Why is My Smartphone Slow?

On The Fly Diagnosis of Underperformance on the Mobile Internet

Chaitrali Amrutkar*, Matti Hiltunen†, Trevor Jim†, Kaustubh Joshi†,
Oliver Spatscheck†, Patrick Traynor* and Shobha Venkataraman†
*Georgia Tech Information Security Center
Georgia Institute of Technology
{chaitrali, traynor.cc}@gatech.edu
*AT&T Labs – Research
Florham Park, NJ
{hiltunen, trevor, kaustubh, spatsch, shvenk}@research.att.com

***Abstract*—The perceived end-to-end performance of the mobile Internet can be impacted by multiple factors including websites, devices, and network components. Constant changes in these factors and network complexity make identifying root causes of high latency difficult. In this paper, we propose a multidimensional diagnosis technique using passive IP flow data collected at ISPs for investigating factors that impact the performance of the mobile Internet. We implement and evaluate our technique over four days of data from a major US cellular provider's network. Our approach identifies several combinations of factors affecting performance. We investigate four combinations in-depth to confirm the latency causes chosen by our technique. Our findings include a popular gaming website showing poor performance on a specific device type for over 50% of the flows and web browser traffic on older devices accounting for 99% of poorly performing traffic. Our technique can direct operators in choosing factors having high impact on latency in the mobile Internet.**[1]

I. Introduction

Cellular networks are increasingly becoming the dominant method for people to access the services provided on the Internet. At the same time, the latency experienced on cellular networks is generally higher than that on wireline networks, which can lead to lower user satisfaction [3]–[5]. While some of these differences in performance can be explained by fundamental limitations of wireless technology, cellular network architectures, and wireless spectrum availability, these do not tell the entire story.

Even within a single cellular network uniformly using the same technology, there are often large variances in the end-to-end latency because of differences in the capabilities of user devices, the behavior of mobile applications, geographical and temporal variations in the provisioning and performance of the cellular infrastructure, and in the performance of the Internet services themselves. For example, Figure 2 shows that end-to-end RTTs measured on the data we obtained from a national US cellular provider exhibit a range that spans over 4 orders of magnitude. Consequently, when a particular user experiences poor network performance, these factors make it difficult to precisely answer the question *why is the performance poor?* Is it truly an issue of network performance, or is it because the user is using an older device, an unoptimized application, a service hosted on a slow Internet website, or is it a combination of these issues?

While offline benchmarking and troubleshooting of phones, applications and websites can help provide some clues, they are not sufficient for a number of reasons. First, performance problems can be caused by a combination of factors (e.g., an app may perform poorly on a specific phone model or some content providers may experience issues only when used by a specific OS version) and there are simply too many combinations of devices, applications, and services to analyze. Furthermore, these combinations change continuously as new elements are added to each set and it is impractical to examine every new device with every available application and application release. Second, performance impacts of these factors can change dramatically with time. This is especially the case for a cellular network, where performance can vary depending on the weather, user workload, failures, and maintenance activities.

In this paper, we propose an alternative online approach to such offline performance troubleshooting. We continuously monitor the network performance by collecting anonymized performance data on IP flows and track their end-to-end latency. Each flow is labeled with the device, OS version, application, the network elements it passes through, and the Internet service (e.g., website) it uses. Using these multi-attribute flow records and their corresponding latency, we use techniques proposed in [21] to compare the worst performing flows against flows that are deemed to have acceptable performance, and isolate the combination of factors (e.g., device, application, network element) that best differentiate the two sets of flows. In doing so, we obtain a number of signatures, each signature indicating a set of factor combinations that is likely to have poor performance. We perform this analysis every hour to track both long term as well as transient causes of poor performance.

These signatures, once produced, have a number of uses. They can be used for quickly identifying applications or Internet services that can be optimized further, or for recommending device upgrades to users complaining about poor performance. They can also be used to direct network maintenance by identifying network elements that are contributory to poor user experience over the long run. Finally, the signatures can also be correlated to network alarms to help better localize

[1]This paper has been cleared through author affiliation.

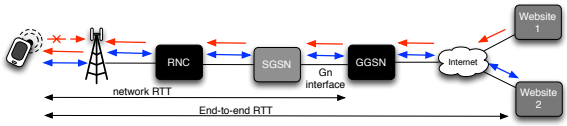

Fig. 1: Architecture of the mobile Internet. Data traffic traverses through the wireless cellular network before connecting to the Internet.

the source of transient network anomalies.

Our approach has several benefits. The first is that the approach is lightweight, zero-touch, and the results automatically reflect new devices, applications, and services without any additional effort. Second, our results achieve high coverage because we test combinations of devices, applications, and services that users actually use. Third, we are able to pinpoint causes of performance degradations that are transient in nature, such as network infrastructure that gets overloaded at the peak hour, flash crowds limited to individual applications, or anomalies caused by maintenance activities. Finally, rather than producing signatures that compare the performance of every combination of factors, we directly identify only the poorly performing ones, thus reducing information overload.

We have implemented our technique and evaluated it over IP flow data covering a major US cellular service provider's national network for a period of 4 days. In our evaluation, we have used radio access network latency as a metric for identifying poor performance. Our results include a popular gaming website showing poor performance on a specific device model for over 50% of the flows, web browsing traffic on older devices accounting for 99% of poorly performing traffic, and a single device responsible for 19% of all the poorly performing flows passing through an RNC supporting hundreds of devices. We perform in-depth investigation of four signatures that corroborates the causes of poor performance picked by our approach: in one such investigation, we find that we can discover automatically, the root cause of a service-impacting incident that took network operators two days of manual investigation to analyze; in another, we pinpoint high latency observed at a small number of devices at one specific gaming website and detect the unusual behavior of these devices. Overall, our results show that a wide variety of causes, ranging from maintenance procedures in Internet servers, to poorly optimized applications must be considered when understanding end-to-end performance of the mobile Internet and our technique can help operators in identifying the factors that have high impact on network latency.

II. Network Architecture and Data Collection

Our study uses anonymized nationwide data collected from a major US cellular operator's 3G UMTS network. Figure 1 shows the path traversed by data traffic inside such a network. Data typically flows through a number of elements, each responsible for specific functions. The RNC (Radio Network Controller) schedules and manages the cell tower radio interface. The SGSN (Serving GPRS Supporting Node) tracks the location of user equipment (UE) and wakes up sleeping UEs on incoming connections. The GGSN (Gateway GPRS Supporting Node) tunnels IP flows and provides NAT and proxy services for each UE. Additional details can be found in [34]. The element of interest to this study is the Gn interface between the SGSNs and the GGSNs. The GGSN is the first IP hop on the path between each UE and the Internet, and cellular providers typically have GGSNs located in a small number of National Data Centers (NDCs). The data we examined was collected through instrumentation deployed by the provider on the Gn interface, and includes anonymized IP flow records. The traces used were collected from March 31^{st} to April 3^{rd} 2012 and from April 29^{th}, 2010, and they did not contain any personally identifiable information.

Flow Records: A flow is defined by the 5-tuple $< srcip, dstip, srcport, dstport, protocol >$. For each flow obtained by sampling all the network flows at a rate of 3%, our traces included the obfuscated 5-tuple (to preserve IP address privacy), the start and end times for the flow, the number of bytes transmitted on it, the device type of the UE, and the RNC to which the UE was connected. Additionally, for the April 2012 data, we record an application identifier ($appid$) derived from the application headers (see [15] for details) of the smartphone app, and the content provider[2] derived from the http header or DNS. For the April 2010 data, we had access to the content provider information, but not to the appids. Instead, we recorded an application *service category* for each flow – this tells us the type of application (e.g., VoIP, browsing) but not the smartphone app itself.

Performance Statistics: To correlate the impact of end-user activity on network performance, we also collect aggregated performance statistics from the UMTS network. In this paper, we focus on the *round-trip time (RTT)*: we define the RTT to be the time between observation of the SYN, SYN-ACK, and ACK packets at the Gn interface. RTT is a good metric to detect performance issues since every element in the data path of the mobile Internet shown in Figure 1 impacts the RTT. Our dataset includes, for every hour, the average RTT observed at each RNC for each device type, appid, and content provider. We record both the network RTT (from the device to the GGSN) and the end to end RTT, as illustrated in Figure 1. For proprietary reasons, all performance values presented in this paper are normalized by an arbitrary (fixed) constant.

III. Methodology

We determine the combinations of factors contributing to high latency by comparing network flows exhibiting high latency with those showing acceptable latency using the methodology and Draco tool developed in [21]. Specifically, we use flow records to construct input "events" consisting of attributes or dimensions (used interchangeable) such as device type, application, RNC, and content provider. Then, we label each event as "good" (acceptable latency) or "bad" (high latency) using aggregate performance statistics collected across all events. Using such labelled multi-attribute events, we

[2]Content providers are Web sites or Internet services that supply different types of online information.

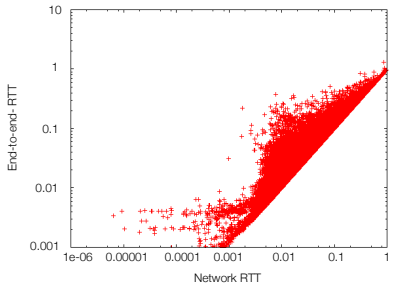

Fig. 2: End-to-end and wireless network RTT (scaled by a constant factor) for sampled flows on 04/02/2012. Our data shows that on the average, the network RTT constitutes 85% of the end-to-end RTT. Thus, network RTT dominates end-to-end RTT.

rank sets of attributes using a scoring function that quantifies the ability of the set to discriminate between good and bad events. In doing so, we obtain a number of *signatures*, each signature consisting of one or more attribute values, and each signature indicating a separate explanation for high latency flows.

A. Construction of input records

We combined flow records and performance statistics to construct the input data for our analysis. We consider each flow to be an event consisting of the RNC, device type, appid, content provider, the NDC, and anonymized device id. We assign an RTT to the event using the average RTT from the performance statistics for the RNC, device type, appid, and content provider in the time bin (one hour) when the flow occurs. Note that the performance statistics include a choice of RTT metrics we could use. The end-to-end RTT for the mobile Internet is the combination of the RTT on the wireless link and the RTT on the wired link from the GGSN to the server accessed by the corresponding connection. However, our results show that the aggregated end-to-end RTT is dominated by the wireless link RTT as shown in Figure 2, and so we focus on the network RTT in the rest of this paper.

We need to label events as good (with acceptable RTT) or bad (with high RTT), but what RTT values can be deemed acceptable is dependent on many factors, such as geographic location (crowded urban regions typically have higher RTTs), time of day/day of week (peak hours have higher RTTs). Further, acceptable RTT values continuously change over time as user behavior and network features change. To account for these factors, we calculate the Cumulative Distribution Function (CDF) for the RTT per regional market areas (defined by network provider) per hour. We then choose a high percentile value (e.g., 85^{th} percentile) from each CDF as the upper threshold of acceptable latency for that market and that hour.

B. Ranking Attribute Sets

We rank sets of attributes using a KL-divergence based scoring function proposed by [21] that quantifies the ability of the set to discriminate between good and bad events. The basic idea is to model an attribute set a as occurring in an event with a fixed, but unknown probability p^a. This attribute occurrence probability is p_f^a for bad events, and p_s^a for good events. The scoring function then uses the labelled event data to model the distributions of these probabilities as Beta distributions $\mathcal{B}$, and computes the conditional entropy of the failure distribution $\mathcal{B}(p_f^a)$ given the success distribution $\mathcal{B}(p_s^a)$. We reproduce the scoring function here as: $KL(\mathcal{B}(p_f^a)||\mathcal{B}(p_s^a)) = ln\frac{B(a,b)}{B(c,d)} - (a-c)\psi(c) - (b-d)\psi(d) + (a-c+b-d)\psi(c+d)$, where B and ψ are the Beta and digamma functions, and a/b and c/d are the numbers of bad events with/without the attribute set a, and the numbers of good events with/without a, respectively.

After picking the set with the highest score as the first signature discovered, we remove all events (both good and bad) that match this attribute combination from the data set, and repeat the process. Doing so removes the impact of the first diagnosed problem and allows one to ask what explains the remaining problems. For more details on the Draco tool that implements this technique, see [21]. The output is a set of signatures, each consisting of a conjunction of multiple attributes, and each explaining a *unique* set of high latency flows on the network. We say a flow (event) *matches* a signature if the flow has all the attribute values contained in the signature. E.g., if the signature consists only of an RNC name, all flows, good or bad, passing through that RNC match that signature. Each signature can be quantified by two metrics:

Coverage: The percentage of all bad flows that are matched by this signature. E.g., a signature with a coverage of 10% explains 10% of all bad flows in the data set.

Precision: The percentage of bad flows out of all the flows that match the signature. E.g., a signature with a precision of 50% means that half of the flows that match this signature are labelled bad.

When choosing signatures for further analysis, we prefer ones with high coverage and precision. When required, we can force the analysis to produce signatures that match a particular pattern, for example, they must contain at least one RNC or at least one content provider. Finally, the fact that an attribute appears in a lot of bad events may be due to some other attribute in these events. E.g., a good device that happens to connect to the network only through a poorly performing RNC could falsely implicate that device. In extreme cases such as these, the signatures produced by the domain agnostic analysis technique need to be further validated to eliminate unnecessary attributes. We illustrate the validation techniques used in our case studies in the next section.

IV. Case Studies

We present the results of evaluating the methodology on our traces. Our four case studies show how we identified the true root cause of high latencies observed in different contexts. The case studies show a range of performance issues ranging from transient, lasting only a few minutes, to chronics, persisting for weeks or months; they may involve extremely specific combinations of multiple attributes (e.g., a specific type of device accessing a specific content provider); they may involve very small number of devices of a certain device model that behave in a specific unusual fashion. Together,

Dimension	S1	S2	Precsn Range
Our approach	6	18	10.1 - 62.5%
Device type	3	8	5.2-29.6%
Content provider	1	7	9.2-22.9%
RNC	0	10	10-30.4%

TABLE I: Comparison of results generated using our technique and those using single dimension analysis. Precsn Range shows the range of the precision of signatures found by each technique. Column S1 corresponds to the number of signatures (out of the top 20) that show at least 1% coverage and 20% precision. Column S2 corresponds to the number of signatures (out of the top 20) that show at least 0.5% coverage and 15% precision.

our results show that our approach accurately isolates the appropriate responsible attributes, allowing network operators to distinguish network problems from problems that stem from device features, app implementation, or end-user behavior.

In each case study, we use different sets of attributes as input to the analysis to detect diverse problems in the same four days of data. Although operators cannot know ahead of time what problems might occur, our case studies demonstrate that different kinds of problems are exposed by examining different sets of attributes.

Comparing with Single-Dimensional Analysis: Before our main results, we briefly demonstrate the need for analyzing multiple attributes in combinations. We do this by comparing the quality of our signatures with the signatures generated by analyzing only one dimension – RNC, content provider, or app, at a time. We use experimental settings that match most of our case studies: one day of data with the 85^{th} percentile as our threshold for marking the high-latency flows. Table I shows our results. We note first that the precision range of our signatures is substantially higher than that of the single-dimensional analysis: our precision ranges from 10.1% to 62.5%[3], while those of the single dimensional signatures is at most 30.4%. Thus, our signatures are substantially more accurate than those of single dimensional analysis.

In each case study, we describe the experimental set-up, the key signatures found, further exploratory analysis to understand the key signatures (unless already clear), and a discussion of the implications. For proprietary reasons, we do not include the name of the device model, network element, app, etc. in our results; instead, we include some relevant aspects of the signatures that we find (e.g., device speed, content provider popularity). To avoid confusion, we use a suffix for certain attribute values representing the corresponding case study. For example, RNC2-CS1 represents the network element RNC2 in case study 1, which is different than RNC2-CS2.

A. Case Study 1: Partial Service-Impacting Incident

We begin by analyzing a previously known transient network incident. As shown in Figure 3, over a period of six weeks, several spikes were observed in the traffic.

Operators required two days of manual analysis to discover that the spikes were caused by a large number of non-malicious connection reset messages directed to the subscribers of the notification service. The reset messages were

[3]A signature with 100% precision was also produced in one case study, but it was ignored because of a very low coverage of 0.1%.

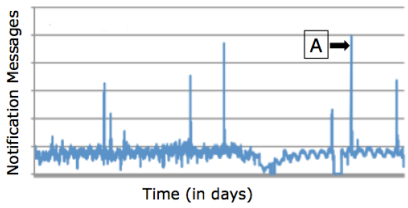

Fig. 3: Traffic from a notification service to its subscribers over a period of six weeks. At the time of the incident (marked as event A), network operators needed two days of manual analysis to identify the true root cause of the spikes in traffic. However, our multidimensional technique automatically identified the cause as increased notifications with no historical data.

generated because of routine server maintenance causing server restarts simultaneously on the majority of the notification servers. Due to a very large number of devices on the network connecting to the service, the simultaneous connection reset messages led to a burst of paging messages from the cellular provider to the devices, causing a service-impacting incident with characteristics similar to [17]. We demonstrate how our methodology can identify the cause of this incident automatically in approximately *45 minutes*.

Experiment: For this study, we use 15 minutes of anonymized network traffic collected from April 2010, covering roughly a third of the network provider's subscribers; the statistics for these 15 minutes are divided into 5 minute intervals. The main part of the service-impacting incident was observed in the last two minutes of the data set, however smaller incidents were observed for five minutes before the main incident. We focus our analysis only on the incoming traffic to end-user devices, since the failure was due to notification messages from an application server to end-user devices[4]. For this analysis, we set the RTT performance threshold at 85^{th} percentile of the provider's nation-wide RTT, and labeled each flow with acceptable or high latency depending on its network RTT value calculated per RNC, device type, and application service category (e.g., VoIP or web browsing). For the analysis, we used the RNC and application service category as the input attributes[5].

Key Signatures: Table II shows the results in each interval. As shown, the signatures contain combinations of input attributes. For brevity, we present only the top five signatures. Our results show that during the main incident (last five minute interval), the devices connecting to the notification servers had many more high latency flows than other application service categories typically responsible for high latencies (e.g., web_browsing and DNS). Smaller incidents (before the main incident) caused by the notification service were discovered in the signatures generated for the first 10 minutes. In particular, out of the 17 different service categories present in the flow data, our analysis generated multiple signatures associated with the notification service events in the last time interval. Additionally, the coverage of high latency flows in the top 5

[4]Our technique is also able to detect the cause of poor performance even when we do not restrict our analysis to just the incoming traffic, as shown in the other case studies.

[5]We had access only to the RNC and service category attribute data from April 2010, when this incident occured.

Attributes	Covrg	Precsn
1) DNS RNC1-CS1	3.3%	100%
2) web_browsing RNC1-CS1	3.2%	100%
3) **notification_service RNC1-CS1**	**3.2%**	**100%**
4) web_browsing RNC2-CS1	3.1%	100%
5) web_browsing RNC3-CS1	3.0%	100%

Attributes	Covrg	Precsn
1) **RNC4-CS1 notification_service**	**5.2%**	**100%**
2) web_browsing RNC5-CS1	5.2%	100%
3) **RNC6-CS1 notification_service**	**5.0%**	**100%**
4) RNC7-CS1 web_browsing	4.8%	100%
5) DNS RNC4-CS1	4.5%	100%

Attributes	Covrg	Precsn
1) **RNC4-CS1 notification_service**	**12.9%**	**100%**
2) **RNC2-CS1 notification_service**	**9.3%**	**100%**
3) **RNC8-CS1 notification_service**	**8.4%**	**100%**
4) RNC4-CS1 DNS	6.6%	100%
5) RNC4-CS1 web_browsing	5.4%	100%

TABLE II: Case study 1: Results from 04/29/2010 from 00:00AM to 00:15 AM from left to right. *Covrg* implies coverage and *Precsn* implies precision. Each table contains the results of 5 minutes of data. Network operators noticed a service impacting incident due to the notification_service events in the last 5 minutes and smaller outages 5 minutes before 00:10AM. Our analysis successfully captured the cause of the performance problems during this time period.

signatures increased significantly from 15.8% to 42.6% from the 1^{st} to the 3^{rd} time bin. 74% of the high-latency flows captured in the top signatures in the 3^{rd} interval corresponded to the notification service across different RNCs. As shown in Table II, all the devices in RNC2-CS1, RNC4-CS1 and RNC8-CS1 connected to the notification service had high latency (indicated by precision 100%) at the time of the incident.

Summary: Our key signatures demonstrate our methodology detected the underlying root cause of the service-impacting incident within minutes, and in an entirely automatic fashion. At the time of the incident, network operators needed two days of manual analysis to discover the problem – this was because the incident caused little change in standard metrics used for monitoring traffic (volume, flows, etc), and further, there were no tools that could automatically pinpoint appropriate combination of multiple attributes.

B. Case Study 2: Poor Performance on an End-user Device

This case demonstrates our technique's ability to pinpoint the key responsible dimensions when a performance problem manifests itself across many different dimensions. Generally, latency monitoring tools operate at the level of a network element such as an RNC [35]. Therefore, if high latency is specific to an app, an end-user device, or a content provider affects enough network transactions, it may be identified in regular network-element monitoring tools as a problem with a network element. We illustrate how our technique discovers that unusually high latencies seen at an RNC were due to a single end-user device.

Experiment: We analyzed the flow events on a single day of data and studied the attributes device type, appid, content provider, RNC, and device id. We forced the RNC attribute to appear in each signature to be able to directly compare results with those of existing monitoring systems running in the provider's network at the RNC level. We generated Cumulative Distribution Functions (CDF) of the RTT values of all the flows per market and selected the 85^{th} percentile value as the threshold to label the high latency flows. The same threshold was used for all the RNCs in the same market.

Key Signature: As expected, the analysis produced signatures containing RNCs with a larger number of high-latency flows than other RNCs. However, some signatures contained attributes such as appid, device type, and anonymized device id, which suggests that problems may be related to these attributes, rather than the RNC itself. One particular signature contained the anonymized device id of a single device as shown in Table III. This end-user device, from here on referred to simply as "deviceF", was the only one selected by our analysis technique out of a few million end-user devices in the data.

Attribute	Coverage	Precision
model1, CDN1 **0128430014607813**, RNC1-CS2	0.1%	100%

TABLE III: Case study 2: The only signature generated containing an anonymized single end-user device identifier. Further investigation revealed that 96% of the events originating from the device showed high latency on 04/02/2012.

Exploratory Analysis of the Key Signature: To investigate the root cause of this signature, we analyzed all the flows associated with deviceF and found that 96% of those flows experienced high latencies over the course of the day. We further examined whether the RNCs that served deviceF had internal failures that might have caused all the flows passing through those RNCs to have high latencies and thus affected the performance of deviceF. However, we found that the two RNCs that served the device had 15.5% and 19% of high latency flows, which is not statistically significant given we chose the 85^{th} percentile as our threshold.

The percentage of poorly performing flows generated from deviceF and passing through the two RNCs (termed RNC1-CS2 and RNC2-CS2 for ease of reference) were 94.8% and 97.5%, respectively. Furthermore, deviceF generated an unusually large fraction of the flows on each RNC; on RNC2-CS2, it was the device with the largest number of flows and on RNC1-CS2, the 3^{rd} largest. We also observed that the devices with more flows than deviceF at RNC1-CS2 mostly showed RTTs below the high latency threshold. These numbers strongly suggest that the deviceF was indeed behaving abnormally on the day of the analysis and the cause of high latency observed at the RNC was not a network element, but the device itself.

We compared our key signature with the alarms generated by a proprietary RNC monitoring tool running on the provider's network. This tool generates a variety of alarms when unusual events or poor performance is observed at an RNC. We studied the alarms generated by the monitoring tool at the RNCs visited by deviceF. We found that the alarms generated for RNC1-CS2 coincided with the times when deviceF had high latency flows. Moreover, there were no relevant alarms in the time windows when deviceF did not

have any high latency flows in RNC1-CS2. The relevant alarms coincided in the same hourly time windows of deviceF's high latency flows. Thus, this analysis provides additional evidence that the behavior and traffic volume of deviceF may have caused the monitoring tool alarms.

Summary: Network operators require techniques that can detect misbehaving devices from within the network to optimize network resources and operator efforts. Our key signature demonstrates that our approach can identify potentially harmful individual devices that generate large amounts of traffic without any prior knowledge of the existence of such devices on the network.

C. Case Study 3: Chronic High Latency on Older Devices

This case study demonstrates that our technique can be used to discover chronic performance issues. In particular, we demonstrate that our methodology is able to pick out a set of multiple attributes often correlated with high latencies, even though none of the individual attributes are, by themselves, correlated with high latencies.

Experiment: We selected 12 hours of data from each of the four days in our anonymized data set and considered three attributes for each flow: device type, content provider, and RNC. Similar to case study 2, we used the 85^{th} percentile value to label the flows as poorly performing. Our technique generated several signatures containing two dimensions: device type and content provider.

Key Signatures: Table IV shows the top five signatures generated on all four days' data. We note that two attributes corresponding to a popular social networking website (`social1`) and a content provider (`CDN1`) are present all four days. Therefore, our initial intuition was that the signatures were generated due to very large number of flows to `social1` and `CDN1`. However, the additional dimension selected in the signature, `model2` device type implied that the percentage of high-latency flows to `social1` and `CDN1` from `model2` devices is higher than from other device types. As seen in Table IV, the percentage of high-latency flows from `model2` devices to `social1` varied from 29% to 39.6%, while the percentage of high latency flows from all devices to `social1` was between 13% to 16.7%. The percentage of high-latency flows from `model2` devices to `CDN1` varied between 37.2% to 48.2%, which is significantly higher than that from all device types to `CDN1`, which varied between 14% to 22.7%.

We found that there were other device types with larger percentages of high latency flows than `model2` devices, but the number of flows was significantly lower. Thus, our analysis did not choose them in the top signatures. We note that the `model1` type devices generated more traffic to `social1` and `CDN1` than `model2` devices, however, the percentages of high latency flows from `model1` to `CDN1` and `social1` were much lower[6].

[6]Both `model1` and `model2` device are HSDPA category 8 (capable of 7.2Mbps downlink), but `model1` device is HSUPA category 6 (capable of 5.76Mbps uplink), while `model2` device is not HSUPA-enabled.

Attributes	Coverage	Precision
1) model4	19.1%	19.2%
2) model2, social1	4.5%	39.6%
3) model2, CDN1	2.7%	41.1%
4) model2, gaming1	**1.1%**	**59.7%**
5) model2, social2	1.0%	46.4%

TABLE V: Case study 4: `Gaming1` shows high latencies for 59.7% of all the flows from `model2` device types and also has the highest precision among all the signatures generated on 04/02/2012.

Exploratory Analysis of Key Signatures: There could be several potential causes for the slow connections of the `model2` devices to `social1` and `CDN1`. For example, `model2` devices may be older with slower processors or model2 devices may run background processes that slow down connections. We began our analysis by exploring the application mix in the flows to content providers `social1` and `CDN1`. While there were a total of 883 applications ids (appids) in these flows, 95% of the total flows and 99% of the high-latency flows were associated with *three* appids corresponding to web browser traffic. We note that the `model2` devices can use `CDN1` from native applications as well as through a web browser. However, the high latency flows were concentrated in SSL and non-SSL web browser traffic.

Summary: The results indicate that `model2` devices may experience unusually high latencies when users engage in web browsing, especially on `social1` and `CDN1`. The manufacturer of these devices has introduced newer devices that do not experience similar high latencies for web traffic, even though they were responsible for twice as much web traffic as that of the `model2` devices in our data. This demonstrates that older versions of devices and browser software significantly slow down web traffic. This case study shows that an operator can use our technique to determine the impact of older devices on the end-to-end RTT and filter high latency events caused due to older devices. The filtering process will allow operators to focus on performance issues that truly originate from network elements or recommend device upgrades to users complaining about poor performance.

D. Case Study 4: Chronic High Latency at a Gaming Website

Our final case study demonstrates how key signatures we discover can help direct operators towards appropriate in-depth analysis, to understand why a very specific combination of factors appears to exhibit high latency consistently.

Experiment: We use the same experiment settings as in case study 3.

Key Signatures: Table V shows the top five signatures generated on the 3^{rd} day's data. We selected the signature that showed the *highest precision* on all four days for further analysis. The signature consisted of the combination of a device type `model2` and a gaming website we call `gaming1`. Although the number of flows from `model2` devices to `gaming1` website was much smaller than flows to other content providers (see the coverage column in Table V), the precision was very high, indicating that a large fraction of those flows have high latency. Specifically, the precision ranged from 51% to 63%, which is significantly higher than

Attributes	Covrg	Precsn
1) model1 cloud1	4.2%	36.6%
2) model4	17.6%	19.3%
3) model1 email1	4.3%	25.6%
4) **model2 social1**	**2.9%**	**29%**
5) **model2 CDN1**	**1.9%**	**37.2%**

Attributes	Covrg	Precsn
1) **model2 social1**	**4.2%**	**34.9%**
2) model1 cloud1	4.2%	30.7%
3) **model2 CDN1**	**2.5%**	**38.3%**
4) search1	13.2%	18.2%
5) model2 gaming1	1.0%	61.5%

Attributes	Covrg	Precsn
1) **model2 CDN1**	**2.8%**	**48.2%**
2) model3	21.8%	18.6%
3) model4	15.5%	19.3%
4) **model2 social1**	**3.4%**	**28.1%**
5) RNC1-CS3 web_browsing	5.4%	100%

Attributes	Covrg	Precsn
1) model4	19.1%	19.2%
2) **model2 social1**	**4.5%**	**39.6%**
3) **model2 CDN1**	**2.7%**	**41.1%**
4) model2 gaming1	1.1%	59.7%
5) model2 social2	1.0%	46.4%

TABLE IV: Case study 3: Results of all four days of the data. Note that the attribute combinations `social1` on `model2` and `CDN1` on `model2` are present in the top signatures on all four days.

the average percentage of high latency flows (12.41%-14.56%) over the entire dataset with all apps and all devices and 15% to 23% over all devices connecting to `gaming1`. These statistics show that the `model2` devices' latencies were the most affected by the `gaming1` website.

Exploratory Analysis of Key Signature: To discover the root cause, we visited the `gaming1` website from the built-in web browser on a `model2` device. To our surprise, we found that the `gaming1` website needs Flash support in the web browser that is typically unavailable on `model2` devices. However, 95% of the anonymized flows in our traffic were identified as web browsing by our application identification algorithms. Of course, the traffic to the `gaming1` website might plausibly have been originated by the native game app on the device. Therefore, we installed the most popular game app offered by the `gaming1` site and analyzed the network traces. The traces showed that the device never connected to the `gaming1` website domain while the app was running, and instead the app used other servers. Together, these indicate that the high latency was observed on `model2` devices to the `gaming1` website when the devices accessed the website through the browser (which, recall, does not render the website).

We also examined the specific `model2` devices that experience high latency to `gaming1` website. We found that only a small proportion of the `model2` devices connecting to `gaming1` experience this high latency: Out of 5203 `model2` devices in our data set, only 317 had high latency flows to `gaming1`, accounting to 6% of the total devices of `model2`. Our study of the volume of requests from each device also showed that the majority of the devices made only *one* connection to `gaming1` and most of the high latency flows were due to the large number of requests from the 6% devices. If these devices were tethered to laptops that run Flash, it would be quite natural for the devices to generate a lot of high-latency flows. We also discovered that certain `model2` devices can be modified to allow them to render Flash websites. Thus, it is likely that most of the `model2` device users visited `gaming1` website on their devices and moved away from it after determining that they cannot play any games in the browser due to the lack of Flash support. However, 6% of the devices were likely tethered to Flash-enabled laptops or Flash-enabled themselves, and thus could play the website's offered game in the browser and generate the bulk of the traffic.

Summary: This case study illustrates how operators may be able to use our signatures to track down the actual root cause of poor performance. In this case, the gaming content provider `gaming1` is responsible for one of the most popular mobile apps today, and has a very large number of users. There were no known complaints of the use of `gaming1` apps on `model2` devices, so it is very unusual to see such a high precision in the signature. Our study demonstrates that while only a small number of devices are affected in this case, these devices have unusual characteristics that cause them to be persistently affected and the affected devices generate the bulk of the high-latency flows.

V. Related Work

Diagnosing problems in distributed systems: Statistical inference techniques have often been studied for problem diagnosis. Data clustering [8], [29] and decision trees [22], [37] can be used on end-to-end request traces to diagnose infrastructural problems. Graph theoretic techniques [7], [19], [28] have also been used for doing localization when the dependencies that led to the failure can be characterized in the form of a graphs. However, both types of techniques are computationally expensive, and the sheer volume of our dataset have generally made them impractical for our system. Signature-based tools [9], [12], [13] are usually scalable, but they cannot discover previously unknown problems.

Localizing end-to-end performance issues: There exist two different techniques to detect and localize end-to-end performance issues in massive networks. Active approaches [1], [2], [16], [27], [36] inject probe packets in the network, whereas passive methods simply monitor network data at the end-user (e.g, Crowdsourcing Event Monitor (CEM) [11]) or from within the network. Deploying end-user based systems at scale is difficult since users do not have a strong incentive to install the systems on their devices. Alternately, network providers can use passive monitoring systems from inside the network to detect performance issues. E.g., [35] propose a system that alarms when RTT crosses its historical norms. Our approach can use the data collected by such a system and provide chronic detection and identification of specific attributes (e.g., an app, a device or a network element) responsible for a deviation in RTT.

Identifying problems stemming from mobile applications: Identifying the right set of components responsible for a failure has become crucial as cellular networks are more connected to the Internet. Internet connected text messaging can be exploited resulting in disruption of services such as calling [14], [24], [31], [34]. Recent research also shows that vulnerable

devices inside the network may prove troublesome [10], [18], [23], [25], [26], [32], [33]. Popular mobile apps have the ability to control a large number of devices on the network and have already been responsible for outages [17], [20]. There have been efforts of running application anomaly detectors on mobile phones [30] and designing infrastructure for failure reporting on mobile networks [6]. However, such efforts are restricted due to the hardware limitations and also rely on third party processing outside the device. To our best knowledge, currently there are no efforts for in-network root cause analysis on cellular networks at scale.

VI. Discussion and Concluding Remarks

While common experience suggests that cellular networks exhibit wide performance variations, it is challenging to understand the origins of these variations. Many such as congestion, application problems, slow websites, slow devices, and network failures affect a metric like RTT differently over different timescales. Furthermore, constant changes in these causes implies that network operators need an online approach with comprehensive end-to-end coverage that helps them classify performance degradations as they happen without any preconceived notions about the dimension along which the problem must lie. We have demonstrated that our technique can troubleshoot previously known incidents automatically as well as identify new high latency issues. Our approach can be used in multiple ways: as a tool for multidimensional localization of the root causes of short lived incidents, as an exploratory tool to identify long term problems (e.g., bad apps, slow devices) plaguing a network, or as a method to filter false-positives out from real-time alarming tools by identifying the right dimension in which to look at a problem and supressing alarms related to other dimensions.

The actionability of the factors responsible for poor performance is at the operator's discretion. For example, a fallacious network element is actionable for operators, but an issue regarding a website or device is not (unless the misbehavior is serious enough that measures need to be taken to exclude the website or device from the network). Issues concerning a device, OS, or app may be actionable by the network company personnel who maintain relationships with the vendors. Our future work will focus on deploying our analysis technique as a real-time support tool for network operators.

References

[1] Gomez inc. http://www.gomez.com/.

[2] Keynote systems. http://www.keynote.com/.

[3] Slow or no internet with a fix... but why? http://support.t-mobile.com/message/9728, 2011.

[4] Very very very slow hspa data speeds on Iphone 4S. http://forums.att.com/t5/Apple-Community-Discussion/Very-very-very-slow-hspa-data-speeds-on-Iphone-4S/td-p/2952617/page/2, 2011.

[5] Extremely Slow 3G Data Connection. http://community.sprint.com/baw/thread/89634, 2012.

[6] S. Agarwal, R. Mahajan, A. Zheng, and P. Nobel. There's an app for that, but it doesn't work: Diagnosing mobile applications in the wild. In *Proceedings of ACM HotNets*, 2010.

[7] 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 *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communications*, 2007.

[8] P. Barham, A. Donnelly, R. Isaacs, and R. Mortier. Using magpie for request extraction and workload modeling. In *Proceedings of the Symposium on Operating Systems Design & Implementation - Volume 6*, 2004.

[9] P. Bodik, M. Goldszmidt, A. Fox, D. B. Woodard, and H. Andersen. Fingerprinting the datacenter: automated classification of performance crises. In *Proceedings of the European conference on Computer systems(EuroSys)*, 2010.

[10] A. Bose and K. G. Shin. On mobile viruses exploiting messaging and bluetooth services. *Proceedings of the IEEE International Conference on Security and Privacy in Communication Networks (SecureComm)*, 2006.

[11] D. R. Choffnes, F. E. Bustamante, and Z. Ge. Crowdsourcing service-level network event monitoring. In *Proceedings of the ACM SIGCOMM conference*, 2010.

[12] I. Cohen, S. Zhang, M. Goldszmidt, J. Symons, T. Kelly, and A. Fox. Capturing, indexing, clustering, and retrieving system history. In *Proceedings of the Symposium on Operating systems principles (SOSP)*, 2005.

[13] S. Duan and S. Babu. Guided problem diagnosis through active learning. In *Proceedings of International Conference on Autonomic Computing (ICAC)*, 2008.

[14] W. Enck, P. Traynor, P. McDaniel, and T. La Porta. Exploiting open functionality in SMS-capable cellular networks. In *Proceedings of the conference on Computer and Communications Security (CCS)*, 2005.

[15] J. Erman, A. Gerber, M. T. Hajiaghayi, D. Pei, and O. Spatscheck. Network-aware forward caching. In *Proceedings of the World Wide Web conference (WWW)*, 2009.

[16] N. Feamster, D. G. Andersen, H. Balakrishnan, and M. F. Kaashoek. Measuring the effects of internet path faults on reactive routing. In *Proceedings of the conference on Measurement and modeling of computer systems*, SIGMETRICS, 2003.

[17] C. Gabriel. DoCoMo demands Google's help with signaling storm. http://www.rethink-wireless.com/2012/01/30/docomo-demands-googles-signalling-storm.htm, 2012.

[18] C. Guo, H. Wang, and W. Zhu. Smartphone attacks and defenses. In *ACM SIGCOMM HotNets*, 2004.

[19] S. Kandula, R. Mahajan, P. Verkaik, S. Agarwal, J. Padhye, and P. Bahl. Detailed diagnosis in enterprise networks. *SIGCOMM Comput. Commun. Rev.*, 39(4), 2009.

[20] R. Karpinski. The LTE signaling challenge. http://connectedplanetonline.com/mss/4g-world/the-lte-signaling-challenge-0919/, 2011.

[21] S. Kavulya, S. Daniels, K. Joshi, M. Hiltunen, R. Gandhi, and P. Narasimhan. Draco: Statistical diagnosis of chronic problems in large distributed systems. In *Dependable Systems and Networks (DSN)*, 2012.

[22] E. Kiciman and A. Fox. Detecting application-level failures in component-based internet services. *IEEE Transactions on Neural Networks*, 16, 2005.

[23] Kingpin and Mudge. Security analysis of the Palm operating system and its weaknesses against malicious code threats. In *Proceedings of the USENIX Security Symposium*, 2001.

[24] P. P. C. Lee, T. Bu, and T. Y. C. Woo. On the detection of signaling dos attacks on 3G wireless networks. In *IEEE International Conference on Computer Communications (INFOCOM)*, 2007.

[25] C. Mulliner, N. Golde, and J.-P. Seifert. SMS of death: from analyzing to attacking mobile phones on a large scale. In *Proceedings of the USENIX Security Symposium*, 2011.

[26] C. Mulliner and G. Vigna. Vulnerability analysis of MMS user agents. In *Annual Computer Security Applications Conference (ACSAC)*, 2006.

[27] V. Paxson. End-to-end routing behavior in the internet. *SIGCOMM Comput. Commun. Rev.*, 36(5), Oct. 2006.

[28] I. Rish, M. Brodie, N. Odintsova, S. Ma, and G. Grabarnik. Real-time problem determination in distributed systems using active probing. In *NOMS*, 2004.

[29] R. R. Sambasivan, A. X. Zheng, M. De Rosa, E. Krevat, S. Whitman, M. Stroucken, W. Wang, L. Xu, and G. R. Ganger. Diagnosing performance changes by comparing request flows. In *Proceedings of the USENIX conference on Networked systems design and implementation (NSDI)*, 2011.

[30] A.-D. Schmidt, F. Peters, F. Lamour, C. Scheel, S. A. Çamtepe, and S. Albayrak. Monitoring smartphones for anomaly detection. *Journal on Mobile Networks and Applications*, 14(1):92–106, 2009.

[31] J. Serror, H. Zang, and J. C. Bolot. Impact of paging channel overloads or attacks on a cellular network. In *Proceedings of the workshop on Wireless security*, 2006.

[32] P. Traynor, C. Amrutkar, V. Rao, T. Jaeger, P. McDaniel, and T. F. L. Porta. From mobile phones to responsible devices. *Journal on Security and Communication Networks (SCN)*, 4(6), 2011.

[33] P. Traynor, M. Lin, M. Ongtang, V. Rao, T. Jaeger, P. McDaniel, and T. La Porta. On cellular botnets: measuring the impact of malicious devices on a cellular network core. In *Proceedings of the 16th ACM conference on Computer and Communications Security (CCS)*, 2009.

[34] P. Traynor, P. McDaniel, and T. La Porta. On attack causality in Internet-connected cellular networks. In *Proceedings of the USENIX Security Symposium*, 2007.

[35] H. Yan, A. Flavel, G. Zihui, A. Gerber, D. Massey, C. Papadopoulos, H. Shah, and J. Yates. Argus: End-to-End Service Anomaly Detection and Localization From an ISP's Point of View. In *IEEE International Conference on Computer Communications*, 2012.

[36] M. Zhang, C. Zhang, V. Pai, L. Peterson, and R. Wang. Planetseer: internet path failure monitoring and characterization in wide-area services. In *Proceedings of the Symposium on Operating Systems Design & Implementation - Volume 6*, 2004.

[37] A. X. Zheng, J. Lloyd, and E. Brewer. Failure diagnosis using decision trees. In *Proceedings of the Conference on Autonomic Computing (ICAC)*, 2004.