

Uma Implementação Distribuída em Névoa do Algoritmo de Detecção de Novidade em Fluxos de Dados MINAS

Luís Henrique Puhl de Souza

Orientador: Prof. Dr. Hermes Senger

05 Julho 2021

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Departamento de Computação

Programa de Pós-Graduação em Ciência da Computação

1. Introdução
2. Fundamentos
3. Estado da Arte e Trabalhos Relacionados
4. Proposta
5. Resultados
6. Conclusão

Introdução

- Crescimento do número de dispositivos IoT e riscos associados;
 - Heterogeneidade de dispositivos;
 - Falta de atualizações de *software*;
 - Exemplo: *Botnet* mirai, infectando cameras e roteadores, gerou 620 Gb/s (KAMBOURAKIS; KOLIAS; STAVROU, 2017).
- Detecção de intrusão em redes:
 - detecção por assinatura *versus* anomalia;
 - ambiente de névoa e redes IoT.
- Um sistema para detecção de intrusão em Redes IoT implementando em névoa;
- A hipótese do trabalho é que o algoritmo MINAS pode ser distribuído em névoa reduzindo a latência e com sem redução na qualidade de classificação.

Fundamentos

- Fluxo de Dados e Métodos Detecção de Novidade;
- Plataformas de processamento distribuído de fluxos.
- Ambientes de computação Distribuída;

Definição de Fluxo de Dados

Um fluxo de dados (*Data Stream*) é uma sequência massiva possivelmente ilimitada de exemplos multi-dimensionais $x_1, x_2, \dots, x_n, \dots$ recebidos em instantes associados $t_1, t_2, \dots, t_n, \dots$

Métodos Detecção de Novidade

Métodos Detecção de Novidade (*Novelty Detection*) lidam com o reconhecimento e classificação de exemplos em padrões que diferem de padrões anteriores (GAMA; RODRIGUES, 2010).

- Evolução de Conceito (*Concept Evolution*): surgimento de um conceito durante o fluxo;
- Mudança de Conceito (*Concept Drift*, deriva ou desvio): modificação da distribuição de um padrão conhecido. A modificação pode ser repentina, incremental ou recorrente;
- Ruído e *Outliers*: que não pertencem a um conceito ou pertencem a um conceito porém estão fora da distribuição conhecida.

Algoritmo MINAS e suas estratégias

- Modelo de aprendizado *Offline-Online*;
- Transformação dos dados analisados para o espaço \mathbb{R}^d ;
- Modelo de classificação com *Clusters*;
- Função de classificação baseada em distância euclideana;
- Algoritmo de agrupamento para identificação de novos padrões;
- Classificação de novos padrões entre recorrência, extensão e novidade;

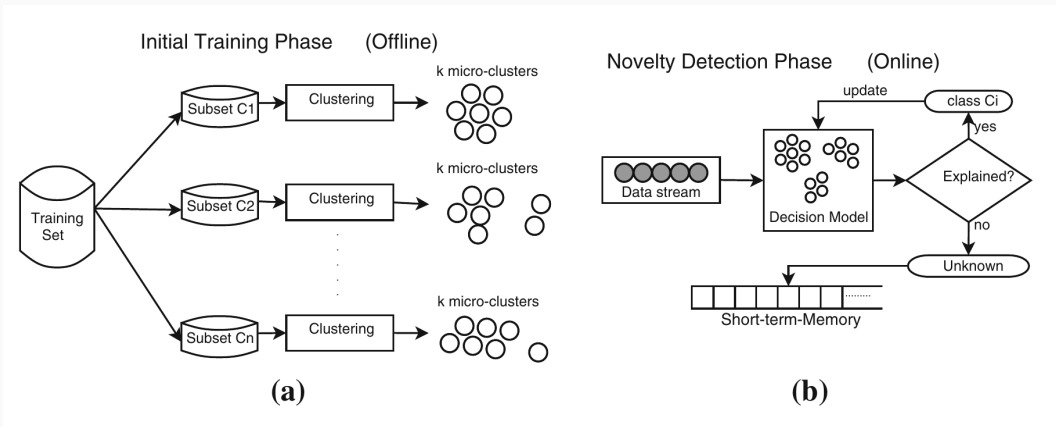


Figura 1: Visão geral do algoritmo MINAS com fases *Offline* (a) e *Online* (b). **Fonte:** Faria, Carvalho e Gama (2016).

```

1 Função MinasOnline(Modelo, fluxoEntrada, fluxoSaída, janelaLimpeza, gatilhoDetecçãoNov):
2   Desconhecidos  $\leftarrow \emptyset$ ; ModeloAntigo  $\leftarrow \emptyset$ ; últimaLimpeza  $\leftarrow 0$ ; proximaNovidade  $\leftarrow 0$ ;
3   para cada exemploi  $\in$  fluxoEntrada faça
4     maisPróximo  $\leftarrow$  clusterMaisPróximo (exemplo, Modelo);
5     se maisPróximo.distância < maisPróximo.cluster.raio então
6       exemplo.rótulo  $\leftarrow$  maisPróximo.cluster.rótulo;
7       maisPróximo.cluster.últimoUso  $\leftarrow i$ ;
8     senão
9       exemplo.rótulo  $\leftarrow$  “desconhecido”;
10      Desconhecidos  $\leftarrow$  Desconhecidos  $\cup$  exemplo;
11      se  $| \text{Desconhecidos} | \geq \text{gatilhoDetecçãoNov}$  então
12        Modelo  $\leftarrow$  Modelo  $\cup$  DetecçãoNovidade (Modelo  $\cup$  ModeloAntigo, *Desconhecidos);
13      se  $i > ( \text{últimaLimpeza} + \text{janelaLimpeza} )$  então
14        Modelo  $\leftarrow$  moveModeloAntigo (Modelo, *ModeloAntigo, últimaLimpeza);
15        Desconhecidos  $\leftarrow$  removeExemplosAntigos (Desconhecidos, últimaLimpeza);
16        últimaLimpeza  $\leftarrow i$ ;
17  fluxoSaída.adicione(exemplo);

```

Algoritmo 1: Interpretação do algoritmo MINAS *online* (FARIA; CARVALHO; GAMA, 2016).

Plataformas de processamento distribuído

- Arquiteturas *Lambda* e *Kappa*;
- Mineração de Dados:
 - *MapReduce* e *Apache Hadoop*;
 - *Apache Spark* com *Resilient Distributed Dataset - RDD*;
- Mineração de Fluxo de Dados:
 - *Apache Spark Streaming* com estratégia de *micro-batching*;
 - *Apache Storm*;
 - *Apache Flink*;
- Não especializadas em fluxo de dados:
 - Não-plataforma (construção dos mecanismos de envio e recebimento);
 - Interface de Troca de Mensagens - *MPI*;

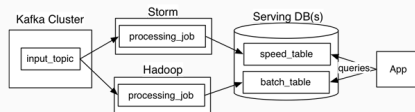


Figura 2: Arquitetura Lambda com Kafka, Storm, Hadoop, SGBD tradicional e aplicação consumidora.

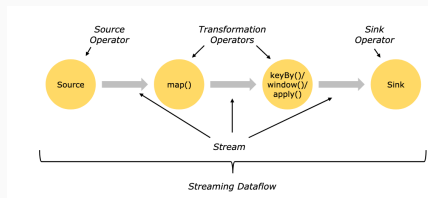


Figura 3: Arquitetura Apache Flink.

- Localidade de dados;
 - Menor número de *page-faults* mantendo o Modelo em cache;
- Memória distribuída e troca de mensagens;
- Padrão MPI - *Message Passing Interface*;
 - Bibliotecas bem estabelecidas;
 - Pares de operações *send/receive*, entre outras operações;
 - Execução gerenciada (*Runtime Environment*, `mpirun`);
- Técnica SPMD - *Single Program Multiple Data*;
 - Construção simplificada;

Ambientes de computação Distribuída

- Computação em Nuvem (*Cloud Computing*) é um modelo que permite acesso conveniente a recursos computacionais compartilhados (MELL; GRANCE, 2012)
- **Características Essenciais:**
 - Auto-serviço sob demanda,
 - Amplo acesso à rede,
 - Agrupamento de recursos,
 - Rápida elasticidade,
 - Serviço mensurado;
- **Modelo de Serviço:**
 - *Software* (SaaS),
 - Plataforma (PaaS),
 - Infraestrutura (IaaS),
- **Implementações:**
 - Nuvem privada,
 - Nuvem comunitária,
 - Nuvem pública,
 - Nuvem híbrida.

Ambientes de computação Distribuída

- Computação de Borda (*Edge Computing*):

Refere-se a qualquer recurso computacional ou de rede entre os dispositivos de borda e centro de dados hospedados em nuvem (SHI et al., 2016).

- Computação em Névoa (*Fog Computing*)

Uma arquitetura horizontal a nível de sistema que distribui funções de computação, armazenamento, controle e rede próximos aos usuários no espaço contínuo nuvem-coisa (IEEE Communications Society, 2018). **Características:**

- Mobilidade,
- Heterogeneidade,
- Baixa Latência,
- Distribuição geográfica,
- Alto número de nós,
- Interoperabilidade e federação,
- Uso de fluxo de dados e aplicações em tempo real.

Estado da Arte e Trabalhos Relacionados

Sistemas de detecção de intrusão em redes

- Ferramenta BigFlow (VIEGAS et al., 2019):
 - + Integração da extração dos descritores de fluxo à emissão de alarmes;
 - + Capacidade de tratamento de grandes volumes;
 - Atualização semanal com avaliação de um especialista;
 - Execução somente em nuvem.
- Ferramenta CATRACA (LOPEZ, 2018; SANZ; LOPEZ, 2018):
 - + Divisão em camadas alocadas em nuvem e névoa;
 - + Modelo de decisão baseado em árvore de decisão;
 - Extração dos descritores de fluxo é feita em névoa, classificação e detecção é feita em nuvem.
- Arquitetura IDSA-IoT (CASSALES et al., 2019):
 - + Avaliação do algoritmo MINAS, ECSMiner e AnyNovel;
 - + Distribuição das tarefas em nuvem e névoa focada em IoT;
 - Implementação e detalhamento da arquitetura em aberto.

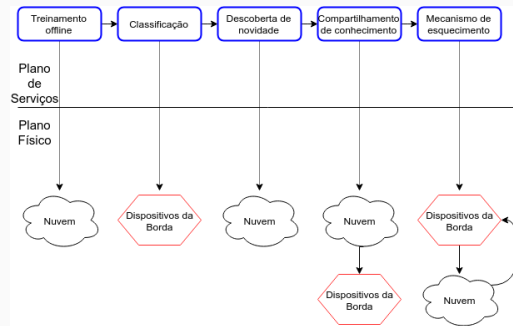
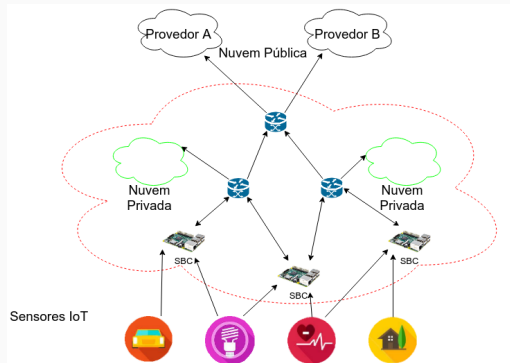


Figura 4: Distribuição de serviços da arquitetura IDSA-IoT. **Fonte:** Cassales et al. (2019).

Proposta

Pergunta de Pesquisa

- É viável implementar a arquitetura IDSA-IoT?
- É viável paralelizar e distribuir o algoritmo MINAS?
- Quais são os efeitos na qualidade de classificação paralelizar e distribuir o algoritmo MINAS?
- Um sistema para detecção de intrusão em Redes IoT implementando em névoa;
- A hipótese do trabalho é que o algoritmo MINAS pode ser distribuído em névoa reduzindo a latência e com sem redução na qualidade de classificação.

Proposta da Pesquisa

- Implementar a distribuição do algoritmo MINAS em nuvem e névoa conforme arquitetura IDSA-IoT;
- Paralelizar o método de classificação do algoritmo MINAS.

Métodologia

- Plataforma de processamento distribuído;
- Estratégias de implementação da arquitetura IDSA-IoT;
- Experimentação com a distribuição do algoritmo MINAS em ambientes;
- Métricas de qualidade de classificação para validação da implementação;
- Métricas de escalabilidade.

O sistema M-FOG é dividido em 5 módulos subdivididos em 2 grupos.

Módulos principais implementam o algoritmo MINAS

- TODO;

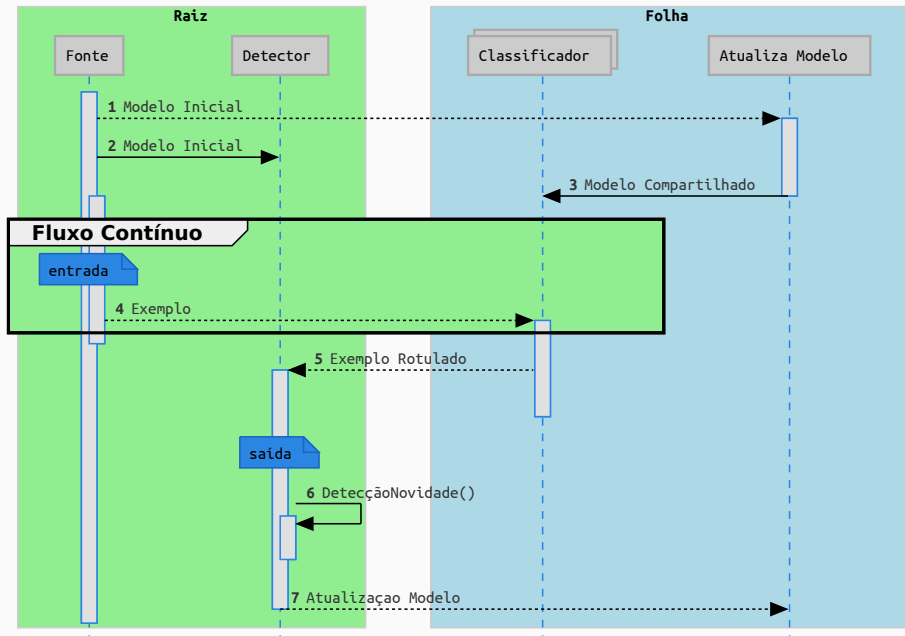


Figura 5: Arquitetura e fluxos de dados do sistema M-FOG.

Resultados

Primeira Implementação com Python e Apache Kafka

- *Python* é acessível e fornece bibliotecas diversas;
- *Apache Kafka* é um sistema de mensagens distribuído;
 - Interface de programação com cliente produtor e consumidor;
 - Mensagens organizadas em tópicos que são distribuídos em partições;
- A hipótese de que a carga seria distribuída entre os consumidores, uma vez que o consumidor pode selecionar uma partição para leitura;
- Em experimento com um produtor, 8 partições e 8 consumidores, observou-se que um consumidor processava a maior parte das mensagens, poucos consumidores recebiam algumas mensagens e a maioria dos consumidores não recebia mensagem alguma.

Segunda Implementação com Apache Flink

- Implementação escrita em Scala ou Java;
- Processamento de fluxos *Stateful*;
- Falta de bibliotecas que distribuam algoritmos base como *K-means*;
- Gerenciador de trabalhos (*job manager*) e gerenciador de tarefas (*task manager*) ocupam mais de 1 GB em execuções consecutivas, portanto não é confiável para dispositivos pequenos.

Métricas e Ambientes

- Métricas de qualidade de classificação:
 - Avaliação do fluxo de saída do classificador;
 - Uso de uma matriz de confusão ou erro;
 - Taxa de desconhecidos;
 - Macro F-score;

$$\mathbf{E}_n = \begin{pmatrix} e_{1,1} & e_{1,2} & \cdots & e_{1,J} \\ e_{2,1} & e_{2,2} & \cdots & e_{2,J} \\ \vdots & \vdots & \ddots & \vdots \\ e_{M,1} & e_{M,2} & \cdots & e_{M,J} \end{pmatrix}$$

$$UnkR_n = \frac{1}{M} \sum_{i=1}^M \frac{\#Unk_i}{\#ExC_i}$$

$$Fscore1_n = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

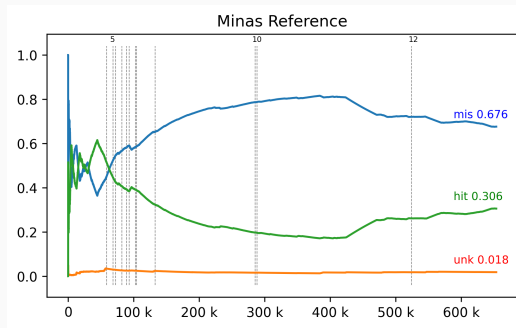
Métricas e Ambientes

- Métricas de escalabilidade:
 - Número e tipo de processadores;
 - Uso de memória;
 - Tempo de processamento;
 - Taxa de eventos;
 - Latência entre a produção e classificação.
- Ambientes de teste:
 - Computador Pessoal (para desenvolvimento);
 - Nevoa composta de SBC (*Single Board Computer*) ARM 4 núcleos;
 - Conjunto de dados para IDS, Kyoto 2006+, segmento dezembro de 2015 como estabelecido por Cassales et al. (2019).

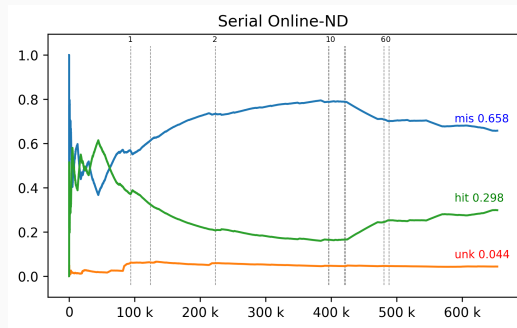
Experimentos

Experimento	Programa	Características
<i>a-Referência</i>	MINAS referência 2013	Raio é a distância máxima.
<i>b-Sequencial</i>	MINAS sequencial para validação	Raio é o desvio padrão das distâncias; Modelo único; Remoção de desconhecidos mais agressivo.
<i>c-Paralelo</i>	sistema M-FOG 1 nó, 4 processadores	Classificadores paralelos; Detecção de novidade assíncrona.
<i>d-Distribuído</i>	sistema M-FOG 3 nós, 12 processadores	Mais processadores; Comunicação em rede.

Tabela 1: Listagem dos principais experimentos.

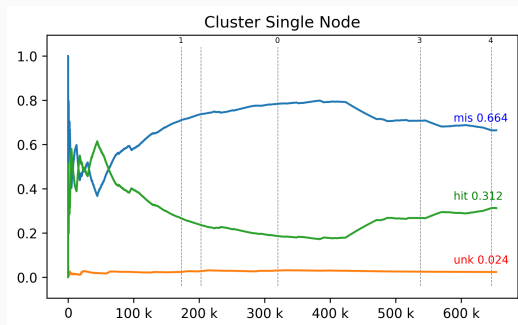


(a) Experimento *a-Referência*, implementação de referência do algoritmo MINAS.

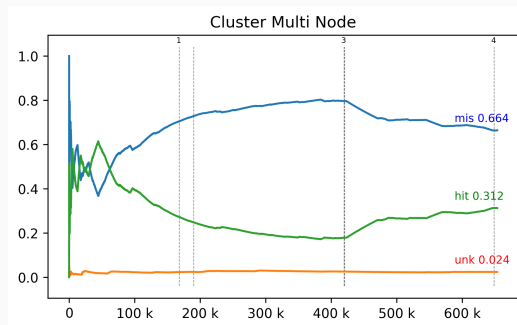


(b) Experimento *b-Sequencial*, sistema M-FOG sequencial.

Figura 6: Visualização de fluxo do conjunto *Kyoto Dez.* 2015.



(a) Experimento *c-Paralelo*, sistema M-FOG com 1 nó e 4 núcleos.



(b) Experimento *d-Distribuído*, sistema M-FOG com 3 nós de 4 núcleos cada.

Figura 7: Visualização de fluxo do conjunto *Kyoto* Dez. 2015.

Resultados - Experimentos Adicionais

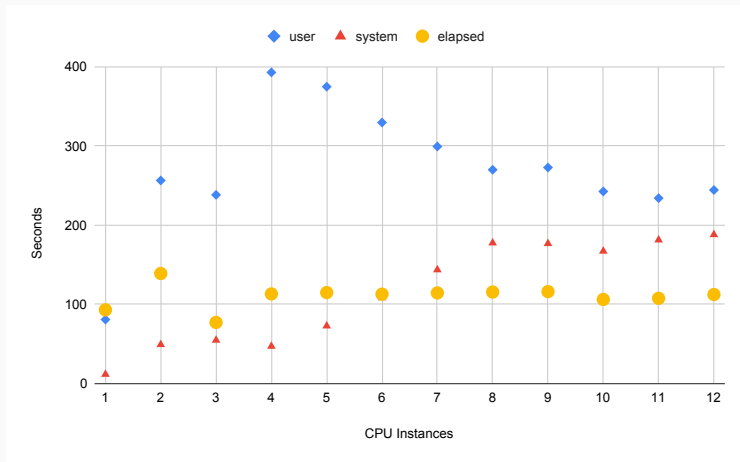
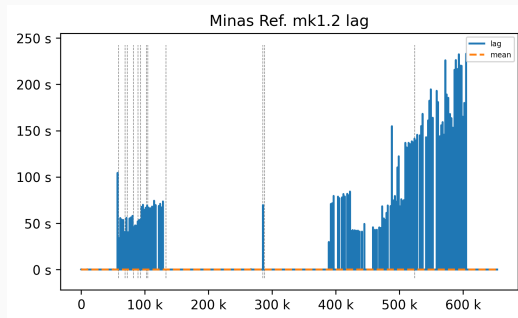
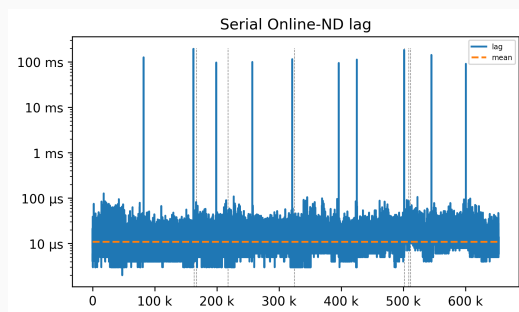


Figura 8: Métricas de tempo para execuções do sistema M-FOG com variação no número de processadores.

Resultados - Experimentos Adicionais

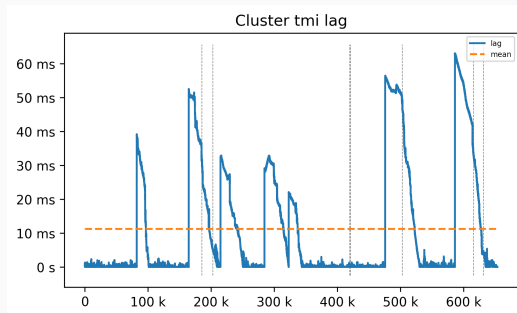


(a) Implementação de referência.



(b) Implementação sequencial.

Figura 9: Visualização de Latência das implementações de referência, sequencial e paralela do algoritmo MINAS.



(a) Implementação paralela.


Figura 10: Visualização de Latência das implementações de referência, sequencial e paralela do algoritmo MINAS.


Conclusão


- sistema M-FOG funciona;
- Distribuição tem efeito mas não é tão drástico;
- Não escala pelo CCR e eficiência;
- Trabalhos futuros:
 - Outros algoritmos de agrupamento (CluStream);
 - Estratégia de otimização da comunicação (micro ou mini batching);
 - Explorar distribuição espacial dos clusters (polígonos sem sobreposição, árvore de busca);
 - Algoritmo com modelo de tamanho fixo (máxima precisão com recursos disponíveis);
 - Modelos com propriedade de conjuntos aditivos para sincronização entre redes distintas.

- ICCSA 2021 (PUHL et al., 2021);
- GitHub;


Obrigado!


 CASSALES, G. W. et al. Ilsa-iot: An intrusion detection system architecture for iot networks. In: *2019 IEEE Symposium on Computers and Communications (ISCC)*. [s.n.], 2019. p. 1–7. ISBN 978-1-7281-2999-0. ISSN 1530-1346. Disponível em: <https://ieeexplore.ieee.org/document/8969609/>.


 FARIA, E. R. de; CARVALHO, A. C. Ponce de L. F.; GAMA, J. Minas: multiclass learning algorithm for novelty detection in data streams. *Data Mining and Knowledge Discovery*, v. 30, n. 3, p. 640–680, May 2016. ISSN 1573-756X. Disponível em: <https://doi.org/10.1007/s10618-015-0433-y>.


 GAMA, J.; RODRIGUES, P. P. *Knowledge Discovery from Data Streams*. [S.l.]: Chapman and Hall/CRC, 2010. ISBN 9781439826119.


 IEEE Communications Society. *IEEE Std 1934-2018: IEEE Standard for Adoption of OpenFog Reference Architecture for Fog Computing*. IEEE, 2018. 176 p. ISBN 9781504450171. Disponível em: <https://ieeexplore.ieee.org/document/8423800>.


 KAMBOURAKIS, G.; KOLIAS, C.; STAVROU, A. The Mirai botnet and the IoT Zombie Armies. In: *MILCOM 2017 - 2017 IEEE Military Communications Conference (MILCOM)*. IEEE, 2017. v. 2017-Octob, p. 267–272. ISBN 978-1-5386-0595-0. Disponível em: <http://ieeexplore.ieee.org/document/8170867/>.


 LOPEZ, M. E. A. *A monitoring and threat detection system using stream processing as a virtual function for Big Data*. Tese (Theses) — Sorbonne Université ; Universidade federal do Rio de Janeiro, Jun 2018. Disponível em: <https://tel.archives-ouvertes.fr/tel-02111017>.

 MELL, P.; GRANCE, T. The NIST definition of cloud computing: Recommendations of the National Institute of Standards and Technology. In: NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. *Public Cloud Computing: Security and Privacy Guidelines*. National Institute of Standards and Technology, 2012. p. 97–101. ISBN 9781620819821. Disponível em: <http://faculty.winthrop.edu/domanm/csci411/Handouts/NIST.pdf>.

 PUHL, L. et al. Distributed novelty detection at the edge for iot network security. In: *Computational Science and Its Applications – ICCSA 2021*. [S.l.]: Springer International Publishing, 2021.

 SANZ, I. J.; LOPEZ, M. A. Um sistema de detecção de ameaças distribuídas de rede baseado em aprendizagem por grafos. In: *Anais do XXXVI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. Porto Alegre, RS, Brasil: SBC, 2018. ISSN 2177-9384. Disponível em: [〈https://sol.sbc.org.br/index.php/sbrc/article/view/2487〉](https://sol.sbc.org.br/index.php/sbrc/article/view/2487).

 SHI, W. et al. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, Institute of Electrical and Electronics Engineers Inc., v. 3, n. 5, p. 637–646, oct 2016. ISSN 23274662. Disponível em: [〈https://ieeexplore.ieee.org/abstract/document/7488250〉](https://ieeexplore.ieee.org/abstract/document/7488250).

 VIEGAS, E. et al. Bigflow: Real-time and reliable anomaly-based intrusion detection for high-speed networks. *Future Generation Computer Systems*, Elsevier, v. 93, p. 473 – 485, 2019. ISSN 0167-739X. Disponível em: [〈http://www.sciencedirect.com/science/article/pii/S0167739X18307635〉](http://www.sciencedirect.com/science/article/pii/S0167739X18307635).

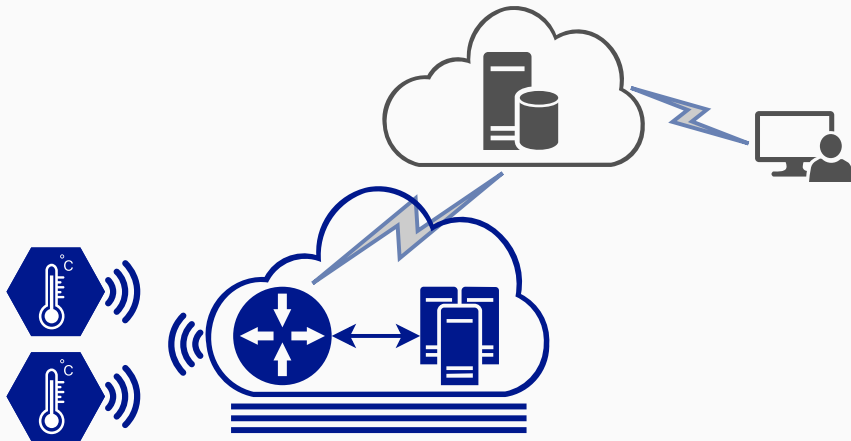


Figura 11: Arquitetura IoT tradicional.