SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •
Special Focus on Convergence Communications

April 2014, Vol. 57 042309:1–042309:11
doi: 10.1007/s11432-014-5084-x

PAIDD: a hybrid P2P-based architecture for improving data distribution in social networks

SHUANG Kai* & SU Sen

State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876*, China*

Received October 17, 2013; accepted December 21, 2013

Abstract Rapid growth in social networks (SNs) presents a unique scalability challenge for SN operators because of the massive amounts of data distribution among large number of concurrent online users. A request from any user may trigger hundreds of server activities to generate a customized page and which has already become a huge burden. Based on the theoretical model and analytical study considering realistic network scenarios, this article proposes a hybrid P2P-based architecture called PAIDD. PAIDD fulfills effective data distribution primarily through P2P connectivity and social graph among users but with the help of central servers. To increase system efficiency, PAIDD performs optimized content prefetching based on social interactions among users. PAIDD chooses interaction as the criteria because user's interaction graph is measured to be much smaller than the social graph. Our experiments confirm that PAIDD ensures satisfactory user experience without incurring extensive overhead on clients' network. More importantly, PAIDD can effectively achieve one order of magnitude of load reduction at central servers.

Keywords social network, data distribution, PAIDD, hybrid P2P-based, interaction-oriented

Citation Shuang K, Su S. PAIDD: a hybrid P2P-based architecture for improving data distribution in social networks. Sci China Inf Sci, 2014, 57: 042309(11), doi: 10.1007/s11432-014-5084-x

1 Introduction

Social networks (SNs), with both global reach, such as Facebook, Orkut, and Flickr, and regional ones like Renren[1)] from China, Hyves[2)] in the Netherlands, and Vkontakte[3)] from Russia, experienced rapid growth in recent years. Such rapid growth presents a unique scalability challenge that draws great research interest from both industry and academic communities [1–5]. The sites of SNs are usually "sticky" to their users, which results in a large number of concurrent online users. For example, checkfacebook site reports that as high as 57% of Facebook users are online, out of 120 million in the US alone[4)]. Additionally, users are very active on SN sites. A request from any user may trigger the server to gather hundreds of latest

*Corresponding author (email: shuangk@bupt.edu.cn)

1) RenRen, http://www.renren.com.
2) Hyves, http://www.hyves.nl.
3) Vkontakte, http://vkontakte.ru.
4) Facebook data tracker, July 15, 2010. http://www.checkfacebook.com.

activities updated by the user's friends, generate a dynamic and customized page for this particular user, and then deliver the page within reasonable time.

Conventional solutions to scale SN sites include extending the server-side infrastructure for data storage and building cache layer for data accessing. The high degree of interconnection among users makes partitions of user data inefficient. Some SN sites keep user data normalized and randomly distribute them amongst thousands of infrastructure servers, and then deploy distributed cache to ensure that most of the active data is kept in memory for fast retrieving. Such solutions require significant investments on the infrastructure and engineering, which becomes a huge burden for SN operators.

P2P technology has been successfully used to address scalability challenges arising from applications relying on centralized servers, such as Skype and Bittorrent. However, we find that it is non-trivial to make P2P work for SNs. It has to be customized based on unique social constraints: Page Load Latency. Contents retrieved through P2P channel have to provide similar experience as that of centralized solution. Since a page reload often triggers a large number of back-end queries, it is a great challenge to perform these queries on-demand in P2P and aggregate all the results with a short delay, even though latency optimization is well studied for DHT-like overlays [6,7]. Besides the above unique problems, using P2P to serve SNs must practically consider known P2P issues such as peer reachability and reliability.

In this study, we analyze the possible P2P-based solutions that address the scalability issues for SNs, and present our design of hybrid P2P-based architecture for improving data distribution in social networks (PAIDD).

The rest of the article is organized as follows. After discussing some related works in Section 2, we introduce our design assumptions and performance metrics in Section 3. We present our PAIDD design and analytical model in Section 4, and present the evaluation results in Section 5. Finally, we give our conclusions in Section 6.

2 Related works

There are efforts in literature to address the scalability of SNs, from the aspect of both central infrastructure efficiency and P2P offload. Regarding improving the performance of central infrastructure, studies focus on efficient resource caching and data storage distribution [1–3,8,9]. Aron et al., proposed a scalable content-aware request distribution, which can cache all of the users' data to improve the central servers' performance [3]. Though it's a better way, it still relies completely on the central infrastructure. Operators usually have to spend plenty of money on the infrastructure and engineering. Clearly, it is not the best way.

As for P2P offloading, it is known that P2P has been investigated as an alternative solution for SNs. Buchegger et al. analyzed the opportunities and challenges of using P2P architecture for SN sites [10]. The authors pointed out that Internet connectivity requirements is one of the most important limitations of such architecture. Abbas et al. proposed a completely decentralized P2P system for both social network operation and the corresponding data storage [11]. The solution is based on Tribler, a Bittorrent based client. It only supports limited functionalities compared to an operational SN site. Compared with previous solutions, the PAIDD combines capacity from the central servers and peers. We take great effort to address the peers' connectivity problem in P2P-based design.

3 Design assumptions and performance metrics

First, we introduce the assumptions in our design and define the performance metrics.

3.1 Design choices and assumptions

In PAIDD's system design, a cluster of central servers are required, named server S. It can be reached with low latency by every node, similar to the central servers of existing SN sites. A node is a machine operated by a user (the terms user, client, peer, and node are used interchangeably in this article). We

Table 1 Top 10 read/write activities from measurement studies [13]

Type of activities	Requests (%)	Users (%)	Traffic (%)
Browse profiles	19.0	17.9	20.0
Browse user posts	18.7	15.9	15.5
Browse photo	15.5	7.3	6.5
Browse homepage	11.8	16.9	21.8
Browse photo albums	8.9	7.3	13.1
Manage invitations	1.1	1.5	0.8
Write posts	1.0	2.1	0.6
Profile editing	0.9	1.2	2.1
Write comments	0.5	0.8	0.4
Join/Leave groups	0.4	0.5	0.2

assume that server S has sufficient bandwidth and computation power as needed. As the foundation of our P2P-based design, we require a presence service (co-located with server S), which shows the online status of users. In addition to online status, the presence service provides some other vital network information to the client, such as friend's IP, network conditions, and latest activity ID.

PAIDD is designed to effectively reduce the workload on servers without impairing user experience. Our hybrid P2P-based model focuses on the user activities that pose heavy workload on S. The studies on measuring and characterizing user activities of several SNs in detail are mentioned in [12,13]. Table 1 summarizes the top 10 activities from one of the measurement study [13]. We do not intend to provide a P2P-based solution for all user activities in SNs, but the most frequent one. These activities cover more than 80% of the server workload.

3.2 Performance metrics

The primary performance metric, server load ratio, is defined as the ratio of the number of messages received and delivered by S with and without PAIDD system. These messages, called client activity messages, contain the information of user activities on SNs. One message represents a discrete activity event such as browsing, updating profiles, blogging, posting comment, etc. We use client activity message as the basic measurement unit for server workload. For example, a post publication will result in one message being sent to S. If N friends receive this post through S, then all activity messages are added to the workload of S.

Another important performance goal is user experience, measured by page load latency. This is achieved by retrieving contents from the central server directly at the right scenarios. We emphasize that page load latency is different with the delay in content update notification, which is defined as how soon one is notified that his/her friend has updates. We define the update notifications as condensed format of the published full content, serving as a summary with a small size. The corresponding full content is named update content.

The update content is not broadcasted by default in most SN networks. It is available upon user requests. Browsing these contents is a major activity in SN sites, e.g., photos. To evaluate the effectiveness of our design for user experience, we use the hit-ratio and wasted bandwidth as the performance metric. As described later, we employ a fine-tuned prefetching mechanism to ensure low page load latency. Hit-ratio is the percentage of user requests being served by prefetched contents. Wasted bandwidth is the traffic incurred by prefetched contents that are not viewed by the user.

4 PAIDD design and analysis

PAIDD consists of two major components: P2P-based data delivery and interaction-oriented content prefetching. The first component deals with user presence, P2P connectivity, and social graph related

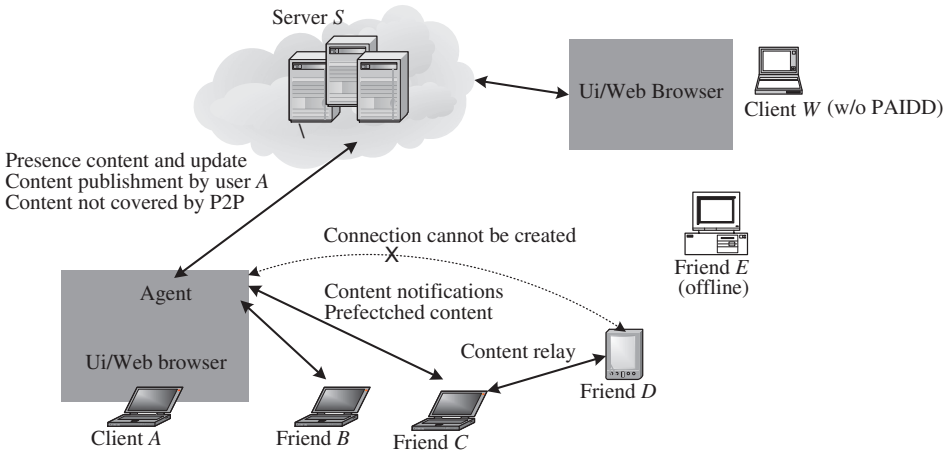

Figure 1 PAIDD architecture.

peering. The prefetching component addresses the user interaction behavior learning, prefetching heuristic, and data prefetching.

4.1 Architecture overview

In our design, all user generated contents are sent to S for two purposes, persistent storage and system failover. S stores, distributes, and caches user contents in similar fashion as the traditional design, such that the system can be compatible with users who have not adopted PAIDD.

The overall architecture is illustrated in Figure 1. We have the central S, which also provides the presence service. A client (user W in Figure 1) without PAIDD always communicates with S directly. A client with PAIDD installed, such as user A, has a local PAIDD agent that transparently processes user's request, retrieves, and presents contents to users as if the user is visiting the SN site directly. The PAIDD agent can be a browser plug-in, or part of a stand-alone application offered by SN sites.

After login, user A registers with the presence server and obtains the network status of online friends. When user A publishes contents, A's local PAIDD agent follows the content publishing process, such that the updated contents are archived by S and update notifications are sent to A's online friends either through direct notification pushing or receiver-driven mechanism.

After receiving an update notification, A's friend D, may decide to read the full update content. Such interaction behaviors are recorded and analyzed by D's local PAIDD agent, serving as the input to determine whether A's future published update contents should be fetched automatically.

4.2 PAIDD agent and P2P communications

A PAIDD agent maintains a list of the user's online friends, obtained from the presence server. The presence server returns the following information for each online peer.

$$\langle \text{IP},\ \text{Port},\ \text{Network Type},\ \text{Content ID},\ \text{Public Key} \rangle.$$

The first three elements, IP, port, and network type, describe a peer's network information. Content ID is the latest ID S assigns to the content published by the user visible to his/her friends. This ID is generated sequentially for each user. Network type is the type of NAT gateway measured by the PAIDD agent with known techniques [14,15]. A peer's network type is classified into six different types based on the NAT classification, shown in Table 2. It is important for PAIDD agents to report their types for that it is reported that these restrictive NAT types are very dominant on the Internet [16,17]. Distributing NAT information effectively improves P2P communication efficiency since a peer can avoid peers with unreachable NAT types. For example, a peer with IP-restricted NAT gateway cannot directly communicate with a peer with symmetric NAT gateway.

Table 2 Nat gateway categorization and reachability among them [15]

NAT category	NAT type	Percentage	Can communicate with
Open	1	9.5%	All NAT types
Full-cone	2	14.7%	All NAT types but 6
IP-restricted	3	1.7%	All NAT types but 6
Port-restricted	4	39.2%	NAT types 1,2,3,4
Symmetric	5	24.0%	NAT types 1,2,3
UDP-disabled	6	11.0%	NAT type 1

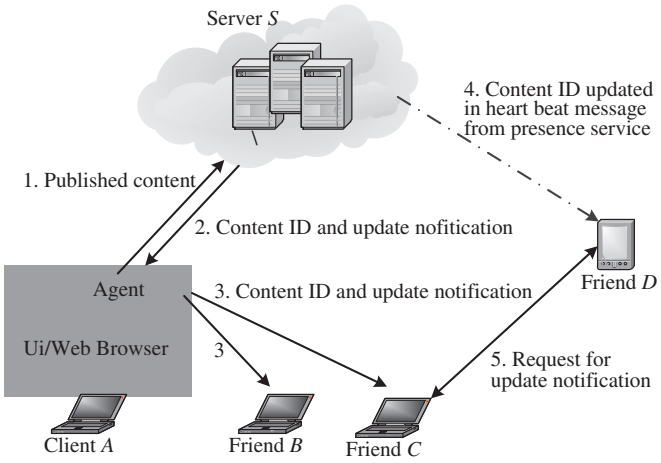

Figure 2 P2P-based delivery of update notifications.

4.3 P2P-based data delivery

A user's profile usually consists of a list of update notifications from all friends. The distribution of update notifications are covered in three aspects: content publishing, active pushing, and receiver-driven retrieval. Figure 2 illustrates the detailed process.

1) Content Publishing: when user A publishes a piece of content, the local agent sends the full content to S. S returns the generated update notification along with the incremented content ID assigned to this content (Steps 1 and 2 in Figure 2). For user generated contents that are not visible to friends, the content ID can be simply set to a reserved value, e.g., zero. S also updates the Content ID field of user A's record in the presence server. Our design keeps update notification generation as part of server side logic, such that content aggregation can be done more efficiently.

2) Active Pushing: upon receiving a valid content ID, A's local PAIDD agent iterates through A's online friends and actively pushes the content ID and the update notification to a subset of them (Step 3 in Figure 2). The subset is selected based on the following criteria: a) user A can reach these friends according to Table 2; b) the rate of message transmission remains below a preset or calculated bandwidth threshold; and c) the active pushing lasts no longer than the heart-beat interval to the presence server. When a peer receives the pushed content ID and update notification from a friend, it stores the notification locally indexed by the user and content ID.

3) Receiver-driven Retrieval: for a peer (D) not selected by user A for active pushing, it will soon learn about the availability of this update in the heart-beat messages from the presence server as the content ID of user A changes. Peer D then starts to obtain this update notification through a receiver-driven mechanism called "two-hop" relay, in other words, relay by "friend's friend".

Peer D selects several of A's online friends to request this update notification from their local cache, shown as Step 5 in Figure 2. Direct request to A is a less preferable option. Similar peer selection heuristic can be used as mentioned above. Maintaining "friend's friend" does not have to go through the presence server. Peers can exchange their online friends list partially or fully with the heart-beat

message for maintaining the NAT traverse capabilities [14]. Additionally, a shared friend can be used to bridge the communication between peers. Our experiment on our incomplete Facebook data shows that 78% of friend pairs share at least one online friend. The "two-hop" relay is quite effective to bypass communication constraint caused by NAT.

4.4 Interaction-oriented content prefetching

After update notifications have been delivered and presented to the user, the user may click on it to request the full update content for further reading or viewing. To serve the update contents with low page load latency, a PAIDD agent tries to prefetch the contents, such that they can be served instantaneously without accessing the central server or other peers.

The content prefetching consists of three parts: interaction preference learning, prefetching decision calculation, and P2P-based data retrieval. We will discuss the prefetching decision heuristic in detail in Section 4.5.

The PAIDD agent records a user's interaction history with his/her friends. An interaction record contains the following fields, ⟨Timestamp, Source User ID, Prefetched, Key Words⟩.

It basically records three Ws: when, who, and what, represented, respectively, by timestamp, source user ID, and key words. The key words currently just record the category of the content, e.g., photo album, photo, URL, blog, etc. The "Prefetched" field is a boolean value stating whether the content is successfully prefetched. In addition to these per-interaction records, a summary of interaction history is also maintained for each friend. A summary record includes three accumulated numbers, the number of prefetched content, user clicks, and prefetched content hits.

The interaction histories are periodically pushed to the server for persistent storage. The copy on the server can be accessed by users migrating to another machine or location. Also, the interaction history can be computed by server offline based on web access log, helping a new PAIDD user to skip the initial learning period. During the learning period, the miss rate is high, which consequently generates higher workload on the central server. Prefetching the full content from peers follows a similar "two-hop" delivery mechanism.

4.5 Prefetching decision calculation

It is a tradeoff between improving prefetching hit-ratio and reducing the incurred overhead (wasted bandwidth). A straightforward heuristic is to prefetch contents based on the social graph of users. However, latest study on SNs [12] indicates that users' interaction circle is much smaller than its full social circle. Based on such discoveries, we propose to leverage the interaction graph to balance the above two prefetching goals. We call it Interaction-Oriented Pre-fetching.

We assume that user n has M_n friends indexed from 0 to $M_n - 1$. The interaction to friend m is $c_{m,n}$, and its proportion among n's friends $p_{m,n} = c_{m,n} / \sum_{m=0}^{M_n-1} c_{m,n}$. The number of updates from friend m is $a_{m,n}$, which presents the activity of friend. The degree of attention to friend m is defined as $d_{m,n} = c_{m,n}/a_{m,n}$. The prefetching decision to friend m is $s_{m,n}(s_{m,n} \in \{0, 1\})$. $s_{m,n} = 1$ denotes that user n prefetches all the updates from friend m while $s_{m,n} = 0$ denotes that user n does not prefetch contents from friend m. The value of $s_{m,n}$ may change as $d_{m,n}$ changes over time.

The prefetching hit-ratio of user n is

$$h_n = \sum_{m=0}^{M_n-1} p_{m,n} \cdot s_{m,n} = \sum_{m=0}^{M_n-1} \frac{c_{m,n} \cdot s_{m,n}}{\sum_{m=0}^{M_n-1} c_{m,n}}. \tag{1}$$

The number of wasted (i.e., un-browsed) prefetching updates from friend m is $a_{m,n} - c_{m,n}$. The normalized wasted bandwidth of user n is

$$w_n = \frac{\sum_{m=0}^{M_n-1} [(a_i - c_i) \cdot s_{m,n}]}{\sum_{m=0}^{M_n-1} (a_i - c_i)}. \tag{2}$$

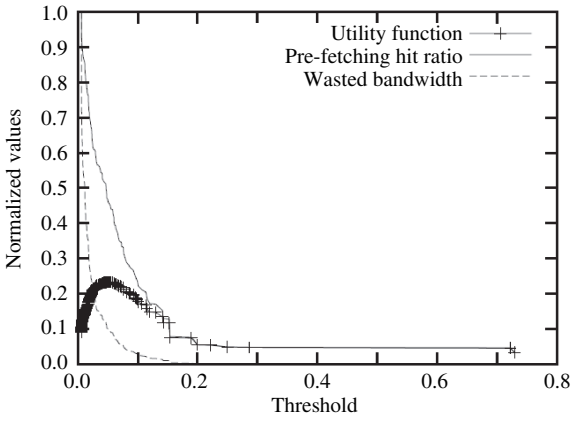

Figure 3 The optimized threshold of a node.

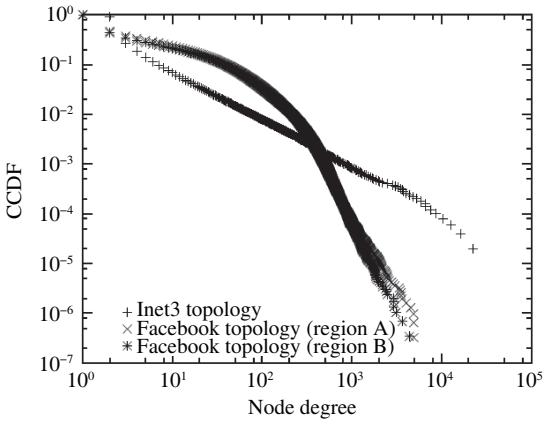

Figure 4 Node degree distribution of Inet3 and Facebook topologies.

The goal is to find a set of $\{s_{m,n}\}$ to maximize prefetching hit-ratio and lowering wasted bandwidth. We reduce the two goal optimization problems to maximize the utility function $U(n)$. With P2P, it is acceptable to have some amount of overhead on the peers. We use control variable b to reflect relative importance of wasted bandwidth compared to hit-ratio.

$$U(n) = \frac{h_n}{1 + b \cdot w_n} = \frac{\sum_{m=0}^{M_n-1} \frac{c_{m,n} \cdot s_{m,n}}{\sum_{m=0}^{M_n-1} c_{m,n}}}{1 + b \cdot \frac{\sum_{m=0}^{M_n-1} [(a_i - c_i) \cdot s_{m,n}]}{\sum_{m=0}^{M_n-1} (a_i - c_i)}}. \tag{3}$$

The complexity of finding all optimized $s_{m,n}$ via exhaustive search is $\mathrm{O}(m \cdot 2^n)$. We develop a greedy algorithm to solve the problem. The optimization problem is equivalent to finding the optimized threshold th_n for user n. If $d_{m,n} > \mathrm{th}_n$, then $s_{m,n} = 1$, otherwise, $s_{m,n} = 0$. We can find the optimized th_n from a limited number of concrete candidates $\{d_{m,n}\}$.The complexity of the greedy algorithm is $\mathrm{O}(m \cdot n)$.

Figure 3 demonstrates the above optimizing problem. User m is selected from the real Facebook data set as a typical example. The figure indicates that $U(m)$ achieves maximizing when $\mathrm{th}_n = 0.052$ ($b = 10$).

To implement the greedy algorithm, we need to collect two information from each node m, the amount of visible activities or update notifications from each friend n (i.e., $\{a_{m,n}\}$) and corresponding clicking history for friend n (i.e., $\{c_{m,n}\}$). Then, each node can calculate which friend's data generates the maximum value for function $U(n)$ based on (3), and uses that friend's attention degree ($d_{m,n}$) as the threshold for prefetching. We can see that it is not complicated to obtain these required information from PAIDD's local agent.

5 Performance evaluation

We study the performance of PAIDD's design from two perspectives. We first demonstrate the effectiveness of P2P-based data delivery. Then, we show that the interaction-oriented prefetching achieves satisfactory user experience without generating significant overhead.

5.1 Experiment setup

1) Topologies: in our experiment, we employ two sets of topologies, one is the artificial topology which is generated by Inet3.0 (Inet3) [18] and the other is the fixed size topology that crawled from Facebook. The artificial topologies are used to study the performance of PAIDD as the social network grows. We use Inet3 topologies with 5 000 to 50 000 nodes. Figure 4 shows the power-law node degree distribution of the Facebook topologies and one Inet3 topology with 50 000 nodes.

The two Facebook data sets are based on crawled user data filtered to two geographic regions of Facebook, with 3.1 and 2.9 million users, respectively [12]. To evaluate the performance of our interaction-

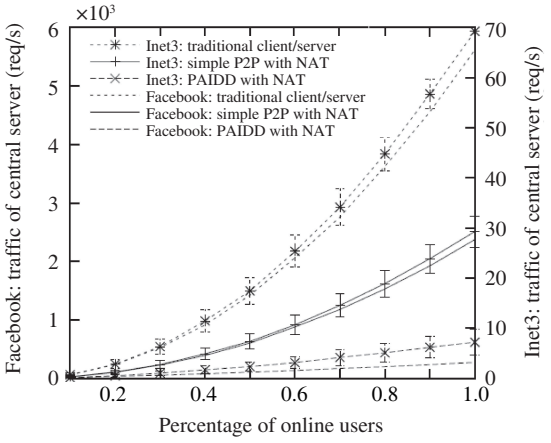

Figure 5 Load of central node on Facebook topology and Inet3 topology with 50 000 nodes.

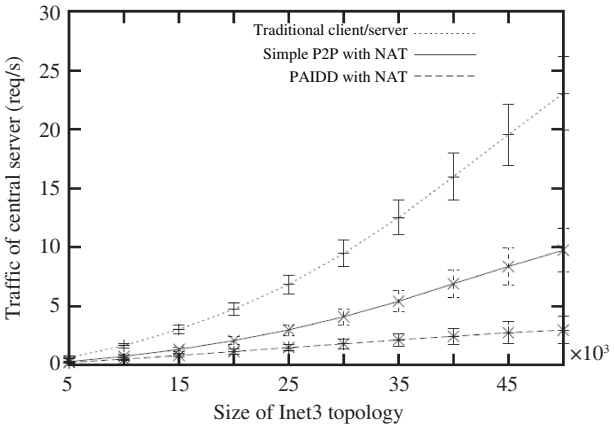

Figure 6 Load of central node on as Inet3 topology grows, with 57.18% of concurrent online users.

oriented prefetching scheme, we also use the crawled interaction data based on the same two sets of Facebook users described above.

2) Network Settings: to simulate realistic network reachability settings in our study, we adopt the data set collected from real deployment of P2P living streaming system [19]. We randomly assign NAT types to nodes based on the distribution available in Table 2.

Besides realistic social degrees and network types, we also need the frequency of activities and online probability for each user.

3) Frequency of Activity per User: the previous measurement work conducted on clickstream [13,20], a social network aggregator that pulls content from multiple SN sites into a single view for users, shows that the ratio between different activities remains relatively stable. It is also confirmed by measurement studies on Facebook [12] that there exists a correlation between users' social degree and his degree of activeness. So we use the user interaction data, which are available from crawled data from Facebook, to infer the overall activities of users in our experiment.

We use linear equation of $F(x) = \sum_{i=0}^{9} (C_i x^i)$ that has been proved to be quite close to the measured distribution and 1.8 requests per day per user on average that is collected from clickstream access logs for Orkut users [13].

4) Online Probability per User: we use a uniform online probability for each node in our experiment. In reality, an active peer may be more likely to be online. We have conducted experiments using probability distribution that favors peers with high social degree. These experiments show improved results than the ones presented. However, since we do not have measured online probability distribution of actual SN sites, we use the results from uniform distribution as the bottom line for PAIDD's performance.

5.2 Effect of P2P-based data delivery

Figure 5 shows the central server load and server load ratio for Inet3 topology and Facebook topology, respectively, with the number of concurrent online users growing to 100%. The amount of message delivered to the central server are reduced by 90% fairly reliably using PAIDD with the number of concurrent users greater than 30%. For comparison reasons, we also include the result for "simple" P2P where peers fall back to central servers for contents if a direct connection to the data source fails. We can see that PAIDD is 81% more effective than a simple P2P design in terms of server load reduction. This confirms that using "friend's friend" for message relay is effective in bypassing NAT constraints and the limitations of user being offline. We only present the data from one Facebook topology (Region A) as the other topology produces nearly identical results.

Figure 6 shows how PAIDD performs as social network grows. Here, we set the percent of concurrent online users to 57.18%, based on checkfacebook's recent data. We observe similar server load ratio as the topology grows.

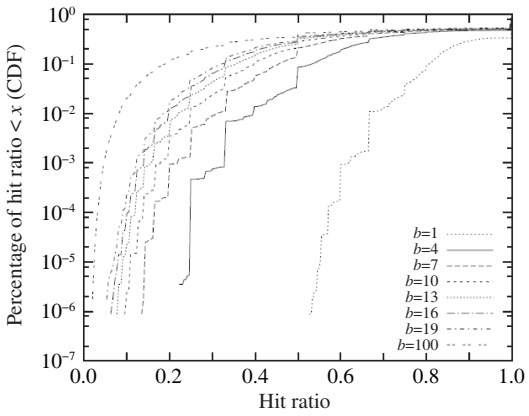

Figure 7 CDF of hit ratio.

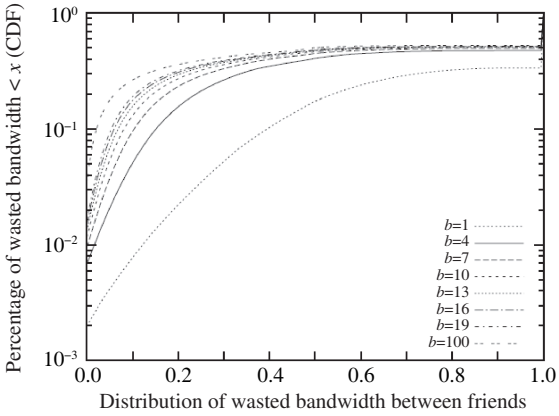

Figure 8 CDF of wasted bandwidth.

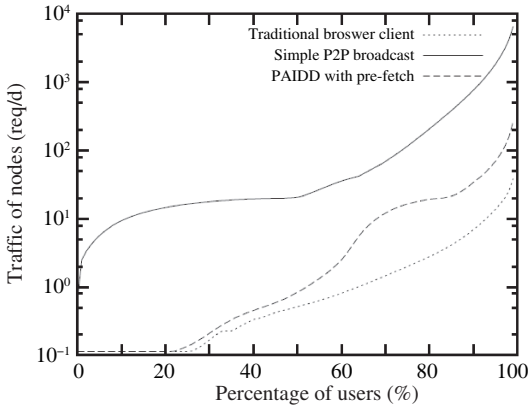

Figure 9 Individual node load comparison with Facebook topology.

We repeated our experiments 1000 times with different NAT and online user distributions. Figures 5 and 6 show that the load reduction from PAIDD is very reliable.

5.3 Threshold and prefetching based user interactions

We first evaluate the prefetching algorithm proposed in Section 4.5. We randomly select 1000 users whose degree are less than 32 from the Facebook data set to get optimal solutions via exhaustive search. The result indicates that 100% of the greedy results equal the optimal solution. Thus, our threshold-based greedy algorithm produces the optimal threshold to balance both hit-ratio and wasted bandwidth. The quality of the greedy algorithm will not be degraded when the user's degree increases.

We calculate the prefetching threshold for each user with greedy algorithm. Figures 7 and 8 show the hit-ratio and wasted bandwidth results, respectively. We see that almost 50% of users obtain 100% hit-ratio. For example, 49.38% of users reach 100% prefetching hit-ratio when $b = 10$.

Figure 8 shows the distribution of wasted bandwidth between user and his/her friends. We see a jump from 51% ($b = 10$) to 100% when x axis reaches 1. This means that nearly half of our users' prefetching content was only from one friend, i.e., the wasted bandwidth comes from the traffic to that friend. This is an artifact caused by the interaction data set used, which only records visible interactions. We have a large amount of users interacting only with one friend. However, the general trend of the results is still valid if we deduct these interactions. That is, when b increases, prefetching hit-ratio will hit diminishing return, because the weight of hit-ratio degrades while that of wasted bandwidth increases according to our designed utility function. Reflected on Figure 8, as b increases, the wasted bandwidth is spread to more friends.

The value of b is control variable that should be tuned by SN operators to control the load distribution

and overhead on their users based on the general activity and interaction frequency between users. Its ideal value can easily be obtained by analyzing a small set of web access logs.

Figure 9 shows the number of messages sent and received by each node with PAIDD. Compared with the traditional client-server solution, less than 40% of nodes in PAIDD have loads increased 10 times. This is acceptable as the traffic of page browsing on SNs from each individual user is generally light. If one goes with a simple pure "P2P" solution, the load on node could be three orders of magnitude higher.

6 Conclusion

In this article, we study the impact of our proposed P2P-based solution to the unique scalability challenge posed by fast growing online social networks. Our solution, P2P-based and Interaction-Oriented data distribution system, is based on our theoretical model and analytical study considering realistic network scenarios and recent measurement discoveries on social interactions. With our design that employs social connections, PAIDD can effectively achieve one order of magnitude of load reduction at central servers. Our experiments confirm that PAIDD ensures user experience without incurring extensive overhead on clients' network.

Acknowledgements

The preliminary work of this paper was presented at the International Conference on Optical Internet (COIN) 2013. This work was supported by Fundamental Research Funds for the Central Universities (Grant No. 2013RC-1102), National Natural Science Foundation of China (Grant No. 61170274), Innovative Research Groups of the National Natural Science Foundation of China (Grant No. 61121061), and Important National Science and Technology Specific Projects: Next-generation Broadband Wireless Mobile Communications Network (Grant No. 2012ZX03002008-002-03).

References

1 de Turck F, Vanhastel S, Volckaert B, et al. A generic middleware-based platform for scalable cluster computing. Future Gener Comput Syst, 2002, 18: 549–560
2 Tang S J, Yuan J, Mao X F, et al. Relationship classification in large scale online social networks and its impact on information propagation. In: Proceedings of 30th IEEE International Conference on Computer Communications, Shanghai, 2011. 2291–2299
3 Aron M, Sanders D, Druschel P, et al. Scalable content-aware request distribution in cluster-based networks servers. In: Proceedings of the Annual Conference on USENIX Annual Technical Conference, San Diego, 2000. 323–336
4 Kevin L, Marco G, Jason K. Social selection and peer influence in an online social network. Proc Nat Acad Sci USA, 2012, 109: 68–72
5 Marcel S, Duy Q V, Shashank K, et al. The dynamics of health behavior sentiments on a large online social network. EPJ Data Sci, 2013, 2: 4
6 Rhea S, Geels D, Roscoe T, et al. Handling churn in a DHT. In: Proceedings of the Annual Conference on USENIX Annual Technical Conference, Boston, 2004. 127–140
7 Godfrey P, Shenker S, Stoica I. Minimizing churn in distributed systems. SIGCOMM Comput Commun Rev, 2006, 36: 147–158
8 Mondal M, Viswanath B, Clement A, et al. Limiting large-scale crawls of social networking sites. In: Proceedings of the ACM SIGCOMM 2011 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Toronto, 2011. 398–399
9 Deng Y. RISC: a resilient interconnection network for scalable cluster storage systems. J Syst Architect, 2008, 54: 70–80
10 Buchegger S, Datta A. A case for P2P infrastructure for social networks: opportunities and challenges. In: Proceedings of 6th International Conference on Wireless On-demand Network Systems and Services, Snowbird, 2009. 161–168
11 Abbas S M A, Pouwelse J A, Epema D H J, et al. A gossip-based distributed social networking system. In: Proceedings of 18th IEEE International Workshops on Enableing Technologies: Infrastructures for Collaborative Enterprises, Groningen, 2009. 93–98

12 Wilson C, Boe B, Sala A, et al. User interactions in social networks and their implications. In: Proceedings of 4th ACM European Conference on Computer Systems, Nuremburg, 2009. 205–218

13 Benevenuto F, Rodrigues T, Cha M, et al. Characterizing user behavior in online social networks. In: Proceedings of 9th ACM SIGCOMM Conference on Internet Measurement, Chicago, 2009. 49–62
14 Ford B, Srisuresh P, Kegel D. Peer-to-peer communication across network address translators. In: USENIX Annual Technical Conference, Anaheim, 2005. 179–192
15 Saikat G, Paul F. Characterization and measurement of TCP traversal through NATs and firewalls. In: Proceedings of 5th ACM SIGCOMM Conference on Internet Measurement, New Orleans, 2005. 199–211
16 Ganjam A, Zhang H. Connectivity restrictions in overlay multicast. In: Proceedings of 14th International Workshop on Network and Operating Systems Support for Digital Audio and Video, Kinsale County Cork, 2004. 54–59
17 Shami K, Magoni D, Chang H, et al. Impacts of peer characteristics on P2P TV networks scalability. In: Proceedings of 28th IEEE International Conference on Computer Communications, Rio de Janeiro, 2009. 2736–2740
18 Jared W, Sugih J. Inet-3.0: Internet Topology Generator. Technical Report CSE-TR-456-02, EECS Department, University of Michigan. 2002
19 Hernandez J M, Kleiberg T, Wang H, et al. A Qualitative Comparison of Power Law Generators. Technical Report 20061115, Delft University of Technology. 2006
20 Schroeder S. 20 ways to aggregate your social networking profiles. Mashable, 2007. http://mashable.com/2007/07/17 /social-network-aggregators/