IMAM - Uma ferramenta para monitoramento inteligente de sistemas e dispositivos em infraestruturas críticas de IoT

David Coelho dos Santos[1], **João Carlos Xavier Júnior**[1], **Gibeon Soares de Aquino Júnior**[2]

[1]Instituto Metrópole Digital (IMD)

[2]Departamento de Informática e Matemática Aplicada (DIMAp)
Universidade Federal do Rio Grande do Norte (UFRN)
59078-970 – Natal – RN – Brasil

david.coelho,jcxavier@imd.ufrn.br,gibeon@dimap.ufrn.br

Abstract. *Information systems are critical systems therefore they need to be reliable and always available. In this sense, periods of inactivity can be costly, and also have quite significant consequences. Most of the monitoring tools are able to identify faults but not able to predict them nor to support proactive actions. Aiming to provide fault prediction in critical systems, this paper introduces IMAM, an intelligent system monitoring tool capable of applying Machine Learning techniques to extract knowledge from system and devices fault logs in an IoT infrastructure.*

Resumo. *É fundamental que sistemas de informação estejam sempre disponíveis e sejam confiáveis. Períodos de inatividade podem custar caro e ter consequências bastante significativas. A maioria das ferramentas de monitoramento conseguem identificar falhas, mas não são capazes de predizê-las e nem de realizar ações de suporte proativas. Por essa razão, este artigo apresenta o IMAM, uma ferramenta de monitoramento inteligente de sistemas capaz de utilizar técnicas de aprendizado de máquina para extrair conhecimento a partir de logs de falhas de sistemas e dispositivos em uma infraestrutura de IoT.*

1. Introdução

Atualmente, devido ao aumento da importância e uso frequente de sistemas informatizados, cresce a preocupação com a manutenção da infraestrutura computacional das organizações, já que este é o principal meio para a disponibilização de acesso aos sistemas e serviços ofertados aos usuários [Silva and Torres 2010]. Nesses ambientes, que geralmente possuem infraestrutura crítica, é fundamental que os sistemas de informação estejam sempre disponíveis e sejam confiáveis. Períodos de inatividade desses sistemas podem custar caro e ter consequências bastante significativas nos mais diversos contextos. Em um sistema de cuidados médicos em saúde (*health care system*), por exemplo, em que os pacientes dependem diretamente do funcionamento adequado e contínuo de sistemas e equipamentos, falhas podem significar vidas. Nesses casos (sistemas críticos), falhas devem ser identificadas e tratadas de maneira rápida e eficiente.

Diante da complexidade envolvida nesses ambientes, falhas podem ocorrer devido aos mais diversos motivos. Por isso, muitas ferramentas de monitoramento são utilizadas

para identificar problemas, e acima de tudo, evitar a indisponibilidade desses sistemas. Contudo, detectar, analisar, e sobretudo obter conhecimento a partir de dados coletados de aplicações em situação de falha, tornam-se muito mais importante à medida que deseja-se prever possíveis novos problemas e realizar ações de suporte proativas visando prover tolerância a falhas em sistemas.

O termo "tolerância a falhas" foi apresentado originalmente por Avizienis [Avizienis 1998] em 1967. Vários autores apresentam suas próprias classificações para as técnicas de tolerância a falhas porém, a mais comum é a classificação em 4 fases de aplicação [Anderson and Lee 1981]: detecção, confinamento, recuperação e tratamento.

Nesse contexto, um sistema de monitoramento e detecção de falhas deve ser confiável e possuir métodos capazes de capturar, armazenar, gerenciar, processar e analisar uma grande quantidade dados. Visando atacar os desafios dessa área, o principal objetivo desse artigo é apresentar o IMAM, uma ferramenta de monitoramento inteligente de sistemas capaz de utilizar técnicas descritivas de aprendizado de máquina para extrair conhecimento a partir de *logs* de falhas de sistemas em uma infraestrutura de IoT.

O restante deste artigo está dividido em 7 seções. A seção II descreve alguns conceitos teóricos importantes. A seção III relata os trabalhos relacionados. A seção IV ilustra a arquitetura proposta para o IMAM, enquanto que a seção V apresenta o protótipo desenvolvido para validar esta proposta. A metodologia utilizada no experimento preliminar do presente trabalho e uma análise dos resultados são discutidas na Seção VI. Finalmente, a seção VII apresenta as considerações finais deste trabalho e os trabalhos futuros.

2. Conceitos Relacionados

Nesta seção são abordados conceitos sobre disponibilidade, ferramentas de monitoramento e técnicas de aprendizado de máquina que são utilizados na concepção, especificação e modelagem do IMAM.

2.1. Táticas de Disponibilidade

Antes que qualquer sistema possa agir em relação a uma falha, esta deve ser detectada ou antecipada. As táticas de disponibilidade, portanto, são projetadas para permitir que um sistema preveja a possibilidade de falhas, de modo que um serviço seja entregue e permaneça em conformidade com suas especificações. Assim, cada tática depende essencialmente da detecção de sinais de vida de vários componentes [Bass et al. 2003]. Algumas táticas para detecção de falhas podem ser definidas como:

- Monitor de Sistema: é um componente que é usado para monitorar o estado da saúde de várias outras partes do sistema: processadores, processos, memória, entre outros.
- Ping/echo: refere-se a um par de mensagens de requisição e resposta (*request/response*) assíncronas trocadas entre os nós. O teste de *ping* é frequentemente enviado ou executado por um monitor de sistema.
- *Heartbeat*: é um mecanismo de detecção de falhas que emprega uma troca periódica de mensagens entre um monitor de sistema e um processo que está sendo monitorado.

- *Timestamp*: é uma tática para detectar sequências de eventos incorretas, principalmente em sistemas distribuídos de transmissão de mensagens.
- *Sanity*: esta tática verifica a validade das operações ou saídas específicas de um componente do sistema. Baseia-se em um prévio conhecimento do design interno e/ou da natureza da informação. É frequentemente empregado em interfaces para examinar um fluxo da informação.

2.2. Aprendizado de Máquina - AM

Há uma grande quantidade de algoritmos de aprendizagem de máquinas na literatura e para vários tipos de domínios de problemas. Em geral, esses algoritmos são escolhidos de acordo com o tipo de problema (classificação ou agrupamento). Classificação e Agrupamento (*Clustering*) são tarefas bem conhecidas de aprendizagem de máquinas.

A classificação de dados é o processo de criação de um modelo de previsão a partir de um algoritmo de aprendizagem. O objetivo deste modelo é prever o valor do atributo classe nas instâncias de teste. Por outro lado, Agrupamento procura explorar ou descrever um conjunto de dados. Em geral, não há rótulos pré estabelecidos nos dados, como é caso dos dados oriundos de falhas de sistemas.

Para as tarefas descritivas, serão analisados, com relação à homogeneidade e compacidade, os seguintes algoritmos de Agrupamento (*Clustering*): Expectation-Maximization (EM) [Xu and Wunsch 2005], k-Means [Faceli et al. 2011] e Hierarchical Aglomerativo [Faceli et al. 2011].

Para as tarefas preditivas, serão analisados, com relação à acurácia, os seguintes algoritmos de Classificação: k-NN (k-Nearest Neighbor) [Faceli et al. 2011], MLP (Rede Neural) [Faceli et al. 2011], J48 (Árvore de Decisão) [Faceli et al. 2011], Naive Bayes [Faceli et al. 2011] e SVM (Máquina de Vetores de Suporte) [Mitchell 1997].

2.3. Índices de Validação

A qualidade dos grupos ou *clusters* gerados a partir da utilização dos algoritmos de agrupamentos precisa ser validada sob alguns aspectos importante, tais como: homogeneidade e separação. Há vários índices de validação na literatura. Contudo, os mais comumente utilizados são Davies-Bouldin (DB) e Silhueta [Halkidi et al. 2002].

3. Trabalhos relacionados

Esta seção apresenta trabalhos relacionados a abordagens, métodos e ferramentas de monitoramento de infraestrutura de IoT que utilizam técnicas de Aprendizado de Máquina (AM).

3.1. Soluções de monitoramento em ambientes IoT

Os trabalhos relacionados revelaram que, na maioria dos ambientes baseados em IoT, a disponibilidade das aplicações e dispositivos é fundamental, assim como a necessidade de ferramentas que possam realizar monitoramento e gerenciamento dessa infraestrutura. Portanto, tais ferramentas realizam, basicamente, a coleta de dados relacionados aos recursos computacionais tais como: CPU, RAM e armazenamento. Além disso, são coletadas informações sobre o fluxo de dados na rede de comunicação para fins de detecção de ataques à segurança.

Nesse contexto, Tokuda [Tokuda et al. 2014] propõe o projeto D-Case que realiza monitoramento de redes de sensores ubíquos com detecção rápida de falhas e a recuperação de serviços. Já Yaseen [Yaseen et al. 2017] apresenta um modelo de monitoramento para detecção de ataques de colusão (*collusion attacks*) a partir de experimentos utilizando uma rede de sensores baseada em IoT. A solução utiliza a arquitetura baseada em *fog computing*, que estende a capacidade computacional para a borda da rede.

Através da ferramenta IoT-PIC, Ferrera [Ferrera et al. 2017] apresenta uma solução que realiza o monitoramento, gerência e configuração em tempo real de dispositivos IoT, tais como: gateways, sensores e atuadores. O protocolo XMPP é utilizado como alternativa ao SNMP na gerência e na configuração desses dispositivos. Além disso, o IoT-PIC também permite a descoberta automática de novos componentes e a troca de informações entre objetos IoT.

3.2. AM em sistemas de monitoramento

No trabalho de Chen [Chen et al. 2015], os autores propõem um sistema de monitoramento e detecção de ameaças em rede que utiliza tecnologias como Hadoop, MapReduce e Spark para auxiliar o processamento dos fluxos de dados em conjunto com os algoritmos k-Means (agrupamento) e Naíve Bayes (seleção de atributos). De acordo com os autores, o algoritmo k-Means é utilizado para separar os dados (*logs*) em grupos distintos, e dessa forma, poder identificar comportamentos anômalos na rede. Mas, nenhum mecanismo de tolerância a falhas foi apresentado no mesmo.

Suthaharan [Suthaharan 2014], por sua vez, discute os problemas e desafios em *Big Data* e na predição de intrusões em rede de computadores através do uso de técnicas de aprendizado de máquina. De acordo com o autor, através do uso de *logs* e de técnicas de Aprendizado de Máquina é possível predizer possíveis ataques às redes, mesmo em se tratando de grandes volumes de dados. Porém, para o autor, a escolha do modelo deve ser muito cautelosa. Novamente, nenhum mecanismo de tolerância a falhas foi apresentado no mesmo.

Por último, no trabalho de Soysal [Soysal and Schmidt 2010], os autores empregam três algoritmos de aprendizado de máquina supervisionados, redes bayesianas, árvores de decisão e Perceptrons multicamadas para a classificação de seis tipos diferentes de tráfego de Internet. Embora aplique algoritmos para predição, o mesmo não faz predição de falhas e nem apresenta algum mecanismo de tolerância a falhas foi apresentado no mesmo.

De forma geral, a maioria dos trabalho faz uso de *logs* decorrentes do monitoramento executado. Contudo, somente a minoria utiliza tais *logs* como subsídio para algum mecanismo que possa deixar o ambiente mais seguro. Dentre os trabalhos que utilizam os *logs* para esse propósito (segurança), apenas um faz uso de algoritmos descritivos para identificar comportamentos anômalos na rede, enquanto que um outro utiliza algoritmos preditivos para predizer possíveis ataques às redes. Logo, diferentemente dos trabalhos que utilizam *logs* e técnicas de Aprendizado de Máquina, este trabalho propõe a utilização de ambas as técnicas descritiva e preditiva para a detecção e recuperação de falhas em sistemas e dispositivos em ambiente de IoT.

4. Arquitetura do Sistema

A ferramenta IMAM é um *middleware* capaz de realizar análises em registros de logs de monitoramento de aplicações e dispositivos em ambientes IoT. É composto por 4 componentes principais, são eles: Coletor de Dados de Monitoramento (CDM), Driver, Analisador de Cenários de Monitoramento (ACM) e um Web Dashboard para oferecer ao usuário acesso a configurações, dados e análises, conforme mostrado na Figura 1.

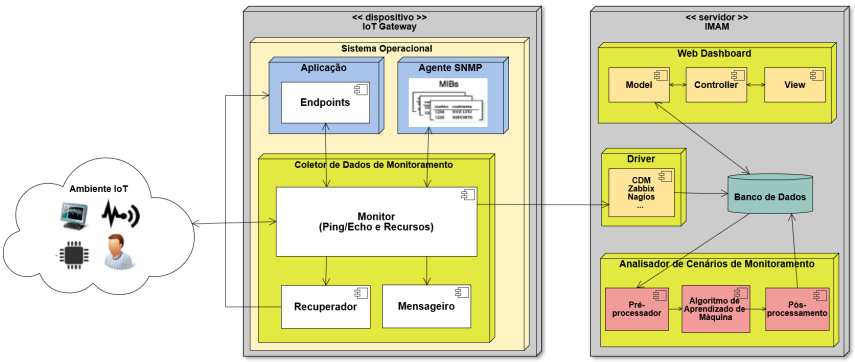

Figura 1. Arquitetura do IMAM.

Além dos componentes principais, o IMAM faz uso de um banco de dados para realizar a persistência dos dados coletados e processados.

4.1. Coletor de Dados de Monitoramento - CDM

A função do CDM é coletar dados, de forma periódica, de sistemas e equipamentos a serem monitorados através de *endpoints* e o protocolo SNMP. Após a coleta, os dados são persistidos no banco de dados. Esse componente também envia mensagens aos usuários e possui função de recuperação de serviços.

4.2. Driver

Tem como objetivo possibilitar a compatibilização do IMAM com as bases de dados de outras aplicações de monitoramento, tais como: Zabbix, Nagios, entre outras. Basicamente, esse componente age como um adaptador, que mapeia elementos contidos no banco de dados de terceiros para o padrão de persistência definido no banco de dados do IMAM.

4.3. Interface Web

O IMAM disponibiliza uma interface Web para permitir que o usuário realize configurações e visualize detalhes relacionados aos dados coletados, processados e analisados através dos demais componentes que compõem o sistema.

4.4. Analizador de Cenários de Monitoramento - ACM

O ACM é o principal componente do IMAM. Ele é responsável por realizar as análises dos *logs* de monitoramento através da utilização de técnicas de aprendizado de máquina. A partir de técnicas de agrupamento o ACM implementa um processo que gera vários grupos, baseado na proximidade das informações contidas a partir dos dados de *logs* recuperados da base de dados. Alertas poderão ser construídos e enviados aos usuários

a fim de informá-los a respeito desses comportamentos, e assim possibilitar a execução de suporte pró ativo em relação a problemas futuros. A Figura 2 mostra os processos necessários para realização de aprendizado, avaliação e predição de falhas.

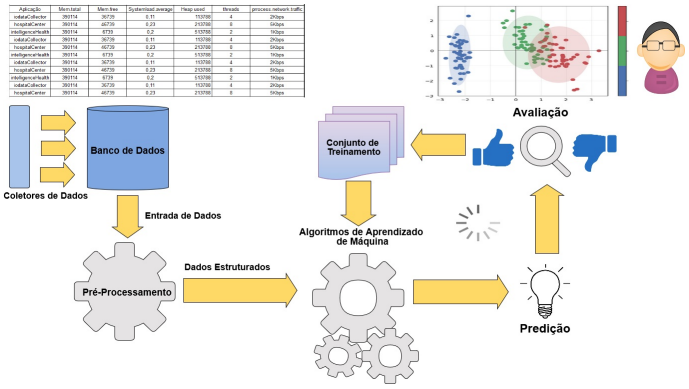

Figura 2. Processo de aprendizado, avaliação e predição de falhas.

O funcionamento do ACM pode ser descrito da seguinte forma:

1. Pré-processa os dados (*logs*) gerados a partir dos Coletores de Dados de Monitoramento (CDM);
2. Aplica técnicas descritivas de Aprendizado de Máquina para gerar grupos distintos, e consequentemente rotular os dados;
3. Utiliza técnicas preditivas para identificar possíveis situações de falhas;
4. Executa ações proativas visando garantir a disponibilidade aos sistemas e dispositivos em um ambiente de IoT.

5. Protótipo

Apesar de algumas ferramentas de monitoramento, como por exemplo o Zabbix, implementarem suporte nativo ao monitoramento de aplicações Java através de JMX (Java Management Extensions), há uma limitação para a realização de captura de dados de *logs* a partir dos vários *endpoints* HTTP disponibilizados por aplicações baseadas em Spring Boot [Webb et al. 2013]. Dessa forma, optou-se pelo desenvolvimento de um protótipo que representa o componente Coletor de Dados de Monitoramento em conjunto com a Interface Web que auxiliarão na configuração do IMAM e na captura, persistência e disponibilização dos dados coletados a partir desses *endpoints*. A Figura 3 mostra esse protótipo em processo de captura de dados de uma aplicação.

Como pode ser visto, a interface web disponibiliza ao usuário a possibilidade de configurações relacionadas a qualquer *endpoint* em aplicações baseadas em Srping Boot. Assim, é possível relacionar os itens desses *endpoints* a aplicações que venham a ser monitoradas pela ferramenta. Além disso, é possível realizar configurações de MIBs(Management Information Base) e OIDs (Object Identifier) para a coleta de dados via protocolo SNMP. A partir dessas configurações, o componente CDM poderá realizar o processo conhecido como *pooling*, em que o dispositivo coletor consulta cada terminal de dados conectado, um por vez, para ler os dados disponíveis para coleta. O usuário poderá relacionar as aplicações e equipamentos a serem monitorados e visualizar o monitoramento em tempo real através de um painel.

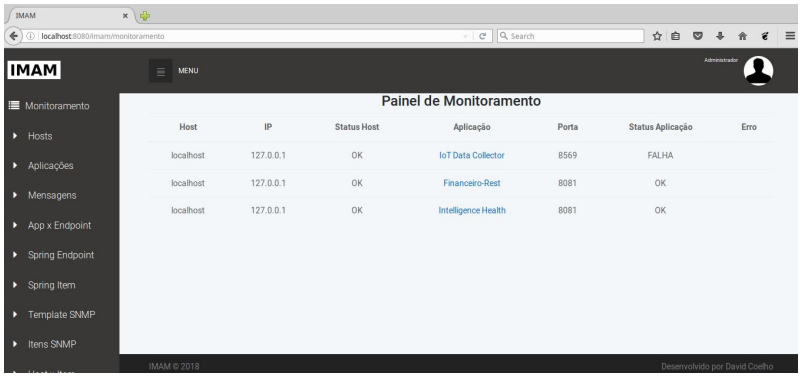

Figura 3. Protótipo do IMAM.

6. Experimento Preliminar

Como ainda não foi possível armazenar uma quantidade razoável de dados de monitoramento de sistemas e equipamentos através da utilização da ferramenta IMAM, optou-se por utilizar uma pequena base de dados de monitoramento de equipamentos de climatização do Datacenter do Instituto Metrópole Digital/UFRN, que foram coletados através do protocolo SNMP e ferramenta Zabbix.

A base de dados, que representa um subconjunto da base principal, é composta por 10 atributos e 8168 registros. Na fase de pré-processamento, procurou-se eliminar as inconsistências e redundâncias nos valores dos atributos, assim como, foi efetuada a normalização de todos os atributos numéricos, mostrados a partir da Tabela 1.

Item	Unidade	Tipo
Equipamento	-	Nominal
Temperatura de sucção	ºC	Numérico
Pressão de sucção	PSI	Numérico
Temperatura de Entrada	ºC	Numérico
Temperatura de Retorno	ºC	Numérico
Temperatura de Suprimento	ºC	Numérico
Velocidade do Ventilador	%	Numérico
Fluxo de Ar	L/s	Numérico
Demanda de Resfriamento	W	Numérico
Status	-	Numérico

Tabela 1. Atributos.

O objetivo desse primeiro experimento foi entender melhor o comportamento desses equipamentos (ar-condicionado) e poder realizar configurações otimizadas, no intuito de obter uma maior eficiência considerando a relação entre a demanda de refrigeração e distribuição da carga em cada célula de racks do Datacenter.

6.1. Resultados

Por não se saber a quantidade ideal de grupos existentes na base a priori, optou-se por usar uma técnica de agrupamento onde o parâmetro k (quantidade de grupos) não fosse determinado, deixando, dessa forma, que o próprio algoritmo encontre esse valor através da distribuição normal dos valores dos próprios atributos.

Devido a esse fato, foi utilizado o algoritmo EM que gerou uma partição com 8 (oito) grupos. A partir dessa descoberta, decidiu-se por avaliar as partições geradas

por outros dois algoritmos de agrupamento (k-Means e Hierárquico Aglomerativo), assim como o próprio EM, variando para tal a quantidade de grupos (k) de 2 até 8. Os algoritmos utilizados estão disponíveis na plataforma de Aprendizado de Máquina Weka [Witten et al. 1999].

Por se tratarem de algoritmos probabilísticos (k-Means e EM), 5 (cinco) execuções diferentes (seeds) foram executadas para cada valor de k, perfazendo um total de 77 (setenta e sete) execuções (k-Means = 35, EM = 35 e Hierárquico = 7). Para cada partição gerada foram utilizadas duas medidas de avaliação (índices internos) que avaliam a compacidade das mesmas. Dessa forma, os resultados dos índices DB e Silhouette foram analisados, visando a escolha da partição mais adequada.

6.2. Validação dos Grupos

Os índices utilizados avaliam como os grupos estão separados, mas possuem características distintas. Porém, é importante esclarecer que o índice DB privilegia valores baixos, ou seja, quanto menor, melhor. Já o índice Silhouette pode gerar valores entre -1 e 1, sendo os valores mais próximos de 1 os valores para os grupos mais bem separados.

Para efeito de melhor entendimento, foram consideradas somente as três melhores partições de cada algoritmo. Dessa forma, como pode ser visualizado na Figura 4, o algoritmo Hierárquico (HR-3K) gerou a melhor partição (três grupos) para ambos os índices (0,79 - DB e 0,61 - Silhouette).

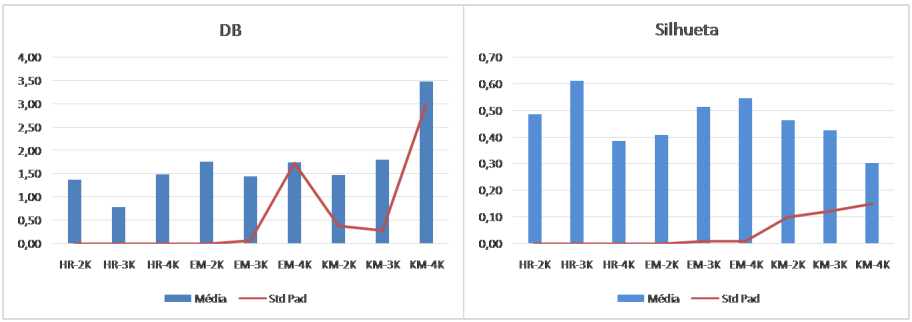

Figura 4. Índices de validação DB e Silhouette.

O gráfico superior central da Figura 5 exibe a relação entre os 3 grupos gerados (HR-3K) e os equipamentos de ar-condicionado. Já o gráfico inferior esquerdo relaciona a demanda de resfriamento com a velocidade dos ventiladores, possibilitando perceber que a demanda de resfriamento é sempre alta para o grupo 3, representado pela cor verde. Por último, no gráfico inferior direito é exibida a relação entre pressão de sucção e temperatura de entrada. Nele, é possível observar que, para o grupo 3, a temperatura de entrada é sempre acima de 50% e a pressão de sucção está constantemente abaixo de 50% quando comparado aos outros grupos.

Em linhas gerais, conclui-se que os equipamentos, representados pelos grupos 1 (azul) e 2 (vermelho), estão trabalhando de forma correta, estável e sem sobrecarga. Os grupos 1 (azul) e 2 (vermelho) representam demandas de refrigeração baixa e média, respectivamente. Por outro lado, o grupo 3 (verde) representa alta demanda por refrigeração, permitindo, dessa forma, que pode haver a ocorrência de problemas em equipamentos e que podem comprometer o bom funcionamento do ambiente do datacenter.

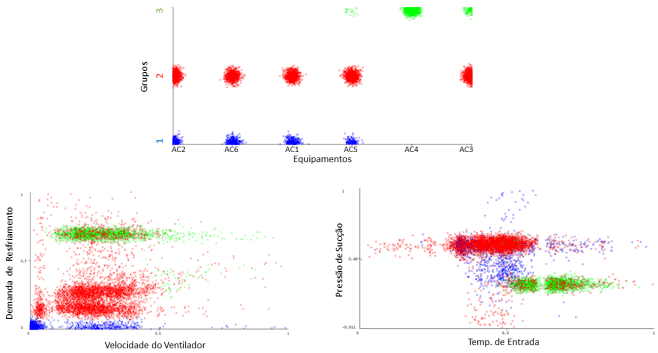

Figura 5. Grupos formados.

A etapa de experimento seguinte seria testar várias técnicas de classificação (Seção 2.2) para a escolha do modelo de classificação mais adequado, que irá compor o componente ACM. Porém, como ainda não temos dados suficientemente gerados pela ferramenta IMAM, decidiu-se executar tal fase um pouco mais tarde.

7. Conclusão e Trabalhos Futuros

A disponibilidade dos serviços de tecnologia é primordial. Dessa forma, sem um monitoramento de infraestrutura em TI automatizado, não há como verificar se algum serviço ou aplicação está indisponível. Logo, automatizar o monitoramento é tão importante quanto otimizar outros processos de gestão.

Visando oferecer suporte a tais exigências, este artigo apresentou a proposta de uma ferramenta de monitoramento que possui a capacidade de monitorar sistemas e equipamentos em uma infraestrutura de IoT, e principalmente, de predizer falhas e tomar ações de suporte proativas. A ferramenta IMAM foi modelada para dar suporte a ações que envolvem tolerância a falhas em sistemas, detectando e corrigindo problemas, reduzindo ou eliminando o tempo de inatividade.

Para tal, utilizaram-se técnicas descritivas e preditivas de aprendizado de máquina para a realização de análises envolvendo sistemas e dispositivos em ambientes baseados em IoT. Dessa forma, a hipótese considera que, a partir da aplicação dessas técnicas, o IMAM possa inferir a respeito de falhas, auxiliando o suporte e a manutenção da disponibilidade e da confiabilidade sobre um conjunto de sistemas e equipamentos.

Um dos desafios na área de tolerância a falhas é o fato de aferir se as técnicas aplicadas irão resultar em um real aumento da confiabilidade de sistemas. Como em grande parte dos sistemas, as falhas podem acontecer de forma aleatória e incontrolável. Uma solução para esse problema é a injeção de falhas.

Assim, com o objetivo de avaliar as funcionalidades do IMAM, é proposto como trabalho futuro a realização de um estudo de caso considerando a utilização de técnicas de injeção de falhas e a atuação da ferramenta IMAM monitorando um conjunto de sistemas e equipemantos de uma plataforma de assistência em saúde baseada em IoT. Dessa forma, a hipótese considera que o IMAM possa detectar falhas, auxiliando o suporte e a manutenção da disponibilidade e da confiabilidade sobre um conjunto de sistemas e equipamentos de uma plataforma de saúde baseada em IoT.

Referências

Anderson, T. and Lee, P. (1981). *Fault tolerance: Principles and practice*. Prentice-Hall International, Inc. Englewood Cliffs, New Jersey, USA.

Avizienis, A. (1998). *Infraestructure-based design of fault-tolerant systems*, volume 98.

Bass, L., Clements, P., and Kazman, R. (2003). *Software Architecture in Practice*. JPearson Education, Inc.,, New York, EUA, 3rd edition.

Chen, Z., Xu, G., Mahalingam, V., Ge, L., Nguyen, J., Yu, W., and Lu, C. (2015). A cloud computing based network monitoring and threat detection system for critical infrastructures. *Big Data Research*, 3:10–23.

Faceli, K., Carvalho, A., Lorena, A. C., and Gama, J. (2011). *Inteligência Artificial: Uma abordagem de aprendizado de máquina*. Grupo Gen-LTC.

Ferrera, E., Conzon, D., Brizzi, P., Rossini, R., Pastrone, C., Jentsch, M., Kool, P., Kamienski, C., and Sadok, D. (2017). Xmpp-based infrastructure for iot network management and rapid services and applications development. *Annals of Telecommunications*, 72(7):443–457.

Halkidi, M., Batistakis, Y., and Vazirgiannis, M. (2002). Clustering validity checking methods: part ii. *SIGMOD Rec.*, 31 (3):19–27.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition.

Silva, P. T. and Torres, C. B. (2010). *Gestão e Liderança Para Profissionais de Ti*. 1yh edition.

Soysal, M. and Schmidt, E. G. (2010). Machine learning algorithms for accurate flow-based network traffic classification: Evaluation and comparison. *Perform. Eval.*, 67(6):451–467.

Suthaharan, S. (2014). Big data classification: Problems and challenges in network intrusion prediction with machine learning. *SIGMETRICS Perform. Eval. Rev.*, 41(4):70–73.

Tokuda, Z., Takuro, Yonezawa, and Nakazawa, J. (2014). Amonitoring dependability of city-scale iot using d-case. *IEEE World Forum on Internet of Things (WF-IoT)*.

Webb, P., Syer, D., Long, J., Nicoll, S., Winch, R., Wilkinson, A., Overdijk, M., Dupuis, C., and Deleuze, S. (2013). Spring boot reference guide. *Part IV. Spring Boot features*, 24.

Witten, I. H., Frank, E., Trigg, L., Hall, M., Holmes, G., and Cunningham, S. J. (1999). Weka: Practical machine learning tools and techniques with java implementations.

Xu, R. and Wunsch, II, D. (2005). Survey of clustering algorithms. *Trans. Neur. Netw.*, 16(3):645–678.

Yaseen, Q., Jararweh, Y., Al-Ayyoub, M., and AlDwairi, M. (2017). Collusion attacks in internet of things: Detection and mitigation using a fog based model. *Sensors Applications Symposium (SAS)*.