Performance Evaluation of Web-based Cloud Services in a Browser-Scripting Approach

Chengwei Zhang, Xiaojun Hei and Wenqing Cheng
School of Electronic Information and Communications,
Huazhong University of Science and Technology
Wuhan, China, 430074
[Email: zhangcw@hust.edu.cn, heixj@hust.edu.cn, chengwq@hust.edu.cn]

Received February 25, 2016; revised April 30, 2016; accepted May 12, 2016;
published June 30, 2016

Abstract

Cloud services are often provisioned to their customers using user-friendly web browsers with flexible and rich plug-in environments. Delay is one of the fundamental performance metrics of these web-based services. Commonly-used network measurement tools usually only measure network delay and it may be difficult to infer the web-delay performance using only network layer measurement approaches. In this paper, we propose to evaluate the application layer delay in a browser-based network measurement platform using engineered scripts. We conducted a delay measurement study using instrumented scripts in the proposed browser-based measurement platform. Our investigation included a comparison study of three browser-scripting delay measurement methods, including Java applet, JSP and Flash ActionScript. We developed a browser-based delay measurement testbed over the Internet so that different delay measurement tools could be evaluated in the same real network environment including typical Internet paths and the Baidu cloud. We also decomposed the components of the end-to-end delay process of the above measurements to reveal the difference and relationship between the network-layer delay and the application-layer delay. Our measurement results characterize the stochastic properties of the application-layer delay over real Internet paths, and how these properties vary from the underlying network layer delay. This browser-scripting measurement approach can be easily deployed on different cloud service platforms to inspect their application-layer delay performance between end clients and the cloud platforms. Our measurement results may provide insights into designing new cloud services with enhanced quality-of-experience perceived by cloud users.

Keywords: Cloud service, Delay, Web scripting, Quality-of-Service, Measurement

The preliminary results in this paper have been presented in the IEEE/CIC International Conference on Communications in China, Shenzhen, China, 2-4 November, 2015 [1]. This work was supported partially by the national natural science foundation of China (No. 61370231) and the fundamental research fund for the central universities (HUST: 2014QN156). The corresponding author is Xiaojun Hei (heixj@hust.edu.cn).

1. Introduction

The emerging vast cloud services and applications have brought forth research interests in the user perceived performance quality [2]. Emerging applications and services can be scaled up flexibly through the cloud platforms, such as cloud-assisted mobile Ad-hoc networks [3], cloud gaming systems [4], mobile cloud streaming systems [5], and so on. These studies focus on improving the application performance using the cost controlling knobs provided by the cloud infrastructure. Cloud service providers are interested in understanding what the user and applications perceive of their service is actually what they are delivering. In heterogeneous networks (such as 3G/4G, Wi-Fi, etc.), the services and applications may exhibit dramatically different service performance [6][7][8]. This drives the need for conducting such application-layer measurements. It is important to enable cloud service providers to evaluate their service performance from the perspective of end-users. To overcome the different system architectures and the Internet firewalls, web-based network measurements have dominated the current application-layer performance measurements in instead of using the prohibitively expensive custom measurement utilities on end-systems [9].

To date, many measurement tools are available to end-users to measure network performance and diagnose problems [10]. Cloud services are often provisioned as web applications, which provide a big virtual web-based environment for the measurement tool [11][12]. Web browsers provide user-friendly interfaces between the service servers and the users to interact with different Internet contents [13], from traditional text-based web pages and emails to ever increasingly streaming media. Modern browsers have now been supporting flexible and rich plug-in environments, which are capable of transferring and rendering multimedia using a bunch of scripts and embedded objects [14]. These scripting and object-embedded capabilities of the dynamic web pages provide the feasibility to conduct such a browser-based measurement platform to evaluate the various network performance metrics [11–15]. In particular, Netalyzr [20] utilizes the Java applet objects which can be embedded in web pages to perform network measurements. Ookla’s Pingtest [16] and Speedtest [17] take advantages of both Java applet and Flash to evaluate the network performance on delay and bandwidth, whereas 17CE [18] and InternetFrog [19] provide the dynamic web pages, such as JavaScript, JSP and ASP, to measure the path properties between service providers and end-users. Given this increasing trend of application-layer measurements, a few studies have been conducted to evaluate the performance such as the accuracy and the overhead of these measurements. For example, Li et al. examined the delay accuracy in browser-based network measurements using HTTP-based Socket, TCP Socket, and WebSocket [21].

In this paper, we conducted a measurement study of evaluating the application-layer delay performance of web-based cloud services using an integrated measurement platform. In addition to the delay accuracy, we also investigated the impact of possible link packet loss, measurement overhead, the time cost of measurement duration using Java applet, Flash, and the JSP dynamic web page method to gain insights into the cloud performance. Our measurement platform includes three web-scripting measurement modules and the classic Ping module operated in real Internet environments. In order to have a deep understanding on the web-based delay measurements, we collected the measurement traces at the packet level in the real network environments. These traces reveal different features compared with the basic web-based measurement. Some preliminary results have been presented in [1] and more measurement results and data analysis are provided in this paper. We also reported the results when our testbed was deployed on the Baidu cloud to investigate the accuracy and overhead for these three different measurement approaches. Our major findings are summarized as follows.

- In an integrated web-based measurement, the application-layer delay can serve as a reference to the network-layer delay under the network conditions which has small disturbance such as the clean lab network environment. The application-layer delay is quite close to the network-layer delay with only 1% - 2% deviation with the lab settings.
- The application-layer measurement packets may be retransmitted due to the TCP retransmission in complex real network environments. These spike measurements may be significantly larger than the real network-layer delay. Based on the analysis of the traced measurement data, these spikes are closely correlated to the link packet loss. In the measurement process, we can utilize a threshold (e.g. 500*ms*) to estimate the lost packets from the overall measurement results.
- Web-based network measurement approaches utilize the HTTP and TCP protocols to transmit the measurement traffic. These measurements may exhibit different characteristics due to diversing measurement packet size, measurement pre-loading time and overhead. Flash shows the most accurate measurement results and the least overhead in the application-layer delay than Java applet and JSP, whereas it has the most pre-loading time among these measurement methods.
- We deployed the proposed integrated application-layer measurement platform on the Baidu cloud infrastructure and analyzed the difference between the network-layer and application-layer delays. We also investigated the accuracy and the overhead of the delay measurements using different popular web browsers. We found that the application-layer delay may exhibit quite different characteristics of different web browsers. From the overall experiments, Google Chrome presents the most accuracy and the least overhead than Microsoft Explorer (IE) and Mozilla FireFox.

The rest of the paper is organized as follows. We proceed to discuss the proposed integrated web-based network measurement platform and compare different application-layer delay measurements in Section 2. In Section 3, we present in details our browser-based measurement testbed on the Internet. We present an analysis of the delay measurements results using our measurement testbed on real Internet paths and the Baidu cloud in Section 4. Finally, we conclude this paper in Section 5.

2. Browser-Scripting Methods

Modern browsers have been equipped with powerful scripting and object-embedded capabilities. Flash ActionScript, Java applet and JSP are able to utilize network socket; JavaScript, HTML5 and other HTML scripts utilize XMLHttpRequest or WebSocket to measure network performance. In this section, we discuss the general browser-scripting measurement process and compare the current widely-used tools with the focus on the delay measurement. The web-based application-layer delay measurement tools are available to end-users to evaluate network performance and diagnose network problems. These web-based tools can usually measure the network round-trip time (RTT) and throughput, such as speedtest [17], netalyzr [15], fathom [9] and so on, all through the web system. As shown in **Fig. 1**, a browser-scripting delay measurement process normally consists of three phases:

initialization, measurement and termination. The details of these three phases are listed as follows:

1) Two primary steps are required to accomplish during the initialization. First, measurement objects are transmitted in HTTP packets between the measurement server and the measurement clients. After HTTP objects are successfully transferred, a client initializes a TCP connection with a three-way handshake. Denote this initialization period as t_i, which is consists of t_{http} and t_{tcp}. t_{http} is the application HTTP page pre-loading time and t_{tcp} is the socket initialization time under different application measurements.
2) In the measurement phase, a client probes the server to measure the connection characteristics on top of TCP, such as delay, bandwidth, packet loss etc., for a period of time. Browser-scripting measurements are conducted at the application layer, and this application-layer delay, denoted as t_m, is usually larger than the network-layer delay which is measured by ICMP probes in practice.
3) A measurement session may last for a fixed or variable duration depending on the convergence conditions. If necessary, the server collects the measurement results from the clients and store them for further analysis. Although this post-processing duration is not a necessary component for the measurement, it can be treated as the application-layer measurement overhead, which is defined as t_d.
4)

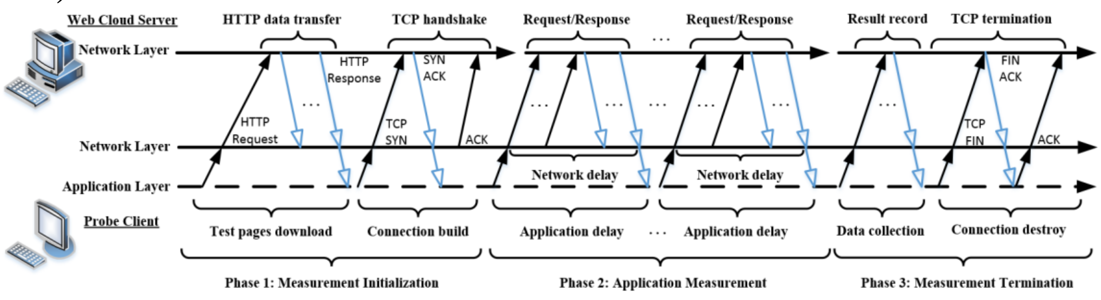

Fig. 1. A browser-scripting delay measurement scenario

Fig. 1 illustrates a browser-scripting delay measurement scenario. The difference is depicted between the network-layer delay and the application-layer in Phase 2 of the measurement process. The network-layer delay characterizes the link latency occurred on the network layers between two end hosts, while the application-layer delay stands for the link latency on the application layers between two end hosts. As shown in **Fig. 1**, compared with the network delay, the application delay consists of the additional application processing latency which can reveal the application service transferring or processing overhead at the application layer. Hence, the application delay is larger than the network delay due to its additional time cost by the application service.

As discussed previously, the whole scripting measurement round time T_r can be defined in **Equation (1)**. In different measurements, the application-delay results (t_m) may exhibit quite similar behaviors on the same link. However, the measurement overhead may deviate very much. Further quantitative discussions based on practical measurements can be found in Section 4.

$$T_r = t_i + t_m + t_d \tag{1}$$

We exmine three basic methodologies: ICMP-based, socket-based and HTTP based approaches in **Table 1**. The measurement overhead of different methods are compared given their suggested parameter settings, including Ping, Flash, Java applet, JSP, JavaScript and HTML5, during one delay measurement round.

Table 1 compares how many packets are transferred for different delay measurement approaches in one measurement round. We assume that the measurement is initiated by end-hosts, which could be a normal computer with a browser, or a mobile device with the embedded browser. Thus, when one measurement finishes, the results must be transferred from the client back to the server for storage. The column “one round packets” shows the total packets generated by each method including the result collection. We find that the Flash ActionScript method generates the largest number of packets due to the requirement that the Flash measurement needs to transfer some security access policies for the “sandbox” between the Flash client and the measurement server. The web-based measurements such as JavaScript and HTML5 which utilize the XMLHttpRequest or WebSocket generate less packets. The column “necessary packets” shows the minimum required packets if the measurement can be optimized to reduce overhead. The Ping method is the most light-weighted while the Flash method is the most heavy-weighted over all the measurement methods. However, Flash can bypass the “sandbox” restriction through the Flash cross-domain policy to optimize the “one round packets” from 27 to 18. Thus, except the ICMP Ping method, the application measurement methods have the similar overhead in the measurement procedure.

Table 1. Measurement overhead comparison

Approach	**Methodology**	**One round packets**	**Required packets**	**Optimization available**
ICMP	Ping	11	2	No
Socket-based	Flash	27	15	Yes
	Java applet	15	13	Yes
	JSP	16	14	Yes
HTTP-based	HTML5	16	12	Yes
	JavaScript	16	12	Yes

3. An Integrated Web-Based Measurement Platform

A web-based measurement platform provides a convenient, fast and suitable environment for mass measurements [22]. It is possible to allow large-scale measurements since the measurement scale is directly proportional to the number of clients visit this web platform. Such a web-based platform does not need any additional measurement configuration support on the client side as long as clients have a web browser.

In order to implement and compare various delay measurement approaches fairly, we first choose the measurement technology which can be supported by most of modern web browsers, such as Chrome, IE, Safari etc., then we construct an integrated web-based measurement platform. This platform implements three browser-scripting delay measurement modules including Flash ActionScript, Java applet and JSP to understand their measurement behaviors. It is deployed on the Internet to collect measurement results and provide a quantitative comparison analysis. It also integrates a Ping variate, a network-layer delay measurement tool, to allow us to study the deviation between the network-layer delay and the application-layer delay.

3.1 Platform Design

As shown in **Fig. 2**, this platform, consisting of a web measurement server and many measurement clients, is deployed on the Internet. The measurement sever is located on a campus network in Wuhan with a Linux Fedora Core system of Intel(R) Xeon(R) CPU 2.40GHz and 32GB memory. A typical measurement client is located on a telecom ADSL network in Wuhan with Windows OS of Intel (R) Core(R) 2.5GHz and 8GB memory. The software installed on the server is Oracle JDK suite 1.60 with Eclipse Jetty Web Server 8, a series of customized measurement tools including the socket adaption, the results storage, the measurement thread control module, etc. A client machine is installed with various web browsers: Chrome, Firefox and Internet Explorer. Li et al. has shown that the application-delay measurement accuracy is only affected slightly with different web browsers [21]. We instrumented a measurement client with various web browsers to access the web server for delay measurement. We also introduced real Internet delay into the experiments in order to investigate the internal relationship between the application-layer delay and the network-layer delay.

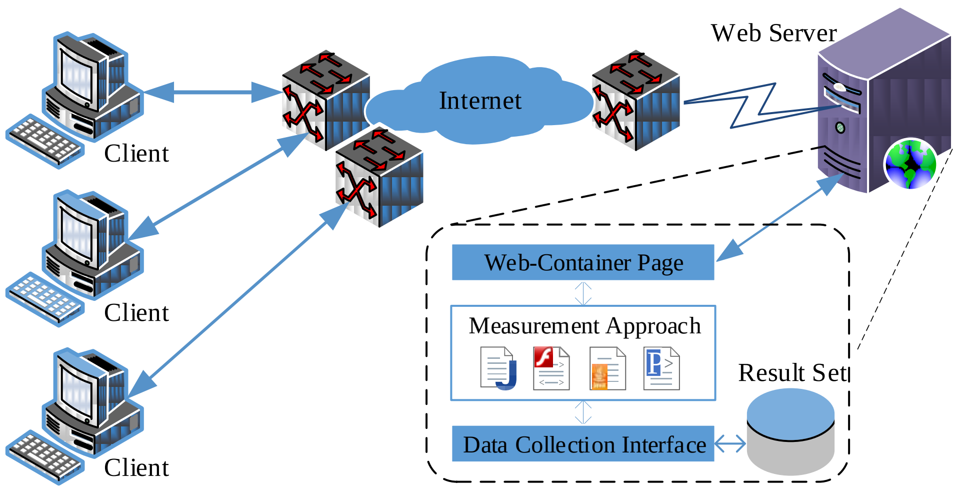

Fig. 2. An integrated web-based measurement platform

3.2 Fair Comparison

We conducted a series of delay measurement experiments during May 2014. To ensure a fair comparison at the same conditions, different measurement modules were executed in a piggy-back fashion round-by-round. The measurement experiments were conducted every 60s interval between each measurement module during one round. During each round, the different measurements were triggered by the scripts to perform the measurement procedures simultaneously. This script control aims to ensure the measurement results observed on the same link during the approximately same period.

To harvest measurement results efficiently, additional control and data collection scripts were developed to automate the above procedures. Upon executing on the client browser successfully, each measurement tool under the measurement interface is invoked and triggered through the web container pages to initiate the different measurement process at different web pages. During each measurement period, each measurement module collects a group of raw delay measurement results; the statistical results such as the average, the minimal and the maximal values were then computed and stored with the raw measurement dataset. In order to ensure that none of the tools were exposed with unfavorable overhead, they were all configured with their suggested optimal parameter settings following the related literature.

When we obtained the link application-layer delays through the web-base measurement platform, the link network-layer delays were also obtained at the same time using the integrated Ping module. Every round measurement results would be tagged with the time stamps at measurement. Then, the data results which have the same time stamps would be chosen as a group delay set. We carefully chose the measurement result groups for the data analysis. The aim of the experiment was to capture the reference data which would provide information on how application-layer delays deviate from the network delays.

4. Analysis

In this section, we present the data analysis of the network delay and application delay behaviors as measured using Java applet, Flash ActionScript and JSP, which are implemented in the integrated web-based measurement platform. Then, we proceed to compare the application-layer delay with the network-layer delay measured by the Ping tool.

4.1 Delay Spikes

In a previous study [23], Sharma and Byers examined the link delay properties and features: these delay characteristics are difficult to capture and predict accurately. Besides, the application-layer delay behavior may deviate far from the network-layer delay 0. With different network conditions and system environments in our dataset, the application delays measured using various web-based tools exhibit much larger delay than the normal Ping results. These significantly large delays, called spikes, are related to the web-based methodologies instead of measurement mistakes or errors. If there exists network congestion on the link or the temporal overloading at client's system, TCP senders may retransmit the lost measurement packets along the path as shown in **Fig. 3** instead of one single measurement round, which cause extrmely large delay measurement results .

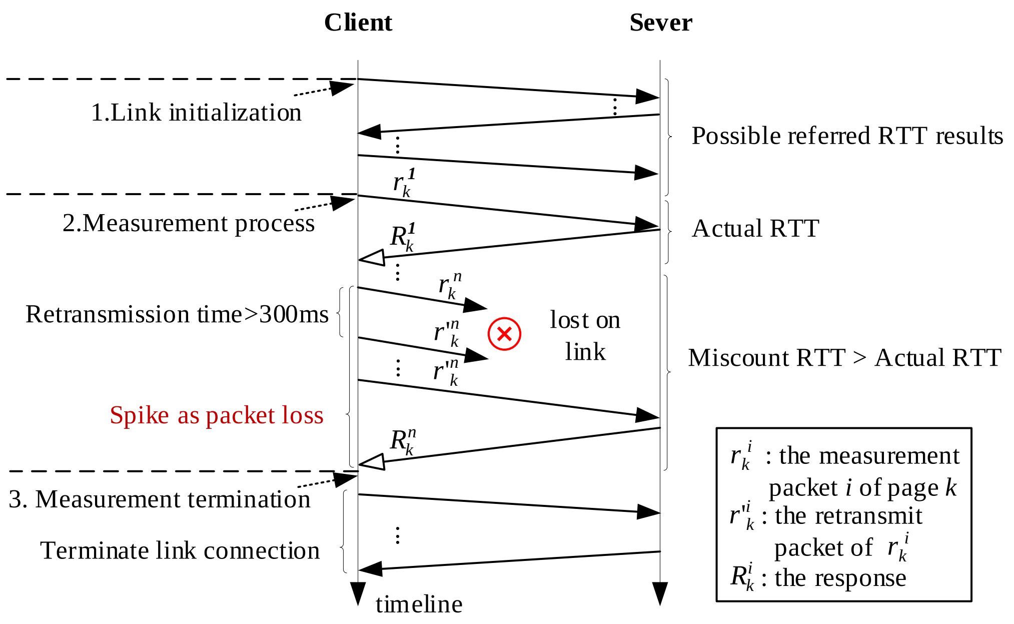
Client
Sever
1.Link initialization
Possible referred RTT results
r_k^1
2.Measurement process
Actual RTT
R_k^1
r_k^n
Retransmission time>300ms
r'^n_k
lost on link
r'^n_k
Miscount RTT > Actual RTT
Spike as packet loss
R_k^n
3. Measurement termination
Terminate link connection
timeline
r_k^i : the measurement packet i of page k
r'^i_k : the retransmit packet of r_k^i
R_k^i : the response

Fig. 3. TCP packet retransmission during measurement

These extremely large delay results can be considered as the outlier measurement samples, which are recommended to be processed as loss [25]. In our dataset, we extracted those outlier samples with threshold > 300*ms* for further analysis as listed in **Table 2**. The Java applet and JSP spikes are mainly located in the range [500, 1000]*ms*, and the Flash ActionScript spikes are mainly scattered in [1000, +∞)*ms*. As the ICMP Ping module packet timeout is normally set at 500*ms*, these web-based measurements spikes (>500*ms*) are treated as packet loss.

Table 2. High link delay comparison

Delay(*ms*)	**> 300**	**> 500**	**> 550**	**> 600**	**> 1000**
Flash ActionScript	2.24%	1.89%	1.81%	1.81%	1.63%
Java applet	5.50%	5.16%	2.84%	1.55%	0.43%
JSP	5.67%	5.59%	3.70%	2.84%	0.26%

We utilize the Ping loss rate as a reference to remove the bias of those extremely high delay in the browser-scripting delay measurements. **Table 3** shows the loss rates of the different web-based measurement modules, which are counted based on the spikes data listed in **Table 2**. Ping has the smallest loss rate, which reveals the network-layer packet loss, while the other three results show the application-layer packet loss.

Table 3. Loss rate comparison

Method	**ICMP Ping**	**Java applet**	**Flash ActionScript**	**JSP**
Loss rate	1.16%	2.84%	1.18%	3.70%

The Flash ActionScript method obtains a smaller application loss rate than the other two web-based measurement methods. The TCP connection generated by the Flash ActionScript is controlled directly by the Flash objects embedded in the HTML page; nevertheless, the Java applet and JSP are quite different. **Table 2 and 3** provide some threshold settings to improve the browser-scripting delay measurement using the web-based modules, and the Flash ActionScript achieves the smallest application-layer loss rate among the three measurement modules.

4.2 Result Comparison

As the spikes indicate the retransmissions of measurement packets, they are considered as loss instead of being included into the end-to-end application-layer delay. As mentioned in Section 2, the application-layer delay t_m is the most focusing metric in these application-layer measurements. Before analyzing the application-layer delay t_m, these spikes should be excluded from the dataset. We utilize the Ping results as the standard scale plate to appraise the measurement results. **Fig. 4** shows that under the normal network conditions, most of the delay measurement results are in the range [40, 80]*ms*. In some worst cases, some delay results are larger than 100*ms* and approach to nearly 300*ms*. The web-based measurement results show that the end-to-end application delay is close to the end-to-end network delay, and the Flash ActionScript measurement results are more centralized than the results of the other two measurement modules. The Java applet and JSP methods show the similar behaviors in the delay measurement.

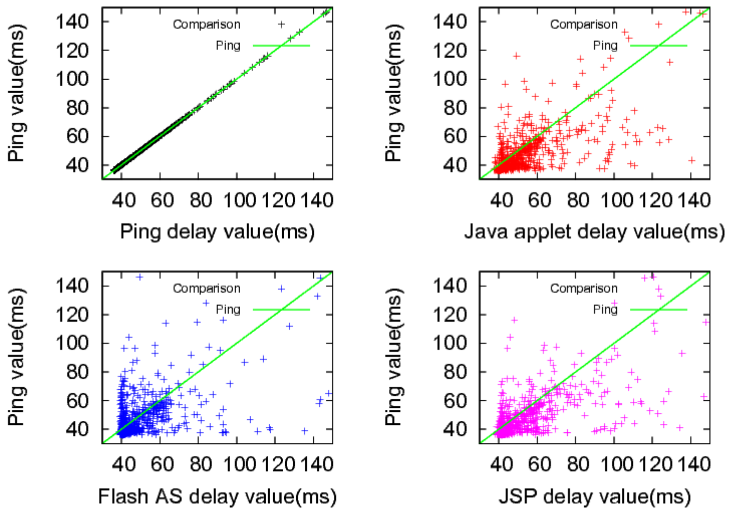

Fig. 4. Delay comparison with different methods

Fig. 5 compares the delay statistics of four different methods including the minimum, average, maximum, median, standard deviation and average deviation. The results show that the normal statistics are very close to each other. In these results, we set the Ping results as the benchmark, the overall statistics results, the average, median, standard deviation and average deviation of the Flash AS results are the closest to the Ping results. However, the minimal and maximal delays are the isolated measurement results, so these two metrics can be used to determine whether these three web-based application measurements are applicable to measure the application-layer delay. The standard deviation and the average deviation show the bias of these measurement methods. The Ping and Flash ActionScript methods have the smaller values for the delay deviation, and the Java applet and JSP have the higher values for the delay deviation. This provides a good indication that the Ping and Flash AS measurements are more accurate than the Java applet and JSP, and the delay errors of the formers are smaller than the latter two.

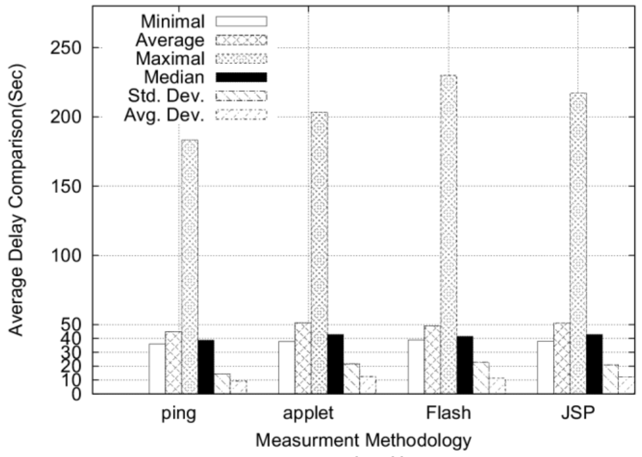

Fig. 5. Delay statistics of different methods

4.3 Dynamic Characteristics

In order to investigate the dynamic characteristics of the web-based measurements, we also tracked the measurement sessions for long durations. The measurements have been automatically executed and recorded in the data group as the original measurement data. **Fig. 6** illustrated the delay measurement results comparison which showed that the measured link delay dynamics under different measurements with respect to the time line. Due to the measurements taken at the same time, the figure utilized the Ping results as the reference and the other three measurement results make the delay comparison with it. In **Fig. 6**, these three web-based application delay measurements demonstrate the proper delay variation at the different time instants. All the application measurement results are higher than the network delay Ping reference results, and the application delay fluctuations are greatly larger than the network delays.

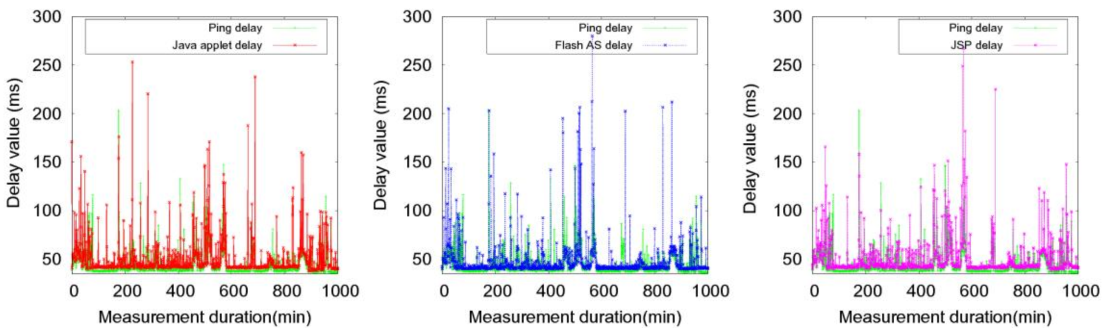

Fig. 6. Dynamic characteristics of different measurement sessions

We set the Ping delays as the benchmark path delays. **Fig. 7** shows the different delay Cumulative Distribution Function(CDF) Curves, in which the red line represents the Ping measurement results, and the Flash AS CDF results are the closest delay curve to the standard delay CDF. The applet and JSP show the similar delay results and characters of the measured path, almost 95% results of them are overlapped with each other. The Flash AS application delay measurement approach shows the better accuracy of the path delay than the other two application delay measurements. All the measurements have the different probabilities to underestimate the application delays, in **Fig. 7**, about 20% Flash results are lower than the standard delay values, and the same criteria to the 15% of the applet and JSP results. For analyzing the accuracy of the measurements, we use the Ping results [0.9, 1.2] interval as the delay measurement criterion, nearly 90% Flash results and 80% applet and JSP results are located in the valid range.

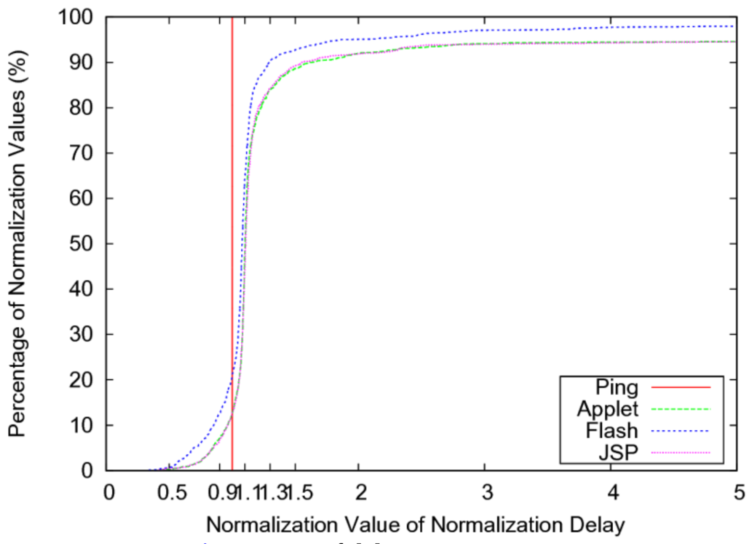

Fig. 7. CDF of delay measurements

4.4 Impact of Script's Pre-loading Time

The web-based delay measurement is advantageous over the classic Ping measurement in that the ICMP ports are commonly blocked on the server side under the security policy; however, the web-based measurement approaches are usually supported due to the need of allowing the web traffic. We conduct additional experiments for analyzing the impact of the script's pre-loading time on the delay basis between the application delay and the network delay. We utilize the Wireshark to capture the measurement packets which are transmitted on the experiment Internet path. Different web-based measurement methods instrument the web pages embedded with the different measurement scripts, the HTTP web page pre-loading time t_{http} and the whole measurement round time T_r should also to be considered.

As discussed previously, the measurement round time T_r contains the t_{http}, t_{tcp}, t_m and t_d, which can be used to represent the overhead of the measurement. t_{http} shows how much time the measurement tool downloads from the server side to the client side, and it is normal to show the method tool size and complexity. Investigating these two metrics can help us understand the deep insights into these application-layer measurement characteristics.

Fig. 8 shows the measurement round time T_r, the HTTP pre-loading time t_{http}, the application-layer link delay t_m using the Java applet, Flash and JSP three web-based measurement modules. In order to observe the application-layer process time during the delay measurement procedure, we also use the Wireshark to capture the measurement packets transferring on the Internet path. These plots first show that the delay results measured by the tools are much closer to the measurement packets captured on the network layer, and the overhead of the delay handle procedure for all methods are acceptable in practice.

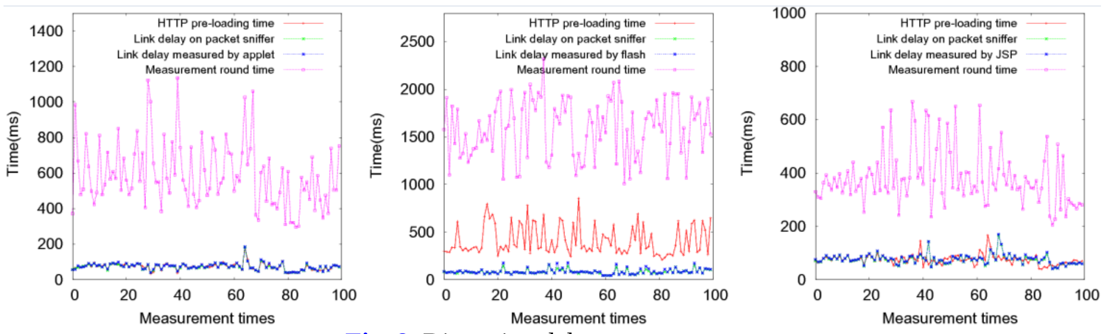

Fig. 8. Dissecting delay measurements

The HTTP pre-loading time t_{http} results in **Fig. 9** show that the applet and JSP have the similar HTTP pre-loading time. It is due to the applet and JSP have the small data and program segment code which can be encapsulated in a single HTTP packet to transfer. This will reduce the measurement pre-loading time before the startup of measurement process. Whereas the Flash needs more time to load some additional “sandbox” security files or cross-domain policy files before the actual measurement session starts, it will require more than one RTT time for the HTTP communication than the other two tools. Hence, the Flash method will cost more time on the measurement pre-loading time than the Java applet and JSP.

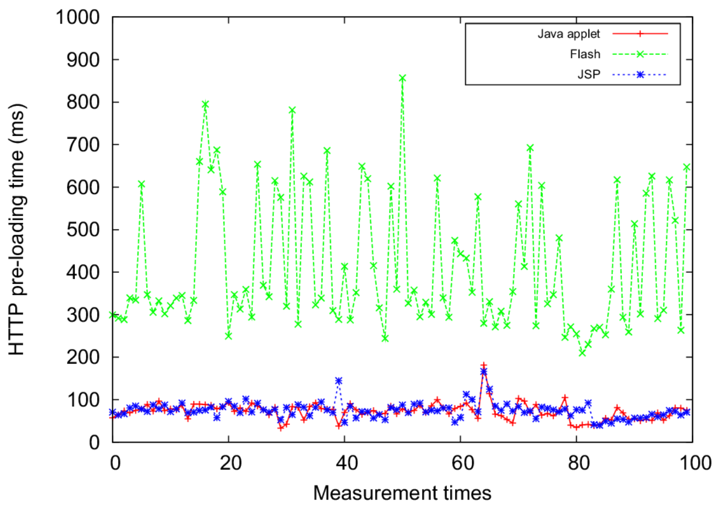

Fig. 9. HTTP pre-loading time comparison

The measurement round time T_r results in **Fig. 10** illustrate the JSP has the mimimal round time whereas the Flash has the highest one. The JSP has the advantage that it does not need to load the JSP measurement scripts to the client which can be directly executed on the server side. Compared with the other two measurements, it will save the data downloading time and the response time in the measurement process. The applet measurement results exhibit the similar status with the JSP, just a little bit higher for loading a very small additional measurment unit to the client side. On the contrary, the Flash measurement tool costs too much time on the pre-loading time and requires the highest overall measurement round time.

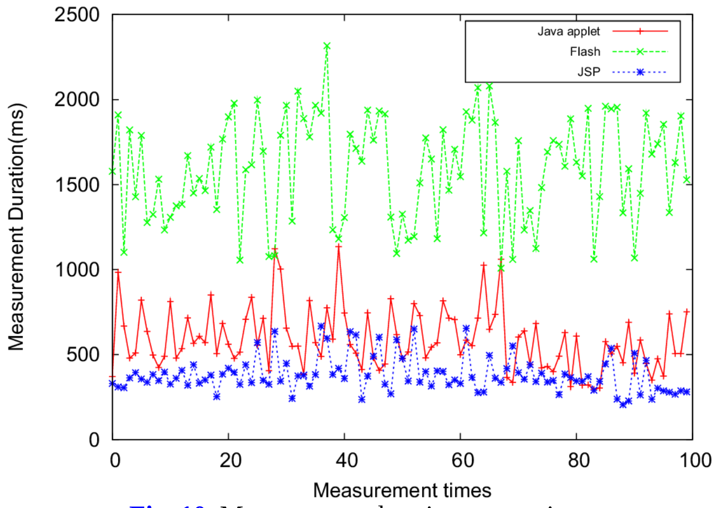

Fig. 10. Measurement duration comparison

In **Table 4**, the Flash ActionScript measurements take the longest duration to accomplish one delay measurement for an end-to-end path which have an average measurement round time about 1589*ms*. Nevertheless, the Flash method can obtain the most accurate delay measurement result than the other two, and the average delay is the closest to the network reference delay measured by the Ping protocol. The Flash delay bias about 4.16*ms* is also the minimum among the three. The applet and JSP have the similar results in the following three aspects: pre-loading time, average delay and the delay bias. The only different factor is the measurement round time. The JSP measurement process is shorter than the applet measurement, and both approaches have much shorter measurement processes than the Flash measurement, but the measurement results are less accurate than the Flash.

Table 4. Statistic results on average (*ms*)

Method	**Pre-loading time**	**Delay**	**Bias(Δd)**	**Round time**
Flash ActionScript	411.13	49.15	4.16	1589.60
Java applet	72.48	51.22	6.23	598.28
JSP	74.01	51.11	6.12	381.27

4.5 Practical Considerations

Based on the overall evaluation experiments, the Flash ActionScript method is recommended if the web browser supports the communication with the Adobe Flash. Although this method requires a relative longer time to obtain the application-layer delay results, it can provide more accurate measurement results compared with the other browser-scripting methods. In addition, the Java applet and JSP present similar ways in the application-layer delay measurement. One difference is that the applet scripts are processed on the measurement client side, whereas the JSP scripts processed on the server side, thus the JSP measurement pre-processing time is shorter than the applet.

During the deployment of the web-based measurement platform, we find that the object creation will affect the measurement results if the measurement method does not handle or control the object creation appropriately. These overhead increases the delay bias Δd siginifcantly. We conducted a simple test to quantify the time overhead due to the object creation using the Java and Flash scripts. In each delay measurement session, the scripts create a new measurement object from the existing class from scratch instead of preloading the exiting one, then we compared the results between the two cases. These results show that the object creation overhead will increase 150 ~ 200*ms* for the delay measurement results if the measurement tool does not well control the object creation. In order to avoid this possible impairment, the simplest way is not to create any objects of any classes during the measurement process.

4.6 Cloud Deployment

Note that many mobile apps utilize the web interface to provide flexible and broad services for a large number of mobile users. Most cloud services are also provided using the web interface. We are motivated to evaluate the proposed browser-scripting delay measurements on the real cloud platforms for examining performance and overhead issues. We deployed our web-based measurement platform on the Baidu cloud to appraise the application-layer delay measurements.

The Baidu cloud platform offers cloud users to create different cloud service instances using the Baidu Cloud Compute (BCC) virtualization utility. Based on this BCC mechanism, we have successfully built a web server running environment for the integrated web-based measurement platform. The web server running environment provides a standard web server container, e.g. Jetty; a common stored database, e.g. MySQL and a Java Running Environment (JRE) to support the service execution of the measurement web platform. Some extra and advanced configurations, such as the system balance, the maximum consumed memory, the constricted CPU utilization, etc., are maintained with the default values.

To understand the diversity of these application delay measurements, we select the widely-used web browsers: Microsoft Internet Explorer (IE, version 11), Google Chrome (version 44.0) and Mozilla FireFox (version 25.0). To support the interpretation and execution of web-based measurement scripts, these browsers are equipped with Adobe Flash plugins, Java Virtual Machine (JVM) and Java plugins compatible with the measurement platform.

As discussed in the previous sections, the application-layer measurements usually require some time overhead to pre-load the measurement scripts. In order to reduce the measurement duration, we ran each measurement module twice within one round to differentiate the network-layer delay, the application-layer delay and the delay bias between two results. As described in **Equation (1)**, the application-layer delay t_m, as shown in **Equation (2)**, can be decomposed into the network delay ΔN and the application processing overhead Δd, which has partly been analyzed in Section 4.4.

$$t_m = \Delta N + \Delta d \tag{2}$$

Fig. 11 presents the measurement results caught by the web-based application delay cloud platform using IE, Chrome and FireFox. ΔN_1 denotes the average network-layer delay through the first time measurement; Δd_1 represents the average delay bias from the first measurement. ΔN_2 and Δd_2 denote the same metrics for the second round time measurement. These time measurements are obtained by analyzing the traffic captured by Wireshark. As discussed previously in Section 4.5, to reduce the application processing overhead Δd influenced by the additional programming overhead, we conduct two consecutive measurements as a complete measurement session for each tool.

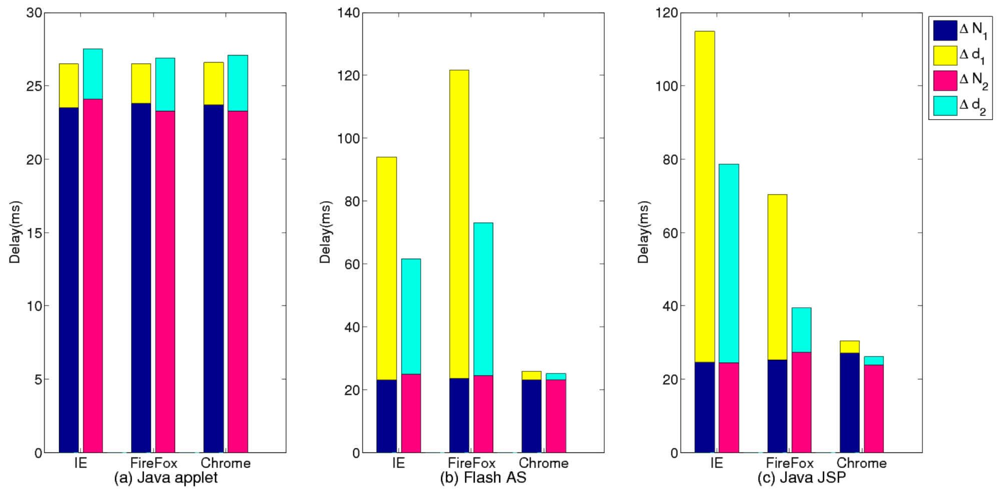

Fig. 11. Network-layer v.s. application-layer delay measurement comparison on the Baidu cloud

The network-layer delay ΔN results on the cloud in **Fig. 11** are quite stable with various measurement tools and browsers, due to the advanced network load balancing policies and the high performance hardwares deployed on the cloud side. However, the application-layer delay t_m and the application processing overhead Δd exhibit quite different characteristics with different web tools and browsers. Google Chrome shows the best performance in the minimal application-layer delay t_m and the inter-process layer time Δd using three measurement tools; whereas the results using IE and FireFox fluctuate greatly using the Flash and JSP methods and show much higher application-layer delays and the application processing overhead.

The applet exhibits quite stable characteristics on the application-layer delay results and is sluggish for different browsers. The Flash and JSP are sensitive for the browser clients, and during the measurement process, they cost much time on the Δd processing using IE and FireFox.

The application-layer delay bias of three measurement tools using different browsers in **Fig. 12** ranges from tens of milliseconds to hundreds of milliseconds, which are significantly far from the stable cloud network-layer delay. In **Fig. 12**, the highest points indicate the maximum values; the lowest points indicate the minimum values. The upper bound of the box indicates the 75% of the data points; the lower bound of the box indicates the 15% of the data points. The Java applet achieves the best accuracy with different browsers, whereas the other two obtain the much more oscillating results with three classic browsers. Note that most cloud and mobile services are instrumented with the web access. The dynamic application-layer delay captures the quality-of-user experiences, which can be used as an important metric to evaluate the cloud services or mobile services.

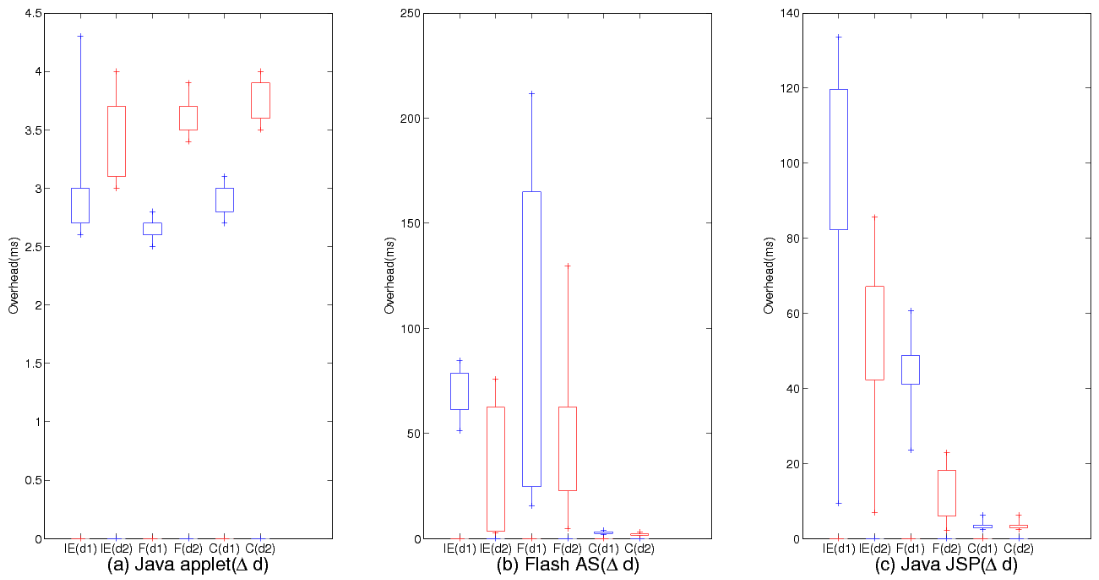

Fig. 12. Measurement time overhead on the Baidu cloud

6. Related Work

Different applications and services deployed on the clouds may need different performance requirements; therefore, different criteria have been proposed to evaluate the performance of the cloud systems, such as service load balance and transaction [26], service workload [27], service delay [28], and so on. Some studies focus on cost reduction in cloud-assisted multimedia systems with required performance requirements. Han et al. proposed a cloud-assisted P2P overlay to reduce the excessive energy consumption in the 5G MANET when the frequent loss of the end links occurs [3]. To avoid the excessive searching and transaction, a cooperative network maintenance mechanism among the super-peers was designed to save the energy consumption in the MANET. Wu et al. are motivated to design and implementation of a cost-effective video gaming system with low latency [4]. Wu et al. proposed cost-effective mobile cloud video streaming with adaptive threshold-controlled cache [5]. Most methods and tools focus on the cloud system performance which can quantify the powerful processing and computing capabilities for cloud clients. However, the network performance between the cloud systems and cloud clients would influence the perceivable performance of end-users. To date, a few studies have been addressing these performance issues using traditional network criteria to evaluate the cloud network performance; nevertheless, these network metrics may not reveal deep insights into the cloud service at the application-layer. In this paper, we are motivated to apply the browser-scripting approach to evaluate the application-layer delay for these web-based cloud services.

As the Internet continues to breed numerous scripting languages such as Flash ActionScripts, Tool Command Language (TCL), Visual Basic scripts (VBS), more and more scripting approaches are developed that subvert these scripting languages themselves to perform delay checks on the pages which they themselves render as typical active measurements [10]. Gummadi et al. in [29] proposed a novel delay measurement tool King by subverting the existing DNS architecture inspired by predecessor tools such as Sting [30]. Turbo King [31] has improved the measurement accuracy and reduced the overhead from the original King tool. [32] and [22] proposed the JavaScript or Flash for measuring the network performance and performed the comparison experiments [2]. Netalyzr [20] utilized the Java applet objects which can be embedded in web pages to perform network measurements. Li et al. [21] examined the delay accuracy in browser-based network measurements using HTTP-based Socket, TCP Socket, and WebSocket with different web browsers in a LAN testbed. [33] constructed an experiment environment to evaluate the TCP connection time of the web services. [5] and [34] studied the accuracy of the application-layer delay measurement on the mobile client side. Wei et al. [35] proposed a model-based approach to identify the dominant congested link along a single path using the periodic end probes. The authors focused on finding the link along the end-to-end path which incurs the most link loss and delay. Two inference algorithms have been proposed to find such a dominant congested link on the path. In this paper, our study mainly focuses on the characteristics and relationship between the network-layer delay and the application-layer delay, and aims to study the factors which impact the application-layer delay on end hosts. Our evaluation study examines three popular scripting languages including Java applet, Flash Actionscript and JSP with an investigation on various practical issues. The performance was evaluated using the same integrated web-based measurement platform. Our experiments were also deployed on the popular Baidu cloud to investigate the accuracy and overhead for different measurement methods.

5. Conclusion

In this paper, we studied the browsing-scripting methods for delay measurement at the application-layer, including Java applet, Flash ActionScript and JSP, in an integrated browser-based network measurement platform to evaluate web-based cloud services. Based on our measurement results, compared with Java applet and JSP, the application-layer delay measurement using the Flash ActionScript is the most accurate. However, the Flash method is not cost-effective among the proposed measurements, and requires the longest pre-loading time and measurement duration. Nevertheless, the Java applet and JSP methods are advantageous in lower overhead and smaller pre-loading time. The browsing-scripting measurement methods can easily cooperate with the cloud-service platforms to provide the network performance monitor by perceived cloud end users. These measurement results may provide some insights into designing new cloud services with enhanced quality-of-experience perceived by cloud users.

Acknowledgement

We would like to thank anonymous reviewers' constructive comments. These comments help us to improve the manuscript significantly. We also thank Fengtao Teng, Xin Huang, Robert Kwadwo Nketia, Botan Suleman Ismail and Yao Yeboah for their contributions in the early stage of this research.

References

[1] C. Zhang, X. Hei, and W. Cheng, "Characterizing the delay performance of web-based cloud services in a browser-scripting approach," in *Proc. of IEEE/CIC Int. Conf. on Communications in China (ICCC)*, November 3-5, 2015. Article (CrossRef Link).

[2] Y. A. Wang, C. Huang, J. Li, and K. W. Ross, "Estimating the Performance of Hypothetical cloud service deployments: A measurement-based approach," in *Proc. of IEEE Conf. on Computer Communications (INFOCOM)*, pp. 2372-2380, April 10-15, 2011. Article (CrossRef Link).

[3] N.D. Han, Y. Chung, and M. Jo, "Green Data Centers for Cloud-assisted Mobile Ad-hoc Networks in 5G," *IEEE Network*, vol. 29, no. 2, pp. 70-76, March, 2015. Article (CrossRef Link).
[4] D. Wu, Z. Xue, J. He, "iCloudAccess: Cost-effective streaming of video games from the cloud with low latency," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 8, pp. 1405-1416, August, 2014. Article (CrossRef Link).
[5] D. Wu, J. Huang, J. He, M. Chen, G. Zhang, Toward Cost-Effective Mobile Video Streaming via Smart Cache With Adaptive Thresholding," *IEEE Transactions on Broadcasting*, vol. 61, no. 4, pp. 639-650, December, 2015. Article (CrossRef Link).
[6] N. Becker, A. Rizk, and M. Fidler, "A measurement study on the application-level performance of LTE," in *Proc. of IFIP Networking Conference*, pp. 1-9, June 2-4, 2014. Article (CrossRef Link).
[7] J. Huang, F. Qian, Y. Guo, Y. Zhou, Q. Xu, Z. M. Mao, S. Sen, and O. Spatscheck, "An in-depth study of LTE: Effect of network protocol and application behavior on performance," *ACM SIGCOMM Computer Communication Review*, vol. 43, no.4, pp. 363-374, October, 2013. Article (CrossRef Link).
[8] J. Wei and C.-Z. Xu, "Measuring client-perceived pageview response time of Internet services," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 5, pp. 773-785, May, 2011. Article (CrossRef Link).
[9] M. Dhawan, J. Samuel, R. Teixeira, C. Kreibich, M. Allman, N. Weaver, and V. Paxson, "Fathom: A browser-based network measurement platform," in *Proc. of ACM Internet Measurement Conference (IMC)*, pp. 73-86, November 14-16, 2012. Article (CrossRef Link).
[10] S. Julian, C. Helle, C. Carlos, and L. M. Ruilope, "A survey on internet performance measurement platforms and related standardization efforts," *IEEE Communications Surveys and Tutorials*, vol. 17, no. 3, pp. 1313-1341, 2015. Article (CrossRef Link).
[11] Q. Li, T. Qin, X. Guan, and Q. Zheng, "Exploring flow characteristics in IPv6: A comparative measurement study with IPv4 for traffic monitoring," *KSII Transactions on Internet and Information Systems*, vol. 8, no. 4, pp. 1307-1323, April, 2014. Article (CrossRef Link).
[12] M. Kaplan, M. Zeljkovic, M. Claypool, and C. Wills, "How's my network? Predicting performance from within a web browser sandbox," in *Proc. of IEEE conf. on Local Computer Networks(LCN)*, pp. 521-528, October 22-25, 2012. Article (CrossRef Link).
[13] M. Butkiewicz, H. V. Madhyastha, and V. Sekar, "Understanding website complexity: Measurements, metrics, and implications," in *Proc. of ACM Internet Measurement Conference (IMC)*, pp. 313-328, November 2-4, 2011. Article (CrossRef Link).
[14] J. P. Kim, D. H. Sung, and J. E. Hong, "Performance Testing of Composite Web-service with Aspect-based WS-BPEL Extension," *KSII Transactions on Internet and Information Systems*, vol. 5, no. 10, pp. 1841-1861, October, 2011. Article (CrossRef Link).
[15] ICSI, "Netalyzr," http://netalyzr.icsi.berkeley.edu/index.html.
[16] Ookla, "Pingtest.net," http://www.pingtest.net/.
[17] Ookla, "Speedtest.net," http://www.speedtest.net/.
[18] 17CE, "17ce.com," http://www.17CE.com/.
[19] InternetFrog.com, "Internetfrog.com speed test," http://www.internetfrog.com/mypc/speedtest/.
[20] C. Kreibich, N. Weaver, B. Nechaev, and V. Paxson, "Netalyzr: Illuminating the edge network," in *Proc. of ACM Internet Measurement Conference (IMC)*, pp. 246-259, November 1-3, 2010. Article (CrossRef Link).
[21] W. Li, R. K. Mok, R. K. Chang, and W. W. Fok, "Appraising the delay accuracy in browser-based network measurement," in *Proc. of ACM Internet Measurement Conference (IMC)*, pp. 361-368, October 23-25, 2013. Article (CrossRef Link).
[22] A. Janc, C. Wills, and M. Claypool, "Network performance evaluation in a web browser," in *Proc. of IASTED Int. Conf. on Parallel and Distributed Computing and Systems (PDCS)*, pp. 370-377, November 2-4, 2009. Article (CrossRef Link).
[23] M. Sharma and J. Byers, "How well does file size predict wide-area transfer time?" in *Proc. of IEEE Global Telecommunications Conference (GLOBECOM)*, pp. 2160-2164, November 17-21, 2002. Article (CrossRef Link).

[24] V. Paxson, “End-to-end routing behavior in the Internet,” *IEEE/ACM Transactions on Networking*, vol. 5, no. 5, pp. 601-615, October, 1997. Article (CrossRef Link).
[25] V. Paxson, M. Allman, J. Chu, and M. Sargent, “IETF RFC 6298: Computing TCPs retransmission timer,” June, 2011. Article (CrossRef Link).
[26] D. Kossmann, T. Kraska, and S. Loesing, “An evaluation of alternative architectures for transaction processing in the cloud,” in *Proc. of ACM Int. Conf. on Management of Data*, pp. 579-590, June 6-11, 2010. Article (CrossRef Link).
[27] A. Iosup, S. Ostermann, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, “Performance analysis of cloud computing services for many-tasks scientific computing,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 6, pp. 931-945, June, 2011. Article (CrossRef Link).
[28] I. A. Moschakis and H. D. Karatza, “Evaluation of gang scheduling performance and cost in a cloud computing system,” *Journal of Supercomputing*, vol. 59, no. 2, pp. 975-992, September, 2012. Article (CrossRef Link).
[29] K. P. Gummadi, S. Saroiu, and S. D. Gribble, “King: Estimating latency between arbitrary Internet end hosts,” *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 3, pp. 11–11, November, 2002. Article (CrossRef Link).
[30] S. Savage, “Sting: a TCP-based network measurement tool,” in *Proc. of the 2nd Conf. on USENIX Symposium on Internet Technologies and Systems (USITS)*, pp. 239-248, October 11-14, 1999. Article (CrossRef Link).
[31] D. Leonard and D. Loguinov, “Turbo King: Framework for large-scale Internet delay measurements,” in *Proc. of IEEE Conf. on Computer Communications (INFOCOM)*, pp. 31-35, April 13-18, 2008. Article (CrossRef Link).
[32] R. Rajamony and M. Elnozahy, “Measuring client-perceived response times on the WWW,” in *Proc. of the 3rd USENIX Symp. on Internet Technologies and Systems*, pp. 16-16, March 26-28, 2001. Article (CrossRef Link).
[33] V. Bajpai and J. Schonwalder, “Measuring TCP connection establishment times of dual-stacked web services,” in *Proc. of the 9th Int. Conf. on Network and Service Management (CNSM)*, pp. 130-133, October 14-18, 2013. Article (CrossRef Link).
[34] W. Li, R. K. P. Mok, D. Wu, and R. K. C. Chang, “On the accuracy of smartphone-based mobile network measurement,” in *Proc. of IEEE Conf. on Computer Communications (INFOCOM)*, pp. 370-378, April 26-May 1, 2015. Article (CrossRef Link).
[35] W. Wei, B. Wang, D. Towsley and J. Kurose, “Model-based Identification of Dominant Congested Links,” *IEEE/ACM Transactions on Networking*, vol. 19, no. 2, pp. 456-469, April, 2011. Article (CrossRef Link).

Chengwei Zhang received the B.S. degrees in computer science and the M.S. degrees in electronics and information engineering from Huazhong University of Science and Technology, Wuhan, China, in 2001 and 2004, respectively, and is currently working toward the Ph.D. degree at the School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan, China. His research interests include network measurement and performance evaluation.

Xiaojun Hei (S'02–M'07) received the B. Eng. Degree from the Huazhong University of Science and Technology, Wuhan, China, in 1998, and the M. Phil. and Ph.D. degrees from the Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong, in 2000 and 2008, respectively. Since 2008, he has been an Associate Professor with the School of Electronic Information and Communications, Huazhong University of Science and Technology. His current research interests include network measurement and software-defined networks.

Wenqing Cheng (M'07) received the B.S. degree in telecommunication engineering and Ph.D. degree in electronics and information engineering from the Huazhong University of Science and Technology, Wuhan, China, in 1985 and 2005, respectively. She is currently a Professor with the School of Electronic Information and Communications, Huazhong University of Science and Technology. Her research interests include information systems and e-learning applications.