Detecting and Localizing End-to-End Performance Degradation for Cellular Data Services

Faraz Ahmed† Jeffrey Erman‡ Zihui Ge‡ Alex X. Liu† Jia Wang‡ He Yan‡
†Department of Computer Science and Engineering, Michigan State University, East Lansing, MI, U.S.A.
‡AT&T Labs – Research, Bedminster, NJ, U.S.A.
Emails:{farazah,alexliu}@cse.msu.edu, {erman,gezihui,jiawang,yanhe}@research.att.com

***Abstract*— Providing high end-to-end (E2E) performance is critical for cellular service providers to best serve their customers. Detecting and localizing E2E performance degradation is crucial for cellular service providers, content providers, device manufactures, and application developers to jointly troubleshoot root causes. To the best of our knowledge, detection and localization of E2E performance degradation at cellular service providers has not been previously studied. In this paper, we propose a holistic approach to detecting and localizing E2E performance degradation at cellular service providers across the four dimensions of user locations, content providers, device types, and application types. First, we use training data to build models that can capture the normal performance of every E2E-instance, which means flows corresponding to a specific location, content provider, device type, and application type. Second, we use our models to detect performance degradation for each E2E-instance on an hourly basis. Third, after each E2E-instance has been labeled as non-degrading or degrading, we use association rule mining techniques to localize the source of performance degradation. Our system detected performance degradation instances over a period of one week. In 80% of the detected degraded instances, content providers, device types, and application types were the only factors of performance degradation.**

I. Introduction

A. Background and Motivation

Internet access through cellular data services has become an essential part of people's everyday life such as usage of emailing, web browsing, video streaming, and online shopping. Nowadays cellular network customers using data services expect seamless service with high performance, i.e., the end-to-end (E2E) performance. To best serve their customers it is critical for cellular service providers to provide high E2E performance and maintain their competitive edge. In the context of cellular data services, E2E performance means the performance that customers experience for a specific location, content provider, device type, and application type. A *user location* means the Radio Network Controller (RNC) that the user's device connects to. A *content provider* means the Internet domain that is serving the user. A *device type* means a specific brand and model of the user's device, such as Apple iPhone 5. An *application type* means the categorical type of the application that the user is running on their cellular device, such as emailing or Web browsing.

In this work, we present a system for detecting and localizing E2E performance degradation (such as slow webpage loading and unsmooth video playing) at cellular service providers across four administrative domains: cellular service providers, content providers, device manufacturers, and application developers. For detection, we want to detect E2E performance degradation before cellular network operators receive complaint calls. For localization, we want to find the problematic domain that is causing performance degradation such as user location, content provider, device type, and/or application type. For example, if all users connecting to an RNC are experiencing performance degradation, regardless of content providers, device types, and application types, then probably the RNC is causing the performance degradation. For another example, if all users of iPhone 5 are experiencing performance degradation for their email application, regardless of user locations and content providers, then probably iPhone 5 is having issues with email applications.

Detection and localization of E2E performance degradation is crucial for all administrative domains to jointly troubleshoot root causes. When users experience E2E performance degradation, they have no clue whom they should "blame". For example, when a user at New York experiences performance degradation for yahoo email on his iPhone 5, he does not know whether it is the cellular provider, or yahoo, or iPhone 5, or his email client app, that is causing the problem. When user experience such E2E performance degradations, they blankly ascribe the fault to their cellular service providers and issue complaints to the cellular network customer service centers, which may result in both reputation damage and financial losses for the cellular service providers. When cellular service providers receive such calls, they have to go through lengthy, manual, labor intensive process to localize the issue, and the actual issue may not be the cellular service provider problem, it may be due to the user device itself (such as device OS, application software) or due to the content provider (such as application servers, datacenter network). For example, the E2E performance degradation maybe caused by a content provider upgrading its service, a device type having updated its OS with incompatibility issues, and an app having been patched with buggy code. The localization findings allow cellular service providers to quickly mitigate service problems, effectively communicate with customers complaining about service problems, and engage content providers, device manufactures, or application developers and operators to jointly troubleshoot for root causes.

There are several key challenges in detecting and localizing E2E performance degradation at cellular service providers.

First, there are a wide range of elements (such as mobile devices, cell towers, radio resource controller, routers, switches, fibers, media gateways, firewalls, multicast servers, name servers, and content servers) at various layers (such as physical layer, link layer, transport layer and application layer) that may cause the E2E performance degradation. Second, it is practically infeasible to gain the complete visibility of E2E performance issues because content providers, devices, and applications belong to different administrative domains. For example, issues on mobile devices and content servers are not visible to cellular service providers. Third, the expected E2E performance varies significantly depending on which application type, content provider, mobile device, geographic location, time of day, and day of a week [6], [7], [15].

B. Proposed Approach

To the best of our knowledge, the detection and localization of E2E performance degradation at cellular service providers has not been previously studied. In this paper, we propose a holistic approach to detecting and localizing E2E performance degradation at cellular service providers across the four administrative domains of user locations, content providers, device types, and application types. Our approach consists of three steps: modeling, detection, and localization.

First, we build models using training data to capture the normal performance of individual *E2E instance*, which means the flows corresponding to a particular user location, content provider, device type, and application type. Each E2E instance has $24 * 7$ models where each model corresponds to a specific hour of a day and a specific day of a week.

Second, we use our models to detect performance degradation for each E2E instance on an hourly basis. For each E2E instance, if the actual performance in the testing phase is too much worse than the expected performance obtained through our models, we label it as degrading.

Third, after marking each E2E instance non-degrading or degrading, we use association rule mining techniques to localize the source of performance degradation. For example, rule $\texttt{iPhone 5}, \texttt{Email} \rightarrow \texttt{degrading}$ shows that at a particular time instant, for all locations and content provides, the cellular users of iPhone 5 are experiencing significant degradation in performance when they use the email application.

C. Technical Challenges and Solutions

To implement our approach, we face three key technical challenges. The first challenge is to tradeoff between model accuracy and model complexity and to deal with data sparsity. We represent E2E performance in a four dimensional matrix that is called *E2E matrix* and is denoted $E_I = [1..L, 1..P, 1..D, 1..A]$, where the four dimensions are L user locations, P content providers, D device types, and A application types. An element $E_I[l, p, d, a]$ in this matrix represents the E2E instances corresponding to user locations l, content provider p, device type d, and application a. On one extreme end, we build only one model for all E2E instances in the E2E matrix, which gives us the least accuracy and the least complexity; on the other extreme end, we build one model for each individual E2E instance, which in theory gives us the most accuracy and the most complexity. To address this challenge, in this paper, we first build a baseline model for all E2E instances, identify the E2E instance groups that have significantly different performance, and then model these groups separately, leaving the rest E2E instances still being modeled by the baseline model. Within each group, the performance of some E2E instances may be relatively different from that of others in the group; thus, we apply this grouping strategy recursively among identified groups. The second challenge is to localize E2E performance degradation issues. To address this challenge, we use association rule mining to summarize the E2E matrix using some simple rules such as $\texttt{iPhone 5}, \texttt{Email} \rightarrow \texttt{degrading}$. The third challenge is to quantitatively evaluate the effectiveness of our approach because we are short of ground truth data. For detection, we do not have pre-labeled testing data because labeling such prohibitive amount of data is practically infeasible. For localization, the customer tickets that we have access to mostly document hard failures such as an RNC is down. When such hard failures happen, typically there is no flow for the affected E2E instances; therefore, our localization scheme will not find such failures. To address this challenge, first, we performed manual inspection for some E2E performance degradation cases; second, we injected some synthetic performance degradation cases into the test data and use those injected cases to serve as the ground truth.

II. Related Work

To the best of our knowledge, the detection and localization of E2E performance degradations at cellular service providers has not been previously studied. Effort related to ours can be categorized into detecting and troubleshooting network or service issues and cellular service E2E performance measurements.

Detecting and Troubleshooting Network or Service Issues: SCORE [11], Shrink [8], and [10] focus on diagnosing network failures through the inference of underlying lower-layer failures using a Shared Risk Link Group (SRLG) model. Pinpoint focuses on diagnosing the root causes of service failures at the application server end, *i.e.*, the content provider end using the terminology in this paper [2]. Sherlock [1] and NetMedic [9] both focus on diagnosing the root causes of service failures within the elements (such as DNS servers, firewall configurations, and application servers) of an enterprise network and both use dependency graphs. Sherlock and NetMedic conduct diagnosis at the granularity of machines and processes, respectively. In [12], Mahimkar *et al.* focused on characterizing and troubleshooting performance issues in one of the largest IPTV networks in North America. In [13], Mahimkar *et al.* designed and implemented a tool for detecting the impact of network upgrades on performance. Our work is different from these efforts in terms of both the problem being solved and the solution being used. From the problem perspective, our problem is on monitoring, detecting, and localizing

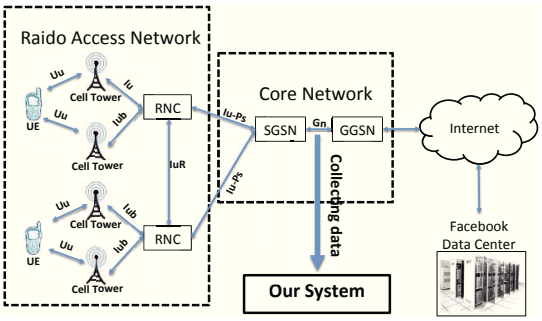


Fig. 3. Data Collection Architecture

E2E performance degradations for cellular data service users along the four dimensions of user locations, content providers, device types, and application types at cellular service providers without the full visibility of the E2E service path. From the solution perspective, our approach consists of passive cellular traffic monitoring, data centric modeling, model based performance degradation detection, and association rule based root cause localization, whereas these efforts uses variations of dependency graphs.

Cellular service E2E performance Measurements: There is work on measuring the impact on cellular service E2E performance by factors such as resource usage by mobile applications [14], content providers [4], and cellular technologies [7]. In [14], Qian *et al.* designed and implemented a tool for discovering inefficient resource usage for smartphone applications using cross-layer interaction among various layers including radio resource channel state, transport layer, application layer, and the user interaction layer. In [4], Finamore *et al.* performed a measurement study that compares YouTube traffic generated by mobile devices with that generated by PCs, and correlated user behavior with system performance. In [7], Huang *et al.* studied the interactions among applications, network transport protocols, and radio layers for the 4G LTE technology and their impact on performance using both active and passive measurements, and showed that E2E performance (such as TCP loss ratio) in cellular services varies across different content providers, device types, applications, connection types, and wireless carriers. Comparing these efforts with ours, we go one step further to localize the causes of E2E performance degradation.

III. Monitoring E2E Performance Dynamics and Data Analysis

A. Data Collection

In this study, we utilize anonymized flow level data collected from the core network of a major cellular service provider in the United States. Figure 3 illustrate the architectural overview of the core of the mobile network, which consists of two main types of nodes: the Serving GPRS Support Node (SGSN) and the Gateway GPRS Support Node (GGSN). The GGSN is the root node in the hierarchy of the cellular data network. GGSN is responsible for sending and receiving Internet traffic to and from the cellular network. SGSN is an intermediate node that connects the lower level nodes to the GGSN through the G_n interface. Typically, a single SGSN is connected to multiple Radio Network Controllers (RNCs) and each RNC serves a geographical region through cell towers.

The data was collected at the Gn interface which connects the SGSNs to the GGSNs. As our goal in this work is to detect and localize the E2E performance degradation in the cellular service, we collect TCP flow level information for each TCP connection which has been aggregated by user equipment (UE) or handheld device type, the particular RNC, SGSN, and GGSN. Each aggregated record in addition contains the timestamp of the (1hr) bin, the application type (e.g. web browsing, streaming video, etc) and content provider (e.g. www.something.com) For each TCP flow, we collect information including standard coordinated universal time (UTC), ID of the serving RNC that describes the user access point, the device type, the application type, and the content provider being accessed. In this paper, we focus on two most important E2E performance metrics in a cellular service: TCP loss ratio and Round Trip Time (RTT). For each flow, we calculate its TCP loss ratio using the following formula:

$$\left(\frac{\texttt{observed \# bytes in the flow}}{\texttt{actual \# bytes in the flow}} - 1\right)$$

The actual number of bytes in a flow means the total number of bytes in the flow, excluding retransmissions. We detect retransmitted packets by tracking packet sequence numbers. In this formula, the observed number of bytes is equal to the actual number of bytes in the flow plus the number of retransmitted bytes. For each flow, we calculate its RTT using two RTT measurements. The E2E RTT of a flow is equal to the sum of cellular network side RTT and internet side RTT. Note that all user/device identifiers (such as IMSI and IMEI) are completely anonymized to protect user privacy.

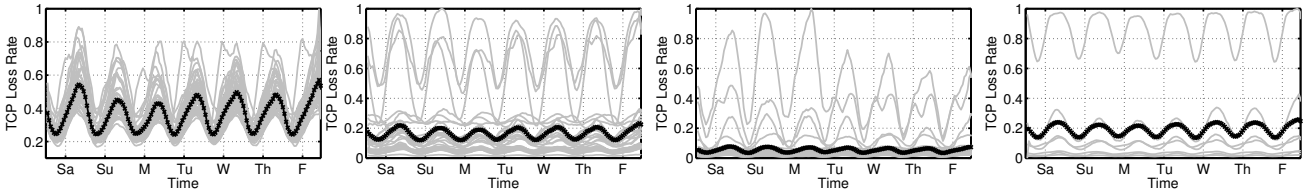
B. Data Analysis

To understand the characterization of TCP Loss Ratio and RTT across the four dimensions (namely user location, content provider, device type, and application type), we examine one week of data collected from one northeast region in the United States. We now analyze the dynamics in TCP loss ratio and RTT for each of the 4 dimensions. The insights that we gain in such analysis will be useful in our modeling of E2E performance.

Figure 1 and 2 shows the normalized hourly TCP loss ratios and RTT values for the 4 dimensions. Figure 1(a) and 2(a) shows the performance of all user locations, where the grey lines represent the individual ratios and the black line represents the average. The *aggregate hourly TCP loss ratio of a user location* is calculated based on the sum of the observed number of bytes transmitted for all downlink flows divided by the sum of the actual number of bytes in all downlink flows using the following formula:

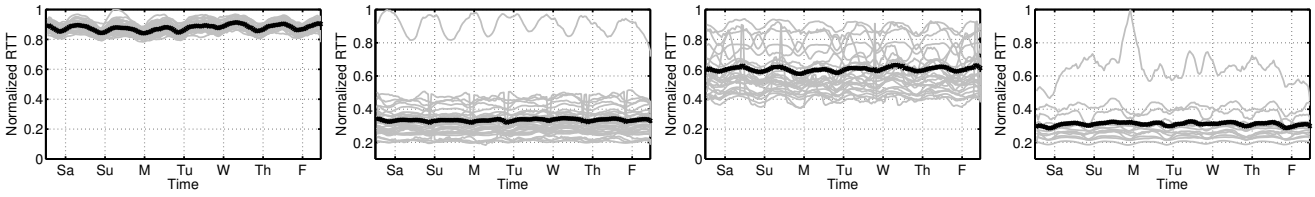
$$\left(\frac{\sum_{f\in\{\texttt{downlink flows}\}} \texttt{observed \# bytes in } f}{\sum_{f\in\{\texttt{downlink flows}\}} \texttt{actual \# bytes in } f} - 1\right)$$

From this figure, we first observe that the aggregate performance of all user locations follow a similar diurnal pattern.



(a) User Location (b) Content Provider (c) Device Type (d) Application Type

Fig. 1. Normalized TCP Loss Ratio



(a) User Location (b) Content Provider (c) Device Type (d) Application Type

Fig. 2. Normalized Round Trip Time (RTT)

This observation concurs with the daily and weekly diurnal pattern discovered in prior studies [15]. Second, we observe that the aggregate hourly TCP loss ratios for all user locations demonstrate moderate deviation from their average. Whereas, the performance of various content providers, device types and applications vary significantly. From these plots, we first observe that the TCP loss ratios and RTT values of these three dimensions follow a similar diurnal pattern as those of user locations. Second, we observe that the performance of these dimensions demonstrate significant deviation from their average, as compared to those ratios of user locations. Third, we observe that a few content providers, device types and applications constantly perform significantly worse than others.

IV. Modeling E2E Performance Dynamics

Our approach is to first build baseline E2E performance models and then detect and localize E2E performance anomalies based on these models. To build models, we first compute a two dimensional aggregate E2E performance matrix that captures the average performance along the dimensions of user locations, content providers, device types, and application types over each hour in a week. We choose a week as the duration because people have weekly diurnal patterns in their use of cellular services [15]. Next we describe our E2E performance modeling approach.

A. Aggregate E2E Performance Matrix

Given TCP flow data for a certain time period of W weeks at a certain region as the training data, let L denote the total number of user locations, C denote the total number of content providers, D denote the total number of device types, and A denote the total number of application types. For each user location, we first calculate the average performance (packet loss rate or RTT) across all content providers, all device types, and all application types, for each hour in the W weeks of $24 * 7 * W$ hours; and then, for each location and for each hour in a week of $24 * 7$ hours, we calculate the median of the W values; thus, for each hour in a week of $24 * 7$ hours, we obtain a vector of L median values. Similarly, for each content provider, we first calculate the average performance across all user locations, all device types, and all application types, for each hour in the W weeks of $24 * 7 * W$ hours; and then, for each content provider and for each hour in a week of $24 * 7$ hours, we calculate the median of the W values; thus, for each hour in a week of $24 * 7$ hours, we obtain a vector of C median values. We apply such calculation for device types and application types as well. In the end, we obtain a two-dimensional aggregate E2E performance matrix $E_A = [1..24 * 7, \{\mathbb{L}, \mathbb{C}, \mathbb{D}, \mathbb{A}\}]$, which has $24 * 7$ rows and four columns with indices denoted $\mathbb{L}, \mathbb{C}, \mathbb{D}$, and $\mathbb{A}$. In this matrix, elements $E_A[i, \mathbb{L}]$, $E_A[i, \mathbb{C}]$, $E_A[i, \mathbb{D}]$, $E_A[i, \mathbb{A}]$ are four vectors of L, C, D, and A median values, respectively, as we calculated above. This matrix E_A will be the input to the robust regression algorithm described below.

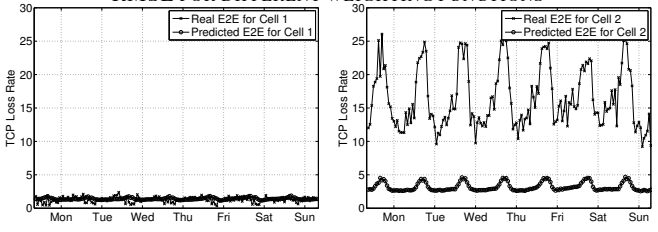
B. Coarse Grained E2E Modeling

Taking the aggregate E2E performance matrix as input, we build a single baseline model for all E2E instances, based on which we can remove extreme outlier data points. Note that these extreme outlier data points are not the E2E anomalies that we are looking for because such data points are mostly errors and noises introduced in our data collection process. We use robust regression for this baseline modeling because it can minimize the impact of extreme outlier data points on the produced model.

Robust regression uses the standard regression model $y = E_A\beta + \epsilon$, where y is the response vector of size $24 * 7$, β is the coefficient vector of size 4, and ϵ is the residual vector of size $24 * 7$. It computes a robust estimate of β such that

Weight Func.	Talwar	Bisquare	Cauchy	Huber	Logistic	Welsch
RMSE	1.23	1.40	1.64	1.99	2.04	1.44

TABLE I
RMSE FOR DIFFERENT WEIGHTING FUNCTIONS



(a) Cell 1 (b) Cell 2

Fig. 4. Real and predicted performance curves

the impact of extreme outlier data points on the produced model is minimized. To compute the coefficient β, we use the well known iteratively re-weighted least squares (IRLS) algorithm [5]. This algorithm first obtains the residual errors based on the standard least square method. Second, it uses a weighting function over the residuals to mark outliers. Third, it ignores outliers and re-estimates regression coefficients. It repeats the above process until the difference of values of estimated coefficients obtained in two successive iterations approaches a minimum threshold. Mathematically, the robust estimates of β after $n+1$ iterations can be represented as:

$$\beta^{n+1} = \text{argmin} \sum_{i=1}^{p} w_i^n |y_i - E_{A_i}\beta|^2 \quad (1)$$

Here w_i^n is the weight assigned to the i^{th} observation at the n^{th} iteration. The weight assignment is done through a weight function $w(r_i)$, where r_i is the scaled residual calculated using the method proposed in [3]. We choose the weighting functions that minimize the overall error by assigning less weights to outlying points. We used our training datasets to evaluate the performance of some well known weighting functions [5]. We calculate the Root Mean Squared Error (RMSE) of the regression models developed using six different weighting functions. Table I gives a comparison of RMSE values obtained through Ordinary Least Square (OLS) method and the IRLS method. It is clear that the *Talwar* weighting function has the least RMSE value. This is because *Talwar* weighting function assigns a weight of 1 or 0 to each data instance. Equation 2 describes the weighting criteria, which is based on the scaled residual values r_i.

$$w(r_i) = \begin{cases} 1 & \text{if } abs(r_i) < 1 \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

Using *Talwar* in IRLS, a data instances with anomalous residual value will get a weight of 0, which means that all data instances that have zero weights will not affect the regression coefficients in the next iteration. Therefore, the baseline developed for the initial training set will not be affected by these outliers.

C. Fine Grained E2E Modeling

We have defined aggregate E2E performance matrix, now we define E2E matrix. An *E2E matrix* denoted $E_I = [1..L, 1..P, 1..D, 1..A]$ is a four dimensional matrix where the four dimensions are L user locations, P content providers, D device types, and A application types. An element $E_I[l, p, d, a]$ in this matrix represents the E2E instances (or say flows) corresponding to user locations l, content provider c, device type d, and application type a.

Describing the performance of all E2E instances in the E2E matrix using only one model is not accurate enough because the performance of some E2E instances is significantly different from that of other E2E instances. For example, Figure 4(a) and (b) show the observed and predicted performance curves for two different cells of the individual E2E matrix E_I. We obtain the predicted performance curve using the single model in the above coarse grained E2E performance modeling. From Figure 4(a), we observe that for E2E matrix cell 1, the predicted performance curve are fairly consistent with the real one but from Figure 4(b), we observe that for E2E matrix cell 2, the predicted performance curve significantly deviate from the real one. Our approach to addressing this issue is to partition E2E instances into groups such that each group has distinct performance. The more groups that we partition, the more accurate models we can obtain, but at the same time, the number of models and the complexity will increase. On one extreme end, we have only one model for all E2E instances, which gives us the least accuracy and the least complexity; on the other extreme end, we have one model for each individual E2E matrix cell, which gives us the most accuracy and the most complexity. Our strategy to tradeoff between model accuracy and model complexity is to identify the E2E instance groups that have significantly different performance and then model these groups separately, leaving the rest E2E instances still being modeled by a single E2E model. Within each group, the performance of some E2E instances may be relatively different from that of others; thus, we apply this grouping strategy recursively among identified groups.

In this work, we first identify E2E instances that perform quite differently from the baseline model; then group such deviating E2E instances using association rule mining; within each group, we apply this strategy recursively. Next, we present the details of these three steps: deviating E2E instance identification, deviating E2E instance grouping, and recursive E2E instance grouping.

1) Deviating E2E Instance Identification: To identify the E2E instances that have deviating performance, we compare the real performance of each E2E instance with its predicted performance based on the baseline model for each hour in the $24 * 7$ hours of a week. To take the standard performance deviation across W weeks into consideration, we compute a two dimensional matrix denoted $\bar{E}_A$ that captures such deviation. Here $\bar{E}_A$ differs from E_A only in that each value in $\bar{E}_A$ is the standard deviation of the W values whereas each value in E_A is the median of the W values. In $\bar{E}_A$, the elements $\bar{E}_A[i, \mathbb{L}]$, $\bar{E}_A[i, \mathbb{P}]$, $\bar{E}_A[i, \mathbb{D}]$, $\bar{E}_A[i, \mathbb{A}]$ are four vectors of L, P, D, and A standard deviation values, respectively. We feed $\bar{E}_A$ to the same robust regression algorithm and obtain the

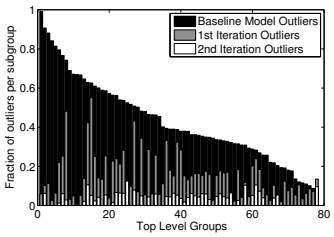


Fig. 5. #iterations vs. #outliers

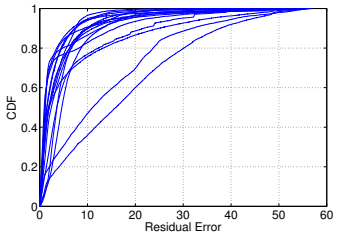


Fig. 6. Single baseline model

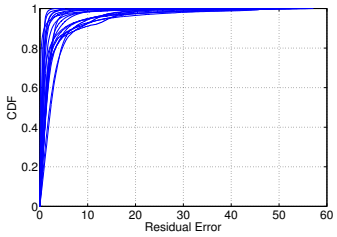


Fig. 7. Multiple models

standard deviation σ. For each E2E instance and each hour of the $24 * 7$ hours of a week, if the real performance is too much out of the range of [predicted performance - σ, predicted performance + σ], we label this E2E instance as deviating for this hour. In this work, we use [predicted performance - $2*\sigma$, predicted performance + $2*\sigma$] as the guideline for quantifying "too much". For each E2E instance, if for a large percentage of the $24*7$ hours of a week, this E2E instance has a deviating performance, we label this E2E instance as deviating. In this work, we use 50% as the guideline for quantifying "large percentage".

2) Deviating E2E Instance Grouping: Now each instance of the E2E matrix E_I has been labeled "not deviating" or "deviating". Visually, we mark non-deviating rules as white and deviating rules as black. The next step is to group such deviating E2E instances together based on their performance. In this work, we model the E2E matrix E_I as a transactional database and use the association rule mining technique to perform grouping. An instance $E_I[l, p, d, a]$ with color c is modeled as a transaction with five items $\langle l, p, d, a, c\rangle$. Given this transaction database as the input, we use the class based Apriori association rule mining algorithm to produce rules with only the color being the consequence. In association rule mining, a rule is of the form $\texttt{antecedent} \rightarrow \texttt{consequence}$ where `antecedent` and `consequence` are disjoint sets of some items. The meaning of the rule is that most people who buy `antecedent` also buy `consequence`. The quality of a rule is measured by two metrics: support and confidence. The support of a rule is the percentage of the transactions that contain all the items in both the `antecedent` and the `consequence` of the rule among all transactions in the database. The confidence of a rule is the percentage of the transactions that contain all the items in both the `antecedent` and the `consequence` of the rule among the transactions that contains all the items in `antecedent`. We are only interested in rules with the color black as the consequence and therefore ignore all rules with color white. Each rule represents a group. For example, a rule

$$\texttt{NewYork}, \texttt{Google}, \texttt{iPhone} \rightarrow \texttt{black}$$

represent a group of all E2E instances whose user location is New York, content provider is Google, and device type is iPhone. This rule means that most E2E instances corresponding to user location New York, content provider Google, and device type iPhone, regardless of application types, have deviating performance from the baseline model. Here New York is just a metaphor for a specific RNC. For simplicity, we use

$$\texttt{NewYork}, \texttt{Google}, \texttt{iPhone}, *$$

to denote this group where wild card $*$ denote all application types. The size of this group is A, which is the number of all application types. The size of each group is exactly the support of the corresponding rule. Note that some of the E2E instances in a group may be white. The percentage of the black E2E instances in a group is exactly the confidence of the corresponding rule. To tradeoff between model accuracy and model complexity, we want to find groups with a large support and a high confidence. In this work, we set the support threshold to be 0.1% and the confidence threshold to be 80%. We pay attention to only the rules whose support and confidence are above these thresholds and ignore the other rules. For simplicity, in the rest of this paper, we use the terms "rule" and "group" interchangeably.

After we filter out rules whos support or confidence is below the corresponding threshold, we still have too many groups to model each individually. We do not choose to reduce the number of rules by simply increasing support or confidence thresholds because a rule with larger support and/or confidence does not necessarily have a larger impact on the baseline model. Instead, we choose to select the groups that have the most impact on the baseline model. Next, we present an exclusion method and an inclusion method to quantify the impact of a group on the baseline model.

In the exclusion method, first, for the set of all E2E instances, denoted E_A, we use the simple regression algorithm to build the baseline model. Second, for each group of E2E instances R, we use the simple regression algorithm to build a model for $E_A - R$. Third, we calculate the absolute difference of the RMSEs of the two models for $E_A - R$ and E_A. The result is used to quantify the impact of group R. Note that here we use the simple regression algorithm, instead of the robust regression algorithm, because robust regression ignores outliers that have large effect on the RMSE.

In the inclusion method, let Σ denote the union of all groups; first, for $E_A - \Sigma$, we use the simple regression algorithm to build the baseline model. Second, for each group of E2E instances R, we use the simple regression algorithm to build a model for $(E_A - \Sigma) \cup R$. Third, we calculate the absolute difference of the RMSEs of the two models for $E_A - \Sigma$ and $(E_A - \Sigma) \cup R$. The result is used to quantify the impact of group R.

After we quantify the impact of each group on the baseline model, we can rank all groups based on their quantified impact.

Note that groups may overlap. For example, the following two groups overlap:

$$\texttt{NewYork}, \texttt{Google}, *, *$$
$$*, \texttt{Google}, \texttt{iPhone}, *$$

For any two groups R_1 and R_2 that overlap, if R_1 is ranked higher than R_2, then for all E2E instances in $R_1 \cap R_2$, we model them using the R_1's model. Given the n ranked groups in the non-ascending order of their impact, denoted by $R_1, R_2, \cdots, R_n$, for each $1 \leq i \leq n$, we build a separate model for group $R_i - \cup_{1 \leq j \leq i-1} R_j$. Note that for group R_i, if there exists $1 \leq j \leq i-1$ such that $R_i \subset R_j$, then $R_i - \cup_{1 \leq j \leq i-1} R_j = \emptyset$.

3) Recursive E2E Instance Grouping: Just like having one model for all E2E instances in E_I is not accurate enough, for some groups, having one separate model is also not accurate enough. Therefore, we apply our fine grained E2E modeling algorithm recursively on each group that we have identified as above. We now empirically show that through recursive E2E instance grouping, we achieve high modeling accuracy. We use a training data collected over a duration of six weeks (*i.e.*, $W = 6$). We calculate the aggregate E2E performance matrix using the six weeks data and build a single baseline model for all E2E instances. We then identify a first set of deviating E2E instance groups. For each of these E2E instance groups, we calculate the total number of E2E instances that are deviating from the single baseline model. In Figure 5, the black bars are the fraction of deviating E2E instances belonging to a particular group, and the grey bars represent the fraction of deviating E2E instances when a separate model is used for each of the E2E instance groups. For each of the E2E instance groups, we develop a separate set of fine grained models through recursion. The white bars are the fraction of deviating E2E instances in the original E2E instance group after separate sets of fine grained models are used for each E2E instance group after recursion. We observe that after at most three levels of recursion, we do not get any further E2E instance grouping. Figure 6 shows the CDF of residual errors obtained when a single baseline model is used. Figure 7 shows the CDF of residual errors obtained when a separate baseline models are used. We observe that recursive E2E instance grouping and fine grained modeling greatly reduce residual errors.

D. Performance Degradation Detection

So far we have discussed how to build E2E performance models based on training data. Now we present solutions to detect and localize degradations in an on-going data feed on an hourly basis. In order to determine whether a particular E2E instance in the on-going data feed has degrading performance in the latest hour, we compare the performance of that E2E instance in the latest hour with its predicted performance based on the fine-grained model for that E2E instance for the same hour. For a given E2E instance at a particular hour, if the value of its performance metric is too much higher than the predicted performance, then we label this E2E instance as "degrading". We use the predicted performance + $2 * \sigma$ as the guideline for quantifying "too much".

E. Performance Degradation Localization

For each new hour in the on-going data feed, we first label each E2E instance as degradation or normal and then apply association rule mining algorithm on the E2E instances labeled as degradation. Unlike the training phase, we do not apply the association rule mining recursively. The output rules show the localization of performance degradations. For example, the following rule shows that for a particular hour, for all content providers, all device types, and all applications, the cellular users in the New York area are experiencing significant degrading performance for their cellular services.

$$\texttt{NewYork} \rightarrow \texttt{degrading}$$

This rule is useful for cellular network operators to investigate potential issues in their cellular services in the New York area. Again, here New York is just a metaphor for a specific RNC. For another example, the following rule shows that for a particular hour, for all locations, for all content providers, the cellular users of iPhone are experiencing significant degrading performance for their email application.

$$\texttt{iPhone}, \texttt{Email} \rightarrow \texttt{degrading}$$

This rule is useful for cellular network operators to contact iPhone manufactures to investigate potential issues in their email application.

V. Evaluation

In this section, we evaluate our approach using the operational data collected from a large US-based cellular service provider. The main challenge in our evaluation is lack of ground truth anomalies. Due to the prohibitively huge size of the data, it is practically infeasible to manually label all anomalous events. To address this challenge, we introduce some synthetic anomalies into the collected operational data to effectively evaluate the accuracy of our approach. Specifically, we first apply our approach on the operational data collected during six consecutive weeks to learn the fine grained E2E performance models and then use these models to detect and localize synthetic anomalies. Besides synthetic anomalies, we also present a few real-world anomalies detected and localized by using the learned fine grained E2E performance models.

A. Synthetic Anomalies

To evaluate our system we introduce synthetic anomalies in the data. We created three sets of scenarios, in the first set we have 19 scenarios and in each scenario we introduce a one dimensional anomaly, the second set involves 26 scenarios with two dimensions and the third set involves 19 scenarios with three dimensions. In each scenario we introduced a one-hour anomaly that is associated with a group of instances in the huge E2E performance matrix. For example, a particular scenario introduces performance degradations at a particular hour, for all users accessing their email (application type) from Motorola V3 Razr (device type). Specifically, we first collect

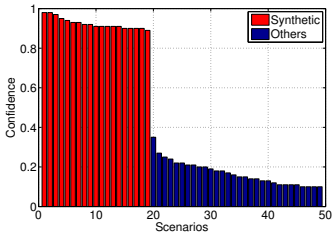


Fig. 8. 1-dimensional scenarios

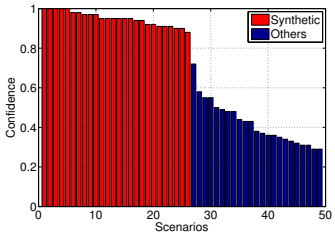


Fig. 9. 2-dimensional scenarios

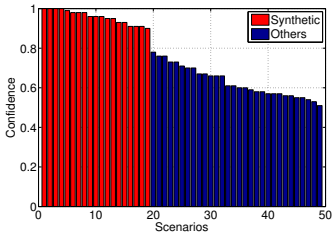


Fig. 10. 3-dimensional scenarios

another week's operational data (different from the six weeks we used to learn the fine grained E2E performance models) and calculate the mean and standard deviation of the TCP loss ratio for each E2E instance using 168 hours of that week. Then for each E2E instance in each particular scenario, we create a one-hour synthetic anomaly by using $Anom = \mu + 2\sigma + |\mathcal{N}(0, 2)|$.

For each scenario, we first used the fine grained E2E performance models learned from six-week data to detect anomalies for each instance of E2E matrix and then use association rule mining to localize the root cause of all detected anomalies. We use confidence as our evaluation metric for the rules obtained through association rule mining and select the top ranked rule obtained from association rule mining according to their confidence values. As we can see from Figure 8, Figure 9 and Figure 10, all 64 synthetic anomalies that we injected into the collected operational data are successfully detected and localized. We get all scenarios as association rules with 90% confidence for one dimensional scenarios. We achieved 0 false positives for a minimum confidence threshold of 70% and 80% for two and three dimensional scenarios respectively.

	Dim	Loc	Dev	App	CP
Loss	Anom	0	5	98	27
	Conf	0	0.80	0.68	0.67
RTT	Anom	0	11	82	0
	Conf	0	0.82	0.68	0

TABLE II
SINGLE DIMENSION ANOMALIES

B. Anomaly Detection in the Wild

In this section, we characterize the performance anomalies detected on an operational network over a period of one week in August 2014. Through this characterization analysis, we will explain that how frequent E2E performance anomalies are and how often multiple dimensions are the root cause of performance degradation. We implemented and deployed our system on an operational network and detected performance anomalies for packet loss ratio and RTT separately. Overall, we monitored 78 different regions, 51 device types, 13 application types and 36 content providers.

It is interesting to see that how often a single dimension is responsible for performance degradation. We use the output rules from the localization step to count the number of single dimensional and multi-dimensional anomalies. For each rule, we count the total number of all E2E instances and the number of E2E instances marked as degrading, which belong to that rule. Table II gives the number of one dimensional anomalies detected on hourly basis. For each dimension the number of anomalies is the number of E2E instances which were marked as degrading in the detection and localization process as discussed in Section IV. We also report the average confidence that is the fraction of all E2E instances which were marked as degrading. We can see that there are no performance anomalies in the user location dimension, which means that a single user location cannot be blamed for the anomalies occurring during the one week's time period. We observe that one dimensional anomalies are rare for content providers and device types. This is mainly because content providers use content distribution networks for serving content to remotely located users. It is very unlikely that the performance of a content provider will degrade across all user locations. Therefore, a problematic content server can only affect the performance of a subset of user locations. For applications, single dimension anomalies are relatively frequent. This observation highlights the performance impact of application patches and bug fixes that are released periodically and frequently and bugs in these updates can cause performance issues at all user locations, across all device types and content providers.

Next we look at multi-dimensional anomalies reported in Table III. We observe that anomalies involving multiple dimensions are more frequent as compared to single dimension anomalies. We detected a total of 8415 E2E instances as anomalous. Our analysis reveals that in 83% of these E2E instances content provider was one of the dimensions, 86% involved device types and 86% involved application types. Overall, in 80% of these instances, the RNC or location dimension was not involved. In other words, 80% of the time the anomalies are not because of problems at the cellular network. This composition of E2E performance anomalies shows that most of the time content providers, device types and applications are involved in the performance degradations. These results are extremely useful for network operators as they highlight the nature of majority of performance anomalies. Additionally, the findings point network operators to the problematic dimensions and reduce the search space of root causes.

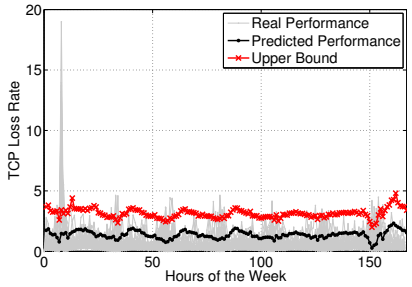


Fig. 11. Real-world Anomaly 1: root cause outside the cellular network

	Dim	Loc,Dev	Loc,App	Loc,CP	Dev,App	Dev,CP	App,CP	Loc,Dev,App	Loc,Dev,CP	Loc,App,CP	Dev,App,CP
Loss	Anom	134	0	215	79	263	0	378	44	233	962
	Conf	0.94	0	0.93	0.76	0.83	0	0.82	0.80	0.76	0.78
RTT	Anom	57	13	56	280	355	0	32	20	444	4627
	Conf	0.80	0.85	0.84	0.71	0.83	0	0.82	0.96	0.72	0.78

TABLE III
MULTI DIMENSION ANOMALIES

C. Real-world Anomalies

In the section, we present two interesting real-word anomalies detected and localized by the learned fine grained E2E performance models (same as the synthetic anomalies) and the possible underlying root causes.

The first real-world anomaly involves three dimension namely, content provider, device type and application type. Our system detected and localized a one-hour anomaly across all 78 user locations (corresponding to 78 RNCs) but specific to apple.com (content provider), iPhone 4s (device type) and browsing (application type). Figure 11 shows the time series plot of the TCP loss ratio for all 78 user locations , apple.com, iPhone 4s and browsing. We also plot the predicted performance and the upper bound of the TCP loss ratio for apple.com, iPhone 4s and browsing across all locations. One can clearly see that the anomaly occurred around the 8th hour in the week for all 78 user locations. As the anomaly is across all locations and specific to apple.com, iPhone 4s and browsing, the root cause is most likely to be outside of the cellular network and might be apple related.

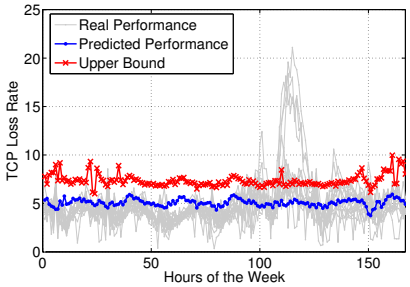


Fig. 12. Real-world Anomaly 2: root cause inside the cellular network

In the second real-world anomaly, all four dimension are involved. Our system detected and localized a 15-hours anomaly for four user locations across all content providers, device types and application types. Figure 12 shows the time series plot of the TCP loss ratio for all locations aggregated on other 3 dimensions. We only picked one user location to plot the predicted performance and the upper bound of the TCP loss ratio for that user location, apple.com, iPhone 4s and browsing due to space limits. One can clearly see only 4 curves (corresponding to 4 RNCs) had a significant spike starting around hour 108. As a contrast to the first real-world anomaly, the anomalies here are only specific to 4 user locations regardless other 3 dimensions. Thus the root cause here is most likely to be something inside the cellular network impacting the 4 user locations (corresponding to 4 RNCs).

VI. CONCLUSIONS

We make the following key contributions in this paper. First, we design and implement a comprehensive and holistic measurement system that monitors E2E service performance across four dimensions - user locations, content providers, device types, and application types. Second, we propose fine grained models that capture the normal performance for every combination of user locations, content providers, device types, and application types, and use the models to detect performance degradation. Third, we propose an association rule mining based approach to localize performance degradation. Fourth, we use both real network traces and synthetic performance degradation to show that our method is highly effective.

REFERENCES

[1] P. Bahl, R. Chandra, A. Greenberg, S. Kandula, D. A. Maltz, and M. Zhang. Towards highly reliable enterprise network services via inference of multi-level dependencies. In *Proc. of the ACM SIGCOMM*, pages 13–24, 2007.

[2] M. Chen, E. Kiciman, E. Fratkin, A. Fox, and E. Brewer. Pinpoint: Problem determination in large, dynamic Internet services. In *Proc. of the Int. Conf. on Dependable Systems and Networks*, pages 595–604, 2002.

[3] W. Dumouchel and F. O'Brien. Integrating a robust option into a multiple regression computing environment. In *Proc. of the 21st Symposium on the Interface of Computer Science and Statistics*, page 41, 1991.

[4] A. Finamore, M. Mellia, M. M. Munafò, R. Torres, and S. G. Rao. Youtube everywhere: Impact of device and infrastructure synergies on user experience. In *Proc. of the ACM IMC*, pages 345–360, 2011.

[5] P. W. Holland and R. E. Welsch. Robust regression using iteratively reweighted least-squares. *Communications in Statistics-Theory and Methods*, 6(9):813–827, 1977.

[6] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck. A close examination of performance and power characteristics of 4g lte networks. In *Proc. of the 10th Int. Conf. on MobiSys*, pages 225–238, 2012.

[7] J. Huang, F. Qian, Y. Guo, Y. Zhou, Q. Xu, Z. M. Mao, S. Sen, and O. Spatscheck. An in-depth study of lte: Effect of network protocol and application behavior on performance. In *Proc. of the ACM SIGCOMM*, pages 363–374, 2013.

[8] S. Kandula, D. Katabi, and J. Vasseur. Shrink: A tool for failure diagnosis in IP networks. In *Proc. of the ACM SIGCOMM workshop on Mining network data*, pages 173–178, 2005.

[9] S. Kandula, R. Mahajan, P. Verkaik, S. Agarwal, J. Padhye, and P. Bahl. Detailed diagnosis in enterprise networks. *Proc. of the ACM SIGCOMM*, pages 243–254, 2009.

[10] R. Kompella, J. Yates, A. Greenberg, and A. Snoeren. Detection and localization of network black holes. In *Proc. of the IEEE INFOCOM*, pages 2180–2188, 2007.

[11] R. R. Kompella, J. Yates, A. Greenberg, and A. C. Snoeren. Ip fault localization via risk modeling. In *Proc. of the NSDI*, pages 57–70, 2005.

[12] A. A. Mahimkar, Z. Ge, A. Shaikh, J. Wang, J. Yates, Y. Zhang, and Q. Zhao. Towards automated performance diagnosis in a large iptv network. In *Proc. of the ACM SIGCOMM*, pages 231–242, 2009.

[13] A. A. Mahimkar, H. H. Song, Z. Ge, A. Shaikh, J. Wang, J. Yates, Y. Zhang, and J. Emmons. Detecting the performance impact of upgrades in large operational networks. In *Proc. of the ACM SIGCOMM*, pages 303–314, 2010.

[14] F. Qian, Z. Wang, A. Gerber, Z. Mao, S. Sen, and O. Spatscheck. Profiling resource usage for mobile applications: a cross-layer approach. In *Proc. of the 9th MobiSys*, pages 321–334, 2011.

[15] M. Z. Shafiq, L. Ji, A. X. Liu, and J. Wang. Characterizing and modeling internet traffic dynamics of cellular devices. In *Proc. of the ACM SIGMETRICS*, pages 305–316, San Jose, California, June 2011.