4-2016

Not you too? Distilling local contexts of poor cellular network performance through participatory sensing

Huiguang LIANG

Ido NEVAT

Hyong S. KIM

Hwee-Pink TAN
Singapore Management University, hptan@smu.edu.sg

Wai-Leong YEOW

DOI: https://doi.org/10.1109/NOMS.2016.7502836

Citation

LIANG, Huiguang; NEVAT, Ido; KIM, Hyong S.; Hwee-Pink TAN; and YEOW, Wai-Leong. Not you too? Distilling local contexts of poor cellular network performance through participatory sensing. (2016). *Proceedings of the NOMS 2016: 2016 IEEE/IFIP Network Operations and Management Symposium, April 25-29, 2016, Istanbul, Turkey*. 392-400. Research Collection School Of Information Systems.
Available at: https://ink.library.smu.edu.sg/sis_research/3328

Published in Proceedings of the NOMS 2016: 2016 IEEE/IFIP Network Operations and Management Symposium, April 25-29, 2016, Istanbul, Turkey. Piscataway, NJ: IEEE. pp. 392-400.
http://doi.org/10.1109/NOMS.2016.7502836

Not You Too? Distilling Local Contexts of Poor Cellular Network Performance through Participatory Sensing

[†] Huiguang Liang, [∞] Ido Nevat, [⋆] Hyong S. Kim, [◊] Hwee-Pink Tan, [ø] Wai-Leong Yeow

[†] Social & Cognitive Computing, Institute of High Performance Computing, Singapore

[∞] Sense & Sense-abilities Programme, Institute for Infocomm Research, Singapore

[⋆] Dept. of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, United States of America

[◊] SMU-iCity Lab, Singapore Management University, Singapore, [ø] Solace Systems, Inc., Singapore

Corresponding Author: [†] `hliang@ihpc.a-star.edu.sg`

***Abstract*—Cellular service subscribers are increasingly reliant on cellular data services for all kinds of mobile applications. Oftentimes, when subscribers experience frustratingly high network delays and timeouts, they like to know whether their experiences are shared by other users nearby. The question that is often asked is essentially this: "*is it just me, or do others around me face the same problem?*"**

In this paper, we describe how we use *Tattle*, a distributed real-time participatory sensing and monitoring framework, to glean network performance information from users nearby. Tattle relies on recent advances in peer-to-peer device networking, such as Wi-Fi Direct, Bluetooth Low Energy, and Apple's iBeacon, to exchange key snippets of diagnostic information using very low-power, very short-range local-area wireless interfaces, between participating devices. We propose and develop a robust statistical algorithm, based on quantile regression, which identifies key points in time where a device experiences high delays and outages that are not observed by its neighbors, and decides if the device is performing "normally", or "abnormally". This directly answers the "*me, or others?*" question. We demonstrate and validate the efficacy of our system through real-world measurements of network delay, consisting of over 7,300 time-series that comprises over 443,500 data samples, using commodity smart devices attached to two different providers' networks.

I. INTRODUCTION

As traffic load increases, cellular networks naturally become increasingly congested, and subscribers tend to experience high network delays, slow speeds, and possibly service outages in cells and areas that are overloaded. This can be a common occurrence and a frustrating experience, especially during periods of congestion, such as rush hours and spectator events.

Whenever a subscriber experiences periods of unsatisfactory network performance, the subscriber may often blame the cellular operator for providing inadequate coverage and capacity. On the other hand, the cellular operator may blame the user's device or the software as the cause. It is often difficult to identify the root cause of network performance issues.

We believe that one way to assist the diagnosis of the root cause from the subscriber's perspective is to ask: "*am I the only one suffering from this, or is this also experienced by others?*" For ease of reference, we shall refer to this as the "*me, or others?*" (MOO) problem. The subscriber's intention (often without consciously realizing) is to determine if the fault lies with his own device (possibly as a result of software/hardware issues), or with the network. If the subscriber is able to determine that the former is true, he can take some limited mitigation steps, such as rebooting his device, closing errant apps, etc. If the latter is true, then the subscriber gains closure in knowing that his device is likely to be working fine, and that other co-located subscribers in the same area are experiencing the same kinds of performance impairments.

A. *Paper contribution and overview*

The contributions of this paper are as follows:

1. We describe how we use *Tattle* [1], a distributed real-time participatory sensing and monitoring framework, to allow participating devices to glean networking performance information from each other in real-time, while preserving the context of co-location.
2. On top of that, we propose a complete and flexible statistical framework, based on *quantile regression* and *outlier classification*, to analyze and systematically identify points in time where a device is not performing as well as its co-located neighbors, and decides if the device is performing "normally", or "abnormally".
3. We report on the characteristics of the delay performance of co-located devices subscribed to two cellular network operators in Singapore, and describe the results of applying our proposed approach to answering the MOO question, on real-world measurements of over 7,300 time-series of network delay, consisting of over 443,500 data points.

With these contributions, this paper provides a framework to comprehensively and systematically answer the key MOO question of whether poor network performances is an isolated problem, faced by one or a few devices in particular, or an endemic condition that affects most devices in a given area.

This paper is organized as follows. In Section II, we describe the general problem and provide a brief introduction to existing work. In Section III, we describe the Tattle system, and how it allows smart devices to leverage on low-power, short-range peer-to-peer information exchange to glean performance measurements from other co-located participants. In Section IV, we introduce the proposed statistical framework that we will build on top of Tattle, in order to answer the MOO problem. In Section V, we describe our experimental methodology, and present some key features of the 7,300 real-world network delay time-series that was collected. We then apply our proposed algorithm on the data, and report on the results. Finally, we conclude and discuss future work in Section VI.

II. PROBLEM DESCRIPTION AND BACKGROUND REVIEW

When a subscriber suffers impairments to the underlying mobile data service, the experience can be exceedingly frustrating. One key question that should be asked is this: "*is it just me, or does the problem lie with the network?*" We introduced this earlier as the "*me, or others?*" (MOO) problem. In this paper, we formalize and propose a system based on low-power, short-range peer-to-peer exchange of network performance information to systematically answer the MOO question. A robust statistical framework is proposed to take into account a participant's, as well as his co-located neighbors' network performance information to determine whether a device is performing relatively "normally", or "abnormally".

A. *Background Review*

Fault management in wireless networks is a fairly-well studied topic, with strong existing contributions addressing the management of Wi-Fi network faults in particular. In [2], the authors propose the use of large arrays of commodity desktop computers, equipped with Wi-Fi cards or dongles, as enterprise network sensors and monitors. In another study [3], its authors suggest that Wi-Fi clients and access points can be instrumented to become diagnostic agents, which can be directed to switch to promiscuous mode to help detect and relay problems.

Cellular networks have conventionally required more centralized approaches to fault management because of its large geographical spread, provisioned QoS, strict centrally-managed infrastructure, and until recently, 'dumb' clients, which cannot be easily instrumented to perform any complex tasks. Existing work focus on detecting network-side faults, down to at most a cell-wide level, by examining key performance indicators (KPIs). In [4][5], the authors provide a brief description of possible faults and symptoms in cellular networks, and propose a Bayesian inference model to compute the probability of a fault based on observed KPIs. In another recent study [6], its authors propose a two-stage detection-diagnosis model where each monitored KPI is compared against normal 'profiles', and deviations are matched to known root causes.

Our work differs fundamentally from existing work in the following ways:

1. The MOO problem is a different problem from conventional fault management. Here, we are only interested in the question of whether one participating device is getting at least as good network performance *as compared to its co-located neighbors*. The motivation is to allow participants to either perform limited mitigation steps to alleviate impairments (due to device-related issues), or simply wait out periods of poor performance as others nearby are also suffering from the same impairments.
2. Our approach requires no instrumentation on the part of participating devices, except to run a mobile app. No hardware modifications, nor any changes in device functionality, are required, unlike those commonly suggested in studies addressing Wi-Fi management. Also, modifications to the network infrastructure are not required.
3. Our approach requires no *a prior* knowledge of the structure, symptoms and causes of faults.

Another area related to our work is that of *participatory sensing* [7]. The proliferation of smart devices has been a key enabler for many interesting research projects in this area. Participatory sensing leverages on the potential collaborative and participative nature of co-located smart devices to achieve some greater goal.

III. TATTLE – DISTILLING LOCAL CONTEXT OF PERFORMANCE THROUGH COLLABORATION

Tattle, as proposed in [1], is a distributed monitoring framework that is scalable, and monitors real-time network performance on large geographic areas with good measurement location fidelity, and requires minimal involvement of subscribers (other than to allow their devices to participate in monitoring by running a background app on their smart device).

The framework primarily involves three generic components:

1. *Peer-to-peer front-end*: Tattle advocates the use of peer-to-peer wireless interfaces to allow participating devices to communicate diagnostic and monitoring information to other nearby participants. This component is critically useful for applications that require the context of *co-location*, i.e., discovering and communicating with other devices that are in close proximity. We are convinced that the barrier to adopting this approach is becoming much lower, with the advent of recent standards in peer-to-peer ad hoc wireless networking, such as Apple's iBeacon [8], the increasingly-pervasive Bluetooth Low Energy [9], as well as Wi-Fi Direct [10], all of which feature convenient, *ultra-low-power* short-range communication capabilities that require minimal participant involvement (other than expressly permitting an app on their smart devices to make use of these interfaces).
2. *Transmission to back-end*: For the purposes of aggregate sensing and monitoring, measurements have to be transmitted to the back-end. The frequency and fidelity of these transmissions are mostly dependent on the needs of the sensing and monitoring application. A common and well-studied approach is to elect a representative, in a cluster of peer-to-peer devices, to upload the requisite information, such as an aggregated measurement (e.g. mean ambient temperature) to the back-end.
3. *Back-end pre-processing and post-processing*: This component is an abstraction of all the necessary processing that is required by the application built on top of the Tattle framework.

In the context of answering the MOO problem, we focus on the peer-to-peer front-end component to gather measurements from the co-located devices, transmit them to the back-end, and use our proposed analytics framework at the back-end to answer the MOO question, and feeding the result back to the user.

A. *The MOO app – Description and operation*

The MOO app, built using the Tattle framework, is a simple prototype to demonstrate the efficacy of our proposed approach in answering the MOO problem. It exploits Wi-Fi Direct as a local communication interface for devices to exchange delay measurements, though the app can be easily extended to use other aforementioned wireless interfaces. Wi-Fi Direct is an attractive choice because it can work in tandem with a user's existing Wi-Fi association to any Wi-Fi Access Point.

In our prototype, each participating device attempts to measure network delay by probing a standard set of servers. We will show in Section V that most of the delay variability observed is not dependent on the choice of server, so long as the servers are hosted in the same wide-area network (WAN) (and hence differences in propagation delay between the provider's core network and the probed servers are negligible). Hence, in a production setting, each participant need only to probe one standard server.

Each probe is a HTTP HEAD request for each server, such as google.com.sg, youtube.com, and our self-managed physical server, which we shall refer to as UNISENSE. Each probing attempt generates a very small amount of data transferred on the cellular uplink and downlink. For example, based on interface packet captures, we found that each fetch to google.com.sg involves only 484 bytes transferred on the uplink, and 1079 bytes downloaded, fully inclusive of HTTP and TCP/IP overheads. We designed our prototype such that each device attempts to measure the network delay every five seconds. This interval is chosen so as to avoid excessive and unnecessary traffic, to conserve power, and more importantly, to avoid the cellular Buffer Bloat problem [11] which can cause an undesirable skew in measured delay. Probing each server in this manner generates just a little over 1 megabyte of data in total per hour. In a production setting, the probe interval can be 'on-demand', or set to a much larger value so as not to congest the network further.

Simple PING probes are not used because many networks and servers actively blocks ICMP traffic, and PING behaviors do not fully reflect the overheads incurred at the transport and application layer.

The devices' times are synchronized to the order of milliseconds using the Network Identity and Time Zone (NITZ) protocol. Every time a network probe is complete, each device will broadcast the result of that probe as a tuple of <Probe Timestamp, Hashed Device ID, Measured Server ID, Measured Delay>. For probes that experience time-outs (where the HTTP connection did not receive a reply after more than 10 seconds), a nominal value of 20 seconds is used. This broadcast is done on the Wi-Fi Direct interface, and any device that overhears this broadcast simply retains it in memory. This will form that device's 'cluster' of readings, consisting of its own as well as those of others' nearby.

In order to answer the MOO question, the goal is to collect these measurements from other nearby devices, as well as a participant's own measurements, for a small window of time. Subsequently, we apply a statistical algorithm that essentially determines the following:

1. In the given window of time, at which periods were a given device performing considerably worse than others?
2. Given those periods that a device was said to be performing worse than others, is the device performing "normally", or "abnormally" on the whole?

If the device is deemed to be performing "normally", then the answer to the MOO question is clearly "*others*". If the device is determined to be performing "abnormally", then the answer to the same question is naturally "*me*". This algorithm can then be repeated as desired using a sliding window approach, or simply at fixed intervals. For our prototype, computation for each cluster of readings is performed at the back-end.

B. A note on power, incentives, security, privacy and trust

In this paper, we focus on the systems aspect of our proposed approach to addressing the MOO question. However, we recognize that power consumption, incentivization, security, privacy and trust are key elements of any participatory sensing framework. We note that there are ongoing research into the aforementioned topics, and refer to recent work such as [1], [12], [13], [14] and [15], which addresses these concerns.

IV. Optimal Classification With Robust Estimation

Our goal is to design a classifier, such that given a frame of L observations of measured network delay at a selected device, denoted by $\mathbf{Y} := \{Y_i\}_{i=1}^{L} \in \mathbb{R}^L$, as well as those of its K neighboring devices, denoted by $\mathbf{X} := \{\mathbf{X}_i\}_{i=1}^{L} \in \mathbb{R}^{L\times K}$, the classifier will detect if the device under investigation performs "normally", or "abnormally".

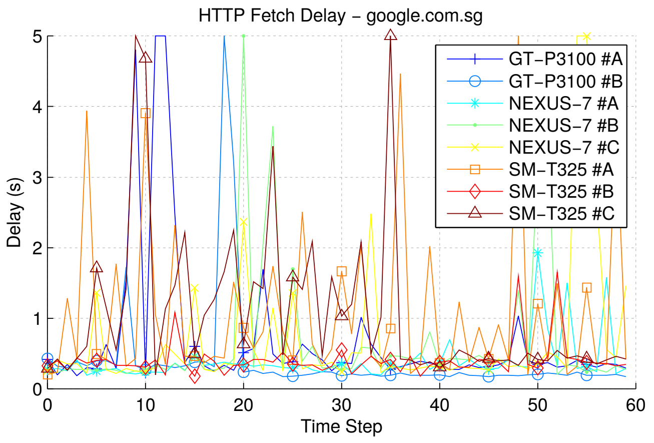

Figure 1: A representative plot of HTTP fetch delay measurements conducted by 8 co-located devices. Each device fetches a small HTML resource from google.com.sg every 5 seconds and measures the delay taken for the fetch to be completed.

To do so, we perform the following steps:

1. Fit a regression model to explain the relationship between the observations $\mathbf{Y}$ from the device under investigation, and the explanatory variables $\mathbf{X}$, which are the realizations from the stochastic processes measured by the neighboring devices.
2. Outlier detection via quantile regression residual analysis, based on the regression coefficients.
3. Count the number of outliers n in a given window. We assume that n is a realization of the random variable $N \sim F_m(n; \theta_m)$ that is an outcome of one of two models, namely "normal", or "abnormal", where m is the model indicator.
4. Perform hypothesis testing to decide if the number of outliers in a given window is consistent with a "normal" behavior (the null hypothesis, $\mathcal{H}_0$) or an "abnormal" behavior (the alternative hypothesis, $\mathcal{H}_1$).

A. Algorithm Description

We now detail each of the steps in our algorithm.

1) Fitting of a regression model

The most standard regression model structure is *mean regression*, in which one would assume for instance a linear relationship between the observed process and the p explanatory variables $\boldsymbol{x}_i = (x_{i1}, \cdots, x_{ip})$, which include the measurements of the other devices, given by,

$$Y_i = \alpha_0 + \textstyle\sum_{j=1}^{p} \alpha_j \boldsymbol{x}_{ij} + \epsilon_i, \tag{1}$$

where ϵ_i is a random variable representing the residual error, which accounts for the fact that the regression model does not capture all variation in the observed process. In the case that the random variables ϵ_i were all i.i.d. with a symmetric zero mean distribution, then this would be equivalent to modelling the conditional mean of the process given a linear function:

$$\mathbb{E}[Y_i|\boldsymbol{x}_i] = \alpha_0 + \textstyle\sum_{j=1}^{p} \alpha_j \boldsymbol{x}_{ij}. \tag{2}$$

The estimation can then be solved via maximum likelihood if the distribution of the errors is assumed, or via least squares. The coefficients are given by:

$$(\hat{\alpha}_0, \cdots, \hat{\alpha}_p) = \arg\min_{\alpha} \sum_{i=1}^{L} \left(y_i - \alpha_0 + \sum_{j=1}^{p} \alpha_j \boldsymbol{x}_{ij}\right)^2. \quad (3)$$

The main limitation of linear regression relates to outliers present in the observed data, which mean regression is highly sensitive to (and hence, not robust). Figure 1 is a representative plot of actual measured delays experienced by eight co-located devices when attempting to fetch a common HTTP resource. With data-sets such as these, mean regression will unbiasedly adapt to the numerous, random 'spikes' in the delay by minimizing the overall square error. This behavior is the reason why mean regression is not robust to outliers.

In contrast to mean regression, quantile regression models are capable of dealing with the presence of outliers in the data. Our approach is hence based on Quantile Regression (QR), which has the following properties:

1. QR is a robust framework that can identify outliers (i.e., spikes in network delay that are not observed by other neighboring devices), and heavy tail realizations from the process.

2. QR is flexible and allows us to introduce any relevant covariates (independent regressor variables) to make inference on the properties of the physical process.

3. QR reconstructs the level sets for the physical process in a consistent and model-based approach, which does not rely upon simplifying assumptions on the distribution of the underlying process properties.

a) Quantile Regression – A brief primer

Under a parametric approach, we assume that $Y_i^* \sim F(y^*|\boldsymbol{\theta})$, where $F(y^*|\boldsymbol{\theta})$ is the conditional cumulative distribution function and $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ is a vector of model parameters, all unknown coefficient parameters, and distributional parameters. The quantile function for the conditional distribution of Y_i^* given $\boldsymbol{x}_i$ at a quantile level $u \in (0,1)$ is:

$$Q_{Y^*}(u|\boldsymbol{x}_i) \equiv \inf\{y^* : F(y^*|\boldsymbol{\theta}) \geq u\} = \arg\min_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \mathbb{E}\left[\rho_u(\epsilon_i)\right], \quad (4)$$

where the loss function in the expectation is given by:

$$\rho_u(\epsilon) = y(u - \prod[y > 0]). \quad (5)$$

Under this formulation, the conditional quantile function in (4) is given by,

$$Q_{Y^*}(u|\boldsymbol{x}_i) = \mu_i + Q_\epsilon(u)\sigma_i, \quad (6)$$

where $Q_\epsilon(u) = F_{z^*}^{-1}(u)$ is the inverse cumulative distribution function for the standardized variable $Z_i^* = \frac{Y_i^* - \mu_i}{\sigma_i}$, and:

$$\textbf{location: } \mu_i = \alpha_0 + \sum_{k=1}^{m} \alpha_k x_{ik},$$
$$\textbf{scale: } \sigma_i^2 = e^{(\beta_0 + \sum_{k=1}^{v} \beta_k s_{ik})}. \quad (7)$$

This optimization problem can be solved efficiently using the approach given in [16].

2) Calculating residuals and classifying outliers

To decide if the ith sample in Y_i is an outlier/inlier, we perform a residual-based outlier detection, as detailed in [17]. To achieve that, we calculate the residual for the ith sample as:

$$r_i = y_i - Q(0.50|\boldsymbol{x}_i), \quad (8)$$

where $Q(0.50|\boldsymbol{x}_i)$ is the 50th conditional quantile for the ith observation. This corresponds to a median regression instead of a mean regression. We perform the following hypothesis test:

$$\mathcal{H}_{\text{inlier}} : r_i < k_r\hat{\sigma} \text{ (Inlier)}$$
$$\mathcal{H}_{\text{outlier}} : r_i \geq k_r\hat{\sigma} \text{ (Outlier)}$$

where k_r is a resistant parameter that controls the cut-off rate, and $\hat{\sigma}$ is the corrected median of the absolute residuals:

$$\hat{\sigma} = \text{median}\left(\frac{|r_i|}{\hat{\beta}_0}, i = 1, \cdots, L\right), \quad (9)$$

where $\hat{\beta}_0 := \Phi^{-1}(p)$ is the inverse cumulative distribution function (CDF) of Gaussian density with the pth quantile. Note that the formulation of Equation (8) distinguishes between positive residuals (where the actual delay is higher than the regressed quantile) and negative residuals. The latter case, where actual delay y_i is lower (better) than the regressed quantile $Q(0.50|\boldsymbol{x}_i)$, is actually desirable, and hence will always be considered an inlier.

3) Counting the number of outliers in a given frame

We define the following random variable N to represent the number of outliers identified in a given window, where,

$$N = \sum_{i=1}^{L} \mathbf{1}(r_i \text{ declared as } \mathcal{H}_{\text{outlier}}), \quad (10)$$

and where $\mathbf{1}(.)$ is the indicator function, and L is the frame length. We used extensive analysis of real data from both "normal" and "abnormal" representations to model N as a realization from a Geometric distribution with different success probabilities p, as follows:

$$\mathcal{H}_0 : N \sim \text{Geo}(p_{\text{normal}})$$
$$\mathcal{H}_1 : N \sim \text{Geo}(p_{\text{abnormal}})$$

where $\text{Geo}(p)$ is the Geometric distribution defined as $\mathbf{P}(N = n; p) = (1-p)^n p$. To find the values p_{normal} and p_{abnormal}, we used 384 time-series as a training data set, based on which we performed a Maximum Likelihood estimation (MLE), as presented next. The likelihood function is given by:

$$\mathbf{P}(N_1 = n_1, \cdots, N_L = n_L) = \prod_{l=1}^{L} (1-p)^{n_l} p. \quad (11)$$

By taking the derivative of the logarithmic transform, setting it to zero and solving, we obtain the MLE estimate as follows:

$$\hat{p} = \frac{1}{\frac{\sum_{l=1}^{L} N_l}{L} + 1}, \quad (12)$$

where L is the number of training examples for each model.

4) Device Classification via Likelihood Ratio Test

The final step of the algorithm involves a second hypothesis test to classify the behavior of the device under investigation. This step tests whether the number of outliers detected is best explained by a "normal" behavior (the null hypothesis, denoted by $\mathcal{H}_0$) or an "abnormal" behavior (the alternative hypothesis, denoted by $\mathcal{H}_1$). To achieve this, we derive the Likelihood Ratio Test (LRT), given by:

$$\Lambda(Y_{1:L}) := \frac{\mathrm{P}(N = n|\mathcal{H}_0)}{\mathrm{P}(N = n|\mathcal{H}_1)} \underset{\mathcal{H}_1}{\overset{\mathcal{H}_0}{\gtrless}} \gamma, \quad (13)$$

where the threshold γ can be set to assure a fixed system false-alarm rate under the Neyman-Pearson approach, or can be chosen to minimize the overall probability of error under the

Bayesian approach [18]. Since the marginal distribution under both hypotheses follows a geometric distribution with different probabilities p, the test statistic is given by:

$$\Lambda(Y_{1:L}) = \frac{(1-p_{\text{normal}})^n p_{\text{normal}}}{(1-p_{\text{abnormal}})^n p_{\text{abnormal}}}. \quad (14)$$

It is important to note that there is no objective way to choose "optimal" values of k_r and γ, as it reflects a trade-off between detection rate and false-alarm rate. The values of both parameters should be chosen by the policy makers to reflect the Quality-of-Service (QoS) that is acceptable in a particular network, and may change as a function of time and location. In our experiments, we chose the values $k_r = 5, \gamma = 1$ as these provided good detection performance with low false alarm rate.

B. *Answering the MOO question*

Once hypothesis testing is complete, we can then answer the "*me, or others?*" question for each investigated device, given its observed network delay, together with those of its co-located neighbors'. If the null hypothesis $\mathcal{H}_0$ is accepted, the device is classified as experiencing "normal" network delays as compared to its neighbors, and hence the answer to the MOO question is "*others*". If the alternative hypothesis $\mathcal{H}_1$ is accepted, then there is evidence to suggest that the device is experiencing "abnormal" network delays that are not observed by others, and hence the answer to the MOO question is "*me*".

We also note that our approach will inherently adapt to situations where most devices experience very high network delay. In such cases, the threshold expressed in Equation (9) will be elevated accordingly, such that only extreme deviations will be labeled as outliers. In this case, the algorithm will consider devices that jointly experience very high network delays as performing "normally".

V. Experimental Setup And Results

In our experimental setup, we used the following devices listed in Table 1.

Manufacturer	Model	No. of Units	Cell Provider
Samsung	GT-P3100	02	#1
Asus	Nexus 7 3G	03	#1
Samsung	SM-T325	03	#1
Samsung	SM-T325	03	#2

Table 1: The devices that were used in our experiments. For units of the same make and model, we updated all devices to their latest official firmwares, with no other after-market apps installed, except for Tattle.

For each of these devices, we collected HTTP HEAD delay measurements every 5 seconds for google.com.sg, and UNISENSE, for a window of 5 minutes, at each of the 24 positions, located on a busy outdoor train station platform, as illustrated in Figure 2.

Each set of collection lasts for two hours, and is repeated twice a day, over a stretch of 7 consecutive days, resulting in over 7,300 time-series of five minutes each, comprising in total 443,500 delay measurements.

This train platform was chosen for its persistent crowdedness, which is typical of the type of congested urban areas that are pervasive in Singapore. Our proposed approach is inherently applicable in such scenarios, where there are large crowds, and unreliable cellular network connectivity.

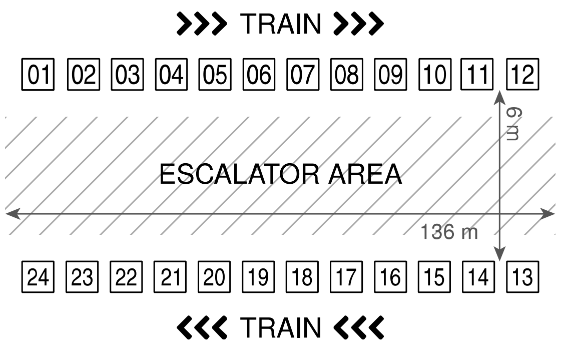

Figure 2: The train station platform, as well as the individual spots where data collection was performed, are illustrated in this figure.

A. *Temporal correlation of data*

In regression analysis of time-series data, one of the core assumptions is that the observations come from an i.i.d. process. In order to validate this assumption, we examine the temporal autocorrelation of each time-series collected by devices on Provider #1's network. Figure 3 illustrates the distribution of the autocorrelations computed for 2,680 time-series (collected for google.com.sg), over a range of lags, where the interval between each timestep is five seconds. The key result here is that even at a lag of one timestep, close to 80% of the time-series demonstrated very little autocorrelation (coefficient ≤ 0.25).

To further validate the assumption, we next apply the parameterized Ljung-Box Q-Test to the data, which tests each time-series under the strict null hypothesis that the autocorrelations under the first x lags are *jointly* zero. The results are given in Table 2. The ratio of non-rejection indicates the percentage of those 2,680 time-series tested that does not reject the null hypothesis at a significance level α of 0.05. Even at a lag of $x = 1$, there is no evidence to reject the null hypothesis, more than 78% of the time.

Null hypothesis, $\mathcal{H}_{\text{null}}$	Ratio of non-rejection
$\rho_1 = 0$	78.02%
$\rho_1 = \rho_2 = 0$	71.79%
$\rho_1 = \rho_2 = \rho_3 = 0$	72.05%
$\rho_1 = \rho_2 = \cdots = \rho_4 = 0$	73.02%
$\rho_1 = \rho_2 = \cdots = \rho_5 = 0$	74.78%
$\rho_1 = \rho_2 = \cdots = \rho_6 = 0$	75.90%

Table 2: The Ljung-Box Q-Test for temporal autocorrelation. The null hypothesis $\mathcal{H}_{\text{null}}$ is constructed such that the autocorrelation coefficients $\rho_1, \rho_2, \cdots, \rho_x$ considering the first x lags are jointly zero (that is, completely uncorrelated).

B. *Choice of probed servers*

In order to demonstrate that the observed variability in delay is mostly due to the performance of the cellular access portion of the network, and not influenced by the choice of probed servers, we provide the cumulative distribution functions of the HTTP HEAD fetch delay for both google.com.sg and UNISENSE in Figure 4. Clearly, the overall trends in performance are almost identical. This is also patently observed on Provider #2's network. Hence, without loss of generality, we focus on the delays measured for google.com.sg henceforth.

C. *Performance differences between cellular providers*

It can be seen in Figure 4 that the difference between the delay performances of Provider #1's and Provider #2's network is drastic, regardless of the choice of probed servers. The median

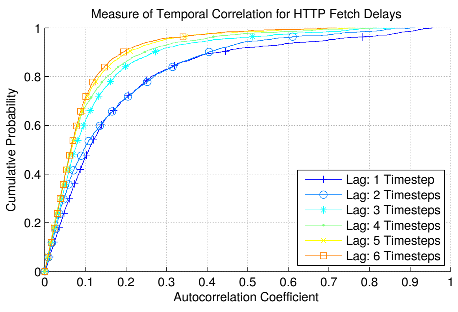

Figure 3: The distribution of autocorrelation coefficients for 2,680 time-series of network delay. Each time-step is 5 seconds in length.

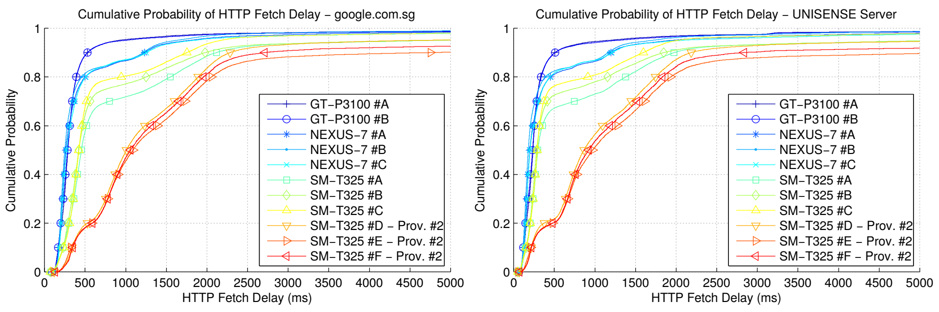

*Figure 4: (**a, left**) A comprehensive characterization of the HTTP fetch delays for* google.com.sg*, experienced by each device in our entire data-set of over 220,000 data points. Provider #2's network performs considerably and observably worse than Provider #1's network in largely every experiment. (**b, right**) The same plot is given, but these devices instead perform HTTP fetches on our self-managed* UNISENSE *server, located in Singapore.*

fetch delay for both google.com.sg and UNISENSE on Provider #2's network takes almost 300% longer than that of Provider #1's. Figure 5 shows an example of this difference in the temporal domain. Delays on Provider #1's network were subjected to much less variability compared to Provider #2's network, and this is true in most of the time-series that we collected.

D. Performance differences between device models

Figure 4 also reveals some interesting relationships between the delays observed and the make and model of the device performing the measurement. The Samsung GT-P3100, introduced in 2012, performs very stably, with around 90% of its measurements coming in below 500 ms for both servers probed. In contrast, Samsung's latest flagship tablet, the SM-T325, performs poorly, with 90% of their measurements coming in between 1500 to 2000 ms, on the same network. This however, does not invalidate our approach of taking all co-located participants' measurements into consideration, regardless of make and model, to determine if a device is doing worse than its neighbors. This exactly is in line with the basic intention of asking the MOO question. However, if there are other co-located devices in the same area with the same make and model, further extensions can be easily made to consider only same device types, though that is beyond the scope of this paper.

E. "Normal" vs. "Abnormal" performance

There are numerous periods in the course of our experiments where sudden, persistent spikes in delay are observed by many devices. These will result in windows that could stretch over several minutes where many devices observe very high delays, or complete outage of data service. This is illustrated in Figure 6, where within a window stretching over more than one minute, devices experience abrupt spikes in their fetch delays, which affected at least 6 out of the 8 devices tested. For applications that are somewhat tolerant of delay, such as web-surfing, this may not appear to be wholly debilitating. However, this could have frustrating consequences for users that are streaming videos from youtube.com, or are having live conversations on Skype. Nevertheless, in situations such as these, many co-located devices indeed suffer together (although to varying degrees), and hence each participant's device that are faced with such delays can be said to be performing "normally".

There are also significant stretches during our experiments where one or two of our devices experience severe outages that are not observed by other devices. Figure 7 illustrates this phenomenon exactly. In this scenario, the device GT-P3100 #A experiences a sudden stretch of high delays and timeouts that lasted more than 3 minutes. Besides NEXUS-7 #A, which also saw some spikes and a short outage of less than 30 seconds, the rest of the devices experienced delays that were fairly normal. In fact, the worst-ever time-series that we observed in a 60 sample window (lasting 5 minutes) had 58 fetch attempts which experienced time-outs, while its 7 other neighbors saw 0, 0, 2, 2, 0, 3 and 0 time-outs respectively. In these types of scenarios, answering the MOO question is obviously useful. When the device is deemed to be performing "abnormally", the participant can proceed to perform limited diagnostic checks, and possibly reboot his device to try and mitigate the outage.

In the real world, there could be many reasons why such outages happen. For example, the device could be physically damaged, or experiencing a rare and very unfortunate prolonged period of deep fading. Other reasons may include bugs in the firmware that manifest themselves after prolonged use (e.g. software aging), having errant malware that are hogging system resources, or simply having left a Virtual Private Network (VPN) connection switched on in the background. Identifying the root cause of such outages is beyond the scope of this paper.

F. Outlier detection performance

In the second step of our statistical framework introduced in Section IV, we take a given delay time-series measured by a device, as well as those of the device's co-located neighbors', and identify points in time where the device is doing considerably worse than its neighbors, using QR. We term those

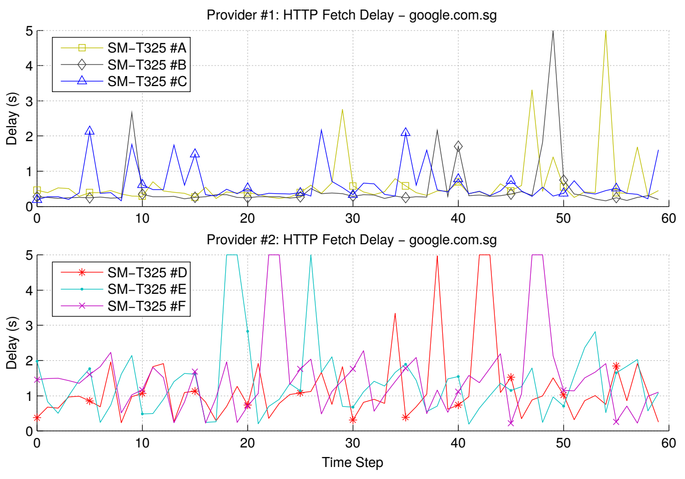

Figure 5: An illustration of two sets of representative time-series, representing the HTTP fetch delay for google.com.sg*. The devices were co-located at the same place, and are the exact same make and model, but 3 were connected to Provider #1's network, while the others were connected to Provider #2's network. The interval between each time step is 5 seconds.*

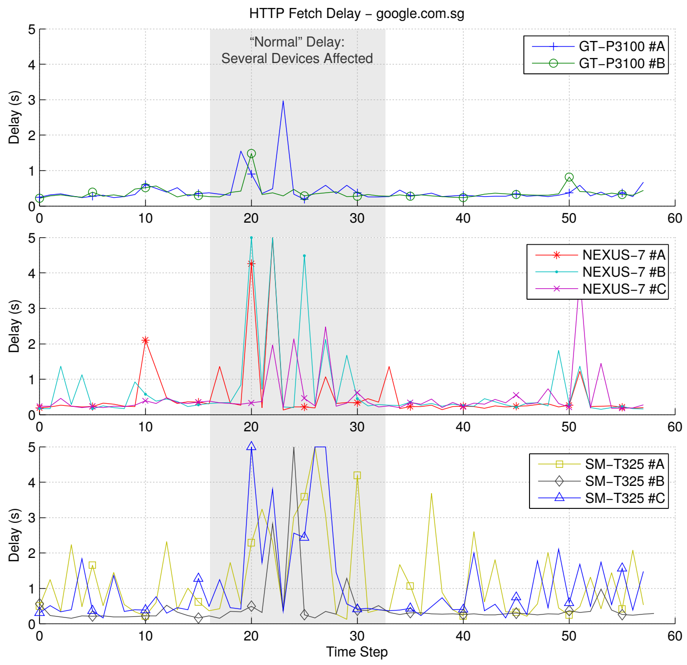

Figure 6: An illustration of three sets of representative time-series, representing the HTTP fetch delay for google.com.sg*. The devices were co-located at the same place, using the same provider, but are arranged according to their makes and models. Periods of high delay can affect many devices in the same area, as shown.*

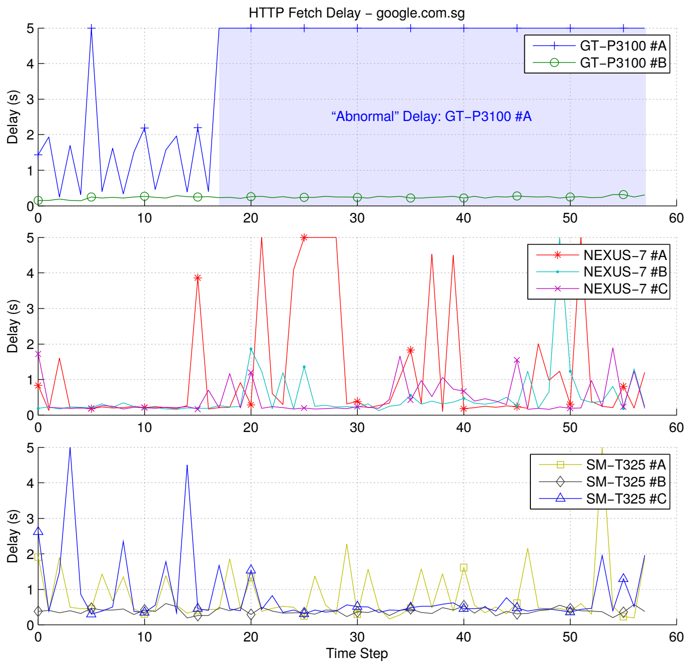

Figure 7: An illustration of three sets of representative time-series, representing the HTTP fetch delay for google.com.sg*. The devices were co-located at the same place, using the same provider, but are arranged according to their makes and models. Oftentimes, a device may experience severe outage of network service, but most other devices are unaffected.*

points as outliers. The result of this approach can be seen in Figure 8. Here, to illustrate clearly how effective our approach is in terms of correctly identifying outliers, we took a time-series of delay measurements collected on Provider #2's network, and used four other time-series collected on Provider #1's network as those of its neighbors, and performed the regression analysis and outlier detection. The outliers that were detected by the algorithm are clearly marked out in Figure 8.

The algorithm patently detected points in time where delay spikes seen by the foreign device were not observed by its neighbors. More importantly, during periods where spikes *were* observed by its artificially-introduced neighbors, the algorithm adapts, and correctly classifies those measurements as inliers. This is evident in time-steps 22, 48, and 51.

G. Effectiveness of the overall algorithm

In the previous subsections, we illustrated results for selected sets of time-series that highlight the effects discussed above. Here, we setup the following experiment to test the effectiveness of the algorithm as a whole, on how accurate it is in detecting time-series that are known to be "abnormal". In order to do this, for each of the set of 8 time-series (belonging to 8 devices) collected per position (over 24 positions), per experiment (over 2 experiments), per day (over 7 consecutive days) on Provider #1's network, we introduce a foreign time-series that were collected at the same corresponding times by a device on Provider #2's network. This resulted in 336 sets of 9 time-series (with 8 devices using Provider #1, and another SM-T325 using Provider #2). Each of the 336 foreign time-series was manually and individually inspected and labeled as "normal" or "abnormal". This is based on whether its median delay is closer to those of the SM-T325s' on Provider #1's network (labeled as "normal"), or those of SM-T325s' on Provider #2's network (labeled as "abnormal"), with reference to Figure 4.

Using a window length of 60 samples (collected in 5 minutes), we apply our 4-step algorithm on each set of 9 time-series, and vary the number of neighbors used in the regression analysis to also examine the effects of increasing the number of co-located participants on detection accuracy. The goal is then to check if the algorithm labels the foreign time-series correctly, or incorrectly (which we term as a mis-detection). Figure 9 shows the results of this experiment. Using our approach, the median mis-detection rate is around 30% when only 2 neighbors are considered in the regression analysis. As more neighbors are introduced, the detection rate improves steadily. When 6 neighbors are used, the median mis-detection rate drops to only around 10%.

We do not show the results of considering 1, 7 or 8 neighbors because these combinations only result in 336 readings, which are insufficient to obtain a smooth cumulative distribution. However, moving up to 7 neighbors, our algorithm can already

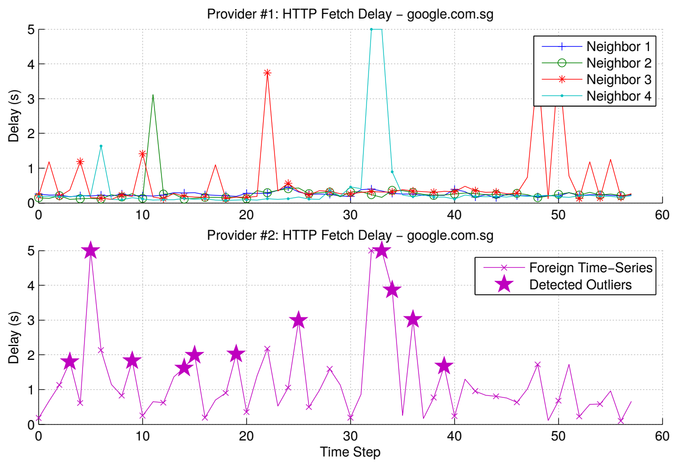

Figure 8: An illustration of two sets of time-series. Here, a time-series of HTTP fetch delays for google.com.sg*, belonging to a device connected to Provider #2's network, is mixed into that of the data set of time-series of fetch delays for devices on Provider #1's network. The goal of our algorithm is to detect points in time where the foreign time-series is deemed to be behaving "abnormally" from others. Those detected points are then marked out as shown.*

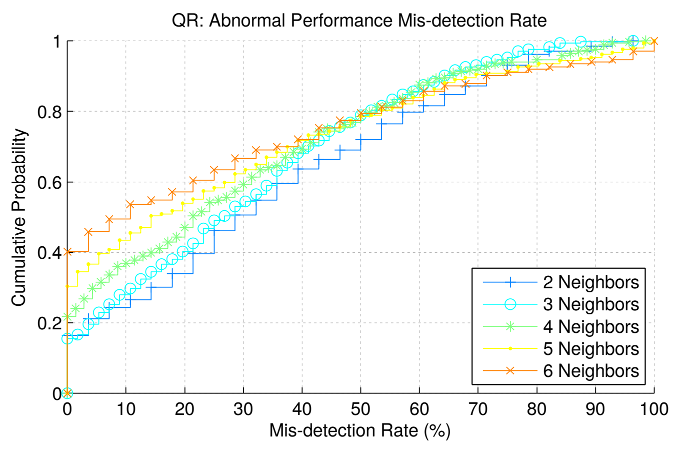

Figure 9: A complete characterization of the misdetection rates of a foreign time-series of HTTP fetch delays, belonging to Provider #2's network, when mixed into the time-series of delays experienced by devices on Provider #1's network. As the number of neighbors considered by the algorithm increases, the detection performance of our algorithm becomes better.

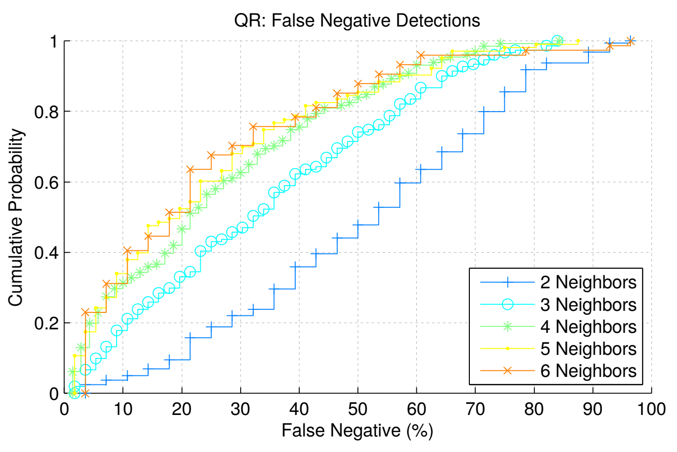

Figure 10: An illustration of the false negative rates of our algorithm. The false positive rate can be inferred by flipping the line-series about the (0,0)-(100,1) diagonal. Of all the times that a misdetection happens, having more neighbors helps to suppress the false negative rate, in favor of the false positive.

reduce the median mis-detection rate to zero. With 8 neighbors, we can in fact detect the foreign time-series with 100% accuracy, in more than 80% of the time-series.

H. False negatives vs. false positive performance

False negative mis-detection happens when a participant experiences very poor network delays compared to its neighbors, but the algorithm returns a "normal" verdict. This can be an exasperating experience because the user's frustrations are not validated. So, we argue that false positives (where the network delay seen by the participant is actually comparable to that of its neighbors', but the algorithm returns an "abnormal" verdict) are preferable in the context of the MOO question. Figure 10 illustrates comprehensively the false negative rates demonstrated by our algorithm, for the same set of experiments. Having just 2 neighbors result in a median false negative rate of over 53%. This improves steadily as more neighbors are added into the regression analysis. With 6 neighbors, the median drops to around 18%, or equivalently, an 82% median false positive rate for all those instances which the algorithm mis-detected the foreign time-series.

VI. Conclusions And Future Directions

In this paper, we first describe the "*me, or others?*" (MOO) question that should be asked by cellular service subscribers when they experience periods of poor network performance. We describe how we use *Tattle*, a comprehensive, large-scale cellular network monitoring framework, to allow participating devices to glean networking performance information from one other in real-time, by leveraging on its capability to opportunistically exchange network measurements to preserve the context of co-location and conserve device power.

We propose a robust statistical framework that builds on top of Tattle, based on Quantile Regression which cumulates into a 4-step algorithm. It first derives a regression model based on a device and its neighbors' network delay measurements, then identifies points in time where a device performs poorly compared to its neighbors. Next, it counts n, the number of these identified outliers in a small finite window, then performs hypothesis testing by deciding on which one of two estimated geometric distributions (from which "normal", and "abnormal", numbers of outliers are drawn) that n most likely belongs to. Through this, we can directly answer the MOO question.

We validate our approach based on real-world measurements of network delay, using several devices of assorted makes and models, to collect over 7,300 time-series of measurements, comprising over 443,500 samples. This includes measurements on 2 different providers' networks for comparison. We first show that at a 5s sampling interval, close to 80% of all the time-series demonstrated very little autocorrelation, so as to fulfill the necessary prerequisite of i.i.d in order to apply regression.

We then illustrate examples where "normal" and "abnormal" performances occur in real networks, and report that there occurs instances where a device can experience complete outage, while none of its neighbors are affected. We give quantitative results on how well our algorithm can detect an "abnormal" time series, with increasing effectiveness as the number of neighbors increase.

We also characterize the false negative and false positive tendencies of our algorithm, and validate that our algorithm favors false positives, which is more desirable than false negatives in the context of the MOO question.

Finally, as future work, we indeed to further expand our algorithm to quantify effects of various parameters such as k_r and γ, as well as extend the framework to include other types of network metrics, such as throughput, which are not as straightforward to measure as network delay.

Acknowledgements

This work was supported in part by CMU-SYSU CIRC, SYSU-CMU IJRI, as well as the Agency for Science, Technology and Research (A*STAR) Singapore.

References

[1] H. Liang, H. S. Kim, H.P. Tan, W.L. Yeow, "*I've heard you have problems: Cellular signal monitoring through UE participatory sensing*", IEEE GLOBECOM '14, 2014.

[2] P. Bahl, J. Padhye, L. Ravindranath, M. Singh, A. Wolman, B. Zill, "*DAIR: A Framework for Managing Enterprise Wireless Networks Using Desktop Infrastructure*", ACM HotNets '05, 2005.

[3] A. Adya, P. Bahl, R. Chandra, L. Qiu, "*Architecture and Techniques for Diagnosing Faults in IEEE 802.11 Infrastructure Networks*", ACM MobiCom '04, 2004.

[4] R. Barco, P. Lazaro, L. Diez, V. Willie, "*Continuous versus Discrete Model in Autodiagnosis Systems for Wireless Networks*", IEEE Transactions on Mobile Computing, Vol. 7, No. 06, June 2008.

[5] R. Barco, V. Wille, L. Diez, M. Toril, "*Learning of model parameters for fault diagnosis in wireless networks*", Wireless Networks, Vol. 16, No. 1, January 2010.

[6] P. Szilagyi, S. Novaczki, "*An Automatic Detection and Diagnosis Framework for Mobile Communication Systems*", IEEE Transactions on Network and Service Management, Vol. 09, No. 02, June 2012.

[7] J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, M. B. Srivastava, "*Participatory Sensing*", ACM WSW '06, 2006.

[8] "*iOS: Understanding iBeacon*", http://support.apple.com/kb/HT6048, Apple Inc., 2014.

[9] "*About Bluetooth® Low Energy Technology*", http://www.bluetooth.com/Pages/low-energy-tech-info.aspx, Bluetooth SIG, Inc., 2014

[10] "*Wi-Fi Direct*", http://www.wi-fi.org/discover-wi-fi/wi-fi-direct, Wi-Fi Alliance, 2014.

[11] H. Jiang, Z. Liu, Y. Wang, K. Lee, I. Rhee, "*Understanding bufferbloat in cellular networks*", ACM SIGCOMM CellNet '12, 2012.

[12] T. Luo, H. P. Tan, "*Profit-Maximizing Incentive for Participatory Sensing*", IEEE INFOCOM '14, 2014.

[13] S. Saroiu, A. Wolman, "*I Am a Sensor, and I Approve This Message*", ACM HotMobile '10, 2010.

[14] Trusted Platform Module Library Specification, Family "2.0", Level 00, Revision 00.99, 2013, Trusted Computing Group, 2013.

[15] X. Wang, W. Cheng, P. Mohapatra, T. Abdelzaher, "*ARTSense: Anonymous Reputation and Trust in Participatory Sensing*", IEEE INFOCOM '13, 2013.

[16] R. Koenker, K. Hallock, "*Quantile Regression: An Introduction*", Journal of Economic Perspectives, Vol. 15, No. 4, 2001.

[17] A. Nardi, M. Schemper, "*New residuals for Cox regression and their application to outlier screening*", Biometrics, Vol. 55, No. 2, Jun 1999.

[18] S. Kay, *Fundamentals of Statistical Signal Processing, Volume II: Detection Theory*, Prentice Hall, 1998.