This paper was presented as part of the 14th IEEE Global Internet Symposium (GI) 2011 at IEEE INFOCOM 2011

Stir: Spontaneous Social Peer-to-Peer Streaming

Anh Tuan Nguyen †, Baochun Li ††, Michael Welzl †, and Frank Eliassen †
† Department of Informatics, University of Oslo, Norway
†† Department of Electrical and Computer Engineering, University of Toronto

***Abstract*—Dealing with a high churn rate is very challenging in live peer-to-peer streaming. State-of-the-art studies try to mitigate the problem by exploiting peer dynamic models, analyzing traces from real world systems, or using enhanced coding techniques, e.g., network coding. Applications of social networking in peer-to-peer systems, especially on file sharing, have recently received research attention. In such systems, the establishment of connections among peers is based on social relationships among users, which are, however, not formed in the context of a peer-to-peer session but, e.g., imported from other social networks. Since friends in such a separate social network do not always have similar interests, they may not necessarily join or stay long in the same peer-to-peer session. We believe that a *tight integration* between the high level social network of users and the low level overlay of peers would bring significant benefits in dealing with high churn rates and providing personalized streaming services. This paper presents *Stir*, the first attempt towards an integrated social peer-to-peer streaming system. The key feature of Stir is that social relationships among users are *spontaneously* formed in a streaming session, and can be exploited *directly* by the underlying streaming protocol. Stir users, who join the same session, can make friends by means of spontaneous communication, e.g., instant messaging. Such social network formation provides a reliable indication to deal with high churn rate. Our simulations with real social data and peer dynamic traces have demonstrated the benefits of Stir and shed light on building such a system in practice.**

I. Introduction

There has been a substantial amount of research on dealing with peer churn. State-of-the-art studies have modelled peer-to-peer (P2P) systems with churn to better understand it [1], analyzed traces of real world systems to understand join and leave patterns [2], or used specialized coding techniques, e.g., network coding [3]. Although they have shown improvements in system performance, the impact of peer churn can not be minimized as its origin has not been considered. The arrivals and departures of users depend on their interests in the session, e.g., users leave quickly because they do not like the content. We believe that knowledge about user interests is critical in dealing with peer churn.

Recently, the popularization of new social network sites that allow individuals to construct personal profiles, connect to people, and keep up with friends has shown that users are indeed interested in sharing their common interests on networked applications. For example, at the time of writing, Facebook [4] is the second most-trafficked website in the world [5]. It has 400 million active users who spend 500 billion minutes per month to interact with over 160 million objects (pages, groups, and events) [6].

This work is partially funded by the Research Council of Norway, grant number 176756/S10.

Applications of social networking to P2P networking have emerged with the objective of improving P2P system performance. The idea is that connections among friends are more reliable than those among strangers. Some studies have shown advantages of social-based P2P systems in file sharing [7]–[9]. In those systems, the establishment of connections among peers is based on social relationships among users, which are not formed in the context of a peer-to-peer session but, e.g., imported from other social networks. Since friends in such a separate social network do not always have similar interests, they may not necessarily join or stay long in the same P2P session. Consequently, users may not acquire enough qualified connections. While P2P-based file sharing applications can suffer from late packets to some extent, the quality of P2P streaming services is seriously affected by such late arrivals.

We believe that a very *tight integration* between the high level social network of users and the low level overlay of peers would bring significant benefits in dealing with peer churn to improve the system performance. This paper presents *Stir*, our new *spontaneous* social P2P streaming system. The distinct feature of Stir is that friendship is spontaneously formed in a streaming session by means of spontaneous communication, e.g., instant messaging (IM) and Twitter-like commenting. Such spontaneous social networks are then directly exploited by the underlying streaming protocol. On one hand, the streaming protocol can get benefits from the social network by being able to establish reliable connections among peers, which are useful in mitigating the impact of peer churn. On the other hand, users have some priority in resource allocation of the streaming protocol run at their friend peers. To the best of our knowledge, Stir is the first social P2P streaming system proposed in the literature which offers satisfactory and personalized streaming services to a large number of users. Simulation results indicate advantages of Stir and shed light on building such a system in practice.

The remainder of this paper is organized as follows. Section II presents the motivation of spontaneous social networking. Section III describes the design of Stir with its architecture. The social-based streaming protocol is presented in Section IV. Section V analyzes simulation results. Related work is presented in Section VI, and Section VII concludes the paper.

II. The Case for Spontaneous Social Networking

In social networking, the term *homophily* is defined as the tendency of people with similar characteristics to be connected [10]. It has also been demonstrated that there is a correlation from social networks to user behavior on the Web [11]. Particularly, people who chat with each other are more likely to

share interests. The more time they spend talking, the stronger this relationship is.

From the above social principles, users who join and stay in a streaming session are likely to have some similar interests in the stream. Therefore, providing means of communication among users will not only offer more entertaining services, but also create social relations among them. From social activities, user behavior can be predicted, e.g., the more friends a user has in a session, the more she is interested in and the longer she stays in the session. Therefore, if connections between peers are established based on such social relationships, they are more reliable and durable. As a result, users are not seriously affected by departures of those who do not have a strong interest in the session and stay for a short period of time. This naturally reduces the impact of churn.

What is a suitable social communication means in the scenario? The foremost requirement for a communication means in our system is that it does not consume much bandwidth, which is a critical resource in streaming. Another criterion is that it is able to achieve a certain level of synchronization because live streaming is highly synchronized among users. Among many means ranging from non-interactive, e.g. emailing, to real time interactive, e.g., voice chatting, IM is probably the most suitable one. Many studies on characteristics of IM [12]–[14] have shown that (1) chat messages exchanged among users are usually very short, but (2) they are expressive enough to support a variety of informal communication tasks in a semi-synchronous way.

III. Stir: Schematic and Architectural Design

Stir is a multi-channel P2P streaming system, which provides live streaming content, e.g., P2P-based IPTV. Each Stir user has a profile containing a unique identifier, a password, and a friend list. A user U needs to log in to the system before watching any channel. There are no privacy concerns in the Stir architecture, as such login information does not have to reveal any personal information — an email address would suffice. If U is authenticated, the system sends the profile back to U. The streaming server sends the channel list to U. When U selects a channel, the server will send an IP address list of some available peers in the channel to U. Now, U can connect to other users for data download.

In Stir, friendships are to be established on-the-fly, as a group of users watch the same channel. While watching, U can post comments to one of the Stir online forums. This forum is only visible to those who are watching the same channel with U. U can have a private chat with another user V, and the chat can be entirely conducted in a web browser, as the channel is being played live. U can add V to her friend list. The list of friends constitutes a state that carries over from one session to the next: being in the friend list of U means that U knows the status (if V is in the system or not, which channel V is watching) of V whenever U logs in to the system.

Stir is a pull-based streaming protocol. Each peer pulls video data from other peers based on buffer map exchanges. Different from traditional P2P streaming which has one list of anonymous peers (called neighbors), each Stir peer has two lists for potential partners. The neighbor list is updated by a gossip-based mechanism, and the friend list contains friends who are also watching the same channel. Each item in the lists contains the IP address and some statistical data collected from the social activities of users, which are stored in the social log. These statistical data will be used by the partner manager and the scheduler. The partner manager selects potential partners from the lists for requesting data based on the status of the playback buffer, social factors and network metrics. The scheduler schedules data requests and sends them to the selected partners. Received packets are stored in the playback buffer and will be sent to the player when the playback deadline is reached. Figure 1 shows the components and their interactions.

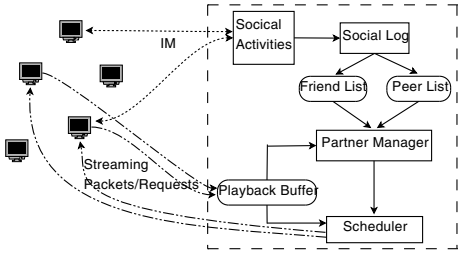


Fig. 1. The architectural design of Stir, with an emphasis on the tight integration between streaming quality and spontaneous social network relationships.

As Figure 1 shows, a user may talk with a group of other users, while video packets are exchanged with yet another different group of users. This is a key difference compared to existing studies on social-based P2P, which assume that there are always enough friends in the system to have trusted connections and discourage connections with strangers. In Stir, both social relationships and network metrics are taken into account to choose partners.

IV. A Social-based P2P Streaming Protocol

With the main components being presented in the previous section, this section goes into details of the partner manager and the scheduler to understand how the underlying P2P overlay relates to the high level social network, and how social data can be used in streaming.

A. Relying on Friendship or Bandwidth: a Tradeoff

In a social-based P2P streaming system, the establishment of network connections among peers is guided by social relationships among users. However, if a protocol is heavily based on friendship and discounts links with others, connections are more reliable but each peer may not acquire enough qualified connections to maintain smooth playback, because their friends may not have sufficient bandwidth. The objective here is to not only give friends some priority in resource allocation but also achieve smooth playback for the peer itself. We propose to use the following utility function, which takes both social factors and network metrics into account as the basis for mitigating the tradeoff.

$$U_Q = (1 - \alpha - \beta) \cdot C(Q) + \beta \cdot S(Q) + \alpha \cdot F(P, Q)$$

In the above equation, C is a capacity-related function, S is a social-related function, and F is a friendship-related function. β is *social coefficient*, as it determines how important social capacity is in the evaluation. α is *friendship coefficient*, as it determines the priority of friendship. In traditional non-social P2P streaming, α and β are 0 as social factors do not exist. In previous social-based P2P systems, α is close to 1 as they discourage connections with strangers. By experimenting with wide ranges of values of α and β, we understand interactions of the social network and the P2P network.

In this paper, the capacity function C, the social function S, and the friendship function F are calculated for Q in a list L as follows:

$$C(Q) = \frac{B_Q}{\max_{i \in L}(B_i)}$$

$$S(Q) = \frac{N_Q}{\max_{i \in L}(N_i)}$$

$$F(P,Q) = \begin{cases} 1, & \text{if P and Q are friends} \\ 0, & \text{otherwise} \end{cases}$$

where B_i is the bandwidth capacity of peer i and N_i is the number of friends of i.

B. Partner Manager

The interaction between the social network and the overlay network happens in the partner manager, which determines a group of active peers for sending and receiving data. When a Stir peer P selects partners, its partner manager calculates utility values of peers in the neighbor list and the friend list. After that, it sends partner requests to a certain number of peers, which has the highest utility values. If a candidate Q accepts the request, P adds Q to its partner list, which is used for buffer map exchanges and video data requests. The partner acceptance check is also based on the utility function.

Each peer has an acceptance list containing peers whose partner requests have been accepted. Being in the acceptance list of Q means that P can send data requests to Q and will be served if Q has sufficient resources. Different from the partner selection process that depends fully on the utility value of candidates, the acceptance check has to give friends higher priority than others, regardless of their social capacity or bandwidth. This is a design principle of Stir to encourage people making friends and sharing their interests. One question here is that if the whole system benefits from this incentive mechanism because it causes less bandwidth to be available for low social capacity peers than a system without the incentive. We believe that users are more willing to share their bandwidth with their friends than with non-friends. Therefore, even if not using the incentive rule brought a little better performance to the whole system, would users be happy if the more friends they have, the more bandwidth they may have to share with non-friends?

C. Packet Scheduler

In traditional pull-based P2P streaming protocols, buffer maps are exchanged between a peer and its partners to decide who will deliver which packets. In Stir, before the buffer map exchange, the peer needs to send a confirmation request to each partner to make sure that it is still in the acceptance list of the partner. After that, buffer maps can be exchanged, and packets can be requested from confirmed partners.

The reason for this step is that the acceptance list of a peer can be changed during the streaming session. For example, at the beginning, a peer is willing to serve non-friend peers because its friends have not joined the session yet. However, when receiving partner requests from friends and the acceptance list is full, it has to remove some non-friends from the list to serve the friends better. As a result, the partner list of those non-friend peers is changed, and needs to be updated by invoking the partner selection. This additional action does not cause much overhead because the size of the partner list is small, $5 - 10$ peers.

V. Stir: Experimental Results

Stir is implemented in our own discrete-event flow-based simulator [16]. Since user behavior and the friend-making process can not be simulated, we focus on interactions between the social network and the P2P overlay, while making assumptions on the social network formation.

A. Data Preparation and Assumptions

For the experiments, we need (1) a real social graph representing the friendship network, and (2) join and leave times of real-world users.

1) Social Graph: We are not aware of the availability of social graphs that are formed spontaneously in a particular context as in the case of Stir. As an alternative, we use a network of people who are interested in a particular topic to represent the network of users joining a streaming session. In particular, we develop our own "graph data provider" plug-in in NodeXL [17] to retrieve friend lists of members of a group in Flickr [18]. With the URL of any public group, the provider can collect user IDs of members of the group and their friends[1]. From the dataset, we form a social network for our experiments as follows:

- ▷ All group members of a group are considered as users of a streaming session.
- ▷ If a member P is in the contact list of another member Q, they are friends of each other in the session. Since we consider relationships among users who join the same session, non-members in the contact lists of the members are removed.

Datasets of different Flickr groups with different sizes have been collected to choose one, which has ~ 1000 members (a medium-large streaming session). It may seem a bit far-fetched to assume that common interest in a topic for pictures in Flickr would correlate with common interest in a live stream in a P2P streaming system. On the other hand, social interaction data from people who are, e.g., watching a stream

[1]The graph data provider is available upon request. Since the member list of a public group and the friend list of a user are public, we do not violate any privacy rules of Flickr.

together were not available to us. We therefore believe that a network of common interests and friendship (as opposed to mere friendship connections) is as close as we can get.

The dataset from the group "Photo Computer Art" is chosen. When we collected this dataset, it had 1280 members. The CDF of the degree of the members in the friendship network is shown in Figure 2, which indicates that $\sim 10\%$ of the members have no friends, $\sim 80\%$ have fewer than 20 friends, and the other 20% have from 20 to 54 friends[2].

2) Peer Dynamics: From the snapshot trace of PPLive [19], we extract the arrival and departure time of users for a period of time (2 hours) on a particular channel. We use three datasets with different levels of peer churn: low, medium, and high. The CDF of the stay duration of peers in the datasets are shown in Figure 3. The high dynamic scenario has up to 60% of peers staying in the session less than 20 minutes and $\sim 90\%$ staying less than one hour. On the other hand, up to 60% of the peers stay longer than one hour in the low dynamic case.

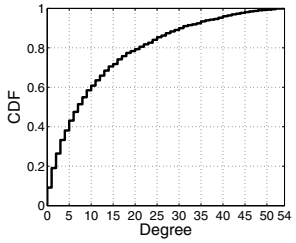


Fig. 2. Friendships among peers

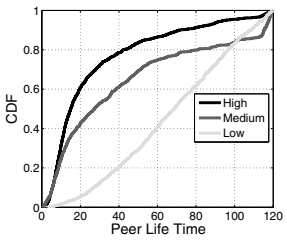


Fig. 3. Peer Dynamic Scenarios

3) Assumptions on User Behavior: We need to combine the peer dynamic datasets and the social dataset to have a complete picture: who are friends of whom, and when they join and leave the session. Based on the discussion in Section II about the spontaneous social networking formation, there is reason to believe that the following assumptions are reasonable: (1) friendship indicates similar interests in the content, and (2) the more friends a user has, the longer she stays in the system. With these assumptions, we can join the datasets as follows:

- ▷ Sort peers on their stay duration.
- ▷ Sort users on the number of friends they have.
- ▷ Assign join and leave times to users so that peers with longer duration have higher number of friends.

In practice, there could be other reasons for user departures, e.g., network availability problems. Therefore, the above assignment is only valid with another assumption: users are not unexpectedly disconnected from the network, i.e., the arrivals and departures are simply from the interest of users in the session. This helps us understand separately the role of social factors in the system performance.

4) Bandwidth Settings: The streaming rate is set to 400 Kbps, and the bandwidth of each peer is randomly assigned one of the following values (download, upload): (450, 300), (550, 450), (700, 650), and (750, 700) Kbps. The server upload rate is set to serve 65 ($\sim 5\%$) peers simultaneously.

[2]Experiments have also been carried out with another dataset, which is also from Flickr but has different degree CDF values. The claims in this paper also hold.

B. Comparison with Existing Work

We implement a CoolStreaming-like protocol based on the description in [15] and a network coding based protocol, called NCStream, in our simulation to evaluate their performance in terms of playback skip rates.

1) On Peer Dynamics: In this experiment, (α, β) in the partner selection, and the acceptance check of Stir are set to (0.2, 0.3), and (0.7, 0.2), respectively. The skip rate of the systems under different peer dynamic scenarios is shown in Figure 4.

Figure 4 shows that the three systems can achieve similarly low skip rates when the network is quite stable (low dynamic). Actually, NCStream is the best protocol in this case because NC helps to utilize the bandwidth better. However, the performance of them are notably different in the high dynamic scenarios. Only $\sim 0.54\%$ of playback segments are skipped by Stir peers in the high dynamic case, while the percentage for NCStream and CoolStreaming is 0.71% and 4.17%, respectively. The reason for the superior performance of Stir under high churn rate is that peers who stay longer in the session are likely to have a certain number of friends and exchange data to each other. In addition, the social-based acceptance check gives friends higher priority in data delivery. Consequently, even when a large number of users who are not interested in the session leave, the communities of friends are not seriously affected. We calculate the traffic (number of packets exchanged) between friends and that between non-friends in the protocols for the high dynamic case: $\sim 60.6\%$ of traffic in Stir is among friends, while, without social knowledge, up to $\sim 65.8\%$ of traffic in CoolStreaming ($\sim 64.2\%$ in NCStream) is among 'strangers'. Although the percentages depend on how dense the friendship network is, this experiment indicates the ability of exploiting social knowledge in Stir.

2) On The Size of Neighbor Lists: In CoolStreaming and NCStream, the neighbor list is the local view of the network. The streaming quality a peer receives completely depends on its neighbors. In Stir, in addition to the neighbor list, a peer has a friend list. It should be noted that peers in the friend list can also appear in the neighbor list because the neighbor list is updated by gossiping, independently from the social network. We would like to answer two questions: *how important is the neighbor list in Stir?* and *How big should it be?* We plot the skip rate of the protocols with the size of the neighbor list ranging from 20 to 60 for Stir, and 40 to 80 for CoolStreaming and NCStream in Figure 5. The reason for larger neighbor lists in CoolStreaming and NCStream is to have a 'fair' comparison between them and Stir, because Stir peers may also have a large friend list.

It can be concluded from Figure 5 that (1) using a larger neighbor list, in fact, does not always improve the performance in CoolStreaming and NCStream, and (2) with the existence of the friend list, Stir can use a small neighbor list. The counter

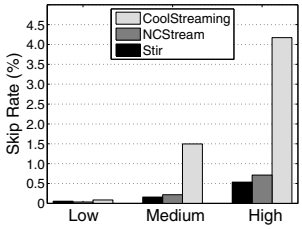


Fig. 4. Stir minimizes the impact of peer churn.

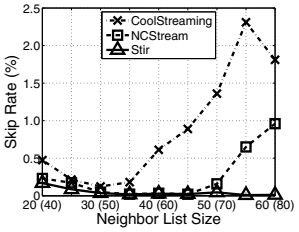


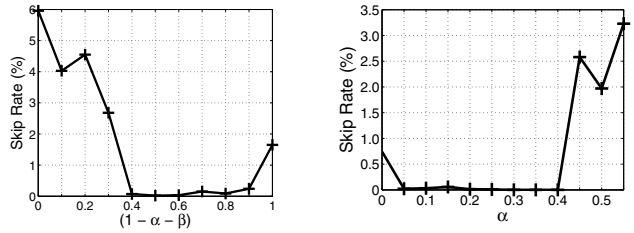
Fig. 5. Skip rates with different sizes of the neighbor list. The numbers in parentheses denote the size of the neighbor list in CoolStreaming and NCStream.

effect of larger neighbor lists in CoolStreaming and NCStream can be explained as follows. The larger the neighbor list is, the higher the probability of more than one peers choosing the same set of high capacity peers, i.e., bottlenecks at high capacity peers are likely to occur. The problem does not occur in Stir due to the acceptance check that guarantees that a peer 'reserves' resources for its friends.

C. Insights of Stir

Convinced that by exploiting social knowledge Stir deals with peer churn much better than previous work, we now turn our attention to the insights of Stir. We experiment with different values of α and β in the partner selection process and the acceptance check to understand the role of friendship, network capacity, and social capacity in choosing partners.

1) On the Partner Selection: We fix the value of α and β in the acceptance check, while using wide ranges of values for the coefficients in the partner selection process. Figure 6 shows the skip rate of Stir when (1) $\alpha+\beta = 0, 0.1, ..., 1$ to understand the role of the network capacity and the social factors (Figure 6a), and (2) $\alpha + \beta = 0.6$ and $\alpha = 0, 0.05, ..., 0.55$ to understand effects of social capacity and friendship in choosing partners (Figure 6b).



(a) Network capacity Vs. Social factors (b) Social capacity Vs. Friendship

Fig. 6. The effect of α and β in partner selection.

Figure 6a indicates that network capacity is important in the partner selection process. If a peer connects to others based heavily on social factors (high values of $\alpha + \beta$), it may suffer playback skips because (1) a high social capacity does not imply a high network capacity, (2) peers with high social capacities have more friends to serve, and (3) its friends may not have sufficient bandwidth. However, a peer should also not depend only on the network capacity (high values of $1 - \alpha - \beta$) because of peer churn: peers with high network capacities may only stay in the session for a short period of time. Very low skip rates can be achieved if peers consider network capacity as important as, or slightly less important than, the social factors ($1 - \alpha - \beta = 0.4$, or 0.5).

Between social capacity and friendship, from Figure 6b, we can see that preferring high social capacity peers gives better results than preferring friends for data requests, as high values of α (> 0.4) increase the skip rate significantly. Although this phenomenon is somewhat contradictory to the idea of connecting friends with each other, it is reasonable from the peers' point of view because the higher the social capacity is, the more durable the peer is. However, friendship has a certain importance as setting α to 0 does not achieve the best performance. The reason is that if a peer chooses its friends as partners, it will have a certain priority at the friend side in their bandwidth allocation. Therefore, if friends of a peer have sufficient bandwidth, the peer will receive higher quality. In addition, being a partner of a high social capacity peer does not guarantee that it will be served. In a nutshell, *network capacity, social capacity and friendship have their own roles in the partner selection process*. However, different from existing work, when a peer chooses partners, friendship is not the only important factor, but bandwidth and social capacity as well.

2) On the Acceptance Check: Since peers have the highest priority to be in the acceptance list of their friends, the role of α no longer exists in the utility function. For non-friends, the acceptance check is based on their social capacity and network capacity. On one hand, a peer could prefer high social capacity peers (by setting high values for β) to reward them as they are 'famous' in the social network. On the other hand, high network capacity peers could have a high priority for the reason that when they can receive packets quickly, they can deliver them quickly to other peers. However, our experiments with different values of β from $0, 0.1$, ... to 1 show no significant differences to the overall system performance (the graph is not shown here). The main reason is that, peers that have a certain number of friends are served well by their friends, so they do not need to be partners of non-friends. For those who have very few friends or no friends, their social capacity is quite similar. Therefore, the role of β is minor. In other words, the case of high social capacity peers compete with low social capacity ones to appear in an acceptance list seldom occurs in our experiments. However, keeping the utility function in the acceptance check is useful in practice, because there are still cases that a peer has a number of friends but some of them may not join the session.

3) The value of being famous in Stir: In Figure 7, we plot the average quality of peers having the same number of friends to find out how the number of friends of a peer relates to the quality that peer perceives.

Figure 7 shows that the more friends a user has, the higher the quality she is likely to receive. This encourages users to join the system, and creates a friendly collaborative network among users. More specifically, there are two noteworthy points. *First*, the average skip rate of socially inactive users who do not have any friend is less than 0.5%, which is still

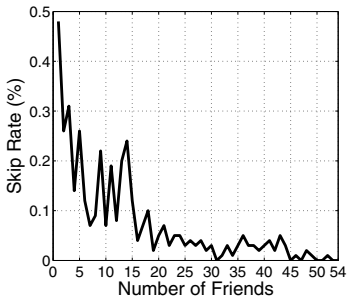


Fig. 7. In Stir: The more famous you are, the higher the quality you receive.

acceptable in live P2P streaming. This means that Stir does not completely discriminate against low social capacity peers. New users who are interested in the session but do not have many friends yet still have chances to enjoy quite a satisfactory level of quality. *Second*, for those who have fewer than 20 friends, higher social capacity does not always bring lower skip rates. The CDF of the number of friends in Figure 2 can help to understand the reason. There are many users ($\sim 80\%$ of the population) having fewer than 20 friends. As a result, many of those users may not receive enough packets from their friends, and have to compete with others. As shown in the experiments of the acceptance check, the social capacity of those peers does not have a strong effect. In a bigger picture, if the overall bandwidth of socially active users is more than enough for the socially active users themselves, than the unused bandwidth can be used for socially inactive users, i.e., it is not wasted. Otherwise, socially inactive users have to suffer degraded quality.

One may concern that a rational peer will make a large number of cheap "spontaneous" friends to improve its quality. As a result, all peers will have a large number of spontaneous friends, and the proposed algorithm can no longer differentiate peers. This is a common issue of social-based applications. For example, Facebook game users may try to make friends with others who they do not really know about to have more friends so that they can get benefits from some game. One possible solution is that some requirements can be used in the friend-making process such as friendship can only be formed after two users sometimes chat to each other.

VI. Related Work

There have been various approaches to dealing with peer churn. Kumar *et al.* introduces a stochastic fluid model that accounts for essential features, including churn, of a P2P streaming system [1]. Vu *et al.* have measured and modelled large-scale P2P overlay graphs in PPlive [2]. Wang and Li are successful in applying NC to live P2P streaming [3]. Thanks to NC, peer bandwidth is better utilized and the system is more robust to peer churn.

Generally, social networking can be applied to P2P systems in two ways. *First*, P2P systems can mimic how people form a social network and how they query, by preference, their friends or acquaintances to construct overlays, which can achieve more efficient routing and data locating [8], [20]. *Second*, they can import social graphs from other social networks to establish more reliable connections among peers [7], or to improve peer collaboration and contribution [9].

While the above social-based systems are designed for P2P file sharing, we are not aware of the existence of any existing designs with respect to social P2P streaming.

VII. Conclusion

This paper presents *Stir*, a new framework towards tightly integrated spontaneous social networking in P2P streaming, as well as a social-based streaming protocol exploiting social relationships of the social network. Under the assumption that social networking characteristics and peer stability are related, our work hints that by offering cheap yet efficient communication means to users, the P2P streaming system can achieve better performance compared to previous systems, especially when dealing with high peer dynamics. Simulations with real social network data and real peer dynamic traces indicate the potential benefits of our approach.

References

[1] R. Kumar, Y. Liu, and K. Ross, "Stochastic Fluid Theory for P2P Streaming Systems," in *Proc. of IEEE INFOCOM*, 2007.
[2] L. Vu, I. Gupta, K. Nahrstedt, and J. Liang, "Understanding the Overlay Characteristics of a Large-scale Peer-to-Peer IPTV System," *ACM Trans. on Mult. Computing, Communications and Applications*, Feb 2011.
[3] M. Wang and B. Li, "R^2: Random Push with Random Network Coding in Live Peer-to-Peer Streaming," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 9, pp. 1655–1666, Dec. 2007.
[4] Facebook. [Online]. Available: http://www.facebook.com/
[5] Alexa Top 500 Global Sites. [Online]. Available: http://www.alexa.com/topsites
[6] Facebook Statistics. [Online]. Available: http://www.facebook.com/press/info.php?statistics
[7] J. A. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D. H. J. Epema, M. Reinders, M. R. van Steen, and H. J. Sips, "TRIBLER: a Social-based Peer-to-Peer System," *Concurrency and Computation: Practice and Experience*, vol. 20, no. 2, 2008.
[8] K.-J. Lin, C.-P. Wang, C.-F. Chou, and L. Golubchik, "SocioNet: A Social-Based Multimedia Access System for Unstructured P2P Networks," *IEEE Trans. on Paral. and Dist. Sys.*, vol. 21, no. 7, Jul 2010.
[9] Z. Liu, H. Hu, Y. Liu, K. Ross, Y. Wang, and M. Mobius, "P2P Trading in Social Networks: The Value of Staying Connected," in *Proc. of IEEE INFOCOM*, 2010.
[10] M. McPherson, L. Smith-Lovin, and J. M. Cook, "Birds of a Feather: Homophily in Social Networks," *Annual Review of Sociology*, vol. 27, no. 1, pp. 415–444, 2001.
[11] P. Singla and M. Richardson, "Yes, There is a Correlation - From Social Networks to Personal Behavior on the Web," in *Proc. of ACM WWW*, 2008, pp. 655–664.
[12] Z. Xiao, L. Guo, and J. Tracey, "Understanding Instant Messaging Traffic Characteristics," in *Proc. of IEEE ICDCS*, 2007.
[13] J. Leskovec and E. Horvitz, "Planetary-Scale Views on a Large Instant-Messaging Network," in *Proc. of ACM WWW*, 2008, pp. 915–924.
[14] B. A. Nardi, S. Whittaker, and E. Bradner, "Interaction and Outeraction: Instant Messaging in Action," in *Proc. of ACM CSCW*, 2000.
[15] X. Zhang, J. Liu, B. Li, and T.-S. Yum, "CoolStreaming/DONet: a Data-Driven Overlay Network for Peer-to-Peer Live Media Streaming," in *Proc. of IEEE INFOCOM*, vol. 3, March 2005, pp. 2102–2111.
[16] A. T. Nguyen, B. Li, and F. Eliassen, "Chameleon: Adaptive Peer-to-Peer Streaming with Network Coding," in *Proc. of IEEE INFOCOM*, March 2010.
[17] NodeXL. [Online]. Available: http://nodexl.codeplex.com/
[18] Flickr. [Online]. Available: http://www.flickr.com/
[19] PPLive Traces. [Online]. Available: http://dprg.cs.uiuc.edu/downloads/
[20] X. Cheng and J. Liu, "NetTube: Exploring Social Networks for Peer-to-Peer Short Video Sharing," in *Proc. of IEEE INFOCOM*, 2009.