Les Cahiers du GERAD ISSN: 0711–2440

eNodeB failure detection from aggregated performance KPIs in smart-city LTE infrastructures

O. Manzanilla-Salazar,
F. Malandra, B. Sansò

G–2018–79

October 2018

La publication de ces rapports de recherche est rendue possible grâce au soutien de HEC Montréal, Polytechnique Montréal, Université McGill, Université du Québec à Montréal, ainsi que du Fonds de recherche du Québec – Nature et technologies.

Dépôt légal – Bibliothèque et Archives nationales du Québec, 2018
– Bibliothèque et Archives Canada, 2018

The publication of these research reports is made possible thanks to the support of HEC Montréal, Polytechnique Montréal, McGill University, Université du Québec à Montréal, as well as the Fonds de recherche du Québec – Nature et technologies.

Legal deposit – Bibliothèque et Archives nationales du Québec, 2018
– Library and Archives Canada, 2018

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

eNodeB failure detection from aggregated performance KPIs in smart-city LTE infrastructures

Orestes Manzanilla-Salazar
Filippo Malandra
Brunilde Sansò

GERAD & Electrical Engineering Department, Polytechnique Montréal (Québec) Canada, H3C 3A7

orestes.manzanilla@polymtl.ca
filippo.malandra@polymtl.ca
brunilde.sanso@polymtl.ca

October 2018
Les Cahiers du GERAD
G–2018–79

Abstract: In this paper we tackle the problem of eNodeB failure detection in LTE networks using Binary Classification techniques under smart-cities Machine-to-Machine (M2M) traffic. We train 20 different classifiers with data from two 24 hrs simulations with different traffic volume levels. Input features for the classification models are built aggregating packet generation and access collisions from the eNodeB on which failures are being detected, as well as from its closest neighbors, by computing statistics for each time-bin. Network service providers generally maintain network performance data by processing real-time data to produce periodic aggregated summaries, which in practice constitutes a filter on the data, reducing the quantity of information available for inference. We explore the effect of different levels of granularity in data aggregation and their effect on our ability to detect failures. We gathered data from M2M traffic along an LTE network simulated using publicly available geographic city data on Montreal, Canada. With Linear Support Vector Machines (L-SVMs) and Bagged Decision Trees (BDT), failure detection rates above 97.5 % were achieved, with false positive rates under 2.8 %, showing that, even in 30 minutes aggregations, it is feasible to extract meaningful data from aggregations of data from LTE networks with M2M traffic.

Keywords: MTC (Machine Type Communications), M2M (Machine-to-machine) communications, failure detection, LTE networks, machine learning, smart city, Internet-of-things

Acknowledgments: This work was funded through grant CRDPJ 520642 between NSERC and Ericsson.

1 Introduction

A smart city relies on a very large number of *smart devices*, such as sensors, actuators, and smart watches, whose number is expected to reach 25 billions by 2020: the interconnection of those devices is usually referred to as the Internet of Things (IoT). What makes these devices *smart* is their ability to communicate without human supervision: this type of communication is commonly grouped under the category of Machine-to-machine (M2M) or Machine Type Communication (MTC) communications.

M2M or MTCs are expected to use existing communication infrastructure and the traffic volume is expected to be non-trivial, especially in densely populated smart cities. Machine-generated traffic has some peculiarities, because it depends on the very different behaviour of machine applications rather than well-studied human behavior. Moreover, the *data size* is usually small in M2M communications, which mainly take place in the uplink direction.

LTE is one of the most popular candidate solutions to support M2M communications, thanks to its large bandwidth availability and its ubiquitous coverage. In the latest specifications of LTE, such as LTE-M and NB-IoT, 3GPP is working on reducing the bandwidth requirements for machines: this would reduce the power consumption, which is particularly sought by small devices in IoT networks. However, a large number of different M2M applications are expected to be using LTE and a diverse set of different machines are requiring LTE access. Therefore, it is important to study the ability of LTE to support M2M traffic [1].

Even though the size of messages in M2M communications is very small, the number of communicating devices is considerably large and this would be a burden for the access nodes in the LTE architecture, i.e. eNodeBs. The increased load can degrade network performance—due to the limited number of available preambles— and also increase the probability of hardware/software failures. Many M2M applications require high reliability and cannot afford long outages or large delays, due to eNodeB failures. Therefore, the prompt detection of those events is crucial for network maintainance and efficiency, and therefore for expedite delivery of M2M messages in an IoT setting.

The massive stream of data generated by network activity is difficult to handle for service providers [2]. Network data are subdivided in time intervals and, for each time interval, only counters and basic statistics, such as minima, maxima and averages are traced. Even though this aggregation permits to considerably reduce the network data, it entails a loss of information. Specifically, original probability distributions are lost, and we are left to study the behavior of the means of variables instead of the variables themselves.

In particular, eNodeB failures can take time and effort to be detected [3]: the delay in the detection can cost money to the network operator and jeopardize the operation of smart city applications. The main objectives of this paper are to analyze traffic patterns associated to an eNodeB failure and to provide models and algorithms to detect failure events. For this purpose, we use the LTE network simulator proposed in [1], which employs real geographic data on the position of the machines and eNodeBs and permits to evaluate the LTE network performance in a large-scale smart city environment.

The simulator allows us to control the different types of events that can take place in the network and to jointly study different types of IoT applications. However, such a fine knowledge is not always available in real-life traffic monitoring products, where only largely aggregated Key Performance Indicators (KPIs) and statistics per minutes, hours or days are available to the operator. Therefore, an important aspect of this paper is to assess the effects of data aggregation on failure detection using machine learning methods.

Summarizing, the paper presents the following original contributions:

1. An assessment of the feasibility of detecting failures in a specific eNodeB of an LTE network carrying M2M traffic for IoT using binary classification techniques.
2. A study of the effects of aggregation levels and traffic volumes on automatic failure detections techniques.

2 Related work

A good deal of research has investigated the problem of failure detection and diagnosis in LTE networks. The large majority of the surveyed work has focused on Human Type Communications (HTCs). We now mention some of the work done in failure detection and diagnosis Self-Organizing Networks (SONs) tasks, mostly in LTE networks. It should be pointed out that all the techniques surveyed are based mainly on HTC, and no distinction between it and M2M or MTC is made.

2.1 Fault detection

Fault detection is defined as the task of identifying the cell, base station, or eNodeB experiencing problems [3]. A fault detection method determines on which network nodes the self-healing functions should be focused. It refers to both outages and degradations. The fault detection methods can be defined as *proactive*, if their goal is to anticipate the effects of the failure in the traffic behavior, or *reactive*, if the detection occurs after the behavior is degraded [3].

A cell in a mobile network can be classified as follows [4]:

- *Degraded cell:* when it is operational but performance values are low with respect to some reference.
- *Crippled cell:* when a failure in the base station severely affects the capacity of the cell. Some traffic is observed both from and towards the cell.
- *Catatonic cell:* when a serious software of hardware failure totally impedes traffic in at least one direction (uplink or downlink).

In general, the detection task is the easiest of the self-healing functions, as alarms, KPIs thresholds, and deviation from normal-behaviour profiles signal the occurrence of many kinds of failures [3]. Failures of sleeping cells, however, are considered a harder problem [2], because of the absence of alarms or messages indicating the degradation or outage of the cell. Operators rely only on the observation of the problematic cell KPIs and the perturbation it generates on its neighborhood, as the User Equipments (UEs) in the area of the failing cell start establishing connections to nearby cells instead. This kind of failures can stay undetected and undiagnosed for days [3].

Random-Access Channel (RACH) originated sleeping cells have the particularity of producing problems of service availability for UEs without noticeable lacks of radio signal coverage [5]. These failures can be caused by configuration problems level, excessive traffic load, software problems, or firmware issues in an eNodeB. At the eNodeB level, KPI thresholds frequently fail to trigger alarms, even though the QoE deteriorates from the user point of view [5]. New users fail to connect while established connections continue working. Eventually, connections end and the eNodeB enters a catatonic state [5]. Given the long time that detection process can take, and the bad experience of the users, RACH sleeping cells are recognized as a hard and relevant for the operators interested in preventing subscriber's churn [6].

Fault detection algorithms are important, as most fault diagnosis automatic methods show a non-trivial quantity of false positives (such as in [7]), and some have good results after receiving data filtered by a fault detector [8].

Data Analysis approaches to fault detection are usually grouped into two categories: those using cell data, and those using UE data. Most of the work on fault detection is based on data from cells [5]. A typical strategy is to observe statistical deviations from the cells "normal" performance [9]. This implies making assumptions about the outage causes and symptoms, which might not be applicable to all situations. Statistics and performance KPIs can also allow a Base Station (BS) to detect a failure in a neighbor BS, by considering the frequency of requests for registration along with Channel Quality Indicator (CQI)ś distribution and time correlation [10]. Most techniques have a specific granularity in their data time-aggregation, however, as performance anomalies can take place at different time-scales, anomaly detection techniques can also be applied simultaneously at different levels of aggregation (*multi-resolution*) [11]. Besides analyzing performance KPIs, data related to interactions with other cells can be used, such as the number of hand-overs with adjacent cells [12] and neighbor cell list reports [13]. Binary classification techniques can also be used as proposed by [10] and [14].

Deviations from "normal" behavior have also been used for UE-based fault detection. In [15] Diffussion Maps are used to reduce the dimensionality of the UE performance data, and data-points that lie in a low density area of the feature-space are considered as possible "faulty" anomalies. A clustering phase is used to identify whether an anomaly corresponds or not to a failure state, which is important because instances of exceptionally good performance can also be considered anomalies. Another clustering approach was proposed in [16] where simulated data is analyzed via an Affinity Propagation algorithm. Reference Signals Received Power (RSRP) and Reference Signals Received Quality (RSRQ) are reported by the UE for serving and adjacent cells and UE geographical location is successfully used to identify the cells at fault. A different approach is given in [5], where events reported by calls in an LTE network are analyzed via an N-grams technique for anomaly detection. They apply dimensionality reduction techniques and k-Nearest Neighbors to estimate the level of probability of each sequence. Sleeping Cells are identified by analyzing aggregations of indicators based on probability scores from adjacent cells. A salient feature of this work is that it takes only 10 minutes of real-time data for this algorithm to start working as a practical detector.

2.2 Fault diagnosis

The diagnosis of a fault involves finding the cause of the network problem [3]. The resulting diagnosis is important for the fault-recovery function, in the case of SONs, and for the experts carrying out the troubleshooting.

One of the problems of using supervised learning for fault diagnosis based on cell KPI values is that service providers seldom register the cause of the faults after troubleshooting [17], and the way KPIs are stored might not be ideal to take advantage of the data [18]. This has motivated research such as [19], with the goal of facilitating the non-intrusive acquisition of experts knowledge during their troubleshooting workflow, creating a knowledge base that can be used by fault-diagnosis algorithms. In [20], KPI values are discretized via thresholds, in order to use Bayesian Networks to perform a diagnosis of the cells, though nearly one third of the non-faulty cells were falsely diagnosed with some type of failure. In [13] the neighbor cell list reports are used to analyze the changes in the visibility graph to detect the failing cells, via binary classifiers (Decision Trees and Linear-Discriminant Functions), achieving good detection results but an impractical level of false positive rate even after penalizing false positive classification errors.

Fuzzy models have been used to mine troubleshooting databases to diagnose [7] network problems. They classify a particular status of the network according to categories corresponding to the nature of the failure. Their approach has false-positives issues, which makes it necessary to include a failure detection stage before the classifier. Fuzzy logic models were also used in [21], where it was shown that individual KPI thresholds are not well-suited for fault diagnosis. They propose a methodology to design and evaluate self-healing systems, and elaborated a simulator that modelled various types of failure.

Self Organising Maps (SOMs) are a kind of neural network architecture whose output creates a low-dimentional representation of the data. If the number of output neurons is low, the result is similar to that of clustering algorithms. In [8] SOM techniques are used to identify clusters of the behavior of network KPIs. The statistical behavior of the data from a cluster is evaluated by experts who determine if its behavior is relevant and coherent with a particular kind of known failure. The technique gave excellent results on real data. The method was tested with data filtered by a fault detector.

In [14], KPIs and Operation Support System (OSS) data are clustered after being discretized. A protocol for knowledge acquisition is also proposed, where experts are asked to match identified clusters with the causes of the failures.

In [18], time intervals where eNodeBs in LTE networks experiment degradation are identified, and the KPI values registered during the intervals are aggregated into one vector describing the performance of the interval. Vectors of each degradation interval are labelled according to the cause registered by the expert who troubleshooted each case. A search engine is proposed to help in diagnosing the failure, finding the most similar past degradations.

AdaBoost is used for failure diagnosis in [22], where it was tested on simulated data. They over-sampled normal traffic, using Synthetic Minority Over-sampling Technique (SMOTE). This strategy allowed them

to address class imbalance. AdaBoost techniques are also used in [17] to perform automatic fault diagnosis. They use Support Vector Machines (SVMs) as the weak learner models, tunning the parameters of the classifier via a genetic algorithm. The method is tested in real data from an LTE network.

3 Models proposed

Differently from the reviewed research, we propose a failure detection strategy based *solely* on M2M / MTC traffic in a smart city. We also consider different levels of aggregation to help the operator choose the best one for their needs.

Even though the data is sequential in nature, we do not approach the problem as one of time series. In order to identify whether there is a failure or not, we only consider aggregated data from one interval at a time, and no information from the system in previous intervals is used.

Data aggregation is performed by estimating several statistical descriptors of the variability of packet and RACH collision counts. Non-overlapping time intervals of different sizes are used to compute the statistics, and supervised learning techniques are adopted to independently classify each of these intervals based on its statistics. In order to extract more information from the KPI variability, we also consider higher order statistics, in order to produce a richer representation of the variability, while still relying on data aggregation to detect failures.

We treat the failure as a sleeping cell problem, assuming that no operation and mainteinance system is providing any kind of alarm. In an ideal context, eNodeB failures are scarce, with respect to normal traffic regimes, which implies that there is very little data from the network during failures, in comparison to data gathered in normal conditions. In machine learning, this situation is called *class imbalance*, and makes the problem difficult to solve using binary classification techniques: in fact, standard training techniques produce models with a tendency to assume the more frequent class for every data-point as the imbalance grows. This is usually addressed by using an anomaly detection approach, under the assumption that most anomalies are also failures. We deal with class imbalance instead, with the binary classification approach, by sub-sampling normal traffic, which is the mayority class, so that both sets of data-points are of equal size.

In this paper, classification performance is measured by focusing more on detection rate rather than on false positive rate, precision or accuracy, as falsely assumming the existence of a failure is less expensive than failing to detect one [22].

We use the LTE simulator both in "normal" operation regime and in failure state. Failures are modeled as the total or partial elimination of Random-Access Opportunities (RAO) for an interval defined by the user. These kind of failures could be caused by a natural disaster, an energy outage or an accident completely affecting the hardware of an eNodeB.

3.1 M2M traffic model

The process of troubleshooting a performance problem in an LTE network begins with the *detection* of the failure, and, ideally, the identification of the cause of the problem [7]. Performing this process in an automated way is part of the functions of SONs, specifically self-healing (Figure 1). Fault detection is a neuralgic part of this process, as the rest of the self-healing functions (compensation, diagnosis and recovery) will only be applied over those nodes indicated by the detection function. In order to perform fault detection Service Providers maintain KPIs that are aggregated over time and across the network, or regions of the network. Low-level fine-grained performance data is costly to keep, as storage requirements are extreme.

Our simulator uses public data to obtain the location of the following types of machines:

- Smart meters (installed in residential addresses).
- Fire alarms.
- Surveillance cameras.
- Bus stops.
- Traffic lights.
- Parking lots.

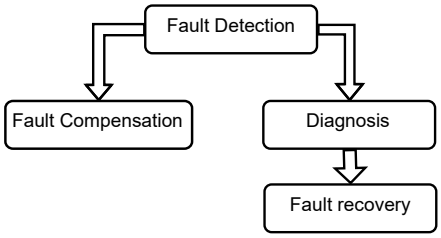

Figure 1: Self-Healing functions in the context of M2M-eNodeB traffic.

In addition to these traffic generators, 50 thousand Micro-Phasor Measuring Units (microPMUs) were randomly placed along the island of Montreal. We also have public data on the location and characteristics of eNodeBs in Canada.[1]

The traffic from the applications was generated following a Poisson distribution, with average times between connection requests varying from 10 seconds for the microPMUs to one hour for the smart meters.

The simulator generates a list with the two closest eNodeBs for each machine. RACH is performed first for the closest eNodeB. Only access from the machines. At the end of the simulation, the model provides the number of packets and the count of the RACH collisions in each cell and globally in the network. Results are also subdivied by the type of M2M applications included in the run.

3.2 Fault model

Our simulator allows the creation of simultaneous eNodeB failures with arbitrary starting time and duration. The intensity of the failure is specified as the percentage of the number of channels at fault.

The intensity of a failure in an eNodeB is modeled by making unavailable for access a specific number of channels in the node. The intensity percentage is also the percentage of channels being turned off (or the closest integer number of channels). In this paper, the pattern classification techniques are trained to recognize 2 different states: (i) 100 % of failure, (ii) normal traffic. We assume that a failure affects the failing eNodeB and its neighbors.

3.3 Granularity

The size of the aggregation of the "real-time" data plays an important role in the ability to observe changes in the patterns of collisions. On one hand, large aggregations imply a loss of information. On the other hand, statistics become more stable as sample size increases, which is a desirable property.

In order to study the effect that granularity size has on our ability to detect failures based solely in the statistic of a particular interval, we perform our analyzis on aggregations of 5, 10, 15, 20 and 30 minutes.

An additional approach we consider regarding granularity, is to use an ensemble classifier where the base classifiers learn at different scales of granularity. In particular we try an ensemble composed as follows:

1. One classifier trained with a granularity of 30 minutes.
2. Two classifiers trained with a granularity of 15 minutes, covering the 30 minutes of the previous classifier.
3. Six classifiers trained with a granularity of 5 minutes, covering the 30 minutes of the first classifier.

3.4 Machine learning methodology

We use the simulator in combination with machine learning models, as depicted in Figure 2. The failure, which is an input parameter of the simulator is sought to be reconstructed by the learner model, by taking as input an aggregation of the collisions and packets counts.

[1]Spectrum Management System Data: https://sms-sgs.ic.gc.ca/eic/site/sms-sgs-prod.nsf/eng/h_00010.html

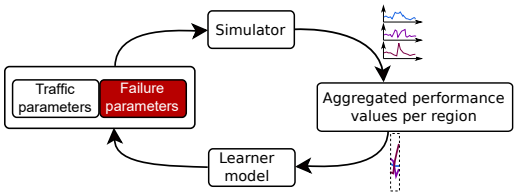

Figure 2: Interaction between simulator and binary classifier.

The problem is formulated as a binary classification problem, where statistics from a single time-bin are used as features or predictors vector by a classification model. The target or label that should be matched by the output of the model, consists in the states of failure or no-failure.

The statistics used to aggregate the data are: average, variance, assymetry, kurtosis, minimum, maximum, range (difference between minimum and maximum), and percentils (5 %, 25 %, 50 %, 75 %, 95 %). In the case the part of the feature vectors describing the behavior of the neighborhood of a node, average, variance, minimum, maximum and range from adjacent cells are added, while kurtosis, skewness and percentiles are averaged.

3.5 The dataset

We generated two scenarios of 24 hours of traffic data for the 553 eNodeBs in the island of Montreal. Each scenario consisted of twelve concatenated 2-hour simulations. For 11 eNodeBs covering emblematic areas of the city, each of the simulations consisted on one hour of normal activity followed by a one-hour failure. This allowed us to consider aggregations of both normal and failing regimes at different times of the day, which is relevant as some machines traffic generation parameters depend on the time of the day. In total we simulated 12 total failures in each of the 11 nodes of choice.

The data pre-processing did not include Principal Component Analysis (PCA) nor standardization, as these techniques worsened the False Positive Rate without any gain in the Detection Rate.

We randomly splitted the data of each node in three sets (Figure 3):

- *Training set:* 70 % of the data was used to train 20 different classification models.
- *Validation set:* 15 % of the data was used to evaluate the 20 models and estimate their generalization ability.
- *Testing set:* 15 % of the data was saved to estimate the classification performance values reported in this paper, after the training, tunning and model selection was finished.

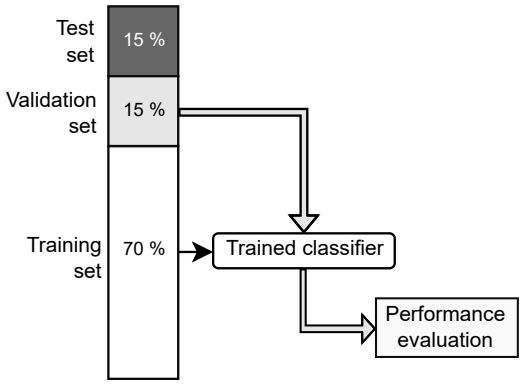

Figure 3: Data split for training validation and testing.

3.6 The learning models

We used 20 binary classifiers implemented in Matlab 2018a for the Classification Learner App and an Ensemble Model, built as an equal-weight voting system among the 20 classifiers from the Classification Learner App, as shown in Figure 4.

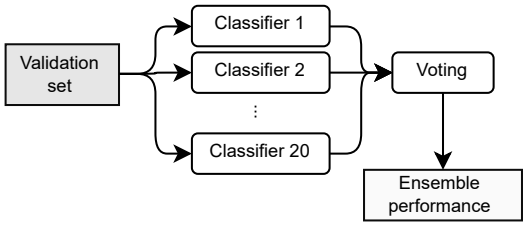

Figure 4: Ensemble classifier performance evaluation.

We implemented the following two training strategies, as depicted in Figure 5:

1. *Node-wise training:* 11 instances of each model were trained, each using as training data the feature vectors of one the 11 experiencing failures. This produced 20 trained models for *each* of the 11 eNodeBs. 11 Ensemble Models were also used, fed with the votes of the 20 models from the node for which a prediction was being made.
2. *Training by set of nodes:* this strategy consists in unifying all the training data from all the nodes to train the 20 models. This implies that the pattern learned by each model is not fitted for the specific behavior or a node.

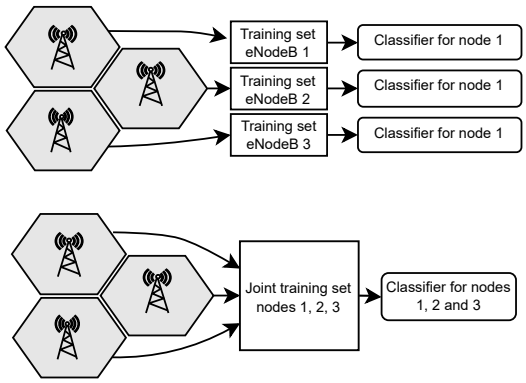

Figure 5: Training by node vs. training from a set of nodes.

In both cases, the aggregation of packet and collision counts from neighboring eNodeBs is part of the feature vector for the target eNodeB.

For the 11 nodes experiencing failures the scenario implies a 50 % of probability of failure, and both *time between failures* and the *time to failure* are exactly of one hour. In order to consider a more general context, we repeated the experiments using the 10 best classifiers, and trained them with a mult-node approach. This time the set of intervals without failures were randomly chosen from the 553 eNodeBs of whole network, instead of using those of normal traffic from the 11 nodes with failures. The number of intervals extracted from normal traffic instances was equal to the number of intervals extracted from failures.

To summarize, by means of the mentioned experiments, we aim at providing an answer to the following research questions:

1. Does node-wise training allow better generalization ability than training in a set of nodes?
2. How does aggregation size affect prediction performance?

3. How does the performance of an Ensemble Model combining different algorithms compare with the individual models?
4. How does the traffic volume affect prediction performance?
5. How will prediction performance will be affected when data from normal traffic intervals is sampled from the whole network?

4 Results and analysis

In order to measure classification performance, we consider the following indicators, sorted in a decreasing priority order:

- *Prediction rate:* defined as the well-classified failures over the total number of failures.
- *False positive rate:* defined as the cases of normal traffic classified as failures over the total of normal traffic instances.
- *Accuracy:* defined as the total of well-classified data-points over the total.
- *F-score:* defined as two times the product of precision (% of detections that are truly failures) and detection rate, divided by the sum of precision and detection rate.

Performance scores of 22 models are shown in Figure 6, for an experiment of high traffic and 5 minutes aggregations, which is representative of the behavior observed in the rest of the experiments. Models trained in multiple nodes (indicated by the orange circles) consistently had detection rates higher than 90 % and false positive rates lower than 30 %, with the exception of one model (subspace k-nearest neighbors, which is the top-right orange circle). There is a clear advantage in multi-node training, even though there are some instances of experiments where occassionally node-wise trained models achieved good combinations of detection and false positive rates.

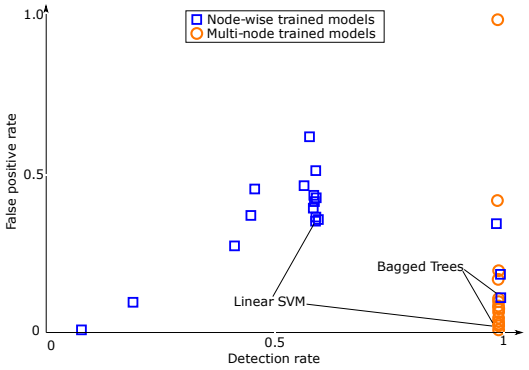

Figure 6: Performance of node-wise and multi-node models in 5 min aggregation and high traffic.

Ensemble learners did not perform systematically well as their false positive rate bested in every experiment by those of BDT and L-SVMs. In Figure 6, all 22 models are placed in a plane according to their performance values of detection and false positive rates. Those shown as blue squares are classifiers trained node-wise, while the red circles show the performance achieved by multi-node trained classifiers. The figure corresponds to the performance values for one particular scenario of 5 minutes of aggregation and high traffic). We point out the positions of L-SVMs and BDT, obtained both via node-wise and multi-node training. Both systematically performed better than Ensemble Learners. It can be seen that with multi-node training their performance is very similar, while node-wise training produces worse results for both algorithms. Considering L-SVMs and BDT in all the training scenarios, we observe that detection rates for multi-node L-SVMs and BDT were higher than 97.5 % and false positive rates were lower than 2.7 % in all aggregation levels (see Tables 1 and 2).

Table 1: Detection and false positive rates on L-SVMs (multi-node training).

	Linear SVM			
	Detection Rate (%)		False Positive Rate (%)	
Aggregation	Low traffic	High traffic	Low traffic	High traffic
5	99.6	99.6	1.6	2.1
10	99.2	99.2	1.3	1.2
15	98.8	98.8	1.0	1.0
20	98.3	98.6	1.0	0.8
30	97.5	97.7	0.9	0.8

Table 2: Detection and false positive rates on Bagged Decision Trees (multi-node training).

	Bagged Decision Trees			
	Detection Rate (%)		False Positive Rate (%)	
Aggregation	Low traffic	High traffic	Low traffic	High traffic
5	99.6	99.6	2.4	2.6
10	99.2	99.2	2.4	2.4
15	98.8	98.8	2.6	2.4
20	98.3	98.6	2.6	1.8
30	97.5	97.7	2.6	1.7

We can also observe in Tables 1 and 2, that larger aggregations produce lower detection rates both in L-SVMs and BDT. In L-SVMs, false positive rate decreased with the increment in aggregation size. The same ocurred in BDT in the high volume traffic scenario.

We also obtained good results when sampling normal traffic intervals from the whole network: in Figure 7, detection rates above 96.3 % were observed for all models except for Fine Gaussian SVM. False positive rates were under 3 % for all 10 classifiers. The figure is representative of the general behavior of multi-node models when sampling from the whole network.

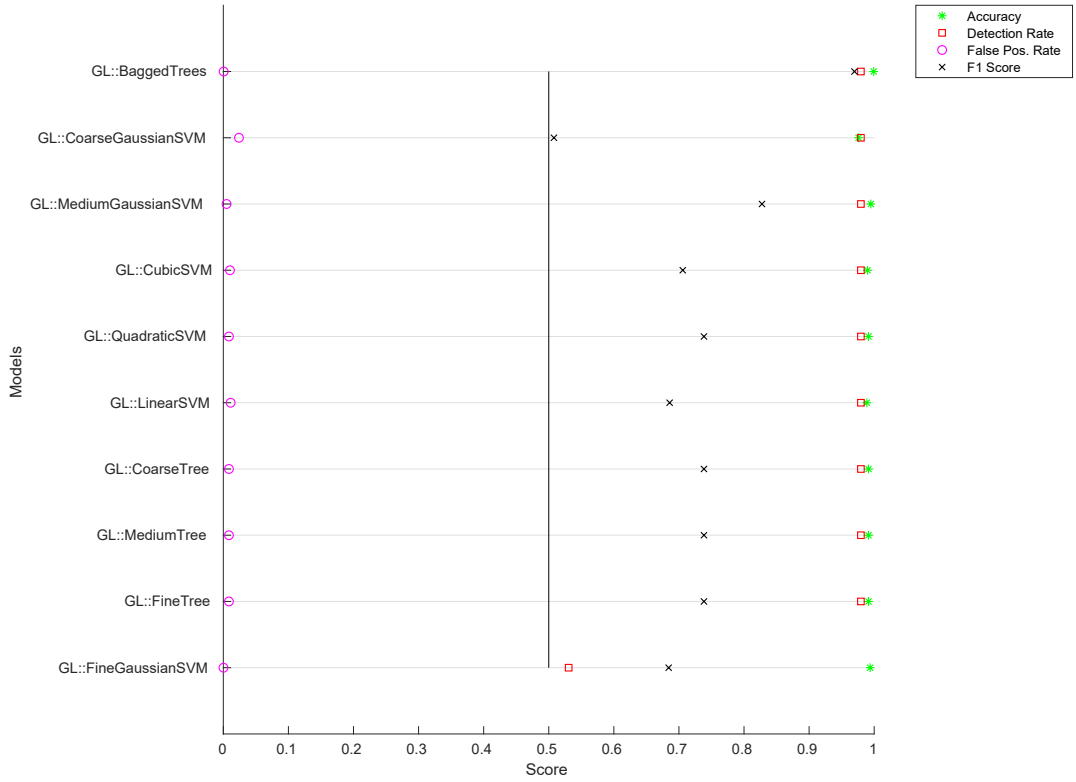

Figure 7: Performance of 10 mult-node models in 30 min aggregation and high traffic, sampling normal traffic from 553 nodes.

5 Conclusions

We performed experiments to detect eNodeB total failures in a simulated LTE network under M2M traffic, with 20 binary classification techniques, following two approaches:

- Training models on only one eNodeB (node-wise training).
- Training models on various eNodeBs (multi-node training).

We found that in both traffic volume scenarios, and in every data aggregation level, most of the best performing models were those trained on a set of eNodeBs. L-SVMs and BDT were the ones showing more consistently good results both in terms of detection rate and false positive rate.

Two traffic volume scenarios were used, one with normal traffic (following templates of normal packet generation by the machines), and one with high traffic and one third of the available channels. On higher aggregations (20 minutes or more), with high traffic conditions the two best models showed a slightly better detection rate. False positive rate appears to improve in the high traffic scenario as well.

Aggregation levels of 5, 10, 15, 20 and 30 minutes were used in our experiments, in both traffic scenarios. As expected, there is a clear trend of decrease in detection rate as the size of the aggregation intervals increases. However, in the false positive rate, L-SVMs performance improved with bigger aggregations, implying that models trained on bigger intervals are more conservative when detecting a failure. On BDT there is not a clear pattern on whether aggregation size is good or bad for false positives rate. Overall, even though detection rates slightly degrade as aggregation size increases, we were able to achieve high detection rates and low false positives in the largest level of aggregation (30 minutes).

Our results suggest that using a set of statistics on the number of packets and collisions, both in a node under observation and its neighborhood, with a small aggregation (5 minutes) gives the best failure detection rate. With respect to the slightly more conservative behavior of models under larger aggregations, one possible explanation is the fact that statistics are computed with a larger sample, making them more stable, which means they are less sensitive to individual outlayers.

Models trained under larger aggregations (30 minutes), despite being the more conservative ones (with lower detection rate), showed the ability to have excellent failure detection rates when using L-SVMs and BDT (higher than 97.3 % in all our experiments), while giving the best rates of false positives (at most 2.6 %).

One possible strategy for Service Providers to take advantage of both the detection rates of smaller aggregations, and the false positive rates of bigger aggregations, is to two aggregation levels at the same time:

1. A model trained with a bigger aggregation, providing the more *conservative* classifier (with lower false positive rates).
2. A model trained with a smaller aggregation, providing the more *sensitive* classifier (with higher detection rates).

For the interval used as input for one classifier of the bigger aggregation, several intervals of the smaller aggregation could feed the more sensitive classifier. As a consequence, one output for a period of time, from the conservative classifier would be complemented with the output of several evaluations of the more sensitive one. When most of the outputs of the more sensitive classifier for the interval under consideration, agree with the output of the conservative classifier, there is lower probability of incurring in a misclassification. On the other hand, if the sensitive classifier gives as outputs several failures within the interval, but the output for the conservative classifier does not indicate a failure, a *warning* state can be triggered, signalling that, while a failure is being detected on several of the small aggregation intervals, there is a higher chance of it being a false alarm.

It is important to point out that the kind of failures we are focusing, in general, does not dissapear, unless there is some activity to fix it. Therefore, if one interval is falsely classified as belonging to “normal traffic”, it is unlikely that in the next interval the same mistake will be made, as there is at most 3 % of probability of missing the detection of a failing interval. As a consequence, failing to detect a failure in one interval, can

be considered as an additional delay in the detection, as the classifier will have one more chance to perform the detection, each time an interval equivalent to the aggregation size passes by. This implies that the overal figures of detection rate are even higher than what is reported when considering more than one interval.

References

[1] F. Malandra, L. Chiquette, L.-P. Lafontaine-Bédard, and B. Sansò, "Traffic characterization and LTE performance analysis for M2M communications in smart cities," Pervasive and Mobile Computing, 48: 59–68, 2018.

[2] E. J. Khatib, R. Barco, P. Muñoz, I. De La Bandera, and I. Serrano, "Self-healing in mobile networks with big data," IEEE Communications Magazine, 54(1): 114–120, 2016.

[3] R. Barco, P. Lazaro, and P. Munoz, "A unified framework for self-healing in wireless networks," IEEE Communications Magazine, 50(12): 2012.

[4] M. Amirijoo, R. Litjens, K. Spaey, M. Döttling, T. Jansen, N. Scully, and U. Türke, "Use cases, requirements and assessment criteria for future self-organising radio access networks," in International Workshop on Self-Organizing Systems. Springer, 2008, pp. 275–280.

[5] F. Chernogorov, S. Chernov, K. Brigatti, and T. Ristaniemi, "Sequence-based detection of sleeping cell failures in mobile networks," Wireless Networks, 22(6): 2029–2048, 2016.

[6] S. Hämäläinen, H. Sanneck, and C. Sartori, LTE self-organising networks (SON): network management automation for operational efficiency. John Wiley & Sons, 2012.

[7] E. J. Khatib, R. Barco, A. Gómez-Andrades, P. Muñoz, and I. Serrano, "Data mining for fuzzy diagnosis systems in lte networks," Expert Systems with Applications, 42(21): 7549–7559, 2015.

[8] A. Gómez-Andrades, P. Muñoz, I. Serrano, and R. Barco, "Automatic root cause analysis for lte networks based on unsupervised techniques," IEEE Transactions on Vehicular Technology, 65(4): 2369–2386, 2016.

[9] B. Cheung, S. Fishkin, G. Kumar, and S. Rao, "Method of monitoring wireless network performance," Mar. 23 2006, uS Patent App. 10/946,255.

[10] Q. Liao, M. Wiczanowski, and S. Stańczak, "Toward cell outage detection with composite hypothesis testing," in Communications (ICC), 2012 IEEE International Conference on. IEEE, 2012, pp. 4883–4887.

[11] A. Coluccia, A. D'Alconzo, and F. Ricciato, "Distribution-based anomaly detection via generalized likelihood ratio test: A general maximum entropy approach," Computer Networks, 57(17): 3446–3462, 2013.

[12] I. de-la Bandera, R. Barco, P. Munoz, and I. Serrano, "Cell outage detection based on handover statistics," IEEE Communications Letters, 19(7): 1189–1192, 2015.

[13] C. M. Mueller, M. Kaschub, C. Blankenhorn, and S. Wanke, "A cell outage detection algorithm using neighbor cell list reports," in International Workshop on Self-Organizing Systems. Springer, 2008, pp. 218–229.

[14] S. Rezaei, H. Radmanesh, P. Alavizadeh, H. Nikoofar, and F. Lahouti, "Automatic fault detection and diagnosis in cellular networks using operations support systems data," in Network Operations and Management Symposium (NOMS), 2016 IEEE/IFIP. IEEE, 2016, pp. 468–473.

[15] F. Chernogorov, J. Turkka, T. Ristaniemi, and A. Averbuch, "Detection of sleeping cells in lte networks using diffusion maps," in Vehicular Technology Conference (VTC Spring), 2011 IEEE 73rd. IEEE, 2011, pp. 1–5.

[16] Y. Ma, M. Peng, W. Xue, and X. Ji, "A dynamic affinity propagation clustering algorithm for cell outage detection in self-healing networks," in Wireless Communications and Networking Conference (WCNC), 2013 IEEE. IEEE, 2013, pp. 2266–2270.

[17] X. Liu, G. Chuai, W. Gao, and K. Zhang, "Ga-adaboostsvm classifier empowered wireless network diagnosis," EURASIP Journal on Wireless Communications and Networking, 2018(1): 77, 2018.

[18] E. J. Khatib, R. Barco, and I. Serrano, "Degradation detection algorithm for lte root cause analysis," Wireless Personal Communications, 97(3): 4563–4572, 2017.

[19] E. J. Khatib, R. Barco, P. Muñoz, and I. Serrano, "Knowledge acquisition for fault management in lte networks," Wireless Personal Communications, 95(3): 2895–2914, 2017.

[20] R. M. Khanafer, B. Solana, J. Triola, R. Barco, L. Moltsen, Z. Altman, and P. Lazaro, "Automated diagnosis for umts networks using bayesian network approach," IEEE Transactions on vehicular technology, 57(4): 2451–2461, 2008.

[21] A. Gómez-Andrades, P. Muñoz, E. J. Khatib, I. de-la Bandera, I. Serrano, and R. Barco, "Methodology for the design and evaluation of self-healing lte networks," IEEE Transactions on Vehicular Technology, 65(8): 6468–6486, 2016.

[22] M. Sun, H. Qian, K. Zhu, D. Guan, and R. Wang, "Ensemble learning and smote based fault diagnosis system in self-organizing cellular networks," in GLOBECOM 2017-2017 IEEE Global Communications Conference. IEEE, 2017, pp. 1–6.