Using Big Data for More Dependability: A Cellular Network Tale

Nawanol Theera-Ampornpunt
Purdue University
ntheeraa@purdue.edu

Saurabh Bagchi
Purdue University
sbagchi@purdue.edu

Kaustubh R. Joshi
AT&T Labs Research
kaustubh@research.att.com

Rajesh K. Panta
AT&T Labs Research
rpanta@research.att.com

ABSTRACT

There are many large infrastructures that instrument everything from network performance metrics to user activities. However, the collected data are generally used for long-term planning instead of improving reliability and user experience in real time. In this paper, we present our vision of how such collections of data can be used in real time to enhance the dependability of cellular network services. We first discuss mitigation mechanisms that can be used to improve reliability, but incur a high cost which prohibit them to be used except in certain conditions. We present two case studies where analyses of real cellular network traffic data show that we can identify these conditions.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: reliability, availability, and serviceability

General Terms

Reliability

Keywords

Big data, data mining, cellular, wireless

1. INTRODUCTION

There are many examples of large infrastructures that instrument everything from regular network performance metrics to detailed user activities. For example, cellular networks collect information about bandwidth usage, handovers, signal strength, connection/disconnection events, etc., to analyze network performance. Web analytics record user clicks, location, page visit time, page dwell time, etc., to help page owners understand their traffic. The collection of these large quantities of analytics from large infrastructures and their use for understanding user behaviors and trends has been called "big data".

Unfortunately, today, these measurements are not used in meaningful ways *online*, if at all. Instead, they are often used to do offline analysis. In this paper, we explore whether real-time data collected from detailed instrumentation of large scale infrastructures can be used to provide real-time services that improve the dependability of the infrastructures and through this, the experience for users using them.

Specifically, this paper presents our vision of how big data analytics can be used in real time to enhance the dependability of cellular network services. We demonstrate how big data can help in the design of adaptive techniques to reduce the incidence of voice or data disconnections in cellular networks and mitigate their effect when they do occur. We show that such mitigation mechanisms come with a cost that prohibits them from being turned on all the time. In order for them to be beneficial, real-time data analysis is necessary because network disconnections depend not only on static factors (such as user locations that are prone to bad network connectivity), but also on dynamic factors (such as current level of congestion in the cell, available radio resources, etc.). We analyze real data collected by a major cellular network and show that we can build a model that identifies conditions that are likely to lead to network disconnections where real time mitigation mechanisms are beneficial.

2. BACKGROUND

2.1 Network Architecture

Figure 1 shows our proposed architecture for the LTE network. The mobile device, called User Equipment (UE), is connected to a cell sector (which we will refer to as cell) in a base station, called eNodeB. A physical base station can have multiple sectors, potentially covering different regions. Cellular traffic from the eNodeB passes through Serving Gateway (S-Gateway) and Packet Data Network Gateway (PDN-Gateway) to external network (e.g., the Internet).

In order to identify (dynamic) conditions that can predict voice or data drops with sufficient confidence, we add two components in the proposed architecture as shown in Figure 1: offline training and online predictor. The offline training component takes data from eNodeBs and construct a machine learning classifier for a specific prediction task. Due to changing network conditions over time, this offline

HotDep '13 Pennsylvania USA

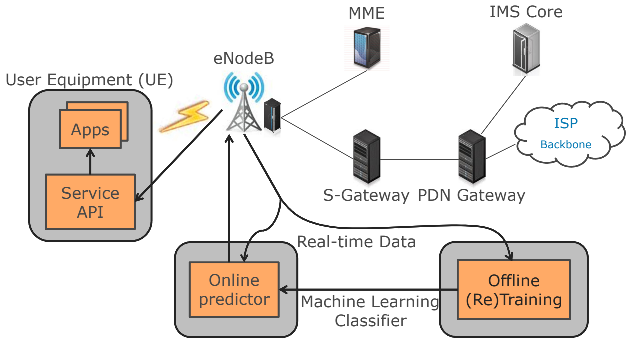

Figure 1: Diagram of the proposed architecture.

training needs to be repeated often (e.g., every day). The learned classifier becomes the online predictor, which makes predictions using real-time data from the eNodeB. When the event of interest occurs (e.g., probability of a drop exceeds the threshold), the online predictor sends a notification to the component(s) responsible for initiating the mitigation actions. This could be the mobile device itself (e.g., to initiate precaching), or a component in the network (e.g., switch to older technology) or both. In case of the device, the notification is first sent to the Service API in the device, which dispatches it to applications that have registered to receive that particular type of notifications.

In this study, we focus on one type of failures: abnormal disconnections (also referred to as "drops"). In this work, abnormal disconnections are defined as abnormal release of Radio Access Bearer (RAB) or Radio Resource Control (RRC) connection [5]. Before performing any voice call or data communication, a phone first needs to establish an RRC connection. This is followed by a RAB connection, which assigns essential network resources to the user device based on the quality of service required by the specific application. Since different applications can have different QOS requirements, a single device may have multiple RAB connections, but only a single RRC connection at any given time. Our first case study focuses on predicting the failures, while the second case study focuses on predicting how long the failure will last (referred to as "drop duration"), given that a failure has occurred.

2.2 Mechanisms that Improve Reliability

Before describing the offline training and online predictor components, we show that the various techniques that can be used to mitigate the effects of possible data or voice disconnections incur some cost. Hence it is important to update offline classifier regularly and perform prediction using real time data to identify conditions that can lead to abnormal connection drops as accurately as possible.

Switching to Older Technology: Since newer Radio Access Technologies (RATs) provide higher bandwidth and/or new features, users tend to prefer them over older technologies. This leads to higher congestion, as each RAT uses its own base stations and other infrastructures. Thus, older RATs generally have higher reliability than newer ones. The cost of using older technology is the lower available bandwidth, which could be more important than the reliability of the connection, depending on the application. For example, a video streaming application may be able to mask a disconnection by buffering the video before it occurs. On the other hand, when making a voice call in a congested region, it may be better to use an older, less congested RAT despite the drop in audio quality.

Prefetching: Prefetching can be done for web browsing (link prefetching) as well as audio/video streaming (downloading the stream at maximum capacity). If used early enough before a degradation of service, and the degradation does not last very long, it can mask the disconnection completely. Otherwise, it still allows the user to access more content before being affected by the degradation of service. This method carries a cost of wasted bandwidth and energy if left on longer than necessary or all the time.

Voice Call Auto-Reconnecting: Currently, if a disconnection occurs during a voice call, the call is simply dropped. It is straightforward to add a functionality of automatic reconnection to voice calls, which will make reconnection more seamless for both parties. However, because most of the time, a drop lasts rather long, this would inconvenience the user as well as the other party of the call while he or she waits.

While the mechanisms mentioned above do improve reliability and/or user experience, they all come with a cost that prohibits them from being used *all the time*. What we need is a way to determine the conditions (e.g., how likely a drop is going to occur in the immediate future), the temporal nature of the failure (e.g., how long the drop is expected to last), take into account the device specific information (e.g., type of applications the user is running), and use one of the mechanisms only when the benefits outweigh the costs. In the next section, we will show how big data analytics can be used to identify these conditions.

3. CASE STUDIES

In this section we explore how we can identify conditions needed by the mitigation mechanisms described in Section 2.2 with enough confidence. We focus on two specific problems: predicting drops and predicting drop duration.

3.1 Data Source

We use real-world 3G cellular traffic data from a tier-1 U.S. cellular network collected on or after June 5, 2012. Although the 3G network has slightly different architecture from LTE as described in section 2.1, the prediction mechanism described here can be applied for LTE networks as well. Data are collected at the NobeB (which is analogous to LTE's eNodeB) during its normal operation, and aggregated at the Radio Network Controller (RNC), which manages multiple NodeB's. All devices and user identifiers are anonymized for our analysis, and packet inspection was not used. The dataset contains various low-level performance events, each of which contains common metrics as well as its own set of metrics. Common metrics include timestamp, user's International Mobile Subscriber Identity (IMSI), cell IDs of the cells the device is connected to. Examples of logged events include connections/disconnections, cell load, primary cell of each UE, and download and upload throughput. Once a model is trained, the original data used to train it are discarded.

3.2 Methodology

Because different events have different reporting interval, we need to combine them by computing aggregate functions

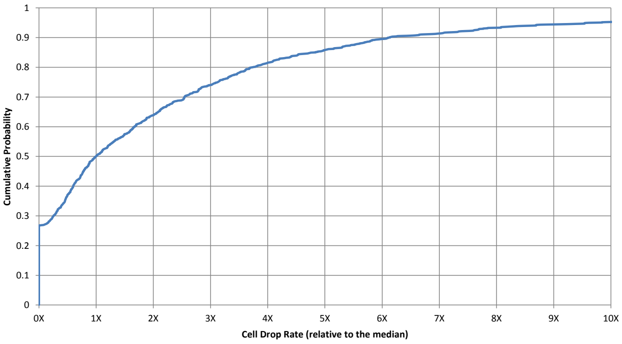

Figure 2: Cumulative distribution plot of cell drop rate.

(e.g., average, count, last report value) of the values within a sliding time window. Specifically, for each window and each metric, we compute 10 aggregate functions of that metric values within the window. If no value is reported within the window, we report it as a special missing value. Aggregated values across all metrics corresponding to the same window form a data point, used to train and evaluate classifiers. For the task of predicting drops, a data point is assigned the "failure" class label if the window it corresponds to precede a failure by up to 20 seconds. It is assigned the "normal" class label otherwise. For the task of predicting drop durations, since we are only making predictions when a drop has just occurred, we only include windows that lead up to, but not including, the drop. We use window size of 10 seconds. We select 12 events with 250 metrics altogether, leading to 2,500 attributes after computing the aggregates.

We use two machine learning algorithms for our data analysis: AdaBoost and Support Vector Machine (SVM). We use Weka's implementation of AdaBoost, with decision stump as the base classifier [4]. For SVM, we use LIBSVM implementation [2]. All prediction accuracy results are generated using separate train and test datasets, each corresponding to one day of operation.

3.3 Predicting Drops

In the first case study, we look at whether we can identify conditions correlated with disconnections. For this purpose we consider disconnections of both voice calls and data connections. There are many ways to partition the models; we can build a single global model, or train a separate model for each cell sector, or build a personalized model for each user. We found that accuracy of personalized models suffer from the lack of sufficient amount of data, because for most users, there is not enough *failure* data to build a good model.

The question then becomes, are the reliability characteristics of each cell different from each other enough to justify partitioning the models by cell? To answer this question, we look at the drop rate of each cell from the same time window. For this study, we define the drop rate to be the number of drops during a period divided by the amount of data traffic used across all users during the same period.

Figure 2 shows the cumulative distribution of cell drop rate from the operational period of April 5 to June 24, 2013. Each cell's drop rate is shown relative to the median drop rate among these cells. The drop rate ranges from zero to 107 times the median. 26% of all cells have no drop at all during this period. From the figure, we can see that there is a high variation among the cells. Thus, it is reasonable to partition the models by cell sector.

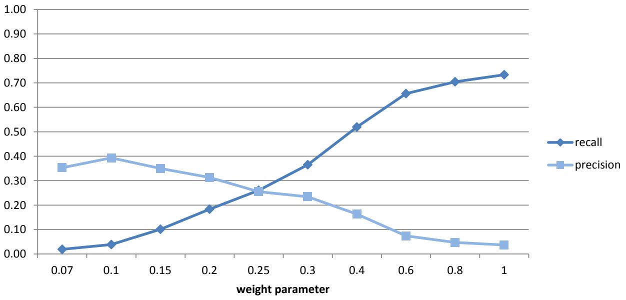

Figure 3: Drop prediction accuracy as a function of weight parameter.

A1	A2	Fraction of failure data points
not greater	not greater	$0.51X$
not greater	greater	$2.04X$
greater	not greater	$1.82X$
greater	greater	$40X$
Any	Any	X

Table 1: Class distribution of data points from one cell given top two attributes compared to the overall fraction of failure data points (labeled X). 'greater' and 'not greater' refers to the outcome of the comparison between the value and the learned threshold.

We separate the data by cell and use AdaBoost to train a model for each cell. Due to the amount of data, we need to sample and include only a fraction of users for each cell. Figure 3 shows the prediction accuracy in terms of precision and recall. The weight parameter is a tunable parameter that controls the relative cost of a false positive versus the cost of a false negative (higher weights mean false negatives cost more). Recall and precision have their conventional meaning. Recall is the proportion of actual failure cases that the classifier is able to predict. Precision is the fraction of actual failure cases to the number of predicted failures. Thus, recall is the complement of false negative rate and precision is the complement of false positive rate.

While the precision of roughly 25% might seem low, it is enough for applications where maintaining a connection is essential. Specifically, when the classifier predicts that a drop will occur, there is a probability of 25% that a drop will actually occur. If we initiate a mitigation action based on this prediction, 25% of the time it would be the correct course of action, while 75% of the time the costs of the mitigation action are incurred unnecessarily. Depending on the action, the costs may be lower available bandwidth, wasted energy and bandwidth, or something else. Our experience shows that prediction accuracy varies based on the operational period from which the data used to train and evaluate the models are collected. Also, there is potential for the accuracy to be improved, for example by using a better sampling method, including more types of logged events in the data analysis, or extracting better features. Next, we illustrate how the attributes used by the classifier correlate to the failures.

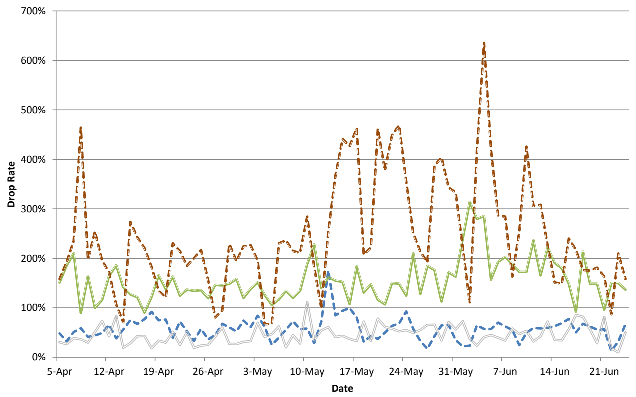

Figure 4: Plot of drop rate of four randomly-picked cells on each day from April 5 to June 24, 2013, compared to the median drop rate.

Each classifier is different, but we found that many classifiers have the same top two attributes that influence the prediction decision the most: 1) the number of UE's uplink throughput records with value of zero within a window, and 2) the sum of the cell's transmit power across all records within a window. We will refer to them as A1 and A2, respectively. Table 1 shows the class distribution of data points from the test dataset from one cell for each combination of outcomes of comparisons between the attribute value and the threshold learned by the Decision Stumps corresponding to A1 and A2. The fraction of failure data point for each combination is shown in relative to the overall fraction of failure data points. We can see that when either If either A1 or A2 (but not both) exceeds its threshold, the fraction roughly doubles. However, when both A1 and A2 exceed its threshold, the fraction becomes 40 times higher. Since the throughput is reported every 2 seconds, even if it is zero, the fact that there are many records with upload throughput of zero indicates that there is some problem with the communication. The cell's transmit power is related to the current load on the cell, which is correlated with drops.

Next, we explore how the conditions change over time, as this will determine how often the classifiers need to be retrained. We randomly pick four cells and plot their drop rate relative to the median drop rate during the period from April 5 to June 24, 2013 in Figure 4. We can see that the drop rate of some cells change significantly over time, and at different time from other cells. In order to capture these changes in conditions, the models need to be retrained frequently.

3.4 Predicting Drop Duration

Although related, predicting drop duration is a separate problem from predicting connection drops. The question we ask here is, given that a drop has occurred, what is the earliest time that the connection can be reestablished. We will refer to the duration between these two events as 'drop duration'. This could depend on the user's mobility pattern and environmental conditions, among other factors. This is important for guiding if a mitigation action is likely to be useful — a short drop duration would mean that it is acceptable to pause the call and then resume it when the connection can be reestablished, while a long drop duration would mean that it is better to simply drop the call.

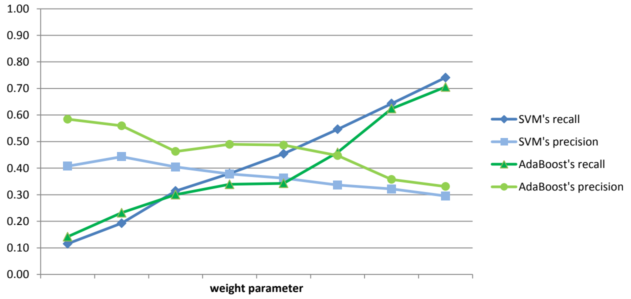

Figure 5: Drop duration prediction accuracy as a function of weight parameter for SVM and AdaBoost. Higher weight means false negatives cost more, relative to false positives.

Unfortunately, we cannot directly determine the earliest time that a reconnection attempt would be successful from our data source. Specifically, the data source only reports successful reconnections (since the network does not know about unsuccessful reconnections). The 3GPP standard does not require the device to attempt to reconnect after a disconnection. Thus, an absence of successful reconnection does not imply that an attempt would have been unsuccessful. However, this is the best method available to us to estimate the drop duration and we use it with an understanding that this is an upper bound of the true drop duration.

Since the goal of drop duration prediction is to make a decision whether to hold the voice call while reconnections are being attempted, instead of predicting the drop duration, we predict whether the drop will be short or long, determined by a threshold $t = 10$ seconds. This decision needs to be agreed on by both the network and the disconnected party. However, no communication is possible once a device is disconnected. Thus, during a voice call, the online predictor needs to keep analyzing real-time data and keep the devices updated about the decision, so that once a drop occurs, they agree on whether to hold the call and reconnect.

Due to variability of drop duration across cells, the classifiers should ideally be partitioned by cell. However, due to time constraints, we only have results from a single global classifier used for all cells. Figure 5 shows the accuracy of drop duration prediction for SVM and AdaBoost with different values of weight parameter. Because the two algorithms have different ranges of weight parameter, the actual values are not shown on the axis. Here, recall is the proportion of short drops that the classifier is able to predict correctly. Precision is the fraction of actual short drops to the number of drops predicted to be short. AdaBoost performs slightly better than SVM, achieving both recall and precision of roughly 45%. We found that the metric that most correlates with short drops is download throughput, with high throughput correlated with short drops.

4. RELATED WORK

There are several proposals for managing faults in cellular networks. However, much of the work focuses on detection of failures [3, 6, 8, 9], and identification of root cause of the failure [1, 10]. Their goal is different from ours, which is to predict failures and proactively try to prevent them, or lessen the effects of failures on user experience in the short term.

Javed et al. propose a machine learning framework for predicting handovers based on data available at the handset [7]. The goal is to notify applications before a short-term disruption in the service due to the handovers so that they can modify their behavior to counter it. This is similar in spirit to our work, although we are not limiting ourselves to handovers. Furthermore, the wealth of data collected at the network provides much more information than data available at the handset. This enables us to predict events that could not have been predicted otherwise.

5. CONCLUSIONS AND CHALLENGES

This paper presents our vision of how big data analytics can be used in real time to improve dependability and user experience. We demonstrated this by analyzing real cellular traffic data from a major cellular network and showed that we can construct a model that predicts drops and drop duration in real time with enough accuracy to enable mitigation actions to be used.

Challenges There are several challenges that need to be solved before such real-time data analysis and mitigation actions within the domain of cellular network become feasible:

Real-time Data Access: Real-time data access is not available for the majority of events logged by cellular network due to reasons including storage requirements and the need to protect user privacy.

Data Volume: Due to the amount of data and number of users involved, such real-time data analysis requires efficient data streaming and processing systems close to the data source.

Lack of unified framework: Because many mitigation actions are application-specific, and some must be initiated by the device, there must be a standard way for the network to send notifications to the device, and for applications to express interests in receiving such notifications.

6. REFERENCES

[1] R. Barco, V. Wille, L. Díez, and M. Toril. Learning of model parameters for fault diagnosis in wireless networks. *Wireless Networks*, 16(1):255–271, 2010.

[2] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[3] B. Cheung, G. Kumar, and S. A. Rao. Statistical algorithms in fault detection and prediction: Toward a healthier network. *Bell Labs Technical Journal*, 9(4):171–185, 2005.

[4] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11(1):10–18, Nov. 2009.

[5] H. Holma and A. Toskala. *Hsdpa/Hsupa For Umts*. Wiley Online Library, 2006.

[6] C.-Y. Hong, M. Caesar, N. Duffield, and J. Wang. Tiresias: Online anomaly detection for hierarchical operational network data. In *Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference on*, pages 173–182. IEEE, 2012.

[7] U. Javed, D. Han, R. Caceres, J. Pang, S. Seshan, and A. Varshavsky. Predicting handoffs in 3g networks. *ACM SIGOPS Operating Systems Review*, 45(3):65–70, 2012.

[8] Y. Liu, J. Zhang, M. Jiang, D. Raymer, and J. Strassner. A model-based approach to adding autonomic capabilities to network fault management system. In *Network Operations and Management Symposium, 2008. NOMS 2008. IEEE*, pages 859–862. IEEE, 2008.

[9] S. Rao. Operational fault detection in cellular wireless base-stations. *Network and Service Management, IEEE Transactions on*, 3(2):1–11, 2006.

[10] Y. Watanabe, Y. Matsunaga, K. Kobayashi, T. Tonouchi, T. Igakura, S. Nakadai, and K. Kamachi. Utran o&m support system with statistical fault identification and customizable rule sets. In *Network Operations and Management Symposium, 2008. NOMS 2008. IEEE*, pages 560–573. IEEE, 2008.