Network Level Footprints of Facebook Applications

Atif Nazir[1,*], Saqib Raza[1,*], Dhruv Gupta[1,*], Chen-Nee Chuah[1,*], Balachander Krishnamurthy[2,+]

[1]University of California–Davis, CA USA
[2]AT&T Labs–Research, NJ USA
[*]{anazir, sraza, dgupta, chuah}@ucdavis.edu
[+]bala@research.att.com

ABSTRACT

With over half a billion users, Online Social Networks (OSNs) are the major new applications on the Internet. Little information is available on the network impact of OSNs, although there is every expectation that the volume and diversity of traffic due to OSNs is set to explode. In this paper, we examine the specific role played by a key component of OSNs: the extremely popular and widespread set of third-party applications on some of the most popular OSNs. With over 81,000 third-party applications on Facebook alone, their impact is hard to predict and even harder to study.

We have developed and launched a number of Facebook applications, all of which are among the most popular applications on Facebook in active use by several million users monthly. Through our applications, we are able to gather, analyze, correlate, and report their workload characteristics and performance from the perspective of the application servers. Coupled with PlanetLab experiments, where active probes are sent through Facebook to access a set of diverse applications, we are able to study how Facebook forwarding/processing of requests/responses impacts the overall delay performance perceived by end-users. These insights help provide guidelines for OSNs and application developers. We have also made the data studied here publicly available to the research community. This is the first and only known study of popular third-party applications on OSNs at this depth.

Categories and Subject Descriptors

C.2.0 [Computer - Communication Networks]: General; H.4.3 [Information Systems Applications]: Communications Applications

General Terms

Measurement

Keywords

Online Social Networks, Social Games, Facebook, Applications, Platform, Delays

IMC'09, November 4–6, 2009, Chicago, Illinois, USA.

1. INTRODUCTION

Online social networking sites command over half a billion users. Sites such as Facebook, LinkedIn, MySpace, Flickr, and Twitter allow users to seek out friends and interact with them in different ways. Besides providing basic communication capabilities (email, instant messaging, and bulletin board writing), some sites also provide other genre of applications such as sharing documents, sending virtual gifts, or gaming. The type and number of applications became unbounded, as popular sites such as Facebook and MySpace opened up their Developer Platforms, allowing external developers to create and launch their own applications. Facebook alone has over 81,000 third-party applications [13].

The open-API model on OSN sites, and the increasing popularity of these third-party applications, can have a profound impact on the Internet. OSNs act as distribution platforms that re-direct traffic between their users and third-party application servers. Each application generates additional traffic between existing users and increases workload flowing through the associated OSNs. This may be compounded by potential upsurge in the number of new users, contributing to a spiral growth. Facebook's site traffic increased by 30% in the week following the launch of its developer platform [9], while Twitter observed traffic increase by a factor of *twenty* after opening up its API [6]. Although OSN traffic volume in bytes is still relatively small compared to P2P networks, new OSN applications that allow uploading of videos would easily change the Internet traffic landscape again. These factors pose new challenges in managing traffic growth from a network infrastructure perspective, especially considering server costs are a major source of headache for large social sites such as Facebook[1]. A better understanding of OSN applications as a new workload is thus critical and overdue.

Last year, we studied **application-level user behavior** on third-party applications and their users on Facebook [8]. This paper focuses *exclusively* on the **network-level effects** of popular third-party applications on Facebook, which has over 150 million monthly *active* users (MAU)[2], and is thus an ideal candidate for just such a study. We perform a comprehensive measurement study from the perspective of third-party applications—an unstudied, interesting, rapidly growing, and diverse part of the OSN landscape. We shed light on the components of interactions between OSN users and third-party applications through the OSN platform, which is treated as a black box due to the lack of access to proprietary information about their internal design details. The key performance metric is end-to-end delay perceived by users, which depends on three main components: (a) the geographical distribution of users and their ac-

[1]http://gigaom.com/2008/05/11/the-rising-cost-of-facebook-infrastructure/
[2]http://www.insidefacebook.com/2009/01/23

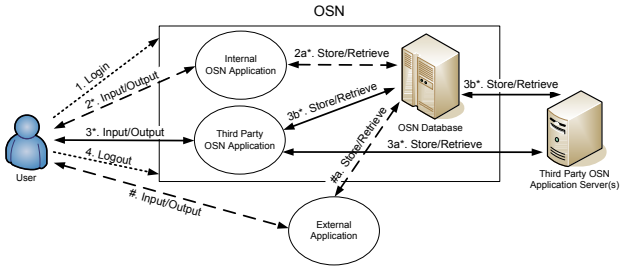

Figure 1: An OSN framework illustrating interactions between users, OSN, and external as well as third-party applications.

cess speeds, (b) processing speed and overhead of OSNs, and (c) bandwidth and processing speed of application servers. One open question we seek to answer is *whether the overheads incurred by Facebook and application developers constitute a significant portion of the end-to-end delays*. The answer is needed by both Facebook and application developers for design decisions (e.g., provisioning strategies), given that their main revenue source is advertisements, which are added into the pages rendered to users, introducing further overheads.

We use an integrated measurement methodology, which combines active tracing from client side with passive measurements at application servers, to infer interactions through Facebook and estimate various network-level as well as data processing and queuing delays in user-application interactions. Designing such a measurement study is a challenging task: unlike the typical run-of-the-mill client-server interaction study on the Web, this involves correlating observations at multiple intermediary components. To better appreciate the complexity and challenges involved, consider the OSN framework shown in Figure 1. There are three main players: the client, the OSN, and the third-party application server. The client can be an OSN user anywhere on the globe, with varying access speed and browser rendering capabilities. The OSN may serve the client's requests from a server farm or CDNs. The third-party application servers are also geographically distributed with different server capabilities. No single entity controls how users running an OSN application are served. This makes predictions of performance impact very difficult.

Our first contribution is a detailed and thorough measurement methodology that encompasses all angles, by collecting data at the application server, and by sending probes through an OSN from numerous vantage points (synthetic clients). We exploit the Facebook open developer platform by becoming one of the players and launching a set of third-party applications. We managed to amass a considerable user-base for all of the applications we launched, of which three applications have achieved more than 1 million MAU at the time of this study, which the other three achieved between 10 thousand to 500 thousand MAU. To provide a reference point, we characterize how our applications are similar to or different from the 200 most used Facebook applications. We analyze network traces, as well as application server logs, which are then correlated to perform self-validations of phenomena observed at application servers. We also conduct a large number of experiments using PlanetLab nodes to model the popular usage cases (as observed in our traces), as well as hypothetical scenarios, where users request content with varying properties. Our comprehensive data allows us to make inferences on the dynamic interactions between users and Facebook through the third-party applications.

The second contribution of the paper is the characterization of various delays involved in user and Facebook third-party application interactions—the first-of-its-kind large-scale study. We demonstrate how the proposed methodology allows us to test our hypothesis about various provisioning strategies at Facebook and application servers. In particular, we address the following questions:

• *Do external developers of popular and viral applications need exorbitantly high resources to serve content to users?* While on-server processing requirements may vary for different applications, we report our findings with regards to other constituents of delays at application servers. For example, request queuing delays are small and stable, response sizes do not vary across time-of-day, etc. For our applications, even non-exorbitant resources are sufficient to support viral application growth.

• *How much do Facebook request forwarding and response processing delays affect user experience? How is this overhead affected by the type and size of user content?* We found that Facebook processing contributes a significant portion of the overall delay experienced by a typical user accessing third-party applications, ranging from 44.4% of 1.7s total delay in case of a less-popular application to 68.8% of 2.21s total delay for the most popular one that we study. We measured how this overhead varies with different type and amount of content that needs to be rendered using PlanetLab experiments. For example, the presence of Javascript in application responses adds substantial delay as Facebook performs additional checks to filter malicious content.

• *What are the possible provisioning strategies at OSNs like Facebook? Does Facebook segregate user data according to user characteristics such as country, network or number of friends? Does it provision resources differently for third-party applications, or differentiate user requests based on properties such as geographical locations?* Given the vast amount of resources at Facebook, and especially due to the extensive use of *caching*, our results showed that Facebook serves user requests generally without preferential treatment (even by locality of request origin).

Based on our study, we conclude that Facebook is well provisioned, even for viral applications. However, significant impact exists on user experiences due to geographical location, which can be resolved by either locating the data center and application servers closer to users, and/or avoiding frequent setup and tear down of HTTP connections that incurs multiple long RTTs. The technical accuracy of this paper has been verified through high-ranking members of the Facebook Platform team through private conversations, and recent actions by the same team serve as a public acknowledgement of the issues highlighted in this paper[3]. We also provide insights aimed to improve resource utilization for application developers, and have made the data studied here publicly available to the research community[4].

The rest of the paper is structured as follows. Section 2 outlines related work, and Section 3 describes the Facebook developer platform architecture and a typical session invoking third-party applications. The proposed measurement methodology is described in Section 4. In Section 5, we present the characteristics of the workload from the perspective of the application server and analyze how load impacts the queuing/processing delays. In Section 6, we analyze delay perceived by Facebook clients based on PlanetLab experiments and infer the Facebook forwarding/processing delays. Section 7 discusses lessons learnt from our measurement study and implications of our findings on third-party application design and for Facebook. Section 8 concludes the paper.

2. RELATED WORK

Online social networks have become increasingly popular over the last few years. Accordingly, there has also been an increase in

[3]http://developers.facebook.com/news.php?blog=1story=222
[4]http://www.ece.ucdavis.edu/rubinet/data.html

research on analysis of OSNs. While some researchers have focused on graph theoretic properties of social networks [4, 7, 8], others have analyzed the usage patterns of individual networks [1, 3]. Another recent work [8] focused on the graph theoretic properties of large OSNs such as YouTube, Flickr, and Orkut. Some works have also focused on privacy and security in OSNs [5].

Retention of users *and* virality are crucial to growth and survival of large online social networks, and consequently there has been great momentum towards social third-party applications. Facebook pioneered this space by opening its Platform to third-party developers for on-site applications, and has most recently expanded itself to mobile platforms such as the iPhone through Facebook Connect. It was imperative, then, that several recent studies focused on one particular OSN, namely Facebook.

A newly published study on characterization of Facebook applications [2] uses profile crawling to explore the high-level characteristics of application users on Facebook, as well as growth patterns of applications using publicly available usage statistics from Adonomics. Another important study [3] on messaging activity inside Facebook highlights Facebook-specific characteristics such as regularities in daily and weekly traffic, and its relation to the use of Facebook by a select demographic (college students).

We previously studied a number of user behavior-related measurements on three of our highly popular gaming and non-gaming Facebook third-party applications [9]. In particular, we emphasized the distinction between user behavior on gaming versus non-gaming applications.

Note that all known previous works that analyze OSN-based applications (especially [9]) have mainly focused on studying user behavior and traffic patterns. We go one step beyond these existing studies by analyzing activity data from several third-party applications on Facebook that we have access to. Given the increasing popularity of OSN applications, it is crucial to understand their impact on the current Internet. We believe that this work is a first attempt to measure and characterize this new Internet workload, and the components of interactions between users and these third-party applications. We focus on investigating factors that impact end-user experience, thus providing insights to third-party application developers and OSNs.

3. OSN APPLICATION FRAMEWORK

The OSN application framework is depicted in Figure 1, showing the three different types of applications, i.e., *internal*, *external*, and *third-party* applications. Third-party applications are characterized by the presence of the OSN server as an intermediary for all communication (shown by the solid lines in Figure 1) between the client and the application server. A client forwards a request to the OSN server, which forwards it to the application server. The application server then sends the response back to the OSN server, which then relays to the client. The focus of our paper is to investigate network level effects of such third-party applications.

The sequence of interactions in a typical user session is shown in Figure 2. In addition to the network transmission and propagation delays, there are two major categories of delays: a) those seen at the application server, and b) those seen at the OSN server.

The application server delays consist of two components:

1. *App. Server Request Queuing Delay* ($d_q = t_5 - t_4$): d_q is the amount of time a request is queued at the network layer before being passed to the application layer for processing.

2. *App. Server Request Processing Delay* ($d_p = t_6 - t_5$): d_p is the amount of time the application server takes to generate a response for a request. Typically, a major constituent of

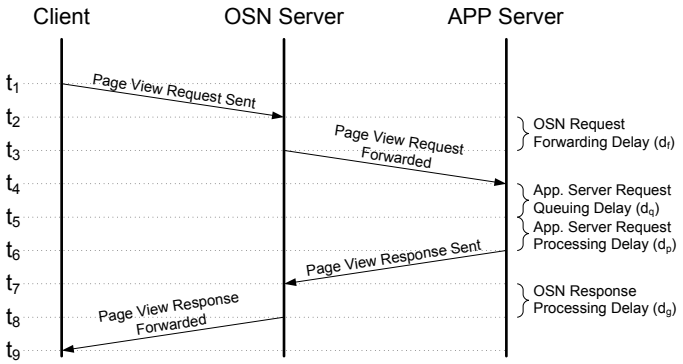

Figure 2: Sequence of interactions between Client-OSN-Application, along with delays incurred at each step.

d_p is delay incurred in executing database queries to generate dynamic HTML content. Also, note that while d_q may decrease with an increase in number of web servers, d_p will remain unaffected.

The OSN server delays include the following:

1. *OSN Server Request Forwarding Delay* ($d_f = t_3 - t_2$): d_f is the amount of time the OSN server takes to pre-process the request received from the user and convert it to be forwarded to the application server. The OSN server will typically process a request to add user-related information to it. For instance, Facebook includes a list of IDs for the user's friends in the forwarded request. The OSN server may also perform certain bookkeeping tasks and security checks. We, however, are not privy to all the tasks undertaken before a request is forwarded by an OSN.

2. *OSN Server Response Processing Delay* ($d_g = t_8 - t_7$): d_g is the amount of time the OSN server takes to post-process a response sent by the application server and convert it to a response that can be forwarded to the user. A response forwarded by the application server typically has placeholders for content that the OSN server populates by retrieving information from the OSN database. As for d_f, d_g is likely to include time taken for additional bookkeeping and security tasks that we are not privy to.

The delay components $t_2 - t_1$, $t_4 - t_3$, $t_7 - t_6$, and $t_9 - t_8$ represent connection setup overheads, and network transmission and propagation delays for interactions between the user, OSN server, and the application server.

4. MEASUREMENT METHODOLOGY

Section 1 discussed the need for data from multiple vantage points to study them, while Section 3 discussed the different intermediate stages of interaction between a client and a third-party OSN application. We developed and launched a set of third party applications on Facebook (Section 4.1), and carried out extensive passive and active measurements using these applications. Section 4.2 describes the information extracted from the network traces and application-layer logs collected at the application servers. We also designed and implemented several PlanetLab experiments to collect data from the clients' perspective (Section 4.3).

4.1 Selected Third Party Applications

We launched six Facebook applications[5] that achieved varying popularity and maturity (i.e., duration of deployment and user base). These include **Hugged**, **iSmile**, and **My Angels** that allow friends to exchange virtual hugs, smiles, or angels, respectively; as well as **Holiday Cheers** (users can send virtual *seasonal* greetings to friends), **Pound Puppies** (lets users adopt virtual pets), and **The Streets** (similar to Fighters' Club [9], where users pick fights with other users).

Table 1 reports the average number of *daily active users* (DAU), *monthly active users* (MAU), and the rank of the six applications among Facebook third-party applications [10]. All six applications are in the top 5% of Facebook applications (ranked by DAU), with Hugged, iSmile, and Holiday Cheers placed in the top 100 (out of over 57,000 applications). Hugged is both the most popular and the longest deployed, followed by iSmile. Holiday Cheers was launched half a month before Thanksgiving, and was tailored specifically towards the upcoming holiday season to achieve 'viral' growth. It attracted more than 1.4 million users in less than two months. The Streets and Pound Puppies grew much slower as can be seen by their metrics.

To gauge the extent our applications are representative of other widely used third-party applications, we installed and manually explored workflows involved in interaction with the 200 most used Facebook applications over a period of two weeks from Jan 10 to Jan 24, 2009. Our findings regarding how our six applications compare to the 200 most used Facebook applications along several dimensions are:

Application semantics: The decision to limit an application's users' interactions to their friends or non-friends impacts its virality as well as user engagement. We refer to the type of interactions allowed on an application as its semantics. In this regard, Hugged, iSmile, My Angels, and Holiday Cheers are similar to 61% of the top 200 applications in that they only allow users to interact with their Facebook friends, while The Streets and Pound Puppies (like the remaining 39%) do not impose such a restriction.

Delay requirements: Lower delays in rendering responses are crucial to enhanced user experience on social (web) applications. However, lower delays might mean higher required processing power for popular applications. We find that a majority (70%) of the top 200 applications utilize the Facebook *canvas* design (as do these six). The canvas design requires application responses to be rendered by the Facebook server, and mandates that application servers respond to a forwarded request within 8 seconds. The rest of the applications are delay insensitive.

Engagement ratio: The last column of Table 1 (ratio of DAU to MAU) shows the *engagement ratio*—an indicator of how many users are returning users. The engagement ratio represents an application's overall 'demand' by its users. A higher value implies more application visits per user. This ratio is high for The Streets and Pound Puppies as these are point-based games that require multiple visits for users to increase their scores. By clustering the top 200 applications according to their engagement ratio, we found that Hugged, Holiday Cheers, and iSmile are similar to 31.6%, The Streets and Pound Puppies are similar to 19.2%, and My Angels is similar to 13.5%, of all applications.

Hence, our six applications represent a diverse mix that is fairly representative of top Facebook applications.

4.2 Passive Measurements

The application servers receive user requests forwarded by the OSN and generate corresponding responses, thereby providing good vantage points for passive measurements. In our case, there are three categories of requests that are forwarded by the Facebook server: (1) *Page View requests (PV)*—regular requests from clients that have installed the application, (2) *Not Installed requests (NI)*—requests from clients that have not yet installed the application (these are redirected to an installation page for the application), and (3) *Inline requests (IR)*—AJAX-based queries for rendering content within the HTML pages.

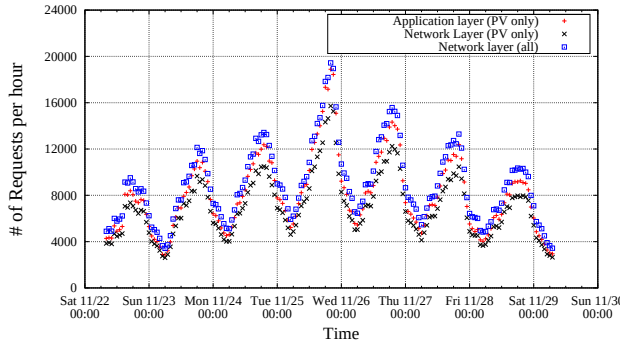

Figure 3: Total and PV requests (per hour) for Hugged. Network- and application-layer logs conform.

We have access to both network and application-layer traces gathered at the application server. Results presented here are based on traces for the period from Nov 18, 2008 to Jan 05, 2009 for Hugged, Holiday Cheers, and The Streets. The *network layer traces* log all request and response packets using *tcpdump*. The payload information contains fields specifying the ID of the request, the ID of the client from which the request originated, and the type (PV/NI/IR) of the request. The tcpdump logs have a time stamp (t_4 in Figure 2) for each request that records when a request is completely received at the network layer. Less than 2.2% of all requests in our network layer traces for Hugged, Holiday Cheers, and The Streets, are IR requests. The fraction of NI requests depends upon how fast an application grows. NI requests form 7.9% of all Hugged requests, and 28.1% of the rapidly growing Holiday Cheers requests. An NI request is re-directed to a Facebook application installation page, and does not involve database access or significant processing at application servers. Thus, in order to minimize processing overhead, the application servers record information only for PV requests, which constitute most of the workload. The *application layer traces* record time stamps when the server starts to process a user request, and when the response is written to the TCP socket. These serve as estimates of t_5 and t_6 in Figure 2, respectively.

For validation, we analyze the request arrival patterns observed at both network and application layers. Figure 3 shows the total number of requests seen at the network layer, and the number of PV request arrivals at both network and application layers, for Hugged. More than 89% of the total request arrivals are of type PV. Ideally, the number of PV requests seen at both layers should be identical. However, we see slight discrepancy due to (1) tcpdump failing to log a fraction (around 8.3%)[6] of arriving requests during high load periods (confirmed by PV requests seen at application-layer but not in tcpdump logs), and (2) a fraction of requests (around 2.6%) are dropped before being processed by the application layer, which also coincides with high load periods. We, therefore, focus on the application-layer PV request arrivals, which give a more accurate representation of the application server workload.

[5]All were developed using Ruby on Rails.

[6]This is due to tcpdump's sampling of logged packets, and does not indicate dropped packets.

Application	Launch Date	Avg. DAU	Avg. MAU	Rank	DAU/MAU
Hugged	Feb 2008	131,292	2.3M	50	0.057
iSmile	Aug 2008	120,361	2.4M	65	0.05
Holiday Cheers	Nov 2008	75,283	1.1M	97	0.068
My Angels	Aug 2008	14,016	370K	339	0.038
Pound Puppies	Jun 2008	1,545	14K	1,368	0.11
The Streets	Nov 2008	1,232	12.5K	1,520	0.099

Table 1: Application Usage Statistics. The average DAU and MAU statistics were calculated from Dec 20 to Jan 24. Application ranks show how popular the six applications are among 81,000+ Facebook third-party applications.

In addition to the PV requests initiated by clients, the application server can make certain API calls to the OSN server (see Figure 1, 3b*). Even though these API calls are not tightly synchronized with client requests, they do impact the overall user-OSN experience. An example of such an API call is a request to update a client's *newsfeed* based upon recent activity. Our application-layer traces log the total duration between making an API call and receiving a response, the sizes of the API call request, and the call response.

4.3 Active Tracing using PlanetLab

In order to collect data from multiple vantage points, we conduct a large number of experiments using PlanetLab (PL) nodes spread across the globe. We use these nodes to send active probes (in the form of synthetic requests), via the OSN, to various application servers. We model various characteristics of user PV requests to understand how they affect OSN request forwarding and response processing delays.

To diversify user locations in our experiments, we selected two PL nodes across 32 different countries, each in a different institution when possible. We launched a set of experiments on every node, twice a day (10AM and 10PM), from Dec 27, 2008 to Jan 27, 2009. We used 3 different Facebook user accounts, *User X*, *User Y*, and *User Z*, having 39, 4, and 208 Facebook friends, respectively. Based on sampled friend-list sizes of users accessing our six applications, we estimate that the average number of friends per user on Facebook is around 35, with a standard deviation of 65. Hence, User X represents the average Facebook user, while users Y and Z represent users with very low and significantly above average number of friends, respectively. From these accounts, we accessed all the six Facebook applications described in Section 4.1.

We launched our experiments with the intention to answer questions listed in Section 1. For each of these concerns, we conducted a set of experiments (coded in Java), which can be described on a high level as follows:

1: A client sends an HTTP GET request to the OSN for a given application, with a parameter specifying an experiment ID. The client logs the time stamp of request departure (T_{dep}) from the PL node and request size ($S_{client-req}$).

2: The given application's server receives the user request (forwarded by the OSN) and logs the arrival time stamp.

3: The application server responds with content specific to the experiment ID, along with the response size ($S_{app-resp}$), the request arrival, and departure time stamps.

4: The client receives the response (forwarded by the OSN), and notes the time stamp of request arrival (T_{arr}) at the PL node, as well as the response size ($S_{osn-resp}$).

A client is one of the PL nodes logged into the OSN as one of our users, and the application is one of our six applications. As shown in Figure 2, any interaction between a user and a third-party application is subject to a certain OSN Request Forwarding Delay (d_f) and a Response Processing Delay (d_g). We expected these delays to vary with $S_{client-req}$, $S_{app-resp}$, and response content. Our goal was to quantify these delays for the different experiments.

To measure the effect of d_f, we vary $S_{client-req}$ from 0 Kb to 50 Kb in our experiments. We achieve this by appending random characters into a single junk parameter with the HTTP GET request. The application response to this experiment is simply a blank page (i.e., $S_{app-resp} = 0$ Kb[7]).

Our experiments for measuring d_g keep $S_{client-req} = 0$ Kb (i.e., excluding the experiment's ID, which is appended to the requested URL), while $S_{app-resp}$ and the application response content vary. The response content and size will determine d_g. The response content for different experiments may be:

• *Non-User-Related*: The response content either has random HTML content or Javascript. Content may also be OSN-specific tags or placeholders that target non-user OSN entities, such as Facebook networks. For example, a Facebook network tag targeting a certain network ID, is replaced with the target ID's Facebook network name during Response Processing at the OSN.

• *User-Related*: The response content will contain different types of OSN-specific tags targeting OSN *users*. For our experiments, these tags may be of the following types: (1) FBML name tags that fetch target users' real names, (2) FBML profile picture tags that fetch target users' profile picture URLs, (3) FBML user status tags that display target users' Facebook status messages.

We diversify the User-Related content experiments by targeting users with differing characteristics, such as different number of OSN friends (popularity), network memberships, and geographical locations. Furthermore, since many large Web sites cache responses, our experiments randomly select target OSN entities from pools of thousands of IDs, gathered at the application servers. We also repeat our experiments (with exactly the same application responses) to gauge the effect of caching more accurately.

Since PL nodes run on virtual machines, updating hardware clocks for synchronization with NTP was not possible. This meant d_f and d_g could not be decoupled cleanly through one-way communication delays; we had to rely on round-trip delays. Calculating d_f and d_g requires elimination of the various network (propagation and transmission) delays associated with each user-to-third-party-application interaction. Propagation delays between two hosts are measured using *ping*[8]. Transmission delays are estimated using the *CapProbe* utility [12] that approximates bandwidth by employing *ping* to send/receive specific-sized packets (1,000 packets of 1,450 bytes each) to/from the target host. We use these approximations[9] to calculate d_f and d_g. Note that PL nodes are not repreentative of all OSN clients, since end-users may have a variety of connection speeds and a richer diversity of geographic locations than PL

[7]Excluding the value for $R_{app-resp}$, which is reported back to the client for calculation of d_f and d_g.

[8]Most countries only have two PL nodes. This limits our ability to estimate propagation delays for different geographical locations.

[9]CPU load on PL nodes vary considerably, impairing CapProbe's accuracy. However, we verified CapProbe's results to be within 5-10kbps of actual average data transfer rates observed on the nodes.

nodes. However, PL nodes provide the best estimate available to us at this time.

The following two sections report our results and findings from measurements at the application servers (Section 5) and PL nodes (Section 6).

5. OBSERVATIONS AND INFERENCES AT APPLICATION SERVERS

As outlined in Section 3, the client-OSN interaction process faces two types of delays at third-party application servers: request queuing delay (d_q) and request processing delay (d_p). Hence, we expect resource provisioning at application servers to have an impact on the overall experience of an OSN user. From the application developer's perspective, one important question is: ***Are exorbitantly high resources needed to ensure satisfactory user experience (e.g., low latency) in case of popular and viral applications?***

To answer this question, we need to understand the workload characteristics (such as request arrival patterns and response sizes), as seen by the application servers, and how they impact d_q and d_p. Furthermore, we need to analyze the delays involved when application servers interact with OSN servers through API calls (described in Section 4.2). In this section, we summarize the key observations and the corresponding analysis of our data. We present results based on our two most popular applications: Hugged and Holiday Cheers, and the least popular application, The Streets. As we established in Section 4.1, these applications are fairly representative of top Facebook applications, and our findings from these applications are therefore relevant to third-party applications on Facebook.

(1) The server loads, measured in terms of number of user requests, follow a diurnal pattern, and show different growth patterns depending on the popularity and seasonal nature of the applications. Already popular applications attract more new users — exhibiting preferential attachment phenomena. Figure 4 shows hourly request arrivals for three applications: Hugged, Holiday Cheers, and The Streets. We observed a diurnal pattern in the arrival process for all three. However, the request arrival rate, and its evolution over time, was vastly different due to the different popularity levels. The Hugged application provides an insight into the request arrival pattern for an extremely popular application that has a relatively mature subscription base. Even though Holiday Cheers and The Streets are examples of recently launched applications, they have widely different growth patterns. Holiday Cheers, being a seasonal application, showed viral growth during the holiday season (Thanksgiving, Christmas, and New Year). The Streets is essentially a gaming application and did not attain the same level of popularity as Holiday Cheers. This shows how targeting application content to time of the year can play a crucial role in the growth of an application.

To further understand what contributes to the busy period (high load) at the application servers, we divided the request arrivals into two categories: requests from users that have already installed the application, and new installation requests. Figure 5 shows a positive correlation between new installation requests and requests from authorized application users. This suggests that increased usage of social network applications begets more users, indicating some kind of preferential attachment model at work. This contributes to the increased load during busy periods, as there is not only a higher number of requests from regular users, but also a larger number of new application installation requests. Another indication of load that we analyzed was the request inter-arrival time at the application servers. We consider all incoming requests (irrespective of their type, or which client they belong to) to get an idea

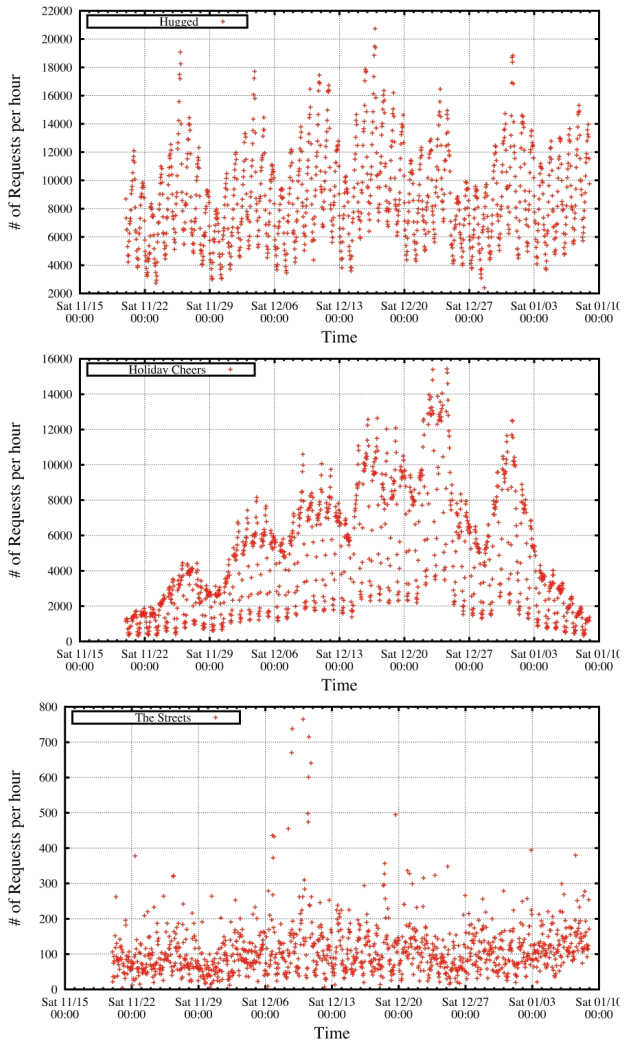

Figure 4: Per hour application-layer request arrivals for Hugged, Holiday Cheers and The Streets. All three applications show a diurnal pattern.

of the global request inter-arrival pattern seen by the application server. We found that for Hugged, the inter-arrival time follows an exponential distribution, with a mean of 0.4 seconds.

(2) Queuing delay is negligible, while processing delays correlate positively with loads and are affected by resource provisioning. We found that d_q at the application servers was less than 20ms on average, and had approximately the same distribution across applications. Figure 6 shows the distribution of d_p for various applications: Hugged had a larger d_p than the other two. Since Hugged is more mature, with a larger user base, we expect that higher server load may be the reason behind the larger processing delay.

Next, we correlated the observed load with the application server delays. While d_q was negligible, we did observe a small number of dropped requests on the application servers. The drop rate peaked during periods of high load ($\sim 2.6\%$). Figures 7 and 8 show how the load on the servers impacts d_p. For Hugged, d_p showed a positive correlation with load and shows a diurnal pattern. While the current provisioning at the application server for Hugged seems to be adequate, an increase in the popularity of the application may mandate more resources. For Holiday Cheers, we saw an initial increase in d_p, with increasing user base, during the first week after

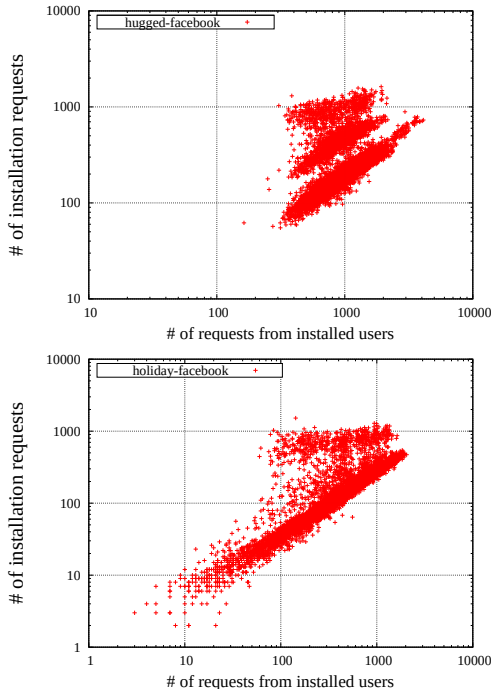

Figure 5: Installation requests vs. Requests from authorized application users for Hugged (top) and Holiday Cheers (bottom). A high correlation coefficient between these two signals some kind of preferential attachment process at work for application growth.

the application was launched ($11/15 \sim 11/22$). On Nov 23rd, application servers were upgraded: from sharing 4GB memory with two other applications, Hugged now shared 8GB with one other application, while Holiday Cheers had dedicated 4GB memory. The impact of this is clearly visible in Figure 8 for Holiday Cheers: subsequent to the upgrade, d_p no longer increased with load or across time, despite the viral growth of the application.

(3) Request response sizes remain stable across time, independent of load. For each application, we extract the response size associated with each individual request from the OSN server to the application server as follows. Using the tcpdump logs, we obtain the time stamp for the request arrival and the source IP/port from which the request was sent. We then look at the packets that were sent back to this IP/port from the application server following the request. These packets constitute the response, and contain the client data. For example, for Hugged, this data would contain previous hugs that the client has sent and received. The response sizes should be larger for more mature applications because (a) they have a larger user base, and (b) individual users may have more application-related data (such as activity history). Our results show that the average response size remains stable for the entire measurement period. The average response size for The Streets (least popular application) is the smallest (1.5–3 KB), and Hugged (most popular application) has the biggest average response size (4–5 KB). Furthermore, we expected average response sizes to *decrease* during periods with high request arrival rates (period during which more new application installation requests arrive), since new users have less (or no) activity data. Our measurement results indicate that no such relationship exists for the applications considered, showing that high request arrival periods are not *dominated* by new application installation requests.

(4) The type of interactions (i.e., API calls) from third-party application servers to OSNs affect application server delays, impacting the overall user experience. We studied the delay associated with making API calls from application servers to OSN servers (termed as the API call duration). Figure 9 shows the distribution of API call durations for various applications. The call duration for The Streets was observed to be fairly small, followed by Holiday Cheers, while Hugged observed the largest API call duration. To understand the role of API calls, we analyzed how the API call duration is affected by server load for two popular applications: Hugged and Holiday Cheers. While average API call duration for Holiday Cheers remains fairly stable across time (Figure 11), call duration for Hugged shows a remarkable increase during peak activity periods (shown by 'All API calls' in Figure 10).

To explain this difference, we compared the specific type of API calls made by these two applications. We found that Hugged and Holiday Cheers make several similar API calls[10], except for a cou-

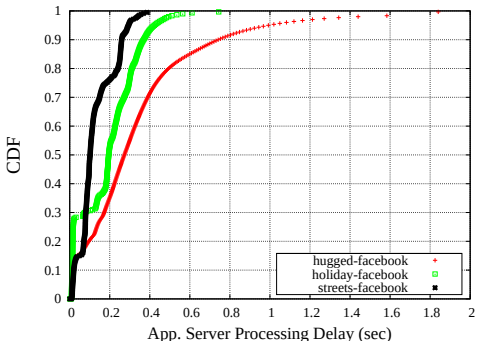

Figure 6: Distribution of application server processing delay. Hugged, being a more popular application, had a larger d_p.

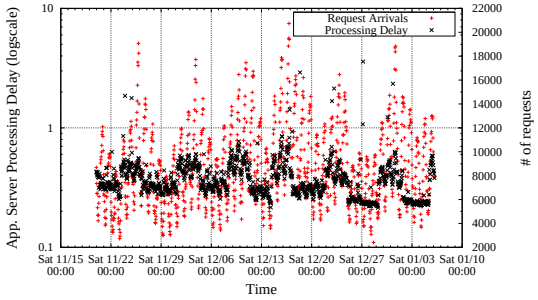

Figure 7: Variation of d_p with load for Hugged. The d_p showed a positive correlation with the load on the application server.

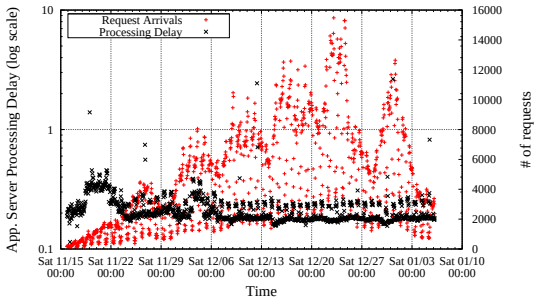

Figure 8: Variation of d_p with load for Holiday Cheers. After the server upgrade (Nov. 23), resources seemed sufficient to handle the viral growth of this application.

[10]profile.setFBML, feed.publishUserAction, notifications.send and users.getInfo. We term these as Type B calls.

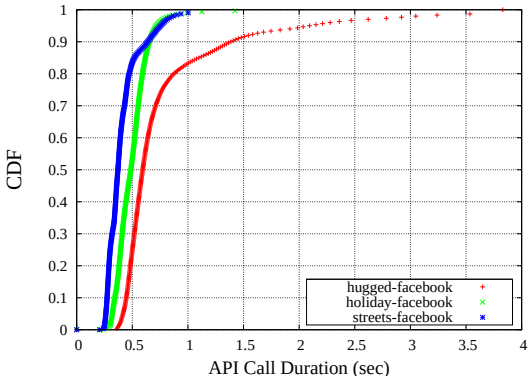

Figure 9: Distribution of API call duration. Hugged had higher call durations than the other two applications.

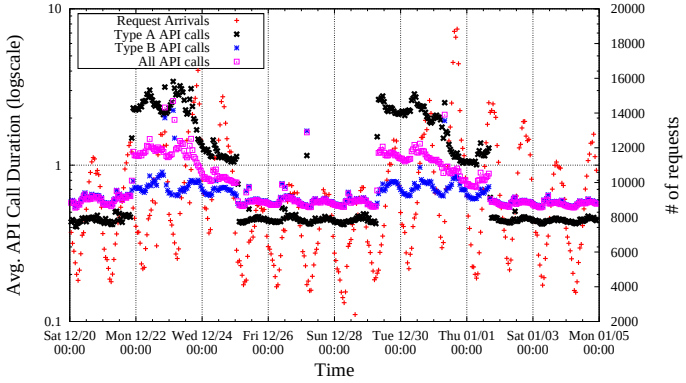

Figure 10: Variation in API call duration with load for Hugged. The Type A calls cause the overall call duration for Hugged to be extremely sensitive to load.

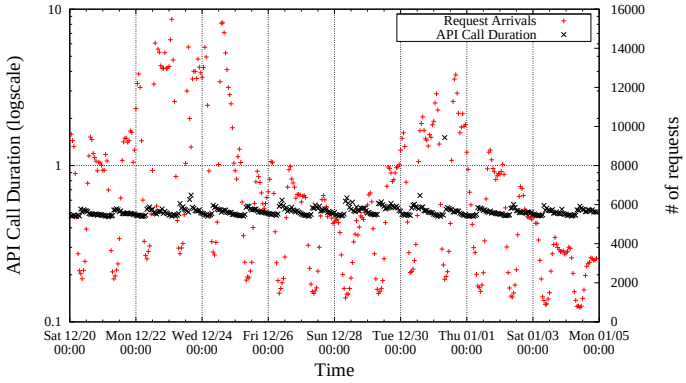

Figure 11: Variation in API call duration with load for Holiday Cheers. The call duration seemed to be fairly stable despite the viral growth of the application.

ple of additional API calls made only by Hugged[11]. We subsequently analyzed the average API call duration for Hugged, separately for the additional API calls (Type A) and the common API calls (Type B) in Figure 10. We observed that Type A calls were responsible for the surge in the average API delays during high load. However, even Type B calls show greater variation with load for Hugged as compared to Holiday Cheers. We suspect this is because of per-application or per-API-call resource budgets at the OSN, that are fully consumed by the relatively higher load for Hugged (or third-party applications in general) during peak traffic periods.

Concluding Remarks: Based on our observations, we conclude that one does not need exorbitant resources to launch and maintain an extremely popular OSN application, despite its viral growth and/or large fluctuations in seasonal usage. We do acknowledge that processing requirements may differ on a per-application basis. Yet, in our case study, a server with Dual Core Xeon 2.0ghz, 4-8GB Memory, and 100 Mbps connection speed is sufficient to serve extremely popular applications like Hugged or Holiday Cheers that attract 100-200K DAU.

6. GAUGING FACEBOOK INTERNALS

After examining results from the perspective of the application servers, we now turn to characterizing delay components from the perspective of end-users. This is motivated by the question: ***Do OSNs such as Facebook introduce significant delay overhead to impact user experience while interacting with third-party applications? If so, what factors influence these delays?***

We first discuss how OSN delays are estimated by sending active probes (synthetic requests) through Facebook from PL nodes, followed by discussions of the results.

6.1 Extracting OSN Delays

Section 3 outlines the two types of delays that are involved when a client interacts with an OSN to access third-party applications: the OSN request forwarding delay (d_f) and the OSN response processing delay (d_g). As explained in Section 4.3, our experiments involve a diverse set of PL nodes that represent the clients. Our data shows that the client requests generated in our PL experiments were forwarded by Facebook to the application servers, from 27 distinct IP addresses, all of which are located in California[12]. We analyzed the round-trip delays from nodes in different countries to these Facebook servers. The average RTT was around 170ms, with nodes farther away geographically having higher round-trip delays. Experiments from nodes in different countries showed similar d_f and d_g values (with differences of less than 10ms on average for the same experiment). However, the outliers among these nodes (most notably in Armenia, India and China) registered larger OSN delays consistently. This is mainly due to the differences in the CPU power and loads on these nodes at the time of the experiment. Since per-country results are diluted by differences in CPU load and CPU power, we do not present results based on geographical location. The variables in our experimental results include the different user accounts used, number of Facebook friends the target user has, the applications being queried, the time of day and week the experiments were performed, and the Facebook network of the targeted users. We attempt to minimize the effect of these variables; our choices of variables are explained along with the results.

Our active measurements using PL nodes provide $t_9 - t_1$ (see Figure 2). By subtracting the Application Processing Delay (d_p) and the estimated transmission and propagation delays, we get the the sum of d_f and d_g. Lacking synchronized clocks in PL, we rely on round-trip times per request rather than one-way delays, for calculating d_f and d_g. As a work-around, for requests of size 0Kb (that require 0Kb responses, except time stamp information necessary for calculating d_f and d_g), we simply remove all (known) network and application layer delays from the total time taken for a request to return (to the client), and estimate 50% of that time as d_f. We use this d_f to calculate subsequent d_g values for response

[11] notifications.sendEmail and users.hasAppPermission. We term these as Type A calls.

[12] Determined using geo-location services such as 'hostip.info'

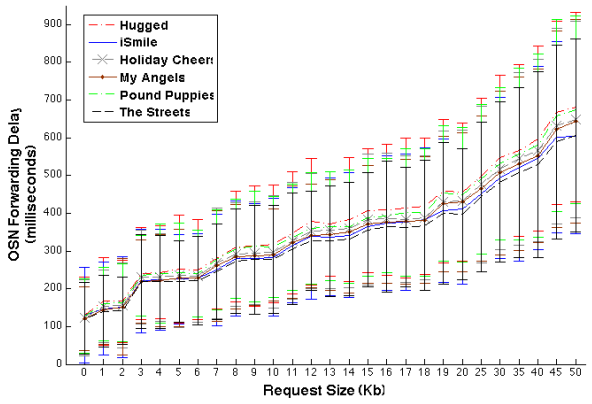

Figure 12: OSN forwarding delays (d_f) with varied request sizes (observed from Dec 27, 2008 to Jan 3, 2009). The vertical bars denote minima and maxima for the averaged measurements.

sizes greater than 0Kb.

6.2 OSN Request Forwarding Delays

Facebook, being the intermediary, must ensure that users' requests are forwarded to the application servers in a timely manner. We first address key questions concerning the forwarding delay (d_f) and summarize our observations:

(1) OSN request forwarding delays are around 130ms for user requests of size 0–1Kb (typical for the six chosen Facebook applications). We gauge d_f (as explained previously) by varying user request sizes (from 0Kb to 50Kb) to application servers. Figure 12 shows how d_f for Facebook applications varies with request size. The minimum d_f (i.e., for a 0Kb request size) was about 130ms, while requests of size 50Kb took 550–650ms to leave Facebook. Since *all* user requests are 0Kb to 1Kb in size, for the considered applications, we conclude that d_f constitutes negligible overhead.

(2) Per-application OSN request forwarding delays increase linearly with request size. Figure 12 shows that for each application, d_f increases steadily as request sizes increase. Note that for every user request to a third-party application, Facebook appends a list of user specific parameters to the URL request. Facebook then calculates a hash of the appended parameters and forwards the modified request to the application server. We are not privy to any additional tasks performed on incoming user requests, and expect that the consistent increase in d_f is due to transmission delays internal to the Facebook network. Note also in Figure 12 that plots for individual applications never overlap or cross. Even though the differences are of a few milliseconds for small request sizes, these differences amplify towards larger request sizes. The differences in delays, however, are not correlated with application popularity.

(3) Per-application OSN request forwarding delays do not vary with load (request arrival rate). We also analyzed the time variation in d_f, from Dec 27th 2008 to Jan 23rd 2009, for the various request sizes. We were unable to observe a relationship with third-party application usage and d_f. Even though d_f fluctuated between times of day and across days, the difference was not appreciable (per-request size) nor consistent (across weeks). However, we cannot relate the absence of a pattern in d_f to overall Facebook Platform usage through our measurements, as they account for only a fraction of overall traffic on Facebook.

Concluding Remarks: Forwarding delays are fairly small when request sizes are between 0–1Kb. They also do not vary consistently with application usage (load) and increase only linearly with increasing request sizes. These delays hence do not have an appreciable effect on users' interactions with third-party applications.

6.3 OSN Response Processing Delays

Being the intermediary, Facebook parses responses from third-party application servers before forwarding them to the users, which incurs additional processing delays (d_g). The extent of this delay may be affected by content type, content size, and characteristics of target entity.

A key feature of the Facebook Developer Platform is its Facebook Markup Language (FBML). It facilitates application developers through quick information rendering for application responses (through FBML tags), and it disallows use of most traditional Javascript features and other HTML content considered insecure for end-users. We study the impact of requesting (allowed) HTML, Javascript, and other FBML content on d_g through a number of experiments. In the following, we summarize key observations with corresponding details of our findings.

(1) Processing HTML content takes significantly less time than processing Javascript. We compared d_g for HTML and Javascript contents by using 200Kb of random HTML and 26Kb of Javascript content in application responses. Our results show that d_g for HTML content was significantly smaller (0.01ms/byte) compared to 0.04ms/byte for Javascript. These stark differences in d_g exist since Facebook must ensure legitimacy of every portion of the Javascript content before forwarding it to the client, Javascript being a more sophisticated language to verify than HTML. Note that we use milliseconds per byte to make a fair comparison without the bias of content size.

(2) OSN response processing delay for FBML content targeting non-user entities is unaffected by the target's popularity. It also remains consistent with time. We consider d_g for FBML tags targeting Facebook networks (non-user entities). In our experiments with FBML network tags, we compared d_g values by targeting Facebook networks with high and low popularity. We used the applications' user base to estimate the popularity of the 15,015 total Facebook networks. We compare d_g values for processing 250 FBML network tags targeting randomly chosen networks out of 3,000 most popular and 3,000 least popular Facebook networks. We found that d_g values for 250 FBML network tags were around 310ms, regardless of networks' popularity. Furthermore, this processing delay did not vary with time in measurements from Dec 27, 2008 to Jan 20, 2009. We had expected FBML network tag processing delays to be small, due to the small number (15,015) of total Facebook networks. However, we expected this delay to vary with overall Facebook Platform usage (i.e., with time of day). The lack of the latter result is plausible in the presence of a caching mechanism for FBML tags. We later confirmed this caching behavior for FBML network tags by running experiments targeting the same networks in quick succession.

A vital set of FBML tags target various content (i.e., name, profile picture, user status) for Facebook users. Given the sheer size of data Facebook must store for its users, one may expect some form of data segregation to speed up processing delays at the OSN. We expected this segregation to occur mainly along three lines: (1) by user geography, represented by a Facebook network, (2) by user popularity, represented by total mutual Facebook friendships, and (3) by type of content for the user (name, profile picture, user status). The following results compared d_g values for these user-related FBML tags along these lines of possible data segregation.

(3) FBML user tag processing delays do not vary with target users' popularity and network membership. If data on Facebook is segregated by user geography, membership in a regional Face-

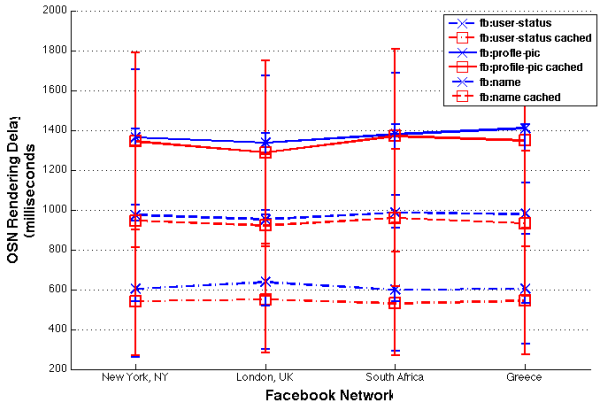

Figure 13: OSN processing delays (d_g) for 250 FBML user tags targeting users in various Facebook networks. Measurements taken from Jan 23-25, 2009 evening periods, through *User Z* on Hugged. Results were similar for other user accounts and applications as well. The vertical bars denote minima and maxima for the averaged measurements.

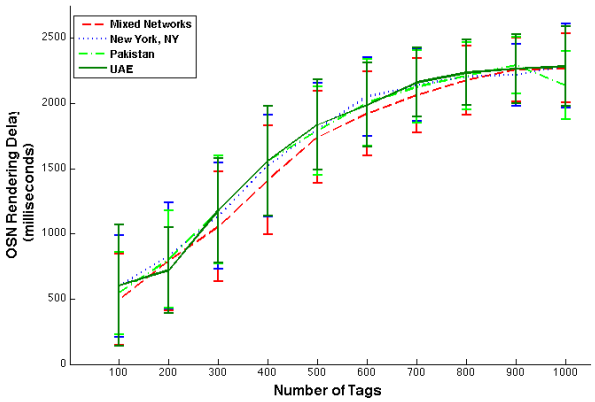

Figure 14: OSN processing delays (d_g) for different number of FBML name tags for users in networks with varied popularity. 'Mixed networks' are chosen from 5,000 random users on Hugged. The New York network is the most popular with 243,162 users on Hugged, followed by UAE with 12,187 and Pakistan with 6,213 users. Measurements were taken Jan 18-22, 2009 through *User Z* on Hugged. Results were similar for other user accounts and applications as well. The vertical bars denote minima and maxima for the averaged measurements.

book network should translate into different d_g values for users in more popular networks and for users in less popular networks. We performed experiments with 250 FBML user tags using five, variably sized Facebook regional networks, both inside and outside the U.S.[13] Our measurements for d_g (from Jan 18 – 25, 2009) did not show any appreciable variation for d_g across these networks. We repeated experiments for FBML user tags targeting Facebook users with different numbers of Facebook friends—measurements (from the six applications we study) show that the average user has 35 friends, with a standard deviation of 65. To capture different ranges of popularity of users, we target users with 15 to 50, 400 to 600, and 1,000 to 5,000 friends. Contrary to our expectations, d_g was similar (average difference of less than 15ms) across the different ranges of Facebook friends for the various FBML user tags.

(4) FBML user tag processing times vary with type of FBML tag. FBML profile picture tags take the longest, whereas FBML user status tags take the shortest times. We compared d_g values for FBML name, profile picture, and user status tags. The results segregated by target user geography (Facebook network), are shown in Figure 13. The figure shows that while FBML name tag processing delays are around $1,000$ms, delays due to FBML profile picture and FBML user status tags are about $1,350$ms and 450ms, respectively. These differences in d_g are quite stark and consistent in our measurements. To understand the reason for the high d_g values for FBML profile picture tags, we examined a sample of 100 public Facebook profiles across a university network and a regional network. We found that on average, each Facebook user has 3.64 profile pictures. Facebook blogs [14] indicate that Facebook stores *four* different sizes of each user's profile pictures. This requires a more elaborate storage mechanism for profile pictures on Facebook (as compared to, say, users' real names and dates of birth), and the high number of total profile pictures explains the higher processing delay for FBML profile picture tags. Furthermore, a user's status on Facebook must be updated more often than, say, their real names and profile pictures. For the 208 Facebook friends of *User Z*, we found that users, on average, updated their status more than twice a day. Because of this higher frequency of updates, user statuses might require a different read/write mechanism. Lacking knowledge of the internal Facebook provisioning and architecture, we surmise this difference is due to a more efficient storage/update mechanism for Facebook users' statuses.

(5) Data caching has significant effect on FBML tag processing delays. To gauge the effect of caching on d_g for FBML user tags, we repeated experiments for each FBML user tag in quick succession with the same set of target Facebook users. The result in Figure 13 (lower delays for subsequent experiments) demonstrates that caching of FBML user tags plays significant role in repeated accesses to the same target user's data, and that content fetches for our measurements are not all being performed using databases. Hence caching significantly affects our ability to estimate d_g.

Furthermore, among all the experiments reported here, we did not find appreciable (nor consistent) variation in d_g for requests made through PL nodes in different geographical locations and through users with varying popularity[14]. We thus believe Facebook does not prioritize user requests based on geography or user popularity. This is plausible considering the very heavy usage of data caching on Facebook [14].

(6) OSN response processing time increases linearly with number of FBML tags. The increased delays show no appreciable variation across third-party applications and target user characteristics. We measured variation in d_g using 100 to 1,000 FBML tags. Figure 14 shows results for FBML name tags for target users in the large New York Facebook network, as well as small regional networks such as Pakistan and UAE, alongside results for mixed networks. We observe an almost linear increase in d_g from 100 to 1,000 FBML tags, with minor (10ms) differences for different networks (that were inconsistent across time). This result suggests that Facebook does not parallelize processing of FBML tags within individual requests, which could significantly decrease d_g. Moreover, this trend is common across our applications with similar results for d_g. While the lack of appreciable variation in d_g across applications may hint at a lack of application-wise resource provisioning, caching at the OSN hampers efforts to ascertain this conjecture.

[13]The highest contributor to Facebook in terms of number of users.

[14]In terms of number of OSN friends.

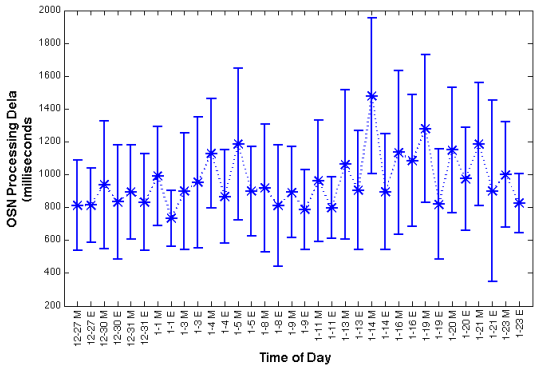

Figure 15: OSN processing delays (d_g) for *User Z* on Hugged, across time from Dec 27, 2008 to Jan 23, 2009 for 300 FBML name tags targeting users in the 'New York, NY' Facebook network. On the x-axis, E and M stand for Evening and Morning, respectively. Results were similar for other user accounts and applications. The vertical bars denote minima and maxima for the averaged measurements.

(7) OSN response processing delays vary with time of day but are not consistent with application usage (load). To examine the effect of d_g with actual application load, we compared d_g values for the 300 FBML user tags across different times of day (mornings or evenings) from Dec 27, 2008 to Jan 23, 2009. The results for FBML name tags are shown in Figure 15. Similar to d_f, we were unable to find a relation with application usage and d_g for FBML tags. Note that even though our applications are among the most popular on Facebook, their traffic may not have been substantial enough to overwhelm Facebook's resource provisioning at any particular time in our trace. Contrary to our finding for d_f, however, we observe that d_g values were higher, on average, for morning (busy) periods and lower for evening periods on the same days[15], barring Jan 3, 2009, which appears to be an anomaly whose cause cannot be ascertained, given our PL experiments utilize only a minute fraction of Facebook's resources. However, this anomaly did not occur because of CPU consumption on the PL nodes.

(8) OSN response processing delays are a significant chunk of total time per user request to third-party applications, for both realistic average workloads and hypothetical scenarios with varying size of content. Our previous experiments quantified d_f and d_g for the various hypothetical cases on Facebook. We observed that while d_f values are merely 130ms (on average), d_g values can be quite significant, and vary appreciably with type of content. To gauge d_g for real scenarios, we modeled the average workload for the most visited pages on the six Facebook applications. The average workload for each application is briefly described below:

• **Hugged, Holiday Cheers, iSmile, My Angels**: Two FBML name tags and one FBML profile picture tag for 25 Facebook users, with 80Kb of HTML content and 2Kb of Javascript. The average workload for all of these applications was similar, differing by one or two targeted users.

• **The Streets**: One FBML name tag and one FBML profile picture tag for 25 Facebook users, with 15Kb of HTML content and 0Kb of Javascript.

• **Pound Puppies**: Two FBML name tags and one FBML profile picture tag for 6 Facebook users each, with 3Kb of Javascript and 25Kb of HTML content.

[15]To make this distinction clear, we only show results for days with complete experiment results.

Using these definitions of average workload, our calculations for d_g are:

• **The Streets**: $d_g = 44.4\%$ of 1.30s total time.

• **Hugged**: $d_g = 68.8\%$ of 2.21s total time.

• **Pound Puppies**: $d_g = 59.9\%$ of 1.77s total time.

These experiments do not involve database accesses at the application servers, hence d_p is negligible (around 4ms). However, the results confirm that indirection via OSNs imposes a significant overhead for user interactions with third-party applications. This fact is further demonstrated in Table 2, which shows the fraction of total time consumed by d_g for certain hypothetical cases for various types of content.

Concluding Remarks: Delays across OSNs (as in the case of Facebook) can dominate the overall latency experienced by users interacting with third-party applications. For Facebook, we found that the OSN forwarding and processing delays are fairly consistent across applications and do not depend on application usage (load). They are affected by the type of content accessed, but not by entities targeted in FBML tags. Caching of data helps stabilize the overall user-perceived delay in their interactions with third-party applications. Even with data caching, however, the stark difference in d_g for different types of FBML tags exists. This could be accounted for by Facebook's internal network delays in accessing different parts of its data center(s), since the sheer data volume at Facebook might be managed through data segregation. This becomes more evident considering different access and storage requirements for various types of data (e.g., profile pictures and user statuses).

7. DISCUSSION

Our measurement study of interactions between users and third-party applications through Facebook has provided us with insights that can be useful for third-party application developers and OSNs.

Insights for Application Developers

Our measurement study revealed some take-home messages from the perspective of third-party application developers. We saw well-defined trends in the request arrival patterns with diurnal, weekly, and seasonal peak activity periods. Application servers should be well-provisioned to deal with such expected surges in activity. The consistency in activity surges allows developers to effectively utilize resources through cloud computing[16]. We also found that request inter-arrival times for a single application were exponentially distributed. This information can be used to provision network layer buffers at the application server.

Our results show that the number and the type of FBML tags can affect the OSN response processing delay. Application developers can keep this in mind to balance content they wish to serve and minimize resulting delays perceived by users. Also, our PL measurements showed that for certain regions (especially those with inferior bandwidth), network delays can have a significant impact on user experience. Application developers can choose to optimize their content based on networking resources available to the client.

Furthermore, we noticed that different API calls made by application servers can take different amounts of time to process, and some API call durations can appreciate significantly with load. This behavior was evident for Hugged, which sees longer API call durations during high user activity periods. This hints at some form of resource quotas (per-application or per-API-call) at Facebook servers. A possible solution to avoid higher API call delays is to queue API calls during high activity periods, and to empty these

[16]Cloud computing services such as Amazon EC2 [11] have become easily accessible.

Content Processed	Avg. Total Time						Avg. Processing Delay					
HTML (200Kb)	3.63s						49.8%					
Javascript (26Kb)	1.99s						66.3%					
	Number of Tags						Number of Tags					
FBML Tags	**25**	**50**	**100**	**150**	**200**	**250**	**25**	**50**	**100**	**150**	**200**	**250**
Name	1.1s	1.2s	1.35s	1.54s	1.63s	1.81s	36.4%	40.5%	47.4%	53.4%	56.4%	60.6%
Profile Picture	1.11s	1.25s	1.51s	1.75s	1.97s	2.3s	35.8%	43.3%	52.8%	58.9%	63.7%	68.6%
User Status	1.05s	1.10s	1.14s	1.23s	1.29s	1.38s	32.5%	35.6%	37.8%	42.3%	45.1%	48.6%
Network Link	1.02s	1.03s	1.05s	1.07s	1.09s	1.1s	31.8%	31.2%	32.6%	32.7%	33.5%	33.9%

Table 2: Response processing delays as fractions of average total time for user requests for different types of content. The HTML content was generated randomly. The reported delays have d_p=4ms.

queues during low user activity periods. Our experiments did not control the rate and schedule of API calls.

Insights for OSNs

Our findings, especially those from the PL experiments, allow us to provide some useful insights for OSNs in general, and Facebook in particular. The following discusses a number of issues and potential areas of improvement for OSNs.

According to recent statistics[17], Facebook has grown, and continues to grow, outside the U.S. at a rapid pace. However, a closer look at the statistics shows that regions farther away from Facebook's server locations in California, such as India and Pakistan, have experienced less than stellar growth. Reasons for slow growth in far-off regions may be attributed to a number of hard-to-measure factors, such as social and technological development. However, one OSN architecture-related reason can be the significant round-trip latencies experienced whilst contacting Facebook from regions far away from the U.S. Since latencies affect user experience even more than bandwidth availability, diversification of data center locations outside regions of traditional growth can benefit OSNs such as Facebook. Note that the overhead introduced by latencies in user interaction with third-party applications will still be significant, unless application servers, too, move closer to the newer data centers, depending on the application's targeted user base.

Allocating data centers closer to far-off regions may improve user experience, but also introduces the complication of replicating data to these data centers. Assuming users in say, India, are more likely to access content generated by users within India[18], the choice of a *single* master[19] data center will not be the most optimal. This is currently the case for Facebook, which has a slave data center in Virginia, and the master data center in California [14]. Note that occurrence of this 'social aspect' in data accesses is still novel to academia as well as the industry.

Each user request to a third-party application results in two HTTP connection setups: 1) between the client and OSN, and 2) between the OSN and application servers. In cases where round-trip propagation delays are high (e.g., far-away clients contacting OSN servers), protocol handshakes involved can contribute significantly to the overall delay experienced by users. This can be remedied through persistent HTTP connections between the application server and OSN. The OSN can multiplex user requests within a few HTTP connections by mandating presence of certain 'response identifiers' in application responses.

Moreover, our results for OSN response processing delays indicate that OSN delays grow in lock-step with the size of content. For the Facebook Developer Platform, the main cause of concern is the high processing delays for FBML user tags, which are pervasively used in third-party applications. Our results in Table 2 (and corresponding average workload measurements) suggest that an OSN could drastically improve user experience by parallelizing OSN tag (e.g., FBML for Facebook) processing within individual application responses. This could, however, turn into a DoS vulnerability for the OSN, and must be implemented with checks in place to avoid such attacks.

[17]http://www.insidefacebook.com/2009/01/23/

[18]Since, users are likely to have more online friends from the same geographical region.

[19]A master data center has the sole authority for modifying data. Slave data centers may only serve content without modification.

8. CONCLUSION

The launch of APIs for OSN applications has brought about thousands of third-party OSN applications. Facebook alone makes over 57,000 applications available to more than 150 million users. Little information is available about the different stages of information flow between users and third-party OSN applications, and the role of the underlying OSN. We described an elaborate measurement methodology, collect data from multiple vantage points, and reconstruct the details of typical client-OSN-third party application interactions. We investigated the different sources of delay and their potential impact on user experience.

Our study showed that even for viral applications, non-exorbitant resource provisioning can bound application server delays. OSNs do introduce significant overhead to user and third-party application interaction. Even with caching, OSNs exhibit different processing delays for different types of content although these delays are relatively stable with application usage (load) and are unaffected by targeted entities (such as in FBML tags). Our findings emphasize the impact of Facebook's processing delays on user interactions with third-party applications. These findings have been verified by Facebook.

9. REFERENCES

[1] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon. I tube, you tube, everybody tubes: Analyzing the world's largest user generated content video system. In *Proc. Internet Measurement Conference (IMC)*, 2007.

[2] M. Gjoka, M. Sirivianos, A. Markopoulou, and X. Yang. Poking Facebook: Characterization of OSN Applications. In *Workshop on Online Social Networks*, 2008.

[3] S. Golder, D. Wilkinson, and B. Huberman. Rhythms of Social Interaction: Messaging within a Massive Online Network. In *International Conference on Communities and Technologies*, 2007.

[4] M. Granovetter. The strength of weak ties. *American Journal of Sociology*, 1973.

[5] R. Gross. Information revelation and privacy in online social networks. In *WPES '05: Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, 2005.

[6] B. Krishnamurthy, P. Gill, and M. Arlitt. A few chirps about twitter. In *Workshop on Online Social Networks*, 2008.

[7] S. Milgram. The small world problem. *Psychology Today*, 1967.

[8] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and Analysis of Online Social Networks. In *Proc. Internet Measurement Conference (IMC)*, 2007.

[9] A. Nazir, S. Raza, and C.-N. Chuah. Unveiling facebook: A measurement study of social network based applications. In *Proc. Internet Measurement Conference (IMC)*, 2008.

[10] Adonomics. http://www.adonomics.com, May 2008.

[11] Amazon ec2. http://aws.amazon.com/ec2/, 2007.

[12] Capprobe. http://www.cs.ucla.edu/NRL/CapProbe/, 2006.

[13] Developer analytics. http://www.developeranalytics.com/, Apr 2008.

[14] Engineering @ facebook blog. http://www.facebook.com/eblog, 2008.