUT e DELETE), sua representação que no caso utiliza o formato JSON e recursos (URLs) que fazem parte dos elementos mais utilizados do barramento de serviços e necessitam de acompanhamento.

Por fim, o levantamento da métrica para obtenção dos registros de informações do LDAP foi definida para que pudesse acompanhar o funcionamento deste recurso, pois barramento de serviços realiza em seu funcionamento a autenticação de aplicações de terceiros implantadas no CPD da UnB, que utilizam o protocolo LDAP para realização com isso espera-se garantir o funcionamento adequado do barramento de serviços por meio das métricas que foram levantadas.

O barramento de serviços vem aumentando a disponibilização de serviços, vale salientar sobre a importância desse acompanhamento, visto que, com aumento dos recursos disponibilizados pelo barramento de serviços é muito grande a chance de algum falhar ou ficar indisponível e que não seja percebido ou não seja tratado imediatamente.

Após análise das métricas levantadas, foi necessário realização da seleção e escolha das métricas à serem implantadas para efetivação da integração entre o barramento de serviços e a ferramenta de monitoramento por meio do agente SNMP.

A seleção e escolha foram necessárias para que se pudesse experimentar o funcionamento da integração, pois antes da realização do monitoramento completo do barramento de serviços optou-se por implementar algumas funções que pudessem coletar informações das métricas levantadas.

Das cinco métricas, duas foram descartas nesse momento, a de Registros do HTTP *(REST Server)* a de Registros de informações do LDAP, por já possuírem seus serviços mape-

ados para o acompanhamento no CPD da UnB, as ou três métricas escolhidas, foram e implementadas, são elas:

- Registros do *LOG* do barramento de serviços;
- Registros de informações de Acesso (OAuth2);
- Registros do *Dispatcher*.

A escolha destas métricas foi definida, devido a importância do acompanhamento dos módulos mais utilizados no barramento de serviços, os registros do *LOG* do barramento de serviços, que estão preparados para exportar e apresentar uma gama de informações podendo não facilitar a identificação de uma falha ou indisponibilidade dos serviços, por isso a configuração para registro das informações dos registros de *LOG* na ferramenta de monitoramento foi escolhido.

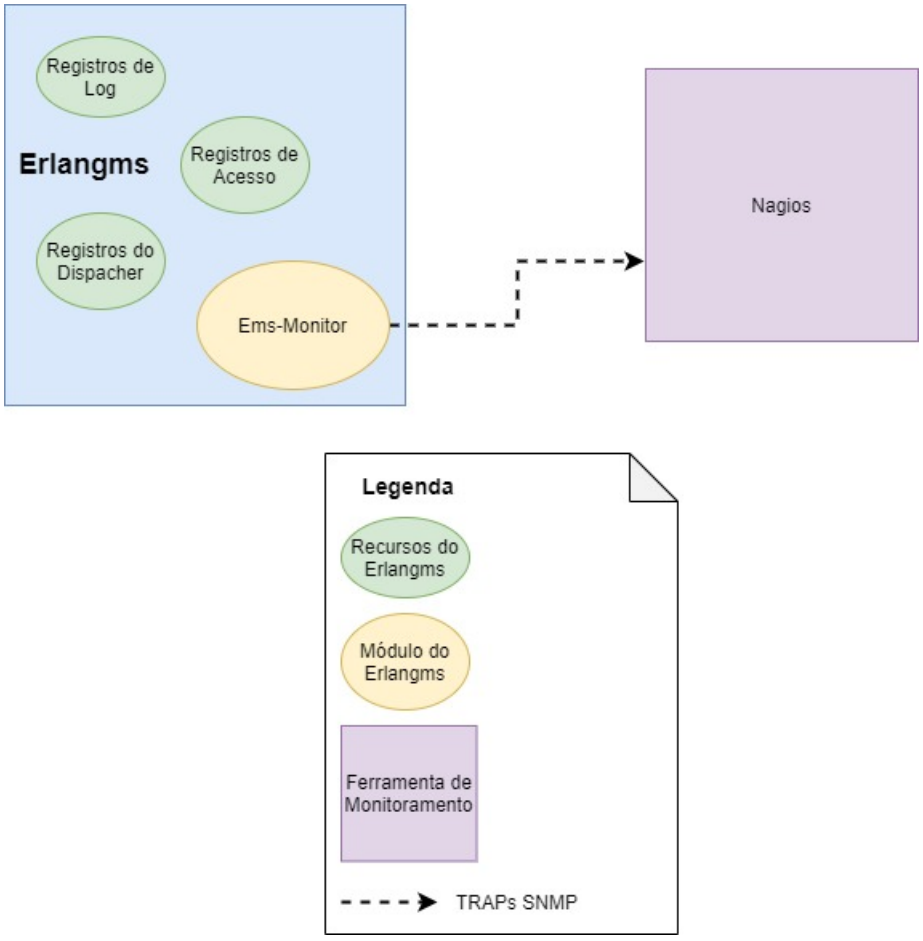

Figura 4.16: Identificação dos recursos do barramento de serviços monitorados.

O motivo para escolha do registro das informações de autenticação promovidas por meio do protocolo OAuth2, foi pelo CPD da UnB, utilizá-lo como meio institucional

de autenticação, trabalho desenvolvido pelo autor Ribeiro [108], e principalmente, pela quantidade crescente de disponibilização de serviços e recursos, que devido ao crescimento dificultou o acompanhamento da autenticação.

Por esse motivo buscou-se obter uma situação real de funcionamento do procedimento de autenticação e autorização, assim como a identificação de ataques, como por exemplo o ataque de negação de serviços por meio do monitoramento.

O funcionamento do *Dispatcher* é essencial para o barramento de serviços, por ele passam todos os serviços executados pelo barramento de serviços, o *Dispatcher* controla e gerencia o recebimento e envio de requisições que são executadas pelo barramento de serviços.

Por esse motivo acompanhar o seu funcionamento é muito importante, ainda mais com a possibilidade da emissão de notificações por meio de seu funcionamento normal e quando houver falha ou indisponibilidade, vale enfatizar que seleções e escolhas das métricas foram realizadas por conta do modelo para realização de monitoramento que a ferramenta de monitoramento selecionada propõe.

4.5 Síntese do Capítulo

Este capítulo apresentou o funcionamento do Ems-Monitor, demonstrando a execução do monitoramento por meio da integração entre o barramento de serviços e a ferramenta de monitoramento com utilização do protocolo de monitoramento SNMP, assim como a execução do agente SNMP e o funcionamento do protocolo SNMP no projeto. Também foram apresentadas, definidas e escolhidas às métricas mais adequadas ao funcionamento das ferramentas e a integração das aplicações para realização do monitoramento, por fim tem-se a exposição da solução da coleta de dados para monitoramento por meio da instrumentação sem comprometer o desempenho durante funcionamento das aplicações e realização do monitoramento.

Capítulo 5

Análise do Ems-Monitor

Com realização da implementação, implantação e integração das aplicações para execução do monitoramento, é necessário a divulgação da análise e dos resultados do trabalho executado. Este Capítulo apresenta a análise realizada após a execução do Ems-Monitor.

5.1 Análise das Aplicações

Com propósito de avaliar a execução do Ems-Monitor, foram analisados alguns requisitos das plataformas utilizadas para monitorar o Erlangms, por meio de uma ferramenta de monitoramento. Inicialmente os requisitos dos recursos computacionais das aplicações foram avaliados, tanto o requisito do barramento de serviços quanto o da ferramenta de monitoramento.

O agente de monitoramento utiliza os mesmos recursos computacionais do *software* onde ele foi implantado, nesse caso o agente de monitoramento é representado pelo *software* Exometer e está implantado no barramento de serviços Erlangms, os requisitos de instalação do barramento de serviços poderão ser visualizados em [109], assim como os requisitos de configuração do Exometer poderão ser visualizados em [110].

A ferramenta de monitoramento também dispõe de requisitos básicos para seu funcionamento, os requisitos do Nagios Core® na versão quatro poderão ser visualizados no portal do Nagios® [111], em ambos os caso os requisitos computacionais não impediram a execução do projeto e consequentemente foram executados sem gerar grandes esforços.

Dispondo da configuração das aplicações foi necessário realizar a avaliação do funcionamento do monitoramento, por meio do conjunto de protocolos TCP/IP, onde sabe-se que o protocolo SNMP é um protocolo da *Internet*, que por sua vez o protocolo SNMP para garantir a transmissão dos dados de monitoramento utilizada por padrão o protocolo UDP especificamente nas portas 161 e 162.

Na porta 161 estão associadas à todas as mensagens enviadas ao protocolo SNMP, e a porta 162 é utilizada para realização das interceptações de todas as mensagens, que transmitirem *TRAPs*, lembrando que de acordo com as especificações da RFC 1157, essas mensagens não serão aceitas caso o tamanho exceda 484 octetos[112].

Dando continuidade na avaliação, percebeu-se que as configurações de agente e gerente SNMP tem o fator presencial de muita importância, nesse trabalho o agente é quem transmite as informações pela porta 4000 e o gerente pela porta 5000, durante a execução do trabalho as configurações das portas ocasionaram um pequeno problema.

Durante o funcionamento do monitoramento no ambiente de desenvolvimento, mesmo que as aplicações possuíssem IPs distintos, as portas geradas com identificações diferentes não funcionariam, pois as portas devem possuir as mesmas identificações tanto para transmitir quanto para interceptar, ou seja, se qualquer transmissão que for realizada pela porta 162 essa transmissão deverá ser interceptada também na porta 162.

Quando as aplicações são instaladas na mesma máquina, a primeira aplicação que iniciar será a detentora da porta, impossibilitando assim o funcionamento das aplicações em um único servidor, validando assim que cada *host* necessita ter a sua configuração individualizada e preparada para que o agente seja responsável pela transmissão dos *TRAPs*.

Uma importante avaliação para realização do monitoramento do barramento de serviços, foi a avaliação feita sobre a construção do arquivo MIB. Na leitura de trabalhos técnicos e verificações em sites de empresas especializadas no ramo de monitoramento o arquivo MIB normalmente é estático, pois contempla especificamente os OIDs para realização do monitoramento dessas aplicações ou ativos de rede.

Neste trabalho a instrumentação provida pelo barramento de serviços gera informações a partir de sua execução, sendo necessário um dinamismo na criação de objetos, o Exometer com funções bem definidas é de fácil compreensão, e correspondeu às expectativas geradas a partir de sua documentação, contribuindo bastante na criação e atualização do arquivo MIB, na criação de métricas, atualização de valores e no *report* dos dados coletados.

A ferramenta de monitoramento utilizada para execução do trabalho demonstrou pontos em que o dinamismo alcançado pela aplicação do agente de monitoramento não poderia ser utilizada de imediato. Após uma breve análise da ferramenta de monitoramento, percebeu-se que ela funciona a partir de módulos (*plugins*) complementares, dependendo do tipo de monitoramento a ser realizado.

Nesse caso o monitoramento que foi realizado foi em modo passivo, para isso foram realizadas as configurações dos *plugins* SNMPTT e o SNMPTRAPD que necessitam dos OIDs mapeados, ou seja, para que a comunicação e transmissão dos dados do monitoramento sejam tratados como confiáveis é necessário informar de qual *host* e quais objetos

terão as mensagens interceptadas.

Esse foi um ponto que não era esperado no projeto, mas que foi de grande utilidade para entender e auxiliar na definição de como realizar o monitoramento em modo passivo.

Por fim, mediante a realização da avaliação e utilização das ferramentas podemos confirmar a partir do estudo o experimento realizado, a possibilidade da realização do monitoramento dos recursos do barramento de serviços por meio de uma ferramenta de monitoramento com utilização do protocolo SNMP.

de acordo com realização do experimento pode-se comprovar que poderá ser utilizada qualquer ferramenta de monitoramento que permita a integração e interceptação de *TRAPs* SNMP, a implantação do agente SNMP permite incluir a funcionalidade com uma das principais características do barramento de serviços Erlangms.

5.2 Resultados

A partir da realização do experimento que possibilitou a integração das aplicações para execução do monitoramento dos recursos do barramento, no trabalho foi realizado a integração com utilização do protocolo SNMP para elaborar a comunicação com entre o barramento de serviços e a ferramenta de monitoramento.

Inicialmente um dos pontos que foi muito importante para contemplar o monitoramento e se demonstrou como resultado satisfatório, foi a criação dinâmica do arquivo MIB, o arquivo MIB com estrutura inicial poderá ser visualizado no anexo I, e o arquivo MIB com os OIDs criados durante a execução do barramento de serviços pode ser visualizado no anexo II.

O dinamismo para criação e atualização desse arquivo possibilita a coleta de qualquer informação ou registro mapeado que é gerado a partir da instrumentação do barramento de serviços, criando objetos identificadores para o envio de *TRAPs*, isso porque as aplicações mais utilizadas no mercado já possuem arquivos MIB definidos estaticamente já apontando o que deve ser monitorado, vale ressaltar que tanto na criação quanto na atualização do arquivo criado o Erlangms não demonstrou ter perda no desempenho do se funcionamento.

Outro ponto importante foi a identificação da estrutura para realização da comunicação via integração, que exige a utilização de *plugins*, fornecido pela ferramenta de monitoramento, para organização e identificação das aplicações e informações a serem monitoradas.

As aplicações em SNMP possuem uma gama de componentes que auxiliam na utilização do monitoramento por meio do protocolo SNMP, neste trabalho o componente executado foi o SNMPTT, no SNMPTT existe um módulo denominado SNMPTTCONVERTMIB que tem como finalidade converter(traduzir) o arquivo MIB em arquivo de

configuração para identificação e interceptação de *TRAPs*, para que esses possam ser interpretados pelo *plugin* SNMPTRAPD, assim como disponibilizado à ferramenta de monitoramento.

Neste trabalho não foi implementada uma solução para automatizar o processo conversão, pois como as métricas que seriam monitoradas foram definidas anteriormente, o SNMPTTCONVERTMIB foi utilizado apenas para conversão do arquivo.

Diante desse cenário podemos confirmar que os componentes SNMP disponíveis fornecem o suporte necessário para realização da integração das aplicações por meio do protocolo SNMP além de garantir o funcionamento do protocolo, de acordo com as recomendações descritas na RFC 1157.

A representação do arquivo MIB convertido com os OIDs e parâmetros de execução, para envio das informações coletadas para ferramenta de monitoramento poderão ser visualizados no anexo III.

Por fim o uso da ferramenta de monitoramento possibilitou idealizar e implantar uma forma de monitoramento para o Erlangms, pois como já informado nesse trabalho a estrutura para transmissão de dados a ser monitorado possui regras definidas, não implicando na evolução ou impedindo o monitoramento, mas restringindo a forma de como o monitoramento deve ser feito a partir da utilização do protocolo SNMP.

A ferramenta de monitoramento por sua vez, restringe que para realizar o monitoramento os *hosts* e serviços devam estar configurados em seus arquivos.

A realização dessa configuração foi executada de forma prática e sem complexidade, mas por conta dessa estrutura de registro e configuração, no momento não foi possível realizar a implementação e implantação do monitoramento de métricas que não foram definidas anteriormente e nem as que não foram compiladas e mapeadas nos arquivos de configuração, a estrutura de registros de configuração de *hosts* e serviços, que nessa situação são o Erlangms e a execução dos registros de *Log*, poderão ser visualizados no anexo IV.

Com a finalização do processo de integração e da configuração da ferramenta de monitoramento pode-se comprovar por meio do experimento realizado nesse trabalho, a integração das aplicações para realização do monitoramento dos recursos do barramento de serviços, com início na coleta de informações geradas por meio da instrumentação do Erlangms, e passando pelo Exometer onde foram criadas as métricas com atualização de valores, assim com o *report* em SNMP e transmitidas por meio de *TRAPs* pelo agente de monitoramento.

Esses *TRAPs* são interceptados pelos plugins da ferramenta de monitoramento e disponibilizado para apresentação em um *dashboard* de serviços da ferramenta de monitora-

mento, o Nagios®, o *dashboard* com a métricas coletadas das informações do recurso de registro de *Log* do Erlangms poderá ser visualizado na figura 5.1

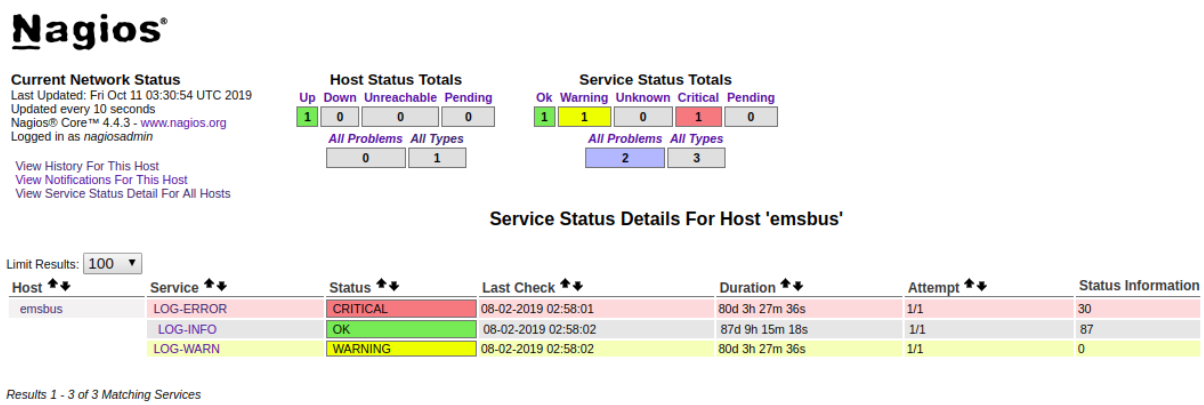

Figura 5.1: *Dashboard* do Nagios® com as métricas do registro de *Log*.

Ems-Monitor

Neste trabalho, foi realizada uma análise com a ferramenta de monitoramento Observer. Pode-se observar que o Erlangms sem a execução do módulo monitoramento, funciona entre cinco e dez porcento de utilização do *Scheduler*, cinquenta **MB** de utilização de Memória e com picos de quatrocentos **B** de utilização de IO. Essa verificação é realizada em tempo real, e pode ser observada na figura 5.2.

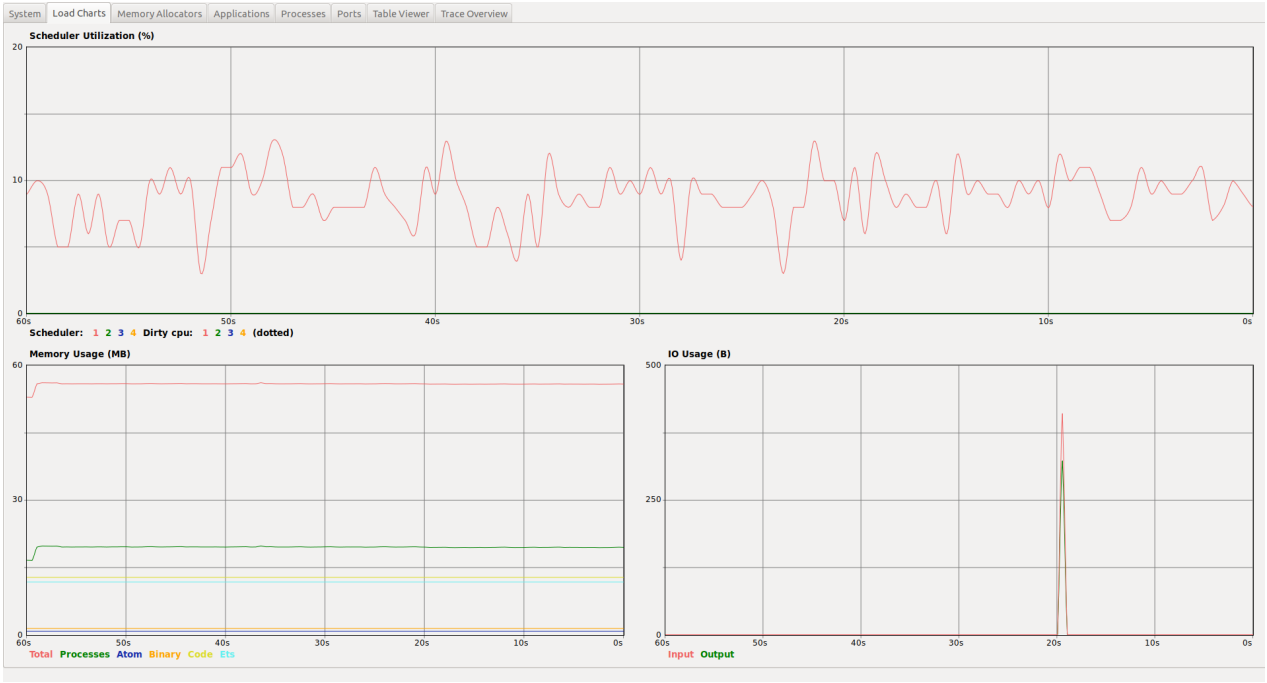

Figura 5.2: Análise executada em tempo real pela ferramenta Observer.

Com a utilização do Ems-monitor ativado no Erlangms, foi possível verificar durante a realização da análise a identificação de *overhead*, no *Scheduler*, Memória e IO. No funcionamento do Ems-monitor o aumento da utilização apresentou as seguintes informações: o *Scheduler* funcionando entre cinco e quinze porcento de utilização, sessenta **MB** de utilização de Memória e com aumento na intermitência que variam de vinte **KB** até cento e quarenta **KB** de utilização de IO.

A análise realizada na ferramenta Observer, para acompanhar o funcionamento do Ems-Monitor pode ser visualizada na figura 5.3.

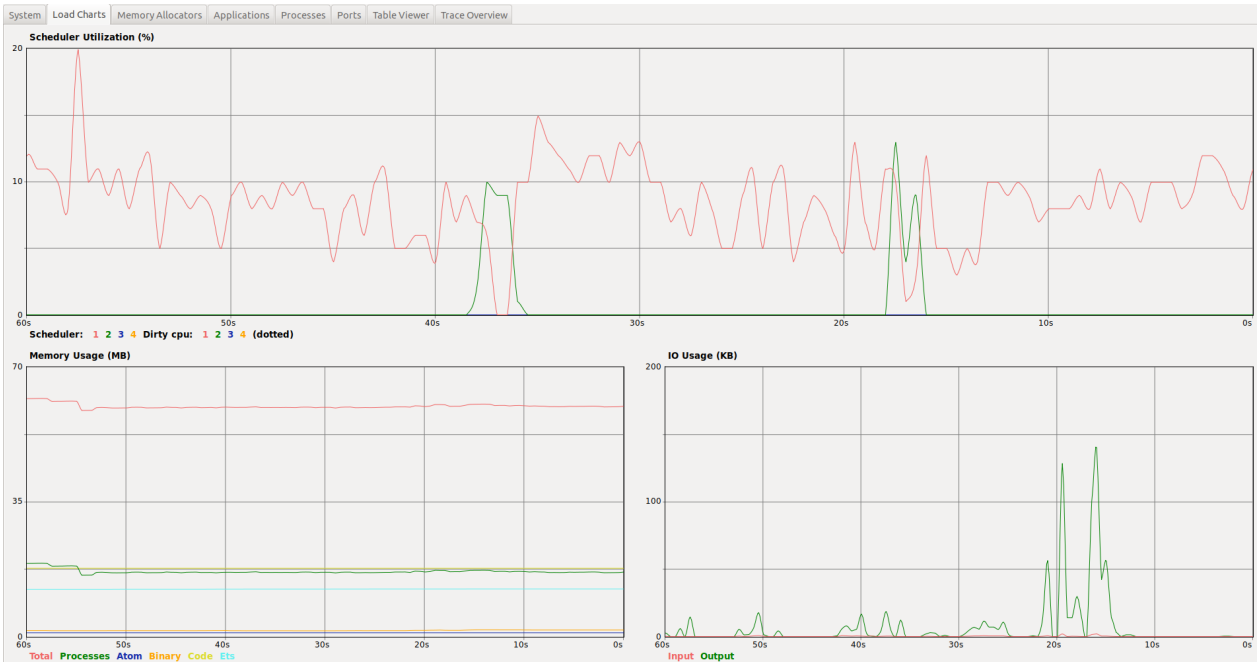

Figura 5.3: Análise executada em tempo real pela ferramenta Observer com o Ems-Monitor.

Por fim, pode-se observar o aumento da utilização dos recursos computacionais. O *Scheduler* precisou gerenciar mais processos, com o aumento de processos foi necessária a utilização de mais memória, que devido ao funcionamento em modo passivo do Ems-Monitor para o envio de *TRAPs*, teve também aumento IO, pois mesmo sem o funcionamento e execução dos recursos do Erlangms, mensagens são enviadas.

O *overhead* gerado pelo funcionamento do Ems-Monitor não demonstrou comprometer de forma impeditiva o funcionamento do Erlangms.

5.3 Síntese do Capítulo

Este capítulo apresentou a avaliação das aplicações integrantes da execução do monitoramento dos recursos do barramento, a partir dessa avaliação foi possível responder com fundamentação obtida dos resultados que a solução está adequada ao ambiente computacional do CPD da UnB, permitindo a possibilidade de implantação após validação pelo colaboradores desse Centro, e disponibilização da próxima versão disponível, para uso exclusivo do barramento de serviços Erlangms.

Capítulo 6

Conclusão

O CPD vem trabalhando nos últimos anos na modernização dos *softwares* por ele gerenciados, desde os sistemas que eram hospedados nos *mainframes*, que precisaram ser migrados para outras plataformas por causa da virada do milênio (*bug* do milênio) até os sistemas desenvolvidos em tecnologias mais atuais.

A partir da experiência adquirida ao longo dos anos, com erros e acertos em definições arquiteturais, finalmente por meio de pesquisas e fundamentação teórica, decidiu-se pela implementação e utilização de uma arquitetura SOA, que devido à sua padronização é muito bem aceita no mercado, instituições e empresas de desenvolvimento de *software*, proporcionando um desenvolvimento mais ágil dos serviços, por parte dos desenvolvedores. No CPD proporcionou o desenvolvimento de vários serviços e sistemas consumidores desses serviços, impactando em uma maior utilização e solicitação de implementação de novos serviços.

Este trabalho proporcionou o estudo, análise e pesquisa de um conjunto de soluções tecnológicas que possibilitaram a prática de ferramentas que auxiliassem na realização do monitoramento dos recursos do barramento de serviços por meio da integração com uma ferramenta de monitoramento via o protocolo SNMP, seguindo abordagens citadas no capítulo 2, que foi possível após a realização do mapeamento sistemático apresentado no capítulo 3 subsidiando a utilização da solução para realização do monitoramento.

A realização deste trabalho facilitou a utilização de novas definições de métricas para execução do monitoramento, visto que o modo para gerar as informações oriundas da instrumentação do barramento de serviços necessitavam de adequações para recebimento ou interceptação das informações pela ferramenta de monitoramento. Essa solução foi desenvolvida para utilização inicialmente no CPD da UnB, mas poderá ser utilizado em outras instituições que fizerem o uso do Erlangms.

O trabalho tem a intenção de suprir uma necessidade em relação à gestão, acompanhamento e monitoramento dos serviços disponibilizados pelo barramento de serviços, que

nos dias atuais já englobam um grande conjunto de serviços (*web services*) que compõem os sistemas estruturantes da UnB.

6.1 Contribuições

A intenção deste trabalho é contribuir com o apoio às soluções de monitoramento executados pelo CPD da UnB e elenca essencialmente a realização da integração das aplicações Erlangms, Exometer e Nagios® para realização do monitoramento por meio do protocolo SNMP, com implementação de um módulo capaz de coletar, armazenar e transmitir informações para execução do monitoramento.

A realização do mapeamento sistemático foi mais um ponto identificado como contribuição para orientação da seleção e utilização das ferramentas, as principais soluções e desafios para realização do monitoramento dos recursos do Erlangms por meio do protocolo SNMP.

Por fim, a implementação do módulo SNMP no Erlangms, proporciona a continuidade na evolução do trabalho [10] e possibilita o barramento de serviços utilizar qualquer ferramenta de monitoramento que funcione, e utilizar como meio de comunicação o protocolo SNMP.

6.2 Trabalhos Futuros

A partir da execução desse trabalho e visando proporcionar a continuidade, pretende-se:

1. Realizar testes de desempenho na solução, a fim de avaliar o funcionamento das aplicações por meio da integração realizada via protocolo SNMP;
2. Implementar uma solução que possibilite através do monitoramento identificar a indisponibilidade, e após a identificação esse serviço possa ser restabelecido automaticamente;
3. Definir um novo grupo de métricas que possibilite realizar o estudo semântico a fim de entender o comportamento do barramento de serviços durante o funcionamento, visando melhorar, ajustar e acompanhar os serviços disponibilizados e executados pelo Erlangms.

Referências

[1] Nadeau, Thomas D: *MPLS network management: MIBs, tools, and techniques*. Elsevier, 2003. x, 6, 8

[2] Manoj, V: *Comparative study of nosql document, column store databases and evaluation of cassandra*. International Journal of Database Management Systems, 6(4):11, 2014. x, 9

[3] LCC, Nagios: *Nagios documentation*, 2012. x, 10

[4] Project, Milenium: *Ganglia monitoring system*. Disponível em: <http://ganglia.info/>, 2000. Acesso em: 20 de janeiro de 2019. x, 11

[5] Group, The Cacti: *Cacti*. Disponível em: <https://www.cacti.net/>, 2004. Acesso em: 22 de janeiro de 2019. x, 12

[6] Oetiker, Tobias: *Rrdtool*. Disponível em: <https://oss.oetiker.ch/rrdtool/index.en.html>, 2009. Acesso em: 22 de janeiro de 2019. x, 12, 13

[7] Vladishev, Alexei: *Zabbix*. Disponível em: <https://www.zabbix.com/documentation/>, 2001. Acesso em: 23 de janeiro de 2019. x, 13, 14

[8] *Snmp,simple network management protocol*. Disponível em: <https://www.gta.ufrj.br/grad/10_1/snmp/componentes.htm>. x, 15

[9] Wiger, Ulf e Magnus Feuer: *Exometer core - erlang instrumentation package, core services*. Disponível em: <https://github.com/Feuerlabs/exometer_core>, 2014. x, 15, 16, 60

[10] Agilar, Everton de Vargas: *Uma abordagem orientada a serviços para a modernização de sistemas legados.* 2016. xvi, 129 f., il. Dissertação (Mestrado Profissional em Computação Aplicada) — Universidade de Brasília, Brasília, 2016., 2016. x, 1, 3, 19, 21, 22, 79

[11] Brereton, Pearl, Barbara A Kitchenham, David Budgen, Mark Turner e Mohamed Khalil: *Lessons from applying the systematic literature review process within the software engineering domain*. Journal of systems and software, 80(4):571–583, 2007. x, 27

[12] Petticrew, Mark e Helen Roberts: *Systematic reviews in the social sciences: A practical guide*. John Wiley & Sons, 2008. 3, 26

[13] Almeida Falbo, Ricardo de: *Mapeamento sistemático*. Retrieved October, 7, 2018. 3, 26

[14] Martino Jannuzzi, Paulo de: *Indicadores para diagnóstico, monitoramento e avaliação de programas sociais no brasil*. Revista do Serviço Público, 56(2):137–160, 2014. 5

[15] Hollingsworth, Jeffrey e Brian Tierney: *Instrumentation and monitoring*. The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, 2003. 5

[16] Penteado, Mauricio G e Luis Carlos Trevelin: *Jmonitor: A monitoring tool for distributed systems*. Em *Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on*, páginas 1767–1772. IEEE, 2012. 6, 34, 40, 42

[17] Roohi, Arman, Khashayar Raeisifard e Suhaimi Ibrahim: *An application for management and monitoring the data centers based on snmp*. Em *2014 IEEE Student Conference on Research and Development*, páginas 1–4. IEEE, 2014. 6

[18] Presuhn, Randy: *Management information base (mib) for the simple network management protocol (snmp)*. Management, 2002. 6

[19] Mankin, Allison: *Towards tactical gosip: performance of tcp/ip over rf links in the itdn*. Em *MILCOM 91-Conference record*, páginas 166–171. IEEE, 1991. 8

[20] Harris, RT: *Tactical communication systems with the gosip architecture*. Em *MILCOM 92 Conference Record*, páginas 731–735. IEEE, 1992. 8

[21] Tanenbaum, Andrew S e Maarten Van Steen: *Distributed systems: principles and paradigms*. Prentice-Hall, 2007. 8

[22] Mohd Shuhaimi, Mohammad Ali Arsyad bin, Irda binti Roslan, Syarulnaziah binti Anawar *et al.*: *The new services in nagios: network bandwidth utility, email notification and sms alert in improving the network performance*. Em *Information Assurance and Security (IAS), 2011 7th International Conference on*, páginas 86–91. IEEE, 2011. 9

[23] Vyas, Ravindra A, Harshad B Prajapati e Vipul K Dabhi: *Embedding custom metric in ganglia monitoring system*. Em *Advance Computing Conference (IACC), 2014 IEEE International*, páginas 793–797. IEEE, 2014. 11

[24] Benincosa, Vallard: *Ganglia and nagios*. 11

[25] Contessa, Diego Fraga e Everton Rafael Polina: *Gerenciamento de equipamentos usando o protocolo snmp*. Relatório Técnico, Technical report, Departamento de Pesquisa e Desenvolvimento-CP Eletrônica ..., 2010. 13

[26] Marik, Ondrej e Stanislav Zitta: *Comparative analysis of monitoring system for data networks*. Em *Multimedia Computing and Systems (ICMCS), 2014 International Conference on*, páginas 563–568. IEEE, 2014. 13

[27] Grover, K. e V. Naik: *Monitoring of android devices using snmp.* Em *2016 8th International Conference on Communication Systems and Networks (COMSNETS)*, páginas 1–2, Jan 2016. 14

[28] Kazaz, T., M. Kulin, E. Kaljić e T. Čaršimanović: *One approach to the development of custom snmp agents and integration with management systems.* Em *2012 Proceedings of the 35th International Convention MIPRO*, páginas 557–561, May 2012. 15

[29] Ribeiro, Marco Antônio: *Instrumentação.* Tek Treinamentos LTDA, 1999. 16

[30] Robinson, William N: *Monitoring software requirements using instrumented code.* Em *Proceedings of the 35th Annual Hawaii International Conference on System Sciences*, páginas 3967–3976. IEEE, 2002. 16

[31] Frota, Mauricio Nogueira e Ludwik Finkelstein: *Educação em metrologia e instrumentação: Demanda qualificada no ensino das engenharias.* Revista de Ensino de Engenharia, 25(1), 2008. 16

[32] Castro, Marcus Vinícius Borela de e Carlos Alberto Mamede Hernandes: *Uma métrica de tamanho de software como ferramenta para a governança de ti.* Revista do TCU, (135):56–75, 2016. 16

[33] Munawar, M. A., M. Jiang, T. Reidemeister e P. A. S. Ward: *Filtering system metrics for minimal correlation-based self-monitoring.* Em *2009 Third IEEE International Conference on Self-Adaptive and Self-Organizing Systems*, páginas 233–242, Sep. 2009. 17

[34] Normas Técnicas, Associação Brasileira de: *Nbr iso/iec 9126-1: Engenharia de software: qualidade de produto: parte 1: modelo de qualidade*, 2003. 17

[35] Meirelles, Paulo Roberto Miranda: *Monitoramento de métricas de código-fonte em projetos de software livre.* Tese de Doutoramento, Universidade de São Paulo, 2013. 17

[36] Fraser, R, Terry Rankine e Robert Woodcock: *Service oriented grid architecture for geosciences community.* Em *Proceedings of the fifth Australasian symposium on ACSW frontiers-Volume 68*, páginas 19–23. Australian Computer Society, Inc., 2007. 17

[37] Sward, Ricky E e Jeff Boleng: *Service-oriented architecture (soa) concepts and implementations.* Em *ACM SIGAda Ada Letters*, volume 31, páginas 3–4. ACM, 2011. 17

[38] Bianco, Philip, Grace A Lewis, Paulo Merson e Soumya Simanta: *Architecting service-oriented systems.* Relatório Técnico, CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST, 2011. 17, 19, 20

[39] Erl, Thomas, A Arsanjani, G Booch, P Boubez, D Chappell, J deVadoss, N Josuttis, D Krafzig, M Little, B Loesgen *et al.*: *Soa manifesto.* Letzter Zugriff am, 19:2010, 2009. 18, 19

[40] Clements, Paul, David Garlan, Len Bass, Judith Stafford, Robert Nord, James Ivers e Reed Little: *Documenting software architectures: views and beyond*. Pearson Education, 2002. 18, 19, 46, 48

[41] Bass, Len, Paul Clements e Rick Kazman: *Software architecture in practice*. Addison-Wesley Professional, 2003. 18

[42] Fielding, Roy T e Richard N Taylor: *Architectural styles and the design of network-based software architectures*, volume 7. University of California, Irvine Doctoral dissertation, 2000. 23

[43] Booth, David, Hugo Haas, Francis McCabe, Eric Newcomer, Michael Champion, Chris Ferris e David Orchard: *Web services architecture, w3c working group note, 2004*. URL: http://www. w3. org/TR/2004/NOTE-ws-arch-20040211, 2013. 23

[44] Bray, Tim: *The javascript object notation (json) data interchange format*. Relatório Técnico, 2017. 23

[45] Abdu, Hasina, Hanan Lutfiyya e Michael A Bauer: *Monitoring overhead in distributed systems: visualization and estimation techniques*. Em *Proceedings of the 1996 conference of the Centre for Advanced Studies on Collaborative research*, página 1. IBM Press, 1996. 24, 34, 39, 44

[46] Repantis, Thomas, Jeff Cohen, Scott Smith e Joel Wein: *Scaling a monitoring infrastructure for the akamai network*. ACM SIGOPS Operating Systems Review, 44(3):20–26, 2010. 24, 33, 39, 43

[47] Subramanyan, Rajesh, José Miguel-Alonso e José AB Fortes: *A scalable snmp-based distibuted monitoring system for heterogeneous network computing*. Em *Proceedings of the 2000 ACM/IEEE conference on Supercomputing*, página 14. IEEE Computer Society, 2000. 24, 34, 39, 43

[48] Phan, Raphael C W: *Cryptanalysis of the application secure alternative to snmp (apssnmp)*. Computer Standards & Interfaces, 31(1):63–65, 2009. 24, 34, 40, 43

[49] Lee, Jin Shyan e Pau Lo Hsu: *Design and implementation of the snmp agents for remote monitoring and control via uml and petri nets*. IEEE Transactions on Control Systems Technology, 12(2):293–302, 2004. 24, 35, 41, 43

[50] Pătruţ, Bogdan e Cosmin Tomozei: *Agent technology in monitoring systems*. International Journal of Computers Communications & Control, 5(5):852–861, 2010. 24, 34, 40, 43

[51] Kitchenham, B e S Charters: *Guidelines for performing structural literature reviews in software engineering*. Empirical Software Engineering, National ICT, páginas 45–56, 2007. 26, 27

[52] Feltrim, Valéria Delisandra: *Uma abordagem baseada em corpus e em sistemas de crítica para a construção de ambientes Web de auxilio à escrita acadêmica em português*. Tese de Doutoramento, 2004. 26

[53] Keele, Staffs *et al.*: *Guidelines for performing systematic literature reviews in software engineering*. Relatório Técnico, Technical report, Ver. 2.3 EBSE Technical Report. EBSE, 2007. 26

[54] Chomsky, Noam e Gabriel Ferrater: *Lingüística cartesiana*. Gredos Madrid, 1969. 30

[55] Petersen, Kai, Robert Feldt, Shahid Mujtaba e Michael Mattsson: *Systematic mapping studies in software engineering.* Em *Ease*, volume 8, páginas 68–77, 2008. 31

[56] Abousharkh, Maha e Hussein Mouftah: *Service oriented architecture-based framework for wban-enabled patient monitoring system*. Em *Proceedings of the Second Kuwait Conference on e-Services and e-Systems*, página 18. ACM, 2011. 33, 38, 42

[57] Abdu, Hasina, Hanan L Lutfiyya e Michael A Bauer: *An investigation of monitoring configurations*. Em *Proceedings of the 1995 conference of the Centre for Advanced Studies on Collaborative research*, página 1. IBM Press, 1995. 33, 38, 42

[58] Cirstoiu, Catalin C, Costin C Grigoras, Latchezar L Betev, Alexandru A Costan e Iosif Charles Legrand: *Monitoring, accounting and automated decision support for the alice experiment based on the monalisa framework*. Em *Proceedings of the 2007 workshop on Grid monitoring*, páginas 39–44. ACM, 2007. 33, 39, 42

[59] Sedayao, Jeff: *Implementing and operating an internet scale distributed application using service oriented architecture principles and cloud computing infrastructure*. Em *Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services*, páginas 417–421. ACM, 2008. 34, 39, 44

[60] Joyce, Jeffrey, Greg Lomow, Konrad Slind e Brian Unger: *Monitoring distributed systems*. ACM Transactions on Computer Systems (TOCS), 5(2):121–150, 1987. 34, 39, 42

[61] Wang, Bo, Ying Song, Yuzhong Sun e Jun Liu: *Improvements to online distributed monitoring systems*. Em *Trustcom/BigDataSE/I SPA, 2016 IEEE*, páginas 1093–1100. IEEE, 2016. 34, 40, 44

[62] Nagorny, Kevin, Robert Harrison, Armando Walter Colombo e Gerhard Kreutz: *A formal engineering approach for control and monitoring systems in a service-oriented environment*. Em *2013 11th IEEE International Conference on Industrial Informatics (INDIN)*, páginas 480–487. IEEE, 2013. 34

[63] Ghorbani, Shirin e Weichang Du: *Personal health service framework*. Procedia Computer Science, 21:343–350, 2013. 34, 40, 42

[64] Hirate, Yu e Hayato Yamana: *Profiling node conditions of distributed system with sequential pattern mining*. Em *Future Dependable Distributed Systems, 2009 Software Technologies for*, páginas 43–48. IEEE, 2009. 35, 40, 44

[65] Casola, Valentina, Andrea Gaglione e Antonino Mazzeo: *Sensim-web: a service based architecture for sensor networks integration.* Em *Industrial Electronics, 2009. IECON'09. 35th Annual Conference of IEEE*, páginas 2665–2671. IEEE, 2009. 35, 40, 42

[66] Kotsopoulos, Konstantinos, Pouwan Lei e Yim Fun Hu: *A soa-based information management model for next-generation network.* Em *Computer and Communication Engineering, 2008. ICCCE 2008. International Conference on*, páginas 1057–1062. IEEE, 2008. 35, 40, 44

[67] Smith, Garry e Mark Baker: *A flexible monitoring and notification system for distributed resources.* Em *Parallel and Distributed Computing, 2008. ISPDC'08. International Symposium on*, páginas 31–38. IEEE, 2008. 35, 41, 43

[68] Feng, JQ, David P Buse, QH Wu e J Fitch: *Distributed mobile communication base station diagnosis and monitoring using multi-agents.* Em *International Conference on Intelligent Data Engineering and Automated Learning*, páginas 267–272. Springer, 2002. 35, 41, 43

[69] Qi-rui, Peng, Wang Cheng, WU Jing, Li Jun, LI Qing e Shao Bei-en: *An authentication and authorization solution supporting soa-based distributed systems.* Em *2010 IEEE International Conference on Software Engineering and Service Sciences*, páginas 535–538. IEEE, 2010. 35

[70] Brattstrom, Morgan e Patricia Morreale: *Scalable agentless cloud network monitoring.* Em *2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud)*, páginas 171–176. IEEE, 2017. 35

[71] Kotari, Manjunath, Niranjan N Chiplunkar e HR Nagesh: *Implementation of secure customized monitoring tool for adapative distributed systems.* Em *2014 International Conference on Contemporary Computing and Informatics (IC3I)*, páginas 562–567. IEEE, 2014. 35

[72] Kistijantoro, Achmad Imam *et al.*: *Multi-layer soa implementation pattern with service and data proxies for distributed data-intensive application system.* Em *2014 International Conference on ICT For Smart Society (ICISS)*, páginas 37–41. IEEE, 2014. 36

[73] Ibrahim, Najhan M, Mohd Fadzil Hassan e M Hussin Abdullah: *Abmom for cross-platform communication in soa systems.* Em *2013 International Conference on Research and Innovation in Information Systems (ICRIIS)*, páginas 107–112. IEEE, 2013. 36

[74] Yong, Sun e Ren Yi-Zhi: *Reliable esb and distributed transactional memory for soa.* Em *2012 Fifth International Symposium on Computational Intelligence and Design*, volume 2, páginas 194–197. IEEE, 2012. 36

[75] Jammes, Francois, Bernard Bony, Philippe Nappey, Armando W Colombo, Jerker Delsing, Jens Eliasson, Rumen Kyusakov, Stamatis Karnouskos, Petr Stluka e Marcel Till: *Technologies for soa-based distributed large scale process monitoring and*

control systems. Em *IECON 2012-38th Annual Conference on IEEE Industrial Electronics Society*, páginas 5799–5804. IEEE, 2012. 36

[76] Pang, Yusong e Gabriel Lodewijks: *Agent-based intelligent monitoring in large-scale continuous material transport.* Em *Proceedings of 2012 9th IEEE International Conference on Networking, Sensing and Control*, páginas 79–84. IEEE, 2012. 36

[77] El Ioini, Nabil: *Web services open test suites.* Em *2011 IEEE World Congress on Services*, páginas 77–80. IEEE, 2011. 36

[78] Yan, Pei e Jiao Guo: *Researching and designing the architecture of e-government based on soa.* Em *2010 International Conference on E-Business and E-Government*, páginas 512–515. IEEE, 2010. 36

[79] Funika, Wlodzimierz, Piotr Pegiel, Marian Bubak e Jacek Kitowski: *Towards role-based self-healing in autonomous monitoring systems.* Em *2010 International Conference on Complex, Intelligent and Software Intensive Systems*, páginas 1063–1068. IEEE, 2010. 36

[80] Ali, Nour e Muhammad Ali Babar: *Modeling service oriented architectures of mobile applications by extending soaml with ambients.* Em *2009 35th Euromicro Conference on Software Engineering and Advanced Applications*, páginas 442–449. IEEE, 2009. 36

[81] Zhang, Jingjun, Fanxin Meng e Guangyuan Liu: *Research on multi-tier distributed systems based on aop and web services.* Em *2009 First International Workshop on Education Technology and Computer Science*, volume 2, páginas 203–207. IEEE, 2009. 37

[82] Zhang, Jingjun, Fanxin Meng e Guangyuan Liu: *Research on soa-based applications based on aop and web services.* Em *2008 International Conference on Computer and Electrical Engineering*, páginas 753–757. IEEE, 2008. 37

[83] Ford, Ray: *Monitoring distributed embedded systems.* Em *Proceedings of the 1990 Symposium on Applied Computing*, páginas 237–244. IEEE, 1990. 37

[84] Mansouri-Samani, Masoud e Morris Sloman: *Monitoring distributed systems.* IEEE network, 7(6):20–30, 1993. 37

[85] Leitner, Philipp, Stefan Schulte, Schahram Dustdar, Ingo Pill, Marco Schulz e Franz Wotawa: *The dark side of soa testing: Towards testing contemporary soas based on criticality metrics.* Em *2013 5th International Workshop on Principles of Engineering Service-Oriented Systems (PESOS)*, páginas 45–53. IEEE, 2013. 37

[86] Bosnjak, Ante, Shihong Huang e James J Mulcahy: *Leveraging service oriented architecture: A case study for ocean energy information management.* Em *2011 IEEE International Conference on Information Reuse & Integration*, páginas 108–112. IEEE, 2011. 37

[87] Benharref, Abdelghani, Mohamed Adel Serhani, Salah Bouktif e Jamal Bentahar: *A managerial community of web services for management of communities of web services.* Em *2010 10th Annual International Conference on New Technologies of Distributed Systems (NOTERE)*, páginas 97–104. IEEE, 2010. 37

[88] Haselböck, Stefan e Rainer Weinreich: *Decision guidance models for microservice monitoring.* Em *2017 IEEE International Conference on Software Architecture Workshops (ICSAW)*, páginas 54–61. IEEE, 2017. 37

[89] Korableva, Olga, Olga Kalimullina e Ekaterina Kurbanova: *Building the monitoring systems for complex distributed systems: Problems and solutions.* Em *ICEIS (2)*, páginas 221–228, 2017. 37, 41, 44

[90] Dinu, Cristian Mircea, Florin Pop e Valentin Cristea: *Pattern detection model for monitoring distributed systems.* Em *2011 13th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, páginas 268–275. IEEE, 2011. 37

[91] Wen, Junhao, Li Zhao e Pan He: *Dynamic agent based monitoring mechanism for web services.* Journal of Convergence Information Technology, 6(10), 2011. 37

[92] Zhang, Xuechai, Jeffrey L Freschl e Jennifer M Schopf: *A performance study of monitoring and information services for distributed systems.* Em *High Performance Distributed Computing, 2003. Proceedings. 12th IEEE International Symposium on*, páginas 270–281. IEEE, 2003. 37

[93] Al-Shaer, Ehab, Hussein Abdel-Wahab e Kurt Maly: *Hifi: A new monitoring architecture for distributed systems management.* Em *Proceedings. 19th IEEE International Conference on Distributed Computing Systems (Cat. No. 99CB37003)*, páginas 171–178. IEEE, 1999. 38

[94] Psiuk, Marek e Krzysztof Zielinski: *Goal-driven adaptive monitoring of soa systems.* Journal of Systems and Software, 110:101–121, 2015. 38

[95] Cao, Jiannong, Kang Zhang e Olivier de Vel: *On heuristics for optimal configuration of hierarchical distributed monitoring systems.* Journal of Systems and Software, 43(3):197–206, 1998. 38

[96] Pras, Aiko e Jean Philippe Martin-Flatin: *What can web services bring to integrated management?* Em *Handbook of network and system administration*, páginas 241–294. Elsevier, 2008. 38

[97] Jaković, Tihomir, Ivan Murat, Filip Klarić e Samir Keitoue: *Transformer fleet monitoring.* Procedia engineering, 202:20–28, 2017. 38

[98] Bai, Xiaoying, Yongli Liu, Lijun Wang e Peide Zhong: *Model-based monitoring and policy enforcement of services.* Simulation Modelling Practice and Theory, 17(8):1399–1412, 2009. 38

[99] Jayashree, K e Sheila Anand: *Web service diagnoser model for managing faults in web services.* Computer Standards & Interfaces, 36(1):154–164, 2013. 38

[100] Junhao, Wen, Zhao Li e Pan He: *Dynamic agent based monitoring mechanism for web services.* Journal of Convergence Information Technology, 6:211–218, outubro 2011. 39, 44

[101] Brattstrom, M. e P. Morreale: *Scalable agentless cloud network monitoring.* Em *2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud)*, páginas 171–176, June 2017. 40, 44

[102] Garlan, David e Mary Shaw: *An introduction to software architecture.* Em *Advances in software engineering and knowledge engineering*, páginas 1–39. World Scientific, 1993. 46

[103] Tanenbaum, AS: *Redes de computadores,[si].* SELEÇÃO TUTORES PRESENCIAIS, 2003. 49

[104] Filgueiras, Renan Costa: *Monitoramento em tempo de execução: a construção de um módulo em erlang para uma arquitetura orientada a serviços.* 2017. xiii, 74 f., il. Dissertação (Mestrado Profissional em Computação Aplicada) - Universidade de Brasília, Brasília, 2017., 2017. 50, 67

[105] *Erlang otp 22.0.* Disponível em: <http://erlang.org/doc/man/observer.html>. Acesso em: 12 de setembro de 2019. 50, 51

[106] *Snmp trap integration.* Disponível em: <https://assets.nagios.com/downloads/nagioscore/docs/nagioscore/4/en/int-snmptrap.html>. Acesso em: 12 de setembro de 2019. 63

[107] *Snmptrapd.* Disponível em: <http://www.net-snmp.org/docs/man/snmptrapd.html>. Acesso em: 12 de setembro de 2019. 63

[108] Ribeiro, Alysson de Sousa: *Uma implementação do protocolo oauth 2 em erlang para uma arquitetura orientada a serviço.* 2017. xv, 78 f., il. Dissertação (Mestrado Profissional em Computação Aplicada)—Universidade de Brasília, Brasília, 2017, 2017. 70

[109] *Erlangms.* Disponível em: <https://github.com/erlangMS/ems-bus>. Acesso em: 15 de setembro de 2019. 71

[110] *Exometer.* Disponível em: <https://github.com/Feuerlabs/exometer>. Acesso em: 16 de setembro de 2019. 71

[111] *Nagios core®.* Disponível em: <http://nagios-br.com/instalando-nagios-core-4-0>. Acesso em: 15 de setembro de 2019. 71

[112] Schoffstall, JD Case M Fedor ML e C Davin: *Rfc 1157: Simple network management protocol (snmp).* IETF, April, 1990. 72

Apêndice A

Código do Módulo do Serviço de Monitoramento

Código da implementação do Módulo SNMP no Erlangms

```
%%%-------------------------------------------------------------------
%% @title Module ems_snmp_server
%% @version 1.0.0
%% @doc Main module SNMP server
%% @author Felipe Evangelista dos Santos <fevansantos@gmail.com>
%% @copyright ErlangMS Team
%%%-------------------------------------------------------------------

-module(ems_snmp_server).

-behavior(gen_server).

-include("include/ems_config.hrl").
-include("include/ems_schema.hrl").

%% Server API
-export([start/1, stop/0]).


%% gen_server callbacks
-export([init/1, handle_call/3, handle_cast/2, handle_info/1,
    handle_info/2, terminate/2, code_change/3]).

% estado do servidor
-record(state, {listener=[],
                                service,
                                name
                }).
```

```
-define(SERVER, ?MODULE).

%%====================================================================
%% Server API
%%====================================================================

start(Service = #service{name = Name}) ->
         ServerName = erlang:binary_to_atom(Name, utf8),
    gen_server:start_link({local, ServerName}, ?MODULE, Service, []).

stop() ->
    gen_server:cast(?SERVER, shutdown).

%%====================================================================
%% gen_server callbacks
%%====================================================================

init(Service = #service{start_timeout = StartTimeout}) ->
         State = #state{service = Service},
         {ok, State, StartTimeout}.


handle_cast(shutdown, State) ->
    {stop, normal, State};

handle_cast(_Msg, State) ->
        {noreply, State}.

handle_call(Msg, _From, State) ->
        {reply, Msg, State}.

handle_info(State) ->
   {noreply, State}.

handle_info(timeout, State = #state{service = S = #service{name = Name,
                                        tcp_listen_address_t = ListenAddress_t,
                                        properties = Props}}) ->
        S2 = ems_config:get_port_offset(S),
        ServerName = binary_to_list(iolist_to_binary([Name, <<"_port_">>,
        integer_to_binary(S2#service.tcp_port)])),

        SnmpAgentConfigPath0 = binary_to_list(
        maps:get(<<"snmp_agent_config_path">>, Props, <<>>)),
        SnmpManagerConfigPath0 = binary_to_list(
        maps:get(<<"snmp_manager_config_path">>, Props, <<>>)),
        SnmpAgentDBPath0 = binary_to_list(
```

```
        maps:get(<<"snmp_agent_db_path">>, Props, <<>>)),
        SnmpManagerDBPath0 = binary_to_list(
        maps:get(<<"snmp_manager_db_path">>, Props, <<>>)),

        SnmpAgentConfigPath = ems_util:parse_file_name_path(
        SnmpAgentConfigPath0, [{<<"PRIV_PATH">>, ?PRIV_PATH}], undefined),
        SnmpManagerConfigPath = ems_util:parse_file_name_path(
        SnmpManagerConfigPath0, [{<<"PRIV_PATH">>, ?PRIV_PATH}], undefined),
        SnmpAgentDBPath = ems_util:parse_file_name_path(
        SnmpAgentDBPath0, [{<<"PRIV_PATH">>, ?PRIV_PATH}], undefined),
        SnmpManagerDBPath = ems_util:parse_file_name_path(
        SnmpManagerDBPath0, [{<<"PRIV_PATH">>, ?PRIV_PATH}], undefined),
        SnmpMetricVerbosity = binary_to_atom(
        maps:get(<<"snmp_metric_verbosity">>, Props, <<"log">>), utf8),



        io:format("AgentSnmpConfigPath: ~p\n", [SnmpAgentConfigPath]),
        io:format("SnmpManagerConfigPath: ~p\n", [SnmpManagerConfigPath]),
        io:format("SnmpAgentDBPath: ~p\n", [SnmpAgentDBPath]),
        io:format("SnmpManagerDBPath: ~p\n", [SnmpManagerDBPath]),
        io:format("SnmpMetricVerbosity: ~p\n", [SnmpMetricVerbosity]),

        ems_util:ensure_dir_writable(SnmpAgentDBPath),
        ems_util:ensure_dir_writable(SnmpManagerDBPath),

        application:set_env(snmp, agent,[
        {config, [{dir, SnmpAgentConfigPath}, {force_load, true},{verbosity,
        debug}]}, {db_dir, SnmpAgentDBPath}, {agent_type, master}]),
        application:set_env(snmp, manager, [
        {config, [{dir, SnmpManagerConfigPath},{db_dir, SnmpManagerDBPath}]}]),
        snmp:start(),
        exometer:start(),
        metric_verbosity(SnmpMetricVerbosity),
        add_reporter(),
        create_metrics(),

        {noreply, State}.

terminate(_Reason, _State) ->
    ok.

code_change(_OldVsn, State, _Extra) ->
    {ok, State}.

%%====================================================================
```

```
%% Internal functions
%%====================================================================

%% @doc Verbosity for SNMP by Exometer
%% Levels: silence | info | log | debug | trace.
metric_verbosity()-> snmpa:verbosity(master_agent, silence).
metric_verbosity(Type)-> snmpa:verbosity(master_agent, Type).


add_reporter()->
        exometer_report:add_reporter(exometer_report_snmp,[]).


create_metrics()->
        exometer:new([ems_logger_write_error], counter, [
                        {snmp, [{value, 20000}]}]),
        exometer:new([ems_logger_write_info],  counter, [
                        {snmp, [{value, 20000}]}]),
        exometer:new([ems_logger_write_warn],  counter, [
                        {snmp, [{value, 20000}]}]).


%% @doc Update metric value by Exometer
inc_counter_metric(Name) -> exometer:update([Name], 1).
dec_counter_metric(Name) ->  exometer:update([Name], -1).
counter_metric(Name, Value) -> exometer:update([Name], Value).

%% @doc Delete a metric
delete_metric(Name)-> exometer:delete([Name]).
```

Apêndice B

Aplicação do Código para a realização da coleta das informações para o Monitoramento

Função do módulo SNMP no Erlangms

```
case UltMsg == undefined orelse UltMsg =/= Msg of
true ->
case Tipo of
    info  ->
        ems_db:inc_counter(ems_logger_write_info),
        ems_snmp_server:inc_counter_metric(ems_logger_write_info),
        Msg1 = iolist_to_binary([?INFO_MESSAGE,  ?LIGHT_GREEN_COLOR,
        ems_clock:local_time_str(), ?WHITE_SPACE_COLOR, Msg, <<"\n">>]);
    error ->
        ems_db:inc_counter(ems_logger_write_error),
        ems_snmp_server:inc_counter_metric(ems_logger_write_error),
        Msg1 = iolist_to_binary([?ERROR_MESSAGE, ?LIGHT_GREEN_COLOR,
        ems_clock:local_time_str(), ?WHITE_SPACE_COLOR, ?RED_COLOR,
        Msg, ?WHITE_BRK_COLOR]);
    warn  ->
        ems_db:inc_counter(ems_logger_write_warn),
        ems_snmp_server:inc_counter_metric(ems_logger_write_warn),
        Msg1 = iolist_to_binary([?WARN_MESSAGE,  ?LIGHT_GREEN_COLOR,
        ems_clock:local_time_str(), ?WHITE_SPACE_COLOR, ?WARN_COLOR,
        Msg, ?WHITE_BRK_COLOR]);
    debug ->
        ems_db:inc_counter(ems_logger_write_debug),
        ems_snmp_server:inc_counter_metric(ems_logger_write_debug),
        Msg1 = iolist_to_binary([?DEBUG_MESSAGE,
        ?LIGHTems_logger_write_debug_GREEN_COLOR,
        ems_clock:local_time_str(), ?WHITE_SPACE_COLOR, ?DEBUG_COLOR,
```

```
        Msg, ?WHITE_BRK_COLOR])
end,
```

Anexo I

Arquivo MIB sem os OIDs

```
    EXOMETER-METRICS-MIB DEFINITIONS ::= BEGIN

IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE, NOTIFICATION-TYPE,
    Counter32, Counter64, Gauge32, Integer32,
    snmpModules, experimental FROM SNMPv2-SMI
    MODULE-COMPLIANCE, OBJECT-GROUP, NOTIFICATION-GROUP
    FROM SNMPv2-CONF;

exometerMetricsMIB MODULE-IDENTITY
        LAST-UPDATED "201401190525Z"
        ORGANIZATION "Feuerlabs"
        CONTACT-INFO "TODO"
        DESCRIPTION
        "This MIB module is used for exposing dynamic exometer metrics."
        REVISION  "201401190525Z"
        DESCRIPTION
                "The initial version"
        ::= { snmpModules 1 }

exometerMetrics OBJECT IDENTIFIER ::= { experimental 7 }

-- CONTENT START

-- CONTENT END

END
```

Anexo II

Arquivo MIB com os OIDs

```
   EXOMETER-METRICS-MIB DEFINITIONS ::= BEGIN

IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE, NOTIFICATION-TYPE,
    Counter32, Counter64, Gauge32, Integer32,
    snmpModules, experimental FROM SNMPv2-SMI
    MODULE-COMPLIANCE, OBJECT-GROUP, NOTIFICATION-GROUP
    FROM SNMPv2-CONF;

exometerMetricsMIB MODULE-IDENTITY
        LAST-UPDATED "201401190525Z"
        ORGANIZATION "Feuerlabs"
        CONTACT-INFO "TODO"
        DESCRIPTION
        "This MIB module is used for exposing dynamic exometer metrics."
        REVISION  "201401190525Z"
        DESCRIPTION
                "The initial version"
        ::= { snmpModules 1 }

exometerMetrics OBJECT IDENTIFIER ::= { experimental 7 }

-- CONTENT START

-- METRIC datapointEmsLoggerWriteErrorValue START
datapointEmsLoggerWriteErrorValue OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION ""
    ::= { exometerMetrics 1 }
```

```
-- METRIC datapointEmsLoggerWriteErrorValue END

-- METRIC datapointEmsLoggerWriteErrorMsSinceReset START
datapointEmsLoggerWriteErrorMsSinceReset OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION ""
    ::= { exometerMetrics 3 }
-- METRIC datapointEmsLoggerWriteErrorMsSinceReset END

-- METRIC datapointEmsLoggerWriteInfoValue START
datapointEmsLoggerWriteInfoValue OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION ""
    ::= { exometerMetrics 4 }
-- METRIC datapointEmsLoggerWriteInfoValue END

-- METRIC datapointEmsLoggerWriteInfoMsSinceReset START
datapointEmsLoggerWriteInfoMsSinceReset OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION ""
    ::= { exometerMetrics 5 }
-- METRIC datapointEmsLoggerWriteInfoMsSinceReset END

-- METRIC datapointEmsLoggerWriteWarnValue START
datapointEmsLoggerWriteWarnValue OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION ""
    ::= { exometerMetrics 6 }
-- METRIC datapointEmsLoggerWriteWarnValue END

-- METRIC datapointEmsLoggerWriteWarnMsSinceReset START
datapointEmsLoggerWriteWarnMsSinceReset OBJECT-TYPE
    SYNTAX Counter32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION ""
    ::= { exometerMetrics 7 }
-- METRIC datapointEmsLoggerWriteWarnMsSinceReset END
```

```
-- OBJECT-GROUP allObjects START
allObjects OBJECT-GROUP
    OBJECTS {
        datapointEmsLoggerWriteErrorMsSinceReset,
        datapointEmsLoggerWriteErrorValue,
        datapointEmsLoggerWriteInfoMsSinceReset,
        datapointEmsLoggerWriteInfoValue,
        datapointEmsLoggerWriteWarnMsSinceReset,
        datapointEmsLoggerWriteWarnValue
    }
    STATUS current
    DESCRIPTION ""
    ::= { exometerMetrics 2 }
-- OBJECT-GROUP allObjects END

-- INFORM reportEmsLoggerWriteErrorValue START
reportEmsLoggerWriteErrorValue NOTIFICATION-TYPE
    OBJECTS {
        datapointEmsLoggerWriteErrorValue
    }
    STATUS current
    DESCRIPTION ""
    ::= { exometerMetrics 8 }
-- INFORM reportEmsLoggerWriteErrorValue END

-- INFORM reportEmsLoggerWriteInfoValue START
reportEmsLoggerWriteInfoValue NOTIFICATION-TYPE
    OBJECTS {
        datapointEmsLoggerWriteInfoValue
    }
    STATUS current
    DESCRIPTION ""
    ::= { exometerMetrics 10 }
-- INFORM reportEmsLoggerWriteInfoValue END

-- INFORM reportEmsLoggerWriteWarnValue START
reportEmsLoggerWriteWarnValue NOTIFICATION-TYPE
    OBJECTS {
        datapointEmsLoggerWriteWarnValue
    }
    STATUS current
    DESCRIPTION ""
    ::= { exometerMetrics 11 }
-- INFORM reportEmsLoggerWriteWarnValue END
```

```
-- NOTIFICATION-GROUP allNotifications START
allNotifications NOTIFICATION-GROUP
    NOTIFICATIONS {
        reportEmsLoggerWriteErrorValue,
        reportEmsLoggerWriteInfoValue,
        reportEmsLoggerWriteWarnValue
    }
    STATUS current
    DESCRIPTION ""
    ::= { exometerMetrics 9 }
-- NOTIFICATION-GROUP allNotifications END

-- CONTENT END

END
```

Anexo III

Arquivo MIB convertido pelo plugin SNMPTTCONVERTMIB

```
MIB: EXOMETER-METRICS-MIB (file:./EXOMETER-METRICS-MIB.mib) converted
on Fri Jul  5 05:29:47 2019 using snmpttconvertmib v1.4

EVENT reportEmsLoggerWriteErrorValue .1.3.6.1.3.7.8 "Status Events" Normal
FORMAT $*
EXEC /usr/local/nagios/libexec/eventhandlers/submit_check_result 127.0.0.1
LOG-ERROR 2 "$*"
SDESC

Variables:
  1: datapointEmsLoggerWriteErrorValue
     Syntax="COUNTER"
EDESC

EVENT reportEmsLoggerWriteInfoValue .1.3.6.1.3.7.10 "Status Events" Normal
FORMAT $*
EXEC /usr/local/nagios/libexec/eventhandlers/submit_check_result 127.0.0.1
LOG-INFO 0 "$*"
SDESC

Variables:
  1: datapointEmsLoggerWriteInfoValue
     Syntax="COUNTER"
EDESC

EVENT reportEmsLoggerWriteWarnValue .1.3.6.1.3.7.11 "Status Events" Normal
FORMAT $*
EXEC /usr/local/nagios/libexec/eventhandlers/submit_check_result 127.0.0.1
LOG-WARN 1 "$*"
SDESC

Variables:
  1: datapointEmsLoggerWriteWarnValue
     Syntax="COUNTER"
EDESC
```

Anexo IV

Registro dos hosts e serviços no arquivo de configuração do Nagios®

```
define service{
        use                             local-service,graphed-service
        host_name                       localhost
        service_description             HTTP
        check_command                   check_http
        notifications_enabled           0
        }

define service{
        name                    emsmetricerror
        use                     generic-service
        register                0
        service_description     LOG-ERROR
        is_volatile             1
        check_command           check-host-alive
        max_check_attempts      1
        normal_check_interval   1
        retry_check_interval    1
        passive_checks_enabled  1
        check_period            none
        notification_interval   31536000
        contact_groups          admins
}

define service{
   host_name               emsbus
   use                     emsmetricerror
   contact_groups          admins
}

define service{
        name                    emsmetricinfo
        use                     generic-service
        register                0
        service_description     LOG-INFO
        is_volatile             1
        check_command           check-host-alive
        max_check_attempts      1
        normal_check_interval   1
        retry_check_interval    1
        passive_checks_enabled  1
```

```
        check_period            none
        notification_interval   31536000
        contact_groups          admins
}

define service{
   host_name               emsbus
   use                     emsmetricinfo
   contact_groups          admins
}

define service{
        name                    emsmetricwarn
        use                     generic-service
        register                0
        service_description     LOG-WARN
        is_volatile             1
        check_command           check-host-alive
        max_check_attempts      1
        normal_check_interval   1
        retry_check_interval    1
        passive_checks_enabled  1
        check_period            none
        notification_interval   31536000
        contact_groups          admins
}

define service{
   host_name               emsbus
   use                     emsmetricwarn
   contact_groups          admins
}
```
