IoT Intrusion Detection with Distributed Novelty Detection: Design, Implementation and Evaluation [DRAFT]

Luís Puhl, Guilherme Weigert Cassales, Hermes Senger, Helio Crestana Guardia
Universidade Federal de São Carlos, Brasil
Email: {luispuhl, gwcassales}@gmail.com, {hermes@, helio@dc.}ufscar.br

***Abstract*—The implementation of the Internet of Things (IoT) is sharply increasing the small devices count and variety on edge networks and, following this increase the attack opportunities for hostile agents also increases, requiring more from the network administrator role and the need for tools to detect and react to those threats. One such tool are the Network Intrusion Detection Systems (NIDS) where the network traffic is captured and analysed raising alarms when a known attack pattern or new pattern is detected. For a network security tool to operate in the context of edge and IoT it has to comply with processing time, storage space and energy requirements alongside traditional requirements for stream and network analysis like accuracy and scalability. This work addresses the construction details and evaluation of an prototype distributed IDS using MINAS Novelty Detection algorithm following up the previously defined IDSA-IoT architecture. We discuss the algorithm steps, how it can be deployed in a distributed environment, the impacts on the accuracy of MINAS and evaluate the performance and scalability using a cluster of constrained devices commonly found in IoT scenarios. We found an increase of *A 0.0* processed network flow descriptors per core added to the cluster. Also *B 0.0%* and *C 0.0%* change in *F1Score* in the tested datasets when stream was unlimited in speed and limited to *0.0 z MB/s* respectively.**

***Index Terms*—novelty detection, intrusion detection, data streams, distributed system, edge computing, internet of things**

I. Introduction

The advent of Internet of Things (IoT) is growing the count and diversity of devices on edge networks, this growth increases network traffic patterns and extends opportunities for cyber attacks presenting new challenges for network administrators. To address those challenges new Network Intrusion Detection Systems (NIDS) and architectures can be explored, especially in Fog Computing and Data Stream (DS) areas.

Expected results: A system that embraces and explores the inherent distribution of fog computing in a IoT scenario opposing regular systems where data streams are collected and centralized before processing; Impact assessment of the impact of distributed, regional flow characteristics, local vs global vs distributed forgetting mechanism and other polices.

IDS characteristics and description of physical scenario.

MINAS characteristics.

Distribution and IDSA-IoT architecture.

CNPq

This paper is structured as follows: Section II presents previous works that addresses related problems and how they influenced our solution. Section IV address our proposal, the work done, issues found during implementation and discusses parameters and configurations options and how we arrived at our choices. Section V shows experiments layouts and results, we compare serial and distributed implementation's metrics for validation, we also evaluate communication delay effects on classification metrics and conclude with the speedup per core and overall maximum stream speed. Section VI summarizes the research results and presents our final conclusions and future works.

II. Related Work

Recent works explored those areas, to name a few: BigFlow [?] employing Apache Kafka and Apache Flink for distributed stream processing evaluating with package stream dataset, CATRACA [?], [?] uses Apache Kafka and Apache Spark for stream processing and

III. Proposal

Fog computing infrastructure aims to offload computing resources from cloud providers by placing edge devices closer to end-users and/or data sources.

Distributed novelty detection in streams using limited hardware.

To achieve

IV. Implementation

The original MINAS algorithm has a companion implementation (*Ref*) written in Java using MOA library base algorithms such as K-means and CluStream. *Ref* employs Java's double, a $64bits$ number whose precision is not absolutely necessary and, as it is often necessary to shuffle between nodes via network and a small economy could be made with only a float number with $32bits$. Another difference between *Ref* and *MFOG* is cluster radius calculation from the distances of elements forming the cluster and the cluster's center, where the former uses the maximum distance, the latter uses the standard deviation of all distances as described in [?].

The stream format for input and output also of note. Input information needed is the value of the item, this value is a

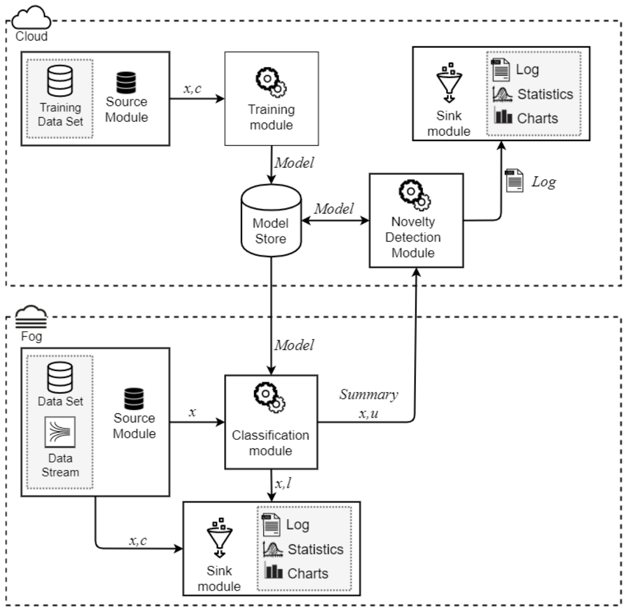

Fig. 1. *MFOG* architecture overview.

number sequence of length d (referenced as dimension). In addition to the value for evaluation and training purposes the class identifier as single character, optimality an unique item identifier (*uid*) can be provided. For output information and format the decision isn't so clear as we can't predict future system integrations needs like only novelty alarms or every item's original value with assigned label so, we have a compromise and put only enough information for the Evaluation Module (where the full information from the testing file or stream can de accessed) meaning the format can be defined as a tuple containing *uid* and assigned label.

Another implementation decision related to the output stream is whether or not to reprocess, and add to the output stream, examples in the unknown buffer after the novelty detection procedure, meaning one item can be classified once as unknown and again with a label. Our tests using this technique had increased true positives when compared to not using it. However this changes the stream operator behavior from a *Map* to a *FlatMap* having duplicate entries on the output stream as previously mentioned. Regardless of choice the classification of the unknown buffer after a model update, using the full model or just the added set of clusters, is done to remove the examples "consumed" in the creation of a new cluster in the internals of the clustering algorithm.

Polices

- Detecção de novidades e manutenção de modelo em ambiente distribuído:

- Mecanismo de ND local (síncrono) vs nuvem quanto à atraso de definição de modelo (nesse ponto é onde a hipótese prevê maior diferença, grande ponto de interesse);

- Mecanismo de esquecimento local vs global (modelo único ou por nó);

- Atraso na reclassificação dos desconhecidos;

The Evaluation Module was also build following reference techniques like multi-class confusion matrix with label-class association [**?**] to extract classification quality metrics.

V. Experimental Setup and Results

The experimental setup is composed of 2 environments and 3 datasets. Kyoto December 2015.

For the experiments, we used the Kyoto 2006+ dataset which contains data collected from 2006 to December 2015. We selected examples from one month, December, 2015. Only the examples of known attack types and known IDS alert code with a minimum of 10,000 occurrences (for significance) were considered. The offline training was performed with 72,000 examples (i.e., 10% of the dataset) using the holdout technique. [**?**]

	C N	**C A**
C N	181391_h	437837_m
N 1	0_m	123_h
N 2	13_m	35_h
N 3	0_m	6_h
N 4	43_m	483_h
N 5	0_m	52_h
N 6	0_m	164_h
N 7	314_h	2_m
N 8	97_m	939_h
N 9	826_m	2133_h
N 10	13887_h	3752_m
N 11	142_m	349_h
N 12	5793_h	1121_m
N 13	35_h	0_m
N 14	10_m	39_h
Unk	3727_u	144_u
Metric	**Value**	**Ratio**
Total output	653457	
Hits	205743	0.314853158
Misses	443843	0.679222963
Unknowns	3871	0.005923879
FNew	12.064786	
MNew	97.910904	
Err	70.811700	

Classes (act) **Labels (pred)**	**A**	**N**
-	3774_u	8206_u
1	123_h	0_m
10	2489_m	4066_h
11	71_m	289_h
12	26_h	0_m
2	145_h	79_m
3	368_h	44_m
4	8_h	0_m
5	52_h	0_m
6	165_h	0_m
7	1_m	229_h
8	1046_h	181_m
9	161_h	154_m
N	438750_m	193030_h
Metric	**Value**	**Ratio**
Total input	653457	
Total output	653457	
Hits	199708	0.30561766
Misses	441769	0.67604907
Unknowns	11980	0.01833326
Reprocessed	0	0.00000000

VI. Conclusion

Acknowledgment

The authors thank CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Código de Financiamento 001). Hermes Senger also thanks CNPq (Contract 305032/2015-1) and FAPESP (Contract 2018/00452-2, and Contract 2015/24461-2) for their support.