Argus: End-to-End Service Anomaly Detection and Localization From an ISP's Point of View

He Yan[1] Ashley Flavel[1] Zihui Ge[1] Alexandre Gerber[1]
Dan Massey[2] Christos Papadopoulos[2] Hiren Shah[1] Jennifer Yates[1]

[1]AT&T Labs - Research [2]Colorado State University
{yanhe,af360w,gezihui,gerber,hiren}@research.att.com
{massey,christos}@cs.colostate.edu

Abstract—**Recent trends in the networked services industry (e.g., CDN, VPN, VoIP, IPTV) see Internet Service Providers (ISPs) leveraging their existing network connectivity to provide an end-to-end solution. Consequently, new opportunities are available to monitor and improve the end-to-end service quality by leveraging the information from inside the network. We propose a new approach to detect and localize end-to-end service quality issues in such ISP-managed networked services by utilizing traffic data passively monitored at the ISP side, the ISP network topology, routing tables and geographic information. This paper presents the design of a generic service quality monitoring system "Argus". Argus has been successfully deployed in a tier-1 ISP to monitor millions of users of its CDN service and assist operators to detect and localize end-to-end service quality issues.This operational experience demonstrates that Argus is effective in accurate, quick detection and localization of important service quality issues.**

I. Introduction

The Internet has become the mainstay of many networked services (e.g., content distribution network (CDN), VoIP, VPN, IPTV). End-to-end service quality is the most important metric in evaluating these networked services and largely decides the reputation and revenue for the service providers. Existing end-to-end service quality management system can be largely divided into two branches: active probing and passive monitoring. Active probing based systems (e.g., Keynote [2], Gomez [1]) that periodically probe the service from agents at different network locations to detect end-to-end performance issues have several limitations. First, without active probes from a vast number of network locations throughout the Internet, the monitoring coverage is limited and some end-to-end service quality issues may not be detected. Secondly, probe packets also place additional overhead on the network and may be treated differently than normal packets.

In passive monitoring based systems (e.g., [5]), first each end-user detects the end-to-end service quality issues individually based on performance metrics extracted from passively monitored traffic and service quality issues detected by individual end-users are correlated spatially and temporally to determine the scope of the problem. Although it overcomes the limitations in active probing based systems, passive monitoring at end-user side has it own limitations. First these systems require end-users to install monitoring software, which may cause a deployment issue as there is no incentive for end-users to help service providers manage their services. Moreover, the effectiveness of these systems is limited by the sparsity of passive end-to-end performance measurements for individual end-users, which further depends how frequently they access the services. For example, if an end-user only accesses the service a few times in a day, systems based on passive monitoring at end-user side may not have sufficient samples to detect service events.

Although networked services are becoming a part of daily life, existing approaches are still quite limited in monitoring end-to-end service quality. Recent trends in the networked services industry see Internet Service Providers (ISPs) leveraging their existing network connectivity to provide end-to-end service. Consequently, new opportunities are available to monitor and improve end-to-end service quality by leveraging the information from inside the network. We argue that the most effective way to manage end-to-end service quality in these ISP-managed services is to passively monitor the traffic to/from end-users from the ISP's point of view.

In this paper, we design a system called "Argus" to detect and localize end-to-end service anomaly events in ISP-managed networked services in a proactive manner. In contrast to existing systems based on active probing or passive monitoring at end-user side, our approach monitors end-to-end service quality from an ISPs point of view in a centralized manner utilizing traffic data passively monitored at the ISP side [7], the network topology, routing information and geographic information. Although our approach is directly applied to ISPs, it can be extended to other general service providers (e.g. Google, Akamai) with proper network topology and routing information.

The organization of the rest of the paper is as follows. Section 2 presents the system design and detailed description of Argus. We describe how to apply Argus in an ISP-managed CDN service in Section 3. Section 4 evaluates the accuracy of service events detected in the CDN service by Argus using a list of labeled events from the CDN service team and Keynote [2] agents. In Section 5, we present the overall results for all detected service events and two representative case examples from our operational experience with Argus in the CDN service. Finally we present the related work in Section 6 and conclude the paper in Section 7.

II. The Design of "Argus"

In this section, we describe the design of "Argus", a generic detection and localization system for end-to-end service anomaly events. "Argus" turns the end-to-end performance measurements for individual end-users into actionable

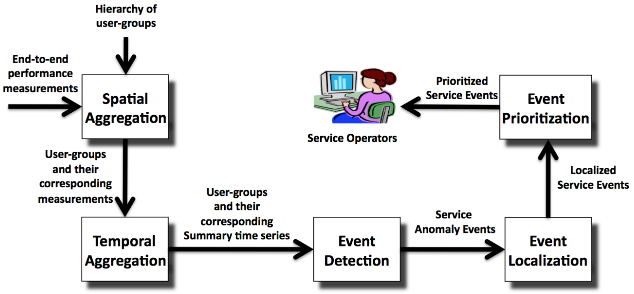


Fig. 1. The architecture of Argus

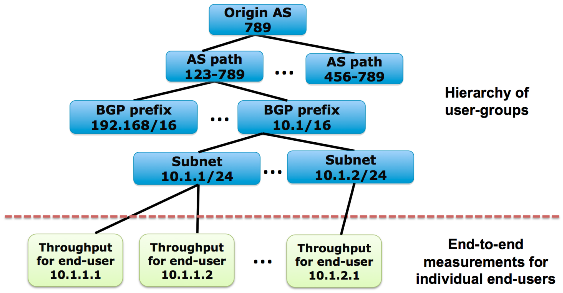


Fig. 2. An example of spatial aggregation in Argus

service anomaly events in real-time. Specifically we adopt a five-stage approach for "Argus" as shown in Figure 1.

A. Spatial Aggreagtion

In order to avoid keeping track of the end-to-end service quality associated with millions of individual end-users and address the sparsity issue in the end-to-end performance measurements for individual end-users, "Argus" first spatially aggregates end-to-end performance measurements associated individual end-users into user-groups. Each user-group is set of end-users that share some common attributes. As shown in Figure 2, all end-users from a "BGP prefix" can form a "BGP prefix" user-group and the users from the same origin AS can form a AS user-group. The attributes on individual end-users can be obtained from different data sources such as network topology, routing information and geographic locations. In practice, how to aggregate the end-to-end performance measurements spatially reflects where operators want to detect and localize service events.

B. Temporal Aggregation

After the step of spatial aggregation, each user-group has a set of end-to-end performance measurements from individual end-users associated with it. The next important question to answer is that how to detect service anomaly events for each user-group. The challenge here is that the end-to-end performance measurements belonging to each user-group could be quite noisy as they are collected from different end-users. Our solution is to focus on the summary statistics (e.g., 50th percentile, 95th percentile, min, max) of the distribution instead of based on individual end-to-end performance measurements. Obviously, we lose some details regarding individual end-users by focusing on the summary statistics. But it is acceptable as the goal is to detect service events that impact the user-groups. The service events detected on user-groups are more likely to be actionable from service provider's perspective compared to the service events detected on individual end-users.

In this step, for each user-group (obtained in the pervious step), we temporally aggregate its end-to-end performance measurements into time-bins. Once time-bins are formed, a summary statistic is selected from all the end-to-end performance measurements in each time-bin to form a **summary time series**. Several summary statistics can be used here – minimum, maximum, average, median or other percentile values. Different statistics may provide an advantage for tracking certain type of issues. For example, if the end-to-end performance measurement is round trip time (RTT), the minimum may well capture baseline RTT due to network propagation delay while being oblivious to varying queuing delay that may be due to network congestion while average can well capture the service event due to network congestion. Since our main goal is to detect service events that impact a relatively large collection of users in each user-group, "Argus" uses the median as the summary statistic for each time-bin by default. We find the median quite effective in tracking service side or network side issues while being robust to variability in performances of individual end-users due to their local processing or local queuing delays.

C. Event Detection

Once we transform the end-to-end performance measurements into **summary time series** of each user-group, we can apply time series analysis techniques to extract service anomaly events from them. There are a wide range of time series anomaly detection algorithm in the literature, ranging from Box-Jenkins linear time-series forecasting techniques, to frequency domain Fourier analysis or Wavelet analysis based approaches, to structural analysis such as principal component analysis. Due to the scale of our application, it is desirable to have online anomaly detection with minimal runtime complexity and memory requirement. We base our approach on the classic additive Holt-Winters (HW) algorithm [3], a widely used one-pass online time series forecasting method. One of the key strengths of HW is that it involves very light-weight computation and has very few states to maintain.

At a high level, HW runs three exponential smoothing processes on three components of the **summary time series**: the baseline, the linear trend, and the seasonal effect to dynamically formulate forecast values based on historical values. Specifically, the forecast value $\hat{y}_t$ at time t is formulated as follows:

$$\hat{y}_t = a_{t-1} + b_{t-1} + c_{t-n}$$

where a_t is the baseline at time $t-1$, b_t is the linear trend at time $t-1$ and c_{t-n} is the seasonal effect at time $t-n$. Note n is the number of cycles in one season (e.g., 24 cycles in one day to model hourly seasonality) and $t-n$ means the same cycle in the previous season (e.g., the seasonal effect for 1pm in the previous day is needed to formulate the forecast value for 1pm today). Given a new value y_t, a_t, b_t and c_t are updated exponentially with parametersα, β, γ respectively.

In order to determine if a new value y_t in the summary time series is abnormal or not, we compare the current residual error (e.g., absolute difference between the actual value y_t and the forecast value $\hat{y}_t$) with the historical residual errors from the same cycle in previous seasons. In this way, seasonal variability in residual errors can be captured. The update

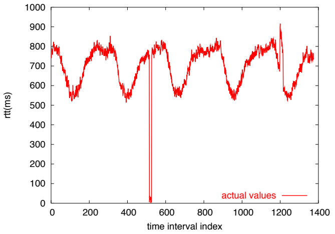


Fig. 3. Anomalies in a summary time series that consists of the average RTTs, one for each 5 minutes

formula for d_t is similar to that of c_t and uses the same parameter γ:

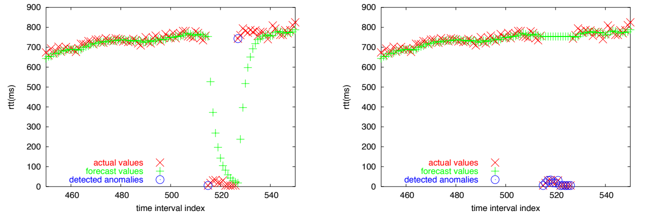
$$d_t = \gamma \times |y_t - \hat{y}_t| + (1 - \gamma) \times d_{t-n}$$

Specifically, for each y_t, we calculate its descretized abnormal level A as follows: $\mathcal{A} = 0, 1, 2, 3, 4, 5$ when $|y_t - \hat{y}_t|$ is in $[0, 0.5 \times d_{t-n})$, $[0.5 \times d_{t-n}, 1 \times d_{t-n})$, $[1 \times d_{t-n}, 1.5 \times d_{t-n})$, $[1.5 \times d_{t-n}, 2 \times d_{t-n})$, $[2 \times d_{t-n}, 2.5 \times d_{t-n})$ and $[2.5 \times d_{t-n}, \infty)$ respectively. Although it is configurable, we typically consider $\mathcal{A}$ of 4 or above as anomalous as suggested in [4]. This is a relatively aggressive setting (i.e., more anomalies). However, it is an appropriate setting as our event localization and prioritization (next two stages) is robust to false positives. We further combine consecutive anomalous values in the summary time series into a single anomaly event and keep track of all on-going anomaly events, with the begin time of the event being the begin time of the first anomalous value.

Although the classic HW has been proven effective in many different application settings, when dealing with the real service performance measurements, we have uncovered several serious deficiencies with HW and proposed corresponding enhancements.

1) Robust Forecast against Dirty Data and Service Disruptions: "Dirty data" is unfortunately unavoidable in real service performance measurements – problems in various software/hardware components of data collectors and ingestion servers can occur, rendering the performance data nonsensical. Furthermore, there are many situations in which network problems can cause service disruptions, driving the performance data out of the norm. For example, in Figure 3, the huge dip (the average RTT is decreased significantly due to missing measurements) is caused by a bug in the data collector and the spike is caused by a link congestion. It is highly desirable that forecast and anomaly detection can be robust against the "dirty data" and service disruptions in our context. In our approach, a_t, b_t and c_t are not updated if the abnormal level A of the current value y_t is above a configurable threshold A_u, which depends on the nature of the summary time series. For a stable time series, A_u should be configured tightly so that most anomalies are excluded in formulating the forecast values. For a noisy time series, less tight A_u is desirable to avoid excessive number of anomalies.

As a head-to-head comparison, we take a close look at the dip in Figure 3 and plot the different behaviors for classic HW and our approach in Figure 4. One deficiency in classic HW is that the forecast values can be easily contaminated by nonsensical performance data as shown here. On the contrast, our approach has a more stable forecast line and thus more robust in detecting anomalies.



(a) Classic HW

(b) Our Approach

Fig. 4. Forecast values and detected anomalies for classic HW and our approach using the same set of parameters. A_u is set to 4 in our approach.

2) Fast Adapt to Permanent Service Performance Changes: Network and service upgrades can often introduce a permanent level-shift on the end-to-end service performance. Once "Argus" is confident that a permanent change has taken place, it is ideal to "forget" about the extended long historical data and adapt the model to capture the recent performance only. To achieve so, a shadow set of baseline a_t^s, linear trend b_t^s, seasonal effect c_t^s and residual error d_t^s is updated in parallel using its own α_s, β_s and γ_s. Compared with the working set of baseline a_t, linear trend b_t, seasonal effect c_t and residual error d_t, the shadow set is updated differently in two aspects. First, the shadow set gives more weights to the recent observations via using relative large α_s, β_s and γ_s while the working set gives more weight to the history compared to the recent observations by using relative small α, β and γ. Secondly, anomalies are used to update the shadow set while they are ignored when updating the working set as we mentioned in II-C1. In this way, the shadow set can quickly adapt to the permanent level-shift.

A moving window of size L is used to keep track the recent values. Once the percent of abnormal values in the moving window exceeds a threshold P (suggesting that a permanent level-shift has occured), the working set is replaced with the shadow set as the shadow set should have adapted to the permanent level-shift. The configuration of L and P determines the trade-off between the timeliness in adapting to permanent changes and the risk of wrongly adapting to temporary abnormal changes.

3) Leverage Temporal Continuity in Residual Errors: In the classic HW, the exponential smoothing is applied on the residual errors d_t on a per cycle basis. For example, if there are 24 hourly cycles in one season, the exponential smoothing only applies to the residual errors of the same hour in different days. In order to leverage temporal continuity in residual errors perform in neighboring cycles (e.g., the neighboring hours in a day), weighted moving average is used for smoothing the residual errors d_t among neighboring cycles. Specifically, given a new value y_t, the residual errors for current cycle and its neighboring cycles $t-w, \cdots, t, \cdots, t+w$ are updated as follows:

$$d_{t+i} = \gamma_w \times (1 - \frac{|i|}{w+1}) \times |y_t - \hat{y}_t| + (1 - \gamma_w \times (1 - \frac{|i|}{w+1})) \times d_{t+i-n}$$

where w is the smoothing window size, i is offset compared with the current cycle ranging from $-w$ to w and γ_w is derived from γ given the window size w. Similar to the working set, the residual errors in the shadow set are also smoothed among neighboring cycles in this way.

Leveraging temporal continuity is critical in particular when the the number of cycles in one season is large (e.g., 288 5-min intervals in a day). Smoothing the residual errors over neighboring cycles restores the continuity and makes anomaly detection more robust.

4) Fine-Grained Detection with Coarse-Grained Model: In the classic HW, the detection interval has to be aligned with the cycles in a season. For example, if the classic HW models a time series with 24 hourly cycles in one season, the anomaly detection interval has to be hourly. In order to trigger anomaly detection more frequently (e.g., once per 5 minutes), the classic HW has to model the time series with more cycles (e.g., 288 5-minutes cycles in one season). The memory consumption in HW is proportional to the number of cycles as one c_t and one d_t need to be maintained per cycle. Thus increase from 24 hourly cycles to 288 5-minutes cycles in one reason can considerably increase memory consumption by 12 times. Considering the millions of time-series need to be monitored in a service, fine-grained model might not be affordable.

In our approach, we enable the fine-grained detection with a coarse-grained model by linear interpolation of the parameters from two consecutive cycles in the model. If the anomaly detection interval is T (e.g., 20 minutes), one hourly bin (e.g., 11:20 - 12:20) may span over two consecutive hourly cycles (e.g., 11:00 -12:00 and 12:00 - 13:00) in the model. The c'_t for the hourly bin (11:20 - 12:20) are derived by the sum of two third of c_t for cycle (11:00 -12:00) and one third of c_t for cycle (12:00 -13:00). d'_t is calculated similarly.

D. Event Localization

As each end-to-end performance measurement contributes to the **summary time series** of multiple user-groups (as shown in Figure 2), a single underlying network failure such as a link failure may manifest itself at multiple user-groups in the spatial aggregation . For example, if an underlying network event has caused an increase of RTT for most of end-users associated with the same "BGP prefix A", "Argus" by design should detect the RTT service anomaly event for the user-group "BGP prefix A". Due to the nature of BGP routing, these end-users should share the same origin AS and AS path, and if these end-users from "BGP prefix A" dominate other end-users associated the same origin AS or AS path, "Argus" would also detect RTT service anomaly events for the corresponding user-groups of origin AS and the AS path. In this case, it is desirable for "Argus" to localize the issue to the "BGP prefix A" and report a single service anomaly event. For another example, if a router failure has caused a throughput service anomaly event for all the end-users that are reached by the service provider via "AS path A", all the children user-groups of "AS path A" at the lower level in the hierarchy (as shown in Figure 2) such as the BGP prefix user-groups would experience the throughput service anomaly event as well. In this case, it is desirable for "Argus" to localize the anomaly to the AS path. Due to space limitations, we present the formulation of event localization problem, its complexity analysis and a heuristic to solve the problem in our technique report [9] .

E. Event Prioritization

After event localization stage, "Argus" employes a ranking function to prioritize the localized service anomaly events. Since each anomaly event may contain multiple anomalous time-bins, we first estimate the severity score of each time-bin and then use the aggregate score of all time-bins as the severity score of the service event. The ranking function used to estimate the severity score of each anomalous time-bin incorporates two factors – the significance of the relative size of the anomaly and the breadth of its impact scope. The significance of the anomaly can be measured by the deviation score $|d|$ from the EHW algorithm. The impact scope can be measured by the the number of distinct end-users observed in the time-bin, which we denote as c. We choose distinct end-users since it is robust against anomalies dominated by a few outlier end-users.

Specifically, for anomaly event e, its baseline ranking score r_e is defined as:

$$r_e = \sum_{b \in \text{ bins of } e} A_b \times C_b$$

where A_b and C_b is the abnormal level and the number of distinct end-user for bin b. In this way, long lasting events are likely given higher priority than short events.

III. Operational Results

In this section, we summarize the results of running the "Argus" system monitoring a CDN hosted in a tier-1 ISP. This ISP is referred to as "local ISP" in the remainder of this section.

A. Overall Results

To convey a basic understanding of how "Argus" works, we focus on the anomaly events detected by "Argus" from 20th July 2010 to 20th August 2010. Note that these anomaly events were detected by running "Argus" in fixed-length bin mode with bin size as 3,600 seconds (1 hour) based on the RTT measurements passively collected at the North-East CDN node.

During this one-month period, "Argus" detected 2,909 anomaly events across all user-groups in the hierarchy (Figure 2). Table I shows the anomaly event distribution across all user-groups. In general, the lower level user-groups are responsible for more anomaly events as they are much more than the higher level user-groups. In addition, for each type of user-group, only a small fraction (bad user-groups) are responsible for the anomaly events and generally there is no heavy hitter among the bad user-groups. According to Table I, the CDN nodes are extremely stable across the month, with no anomalies detected at their user-groups. Anomaly events localized to "Origin AS", "City", "BGP prefix" and "City+BGP prefix" are most likely attributable to events outside the local ISP. In contrast, the anomalous events localized to "Egress Router", "Nexthop AS", "AS path" are more likely to be attributable to events within the local ISP (although they could still have been caused within other ISPs).

We now we focus on the time durations of the 2,909 anomaly events. Our results clearly show that the majority of the anomalies are very short in duration, whilst long-lasting events are rare. Specifically, the events with duration 3,600 seconds are the most common and represent 90% of all

User-group Type	# Total User-groups	# Events	# Bad User-groups
CDN Node	1	0	0
Egress Router	185	66	36
Nexthop AS	125	54	25
Origin AS	820	172	97
AS path	1055	213	119
City	1758	593	289
BGP prefix	4646	600	425
City+BGP prefix	8784	1211	851

TABLE I
ANOMALY EVENTS BREAKDOWN BY USER-GROUP TYPE

anomaly events. In addition, this statement holds true for every user-group type in the hierarchy (due to space limitations, we don't discuss the details of per user-group type distribution here).

As the final step towards understanding the overall results, we look into the details of the top 100 events, as defined using the ranking function described in Section II-E. We spatially and temporally correlate the detected service events with the underlying network events from within the local ISP in a bid to identify the root cause by using a system called G-RCA [10] . Table II shows the root causes of the top 100 anomaly events, as identified using our correlation analysis. 13 of the service events are identified as caused by either events that happened within the local ISP (e.g., link failures, link congestion and CDN assignment change) or from events that are visible within the local ISP (such as BGP routing changes announced by other ISPs). For the rest (majority) of them, we couldn't find any evidence from inside the local ISP, which suggests that the service anomalies may be caused within other ISPs.

User-group Type	Root Cause	# events
Egress Router	CDN assignment change	1
	Link Congestion	1
	Link Failure	1
	Outside the local ISP	2
Nexthop AS	CDN assignment change	1
	BGP routing change	1
	Link Failure	1
	Outside the local ISP	1
Origin AS	CDN assignment change	1
	BGP routing change	2
	Outside the local ISP	8
AS path	BGP routing change	2
	Outside the local ISP	13
City	Outside the local ISP	21
BGP Prefix	BGP routing change	2
	Outside the local ISP	22
City+BGP prefix	Outside the local ISP	20

TABLE II
TOP 100 ANOMALY EVENTS AND THEIR ROOT CAUSES BREAKDOWN BY USER-GROUP TYPE

IV. RELATED WORK

There has been extensive prior work detecting and localizing the end-to-end performance issues. Broadly speaking, they are classified into two categories: active and passive.

Active approaches require the injection of probe packets into the network. The pioneering active approach [8] traceroutes between 37 participating sites are collected and analyzed to characterize the end-to-end performance issues. Similar to [8], [6] detects path outage among hosts using ping and localizes the observed path outage using traceroute. PlanetSeer [11] relies on active probes to diagnose the root cause of Internet path failures that are detected by passive monitoring the end-users of a CDN service deployed on PlanetLab. Commercial systems such as Keynote [2] and Gomez [1] are also available to detect issues from the end-user's perspective by active probing. All these work employ active probing while "Argus" purely depends on passive monitoring.

Passive approach purely depends the existing traffic. There are two sub-categories: passive monitoring at end-user side and passive monitoring at service provide side. A recent work [5] based on passive monitoring at end-user side proposed to push passive monitoring to the end systems themselves and implemented such a prototype system based on BitTorrent. The effectiveness in [5] actually depends the sparsity of passive measurements for individual end systems. An end system with very few measurements would be able to detect event effectively. The deployment is another issue as there is no strong incentive for end systems to corporate. Different from [5], "Argus" is based on passive monitoring at service provide side. It collects the end-to-end performance measurements corresponding to individual end-users at service provider side. Unlike [5], the benefit with our approach is that it is easy to deploy and see data from a wide range of users.

V. CONCLUSION

Detecting and localizing user-perceived service events is critical for service providers. We argue that the most effective way to do that in an ISP managed service is to passively monitor the end-to-end performance associated with end-users from inside the ISP network. In this paper, we presented the design of "Argus", a generic user-perceived service event detection and localization system. We demonstrate the effectiveness of "Argus" by applying it in a CDN service managed by a tier-1 ISP. Our experience with applying "Argus" in the CDN service has been very positive.

REFERENCES

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

[2] Keynote systems, inc. website. `http://www.keynote.com/`.

[3] P. Brockwell and R. Davis. *Time series: theory and methods*. Springer Verlag, 2009.

[4] J. Brutag. Aberrant behavior detection and control in time series for network monitoring. In *Proceedings of the 14th Systems Administration Conference (LISA 2000)*.

[5] D. Choffnes, F. Bustamante, and Z. Ge. Crowdsourcing service-level network event monitoring. *ACM SIGCOMM Computer Communication Review*, 40(4):387–398, 2010.

[6] N. Feamster, D. Andersen, H. Balakrishnan, and M. Kaashoek. Measuring the effects of internet path faults on reactive routing. In *Proceedings of the 2003 ACM SIGMETRICS*, page 137. ACM, 2003.

[7] A. Gerber, J. Pang, O. Spatscheck, and S. Venkataraman. Speed testing without speed tests: estimating achievable download speed from passive measurements. In *Proceedings of the 10th annual conference on Internet measurement*, pages 424–430. ACM, 2010.

[8] V. Paxson. End-to-end routing behavior in the Internet. *ACM SIGCOMM Computer Communication Review*, 36(5):56, 2006.

[9] H. Yan. Passively Monitoring Crowds to Detect and Isolate End-to-End Performance Issues in Wide-Area Services. Technical Report 10-102, Colorado State Univeristy, 2010.

[10] H. Yan, L. Breslau, Z. Ge, D. Massey, D. Pei, and J. Yates. G-rca: a generic root cause analysis platform for service quality management in large ip networks. In *Proceedings of the 6th International COnference*, page 5. ACM, 2010.

[11] M. Zhang, C. Zhang, V. Pai, L. Peterson, and R. Wang. PlanetSeer: Internet path failure monitoring and characterization in wide-area services. In *Proc. USENIX OSDI*, 2004.