On the Network Characteristics of the Google's Suggest Service

Zakaria Al-Qudah, Mohammed Halloush, Hussein R. Alzoubi, Osama Al-kofahi
Yarmouk University, Dept. of Computer Engineering, Irbid, Jordan
Email:{zakaria.al-qudah, mdhall, halzoubi, osameh}@yu.edu.jo

***Abstract*— This paper investigates application- and transport-level characteristics of Google's interactive Suggest service by analyzing a passively captured packet trace from a campus network. In particular, we study the number of HTTP GET requests involved in user search queries, the inter-request times, the number of HTTP GET requests per TCP connection, the number of keyword suggestions that Google's Suggest service to users, and how often users utilize these suggestions in enhancing their search queries. Our findings indicate, for example, that nearly 40% of Google search queries involve five or more HTTP GET requests. For 36% of these requests, Google returns no suggestions, and 57% of the time users do not utilize returned suggestions. Furthermore, we find that some HTTP characteristics such as inter-request generation time for interactive search application are different from that of traditional Web applications. These results confirm the findings of other studies that examined interactive applications and reported that such applications are more aggressive than traditional Web applications.**

***Index Terms*— Ajax, Web 2.0, Network measurments, Performance, Google search engine**

I. Introduction

Interactive Web applications have become extremely common today. The majority of Web sites including major Web-based email services (e.g., Gmail, Yahoo mail, etc.), map services, social networks, and web search services support an interactive user experience.

One of the enabling technologies for this interactive user-engaging experience is Asynchronous Javascript and XML (AJAX) [1]. AJAX allows the web client (browser) to asynchronously fetch content from a server without the need for typical user interactions such as clicking a link or a button. Due to this asynchronous nature, these interactive applications exhibit traffic characteristics that might be different from those of classical applications. In classical (non-interactive) applications, requests for content are usually issued in response to human actions such as clicking a link or submitting a Web form. Thus, the human factor is the major factor in traffic generation. With interactive applications, however, requests can be issued in response to user interactions that typically would not generate requests such as filling a text field or hovering the mouse over a link or an image. Furthermore, these requests can be made even without user intervention at all such as the case of fetching new email message with Gmail or updating news content on a news Web site. Therefore, traffic generation is not necessarily limited by the human factor.

In this paper we focus on one such interactive application which is the Google interactive search engine. Google search engine has many interactive features that are aimed at providing a rich search experience implemented using the AJAX technology [2]. For example, Google Suggest (or Autocomplete) [3] provides suggested search phrases to users as they type their query (See Fig. 1). The user can select a suggested search phrase, optionally edit it, and submit a search query for that search phrase. Suggestions are created using some prediction algorithm to help users find what they are looking for. Google Instant Search service [4] streams a continuously updated search results to the user as they type their search phrases (See Fig. 2). This is hoped to guide users' search process even if they do not know what exactly they are looking for. The other interactive feature that Google provides is Instant Previews [5]. With Instant previews (Shown in Fig. 3), users can see a preview of the web pages returned in the search results by simply hovering over these search results. This service is aimed at providing users with the ability to quickly compare results and pinpoint relevant content in the results web page.

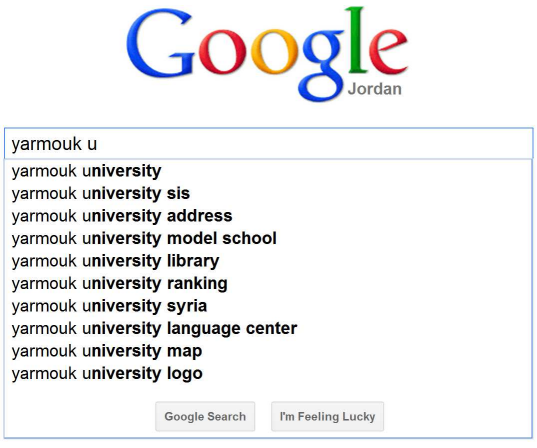

Figure 1. A snapshot of the Google Suggest feature

In this paper, we study the characteristics of the Google Suggest service by analyzing a passively captured packet trace from Yarmouk University campus in Jordan. We look into the number of HTTP GET requests a search query generates, the inter-request generation time, the number of suggestions Google typically returns for a request, and the percentage of time the returned suggestions

doi:10.4304/jetwi.4.3.278-284

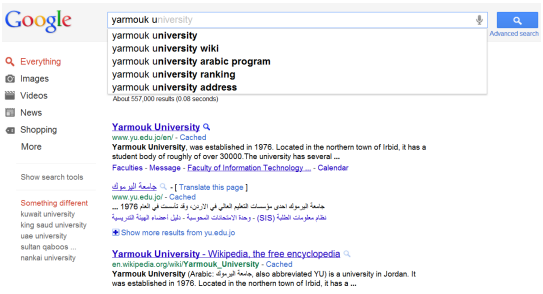

Figure 2. A snapshot of the Google Instant search feature. Note that the search results for the first suggestion is displayed before the user finishes typing his/her query

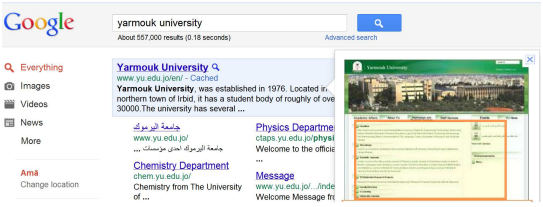

Figure 3. A snapshot of the Google Instant Preview feature. Note the displayed preview of the web page of the first search result entry "Yarmouk University"

are actually utilized by users. We have already begun to study the Google Instant feature and plan to study the Instant Previews in the future.

The rest of this paper is organized as follows. Section II highlights some background information related to our work. Section III motivates our work. Section IV presents the related work. Section V highlights the packet trace capturing environment and characteristics. Section VI presents our results and discuses our findings. We conclude and present our future work plans in Section VII.

II. Background

When browsing the web, one normally uses web search engines several times a day to find the required information on the web. Web search engines therefore are visited by huge number of people every day. Web search can use query-based, directory-based, or phrase-based query reformulation-assisted search methods. Google is considered among the most popular search engines on the web. The Google search engine uses the standard Internet query search method [6], [7].

In 2010 Google announced Google Instant, which live-updates search results interactively during the time at which users type queries. Every time the user hits a new character, the search results are changed accordingly based on what the search engine thinks a user is looking for. This can save substantial user time, since most of the time, the results that a user are looking for are returned before finishing typing. Another advantage of Google Instant is that users are less likely to type misspelled keywords because of the instant feedback. The public generally provided positive feedback towards this new feature [8].

Google Instant Preview is another feature provided by Google. This feature allows users to get snapshots of web pages for the search results without leaving the search results. This feature enhances researcher's search experience and satisfaction. Google Instant Preview provides an image of the web page in addition to the extracted text. Previews are dynamically generated because content is continuously changing. Google users are 5% better satisfied with this new feature [9].

III. Motivation

One motivation of this study is that we believe that measuring such services is extremely important due to the recent popularity of interactive web features. Characterizing new trends in network usage help the research community and network operators update their mental model about network usage. Another motivation is that such characterization is quite important for building simulators and performance benchmarks and for designing new services and enhancing existing ones.

Moreover, Google interactive features may produce large amount of information, which may result in bad experience for users on mobile devices or over low-speed Internet connections [8]. With the prevalence of browsing the web via mobile devices today, we believe that characterizing these services is vital to understanding the performance of these services. To the best of our knowledge, this is the first attempt to characterizing interactive features of a search engine from the application- and the transport-level perspective.

IV. Related Work

The AJAX technology suit enables automated HTTP requests without human intervention by allowing web browsers to make requests asynchronously. This has been made possible through the use of advanced features of HTTP 1.1 like prefetching data from servers, HTTP persistent connections, and pipelining. These features mask network latency and give end users a smoother experience of web applications. Therefore, AJAX creates interactive web applications and increases speed and usability [10].

The authors of [10] performed a traffic study of a number of Web 2.0 applications and compared their characteristics to traditional HTTP traffic through statistical analysis. They collected HTTP traces from two networks: the Munich Scientific Network in Munich, Germany and the Lawrence Berkeley National Laboratories (LBNL) in Berkeley, USA and classified traffic into Web 2.0 applications traffic and conventional applications traffic. They have used packet-level traces from large user populations and then reconstructed HTTP request-response streams. They identified the 500 most popular web servers that used AJAX-enabled Web 2.0 applications. Google Maps is one of the first applications that used AJAX. Therefore, the authors have focused on Google Maps Traffic. The presented findings of this study show that Web 2.0 traffic is more aggressive and bursty than classical HTTP traffic. This is due to the active prefetching of data, which means

many more automatic HTTP requests, and consequently greater number of bytes transferred. Moreover, they found that sessions in AJAX applications last longer and are more active than conventional HTTP traffic. Furthermore, AJAX inter-request times within a session are very similar and much shorter because they are more frequent than all other HTTP traffic.

Besides [10], some work exists in the literature for characterizing HTTP traffic generated by popular Web 2.0 websites. In [11] for example, the authors examined traces of Web-based service usage from an enterprise and a university. They examined methodologies for analyzing Web-based service classes and identifying service instances, service providers, and brands pointing to the strengths and weaknesses of the techniques used. The authors also studied the evolution of Web workloads over the past decade, where they found that although the Web services have significantly changed over time, the underlying object-level properties have not.

The authors of [12] studied HTTP traffic from their campus network related to map applications. Their work examined the traffic from four map web sites: Google Maps, Yahoo Maps, Baidu Maps, and Sogou Maps. In their paper, they proposed a method for analyzing the mash-up (combing data from multiple sources) characteristics of Google Maps traffic. They found that 40% of Google Maps sessions come from mash-up from other websites and that caching is still useful in web based map applications.

Li et al. [13] studied the evolution of HTTP traffic and classified its usage. The results provided are based on a trace collected in 2003 and another trace collected in 2006. The total bytes in each HTTP traffic classes in the two traces were compared. The authors found that the whole HTTP traffic increased by 180% while Web browsing and Crawler both increased by 108%. However, Web apps, File download, Advertising, Web mail, Multimedia, News feeds and IM have shown sharp rise.

Maier et al. [14] presented a study of residential broadband Internet traffic using packet-level traces from a European ISP. The authors found that session durations are quite short. They also found that HTTP, not peer-to-peer, carries most of the traffic. They observed that Flash Video contributes 25% of all HTTP traffic, followed by RAR archives, while peer-to-peer contributes only to 14% of the overall traffic. Moreover, most DSL lines fail to utilize their available bandwidth and that connections from client-server applications achieve higher throughput per flow than P2P connections.

In [15], a study of user sessions of YouTube was conducted. The results obtained from the study indicate longer user think times and longer inter-transaction times. The results also show that in terms of content, large video files transferred. Finally, in [16] [17], the authors proposed the AJAXTRACKER, a tool for mimicking a human interaction with a web service and collecting traces. The proposed tool captures measurements by imitating mouse events that result in exchanging messages between the client and the Web server. The generated traces can be used for studying and characterizing different applications like mail and maps.

V. Data set and methodology

As mentioned, this study is conducted based on a packet-level trace captured at the edge of the engineering building at Yarmouk University, Jordan. The engineering building contains roughly 180 hosts that are connected through a typical 100Mbps ethernet. The trace is collected over a period of five business days. The trace contains a total 31490 HTTP transactions that are related to Google search. To extract the transactions that are related to Google search, the URL or the "HOST:" HTTP request header has to contain the word "google". The search query is contained in the URL in the form of "q=xyz" where "xyz" is the query. After identifying the HTTP request as a Google search request, the corresponding HTTP response is also extracted. The returned suggestions are extracted from these HTTP responses. We also collect the type of the returned HTTP response in order to separate queries from one another as explained later in Section VI.

VI. Results

In this section, we measure various parameters related to Google search queries. To identify the boundaries of a search query, we manually analyze a portion of the collected trace. We found that throughout the process a user is typing the search phrase, the browser generates HTTP GET requests. For these HTTP GET requests, the type of the HTTP response is either "text/xml" or "text/javascript". When the user hits the Return key (to obtain the search results), the browser generates another HTTP GET request, for which the type of the returned HTTP response is "text/html". We verify this observation by actively performing a number of search queries and observing the captured traffic.

In our trace, however, we found a number of occurrences of a scenario where a series of HTTP GET requests from a user appear to be related to two different queries, yet this series of HTTP GET requests is not split by an "text/html" response separating the boundaries of the two different queries. There is a number of usage scenarios that could result in such a behavior. For example, a user might type in a search phrase and get interrupted for some reason. Therefore, the search query will end without the Return key being hit. Furthermore, the TCP connection that is supposed to carry the last HTTP response might get disrupted after the user hit the Return key and before the HTTP response is delivered back to the user.

To handle such cases, we consider two HTTP GET requests that are not split by an "text/html" response belong to two different search queries if the time separation between the two requests is greater than t seconds. To find a suitable value for this parameter, we plot the percentage of queries found using the time separation to the overall number of search queries for different values of t in Fig.

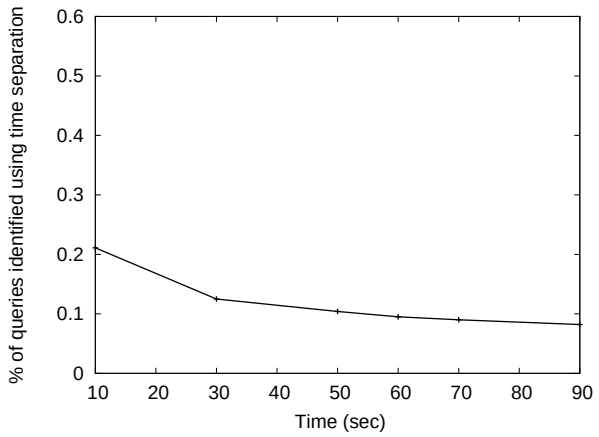

Figure 4. Setting of parameter t

4. The figure suggests that, in general, the percentage of queries identified using the time separation heuristic is insensitive to the setting of the parameter t when t is above 30 seconds, an $t = 60$ is an appropriate value since the percentage of search queries identified using the time spacing heuristic to the number of overall search queries remains stable around this value. Therefore, we choose this value throughout our evaluation below.

A. HTTP GET Requests

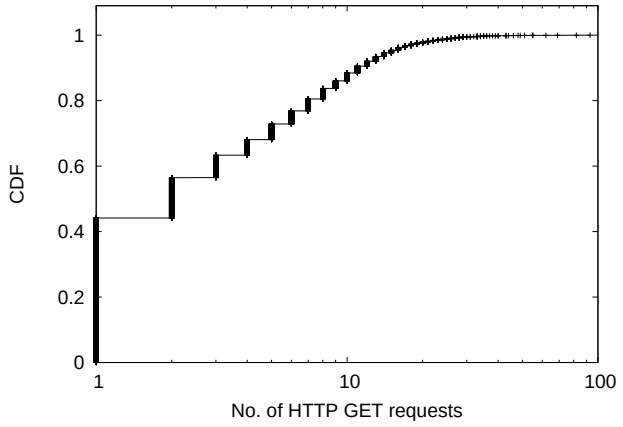

Figure 5. No. of HTTP GET requests per search query

This subsection investigates the number of HTTP GET requests a search query typically involves. We identify a total of 7598 search query. We plot the Cumulative Distribution Function (CDF) of the number of HTTP GET requests in a query in Fig. 5. As shown, over 40% of search queries involve only one HTTP GET request. The possible reasons for the existence of these queries include (*i*) users not turning on Google Suggest and (*ii*) users copying search phrases and pasting them into Google and hitting the Return key. The figure also shows that arround 30% of search queries involve five or more HTTP GET requests with some search queries involving over 90 HTTP GET requests. These results show that the "chatty" nature of AJAX-based applications reported in [10] for the map and email applications also applies to the Google suggest application.

Among the near 60% of search queries for which we believe that users are enabling Suggest (i.e., number of GET requests is greater than one), the vast majority of queries seem to not utilize the suggestions for the first few characters. This is because users continue to type despite the returned suggestions. A possible reason for this might be that for a small number of characters of the search phrase, Google returns quite general suggestions that are usually not selected by the user. We believe there is a room for improvement in the service design by not returning suggestions for the initial few characters of the query.

B. Inter-Requests Time

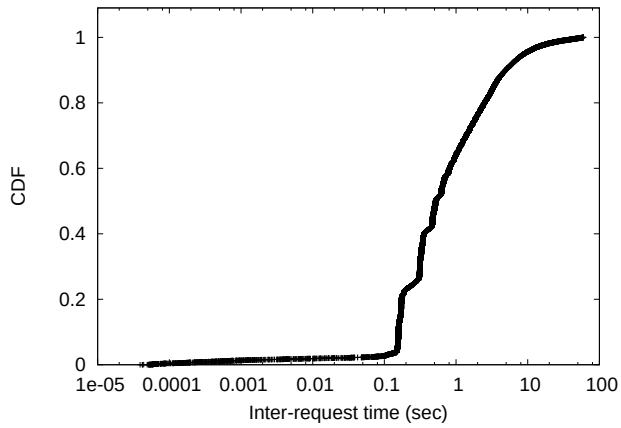

Figure 6. HTTP GET inter-request times within a search query

Next, we turn our attention to investigating the time spacing between HTTP GET requests within the same search query. Fig. 6 shows the results. As shown, 64% of HTTP GET requests involved in a search query are separated by less than one second. We contrast these results with our mental model of traditional HTTP interactions. Normally, a number of HTTP requests are made to download a Web page along with the embedded objects. Then, a think time elapses before new requests are made to download a new page [18]. The interactive search application generates a radically different pattern of HTTP GET requests. This is due to the fact that requests for new set of Google suggestions are automatically made while the user types the search query. This also confirms the results of [10] indicating that inter-request times are shorter in AJAX-based applications than it is in traditional applications. We however believe that the traffic characteristics of these interactive applications are generally application-dependent and not technology-dependent. That is, the characteristics of the traffic that is

generated by an application employing AJAX technology depends on the type of the application and not on the fact that it uses the AJAX technology. This is because AJAX enables the application to generate traffic automatically without user intervention (or in response to user actions that typically do not generate traffic such as hovering over a link), however, it is up to the application logic to decide whether to generate HTTP requests and when to generate these requests.

C. TCP Connections

In our trace, we find that each HTTP GET request is carried over its own TCP connection. To verify if this is a result of the deployed HTTP proxy, we have performed a number of search queries from the author's houses (i.e., using residential broadband network connections). This experiment involves performing a number of search queries for different web browsers (Microsoft Internet Explorer, Mozialla Firefox, and Google Chrome) on a Microsoft Windows 7 machine. We have captured and examined the packet trace of these search queries. Our findings indicate that, contrary to what we find in our trace, various HTTP GET requests can be carried over the same TCP connection. We note here that HTTP proxies are commonly deployed in institutional networks. Therefore, our network setting is not necessarily unique, and we believe that it is totally legitimate to assume that many other institutions are employing similar network settings.

We note that having a separate TCP connection per request might have significant impact on the performance of this service. In particular, each new TCP connection requires the TCP three-way handshake which might add a significant delay. Furthermore, if an HTTP request needs to be split over many packets, the rate at which these packets are transmitted to the server is limited by the TCP congestion control mechanisms. Therefore, these added delays might limit the usefulness of the Suggest service since suggestions usually become obsolete when the user types new text.

D. Suggestions (Predictions)

In this section, we investigate the number of suggestions Google returns in HTTP responses for each HTTP GET request. We observe that the maximum number of returned suggestions is ten. Fig. 7 plots CDF of the number of returned suggestions per HTTP GET request. As shown, close to 40% of HTTP responses involve zero suggestions. This includes cases where the suggest algorithm returned no suggestions and connections disrupted before responses arrive. Furthermore, around 50% of HTTP responses the Google Suggest service returns the full ten suggestions. For the remaining small fraction, the Suggest service returns between one and nine suggestions. A likely reason for returning between one and nine sugestions is the inability of google to find ten suggestions for the specific user's search phrase.

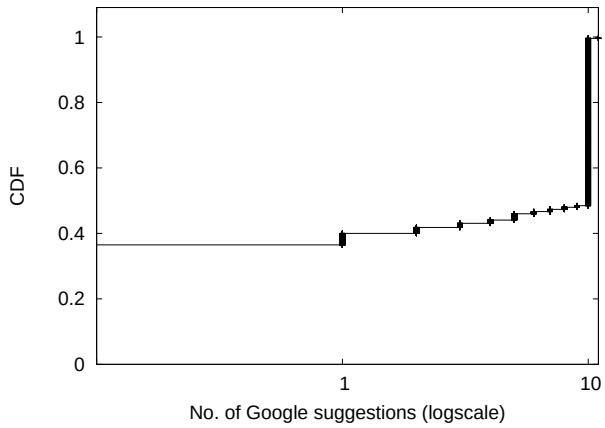

Figure 7. No. of suggested search phrases returned for an HTTP GET request

E. Prediction Usefulness

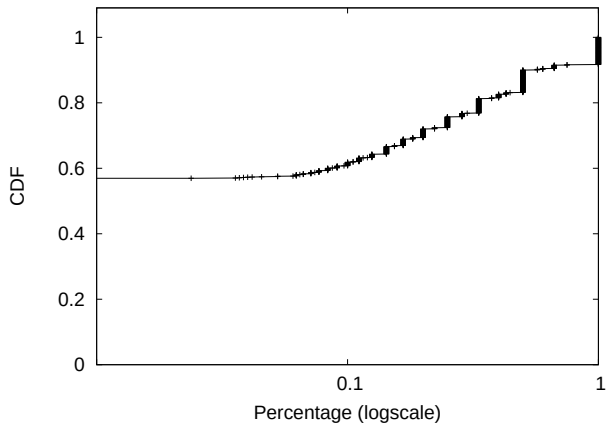

Figure 8. Percentage of time users actually use the returned suggestions of the search query

Next, we investigate the percentage of time users do actually use the returned suggestions during the search process. To assess this, within a search query, we assume that the user has utilized the returned suggestions if the search phrase in the current request matches one of the suggestions appeared in the response for the previous request. To illustrate this, consider the following scenario from our trace. An HTTP GET request was sent to Google with "you" as a partial search phrase. Google responded with "youtube, you, youtube downloader, yout, youtube to mp3, youtu, youtube download, youtube music, you top, you born" as search suggestions. The next HTTP GET request was sent to Google with "youtube" as the search phrase. In this case, we assume that the user has utilized the returned suggestion since the current HTTP GET request involves one of the suggestions that were provided as a response to the previous HTTP GET request. That is, the user is asking for "youtube" which was one of the suggestions made by Google in the previous response.

We note that this is an upper limit on the usage of this service because a search phrase in the current request may match the suggestions in the previous request, yet, the user might have typed the phrase instead of selecting it from the list of returned suggestions.

The result is plotted in Fig. 8. As shown, nearly 58% of the time, users do not use the returned suggestions at all. On the other hand, nearly 10% of queries are constructed with complete guidance of the Suggest service. The following scenario from our trace illustrates a case where a query can be constructed with complete help of Google suggestions. A user typed "f" which triggered a request for suggestions to Google. Google responsed with "facebook,face,fa,friv,fac,firefox,faceboo,farfesh,factjo,fatafeat" as suggestions. The next and final request was for "facebook" which is among the suggested search phrases. In this case the user selected a search phrase from the first set of returned suggestion to complete the search query. This means suggestions are fully utilized. Hence, full utilization of google suggest service is acheived when the user selects a suggested phrase from each returned list of suggestions for a particular search query.

VII. Conclusions and Future Work

In this paper, we have investigated the application- and transport-level characteristics of Google's Suggest interactive feature as observed in a passively captured packet trace from a campus network. We find that a large number of HTTP GET requests could be issued to obtain suggestions for a search query. Interestingly, the characteristics of the HTTP GET requests deviate significantly from those of HTTP GET requests issued for classical Web interactions. In particular, while classical Web interactions are limited by the human factor (think-time), interactive applications are not necessarily limited by this factor. Furthermore, we have characterized the number and usefulness of suggestions made by Google. To this end, we have found that Google responds to the majority of requests for suggestions with either zero or 10 suggestions (the number 10 is the maximum number of suggestions returned per request). However, nearly 58% of users do not utilize the returned suggestions at all.

We have already begun to investigate the characteristics of other Google interactive search features such as Google instant search and plan to evaluate the Google instant preview as well.

References

[1] J. J. Garrett, "Ajax: A new approach to web applications," http://adaptivepath.com/ideas/essays/archives/000385.php, February 2005, [Online; Stand 18.03.2008]. [Online]. Available: http://adaptivepath.com/ideas/essays/archives/000385.php

[2] "Ajax:A New Approach to Web Applications," http://adaptivepath.com/ideas/ajax-new-approach-web-applications.

[3] "Google Suggest (or Autocomplete)," http://www.google.com/support/websearch/bin/static.py?hl=en&page=guide.cs&guide=1186810&answer=106230&rd=1.

[4] "Google Instant," http://www.google.com/instant/.

[5] "Google Instant Previews," http://www.google.com/landing/instantpreviews/#a.

[6] P. Bruza, R. McArthur, and S. Dennis, "Interactive internet search: keyword, directory and query reformulation mechanisms compared," in *SIGIR'00*, 2000, pp. 280–287.

[7] S. Dennis, P. Bruza, and R. McArthur, "Web searching: A process-oriented experimental study of three interactive search paradigms," *Journal of the American Society for Information Science and Technology, vol. 53, issue 2*, pp. 120–130, 2002.

[8] http://dejanseo.com.au/google-instant/.

[9] http://dejanseo.com.au/google-instant-previews/.

[10] F. Schneider, S. Agarwal, T. Alpcan, and A. Feldmann, "The new web: Characterizing ajax traffic." in *PAM'08*, 2008, pp. 31–40.

[11] P. Gill, M. Arlitt, N. Carlsson, A. Mahanti, and C. Williamson, "Characterizing organizational use of web-based services: Methodology, challenges, observations and insights," *ACM Transactions on the Web*, 2011.

[12] S. Lin, Z. Gao, and K. Xu, "Web 2.0 traffic measurement: analysis on online map applications," in *Proceedings of the 18th international workshop on Network and operating systems support for digital audio and video*, ser. NOSSDAV '09. New York, NY, USA: ACM, 2009, pp. 7–12. [Online]. Available: http://doi.acm.org/10.1145/1542245.1542248

[13] W. Li, A. W. Moore, and M. Canini, "Classifying http traffic in the new age," 2008.

[14] G. Maier, A. Feldmann, V. Paxson, and M. Allman, "On dominant characteristics of residential broadband internet traffic," in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, ser. IMC '09. New York, NY, USA: ACM, 2009, pp. 90–102. [Online]. Available: http://doi.acm.org/10.1145/1644893.1644904

[15] P. Gill, M. Arlitt, Z. Li, and A. Mahanti, "Characterizing user sessions on youtube," in *In Proc. of 15th Annual Multimedia Computing and Networking Conference*, San Jose, CA, USA, 2008.

[16] M. Lee, R. R. Kompella, and S. Singh, "Ajaxtracker: active measurement system for high-fidelity characterization of ajax applications," in *Proceedings of the 2010 USENIX conference on Web application development*, ser. WebApps'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 2–2. [Online]. Available: http://dl.acm.org/citation.cfm?id=1863166.1863168

[17] M. Lee, S. Singh, and R. Kompella, "Ajaxtracker: A tool for high-fidelity characterization of ajax applications," 2008.

[18] M. E. Crovella and A. Bestavros, "Self-similarity in world wide web traffic: Evidence and possible causes," *IEEE/ACM Transactions on Networking*, 1997.

Zakaria Al-Qudah received his B.S. degree in Computer Engineering from Yarmouk University, Jordan in 2004. He recived his M.S. and Ph.D. degrees in Computer Engineering from Case Western Reserve University, USA, in 2007 and 2010 respectively. He is currently an assistent professor of Computer Engineering at Yarmouk University. His research interests include internet, content distribution networks, and security.

Mohammed Halloush received the B.S. degree from Jordan University of Science and Technology, Irbid, Jordan in 2004, the M.S. and the Ph.D. degrees in Electrical Engineering from Michigan State University, East Lansing, MI, USA in 2005, 2009 respectively. Currently he is an Assistant professor in the department of Computer Engineering at Yarmouk University, Irbid Jordan. His research interests include network coding,

multimedia communications, wireless communications and networking.

Hussein Al-Zoubi received his MS. and Ph.D. in Computer Engineering from the University of Alabama in Huntsville, USA in 2004 and 2007, respectively. Since 2007, he has been working with the Department of Computer Engineering, Hijjawi Faculty for Engineering Technology, Yarmouk University, Jordan. He is currently an associate professor. His research interests include computer networks and their applications: wireless and wired, security, multimedia, queuing analysis, and high-speed networks.

Osameh Al-Kofahi received his B.S. degree in Electrical and Computer Engineering from Jordan University of Science and Technology, Irbid, Jordan in 2002. He received his Ph.D. degree from Iowa State University, USA. in 2009. His research interests include Wireless Networks, especially Wireless Sensor Networks (WSNs), Wireless Mesh Networks (WMNs) and Ad hoc networks, Survivability and Fault Tolerance in wireless networks and Practical Network Coding.