$\mathcal{SMS}$: Collaborative $\mathcal{S}$treaming in $\mathcal{M}$obile $\mathcal{S}$ocial Networks

Chenguang Kong[1], Chuan Wu[1], and Victor O.K. Li[2]

[1] Department of Computer Science, The University of Hong Kong
{cgkong,cwu}@cs.hku.hk
[2] Department of Electrical and Electronic Engineering, The University of Hong Kong
vli@eee.cs.hku.hk

Abstract. Mobile social applications have emerged in recent years. They explore social connections among mobile users in a variety of novel scenarios, including friend finding, message routing, and content sharing. However, efficiently supporting resource-demanding delay-sensitive streaming applications on the mobile platform remains a significant challenge. In this paper, we study collaborative VoD-type streaming of short videos among small groups of mobile users, so as to effectively exploit their social relationships. Such an application is practically set in a number of usage scenarios, including streaming of introductory video clips of exhibition items to visitors' mobile devices, such as in a museum. We design *SMS*, an architecture that engineers such Streaming over Mobile Social networks. SMS constructs a collaborative streaming overlay by carefully inspecting social connections among users and infrastructural characteristics of Bluetooth technologies. We evaluate our design based on prototype implementation on the Android platform.

Keywords: Applications and Services, Mobile Social Networks, Peer-Assisted Streaming

1 Introduction

Novel mobile social applications have emerged in recent years. They explore advanced mobile technologies and social connections among mobile users for interactions and exchanges anywhere at any time. Representative mobile social networks can be grouped into two categories. Mobile social networks in the first category are a natural extension of online social networks over the Internet, where users participate in Facebook-like social applications using their mobile devices, *e.g.*, Tencent QQ [3], Cyworld [2], both of which are enabled with web and mobile access. Exploiting geographic positions of the mobile users, location-based features may be added, such as sharing of geotagged photos and news captured by mobile devices. In the second category, mobile users in physical proximity directly connect with each other, for friend making based on common interest (*e.g.*, MagnetU [1]), or information searching by propagating queries (*e.g.*, PeopleNet [15]) as well as blog and photo sharing (*e.g.*, Micro-Blog[9]).

Internet access via 3G or WiFi is required for mobile devices to participate in social networks of the first category, while Bluetooth is largely exploited in the second category. We target the latter type of self-organized mobile social networks, where more challenges must be overcome to enable richer functionalities (that are comparable to those via Internet access) at cheaper device costs. In particular, we are interested in effective designs for supporting delay-sensitive streaming applications, which are more resource-demanding than sharing of blogs and photos with typical sizes of a few tens or hundreds of Kbytes.

We seek to design an efficient architecture and detailed protocols for collaborative VoD-type streaming of short videos in a peer-to-peer (P2P) fashion, which effectively exploit participants' social relationships. Such an objective is representative of a number of practical usage scenarios. For example, in a museum or a botanic garden, introductory videos are commonly seen accompanying items on exhibition to give the visitors rich information about the items. Compared to the current way of broadcasting each video with a dedicated screen, a superior solution is to implement cooperative streaming of the video to mobile devices of the group of audience surrounding the exhibition item in a P2P VoD fashion, to save hardware cost and provide viewing flexibility (a visitor can begin watching from the beginning whenever he approaches the item and drag to view any part of the video at will). There are other usage scenarios such as sharing amusing video clips one has in his cellphone to friends in a party.

To achieve the above objective, we design and implement *SMS*, an architecture to engineer VoD-type Streaming over Mobile Social networks, consisting of virtual communities of mobile users interested in particular videos. Our contributions in the novel design of *SMS* are three-fold: *First*, we explore social connections among participants in the virtual community, to facilitate voluntary collaborative streaming according to users' personal preferences. Specifically, we quantify social connections into different levels, namely friends, matches of social attributes, and others. Users are allowed to specify personal preferences in video streaming, to friends only, to friends and attribute matches, or to anyone, based on their own levels of social selfishness, altruism, or practical concerns such as the current battery levels. *Second*, we construct streaming overlays by synergistically combining participants' preferences, video segment availability, and characteristics of Bluetooth infrastructures. The broadcast nature of wireless transmissions is also exploited in efficient streaming of segments of common interests. *Third*, we design detailed protocols to implement *SMS*, which effectively tackle dynamics of the mobile users, including movements and VCR operations. We also implement our design on the Android platform and carry out extensive evaluations on streaming performance and battery consumption.

The remainder of the paper is organized as follows. We present the architecture and detailed design of *SMS* in Sec. 2 and Sec. 3, respectively. Sec. 4 gives results of the experimental evaluation. We discuss related work in Sec. 5 and conclude the paper in Sec. 6.

2 System Architecture

We focus on collaborative streaming of short videos among small groups of mobile users. For each video, there is one source **S**, which could be the video-emitting device besides an exhibition item in the museum scenario, or the mobile user who owns the video to be shared in the party scenario. The video is partitioned into n consecutive *segments* for dissemination. A set of mobile users, $\mathbb{V}$, in close proximity of the source and interested in viewing the video, form a virtual community and retrieve the video segments in a P2P fashion via Bluetooth connections. We assume the source and mobile users are within the Bluetooth transmission ranges of each other. Each mobile user may start to view the video at any time, *e.g.*, when he approaches the exhibition item or first joins the group of friends who are sharing the video, and can drag the sliding bar on his video player to view any part of the video at will. Therefore, Video on Demand (VoD) type of streaming is investigated in our design.

We consider mobile users are social entities, with different levels of selfishness and altruism. In collaborative streaming within each virtual community, the mobile users may have different preferences in contributing resources (battery, bandwidth) to upload video streams to others, which we classify into three categories: (1) strongly socially selfish type, corresponding to mobile users who wish only to upload to their friends; (2) weakly socially selfish type, for those who are willing to upload to friends as well as strangers with certain social attributes (*e.g.*, the same hometown, the same hobby, etc.); (3) altruistic type, for users willing to help anyone else in streaming. Completely selfish users, who do not wish to upload to anyone else, are excluded from the system, by enforcing that only users who pick one of the above options and enable at least one upload connection can participate in streaming.

To achieve efficient streaming over such mobile social networks, we design *SMS*, an architecture for collaborative streaming with voluntary resource contributions, by exploring social preferences of the mobile users. Our architecture consists of four functional modules.

Bluetooth Protocols. Video transmissions among mobile users are based on Bluetooth in *SMS*. The Bluetooth protocols deal with connection setup and data transfer between Bluetooth devices. The detailed functions include device discovery, service discovery, connection establishment, and data transmission. Specifically, we carefully explore characteristics of Bluetooth infrastructures, *i.e.*, piconet and scatternet, when constructing the P2P streaming topology among mobile users, for high transmission efficiency and low latency during streaming.

Social Preferences. The social preference module handles inquiry and matching of preferences and attributes among the mobile users. Each user maintains a profile, which records his upload preference, friend list, as well as a number of attributes he wishes to share with others (*e.g.*, hometown, university/school graduated from, hobbies, etc.). Potential streaming links will be established only between users with matching preference and attributes.

P2P Streaming. The P2P streaming module constructs P2P streaming topology among mobile users in the virtual community, and implements request

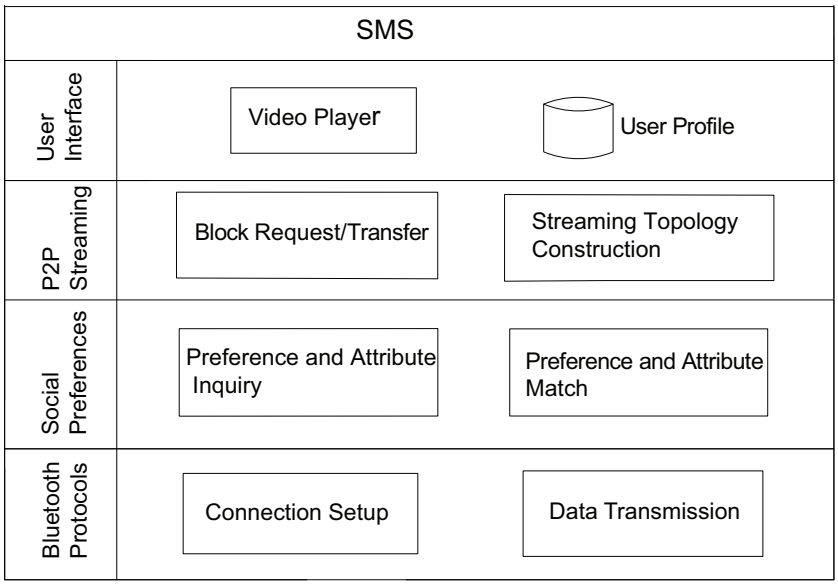

Fig. 1. Architecture of *SMS*.

and upload of video segments among users. The establishment of streaming links from one mobile user to another is contingent on whether the upstream peer has the video segments the downstream peer needs, as well as the upload preference of the upstream peer, social connections between the two, and the transmission efficiency of the resulting Bluetooth infrastructures. In particular, broadcast opportunities are explored as much as possible among sub groups of users with similar viewing progresses, by organizing them into a piconet, for the best dissemination efficiency.

User Interface. This module provides interfaces on the mobile devices for users to maintain their profile, *i.e.*, indicating their upload preferences, friend list, and attribute values. It also provides the video player interface for users to view the video and to control the playback progress.

An illustration of the *SMS* architecture is given in Fig. 1. We design detailed protocols to implement each module in the following section.

3 *SMS*: Detailed Design and Protocols

We first present detailed approaches to construct efficient collaborative streaming topology, based on social preferences and Bluetooth characteristics. We then present practical protocols to implement the design on individual dynamic users.

3.1 Collaborative Streaming Design

The key design issue is to decide which other peers each mobile user streams from and uploads video segments to at each time, *i.e.* the dynamic construction of collaborative streaming topology. Three aspects are investigated.

Segment Availability. To minimize the number of times for costly teardown and setup of Bluetooth connections, a mobile user should connect to a peer owning the largest number of video segments it needs. The suitability for user y to establish a download connection from user x can be decided as follows.

Let $S_x = (s_x^1, s_x^2, \cdots, s_x^n)$ be the bitmap indicating which video segments user x has, where

$$s_x^i = \begin{cases} 1 & : \quad X \text{ holds video segment i} \\ 0 & : \quad \text{otherwise} \end{cases}, \quad i = 1, \ldots, n.$$

Let $R_y = (r_y^1, r_y^2, \cdots, r_y^n)$ denote the segment request list at a mobile user, where

$$r_y^i = \begin{cases} \rho^{i-p} & : \quad i \geq p \\ 0 & : \quad \text{otherwise} \end{cases}, \quad i = 1, \ldots, n,$$

with parameter $\rho < 1$ and p indicating the index of the segment the user is currently playing. Here different segments are assigned different weights according to how close they are to the playback deadlines, to be used to prioritize request sequence of the segments.

The suitability for user y to stream from user x can then be calculated as

$$F(x, y) = \sum_{i=1}^{n} r_y^i \times s_x^i.$$

A larger $F(x, y)$ means that user x may supply user y more segments it urgently needs.

Social Preferences in Uploading. Whether y can download segments from x depends on social preference of x. There are three scenarios in which x is willing to upload to y:

(1) y is a friend of x;
(2) y shares the same social attributes with x, while x is willing to upload to users with matching social attributes;
(3) y is neither a friend nor matches the social attributes, but x is willing to upload to anyone.

Let $W(x, y)$ denote the level of preference for x to upload to y. Let a, b, and c be three system parameters satisfying

$$a + b + c = 1, a > b > c > 0.$$

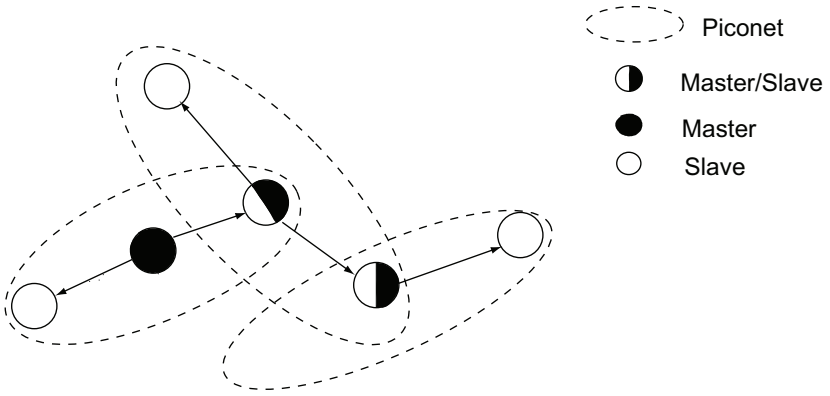

Fig. 2. Bluetooth network structure: an illustration.

We define

$$W(x,y) = \begin{cases} a & : \text{ Case (1)}, \\ b & : \text{ Case (2)}, \\ c & : \text{ Case (3)}, \\ 0 & : \text{ otherwise.} \end{cases}$$

Combining segment availability with social preferences, we derive the following *download preference* for y to prioritize his connection requests towards potential supplying peers:

$$P(x,y) = W(x,y) \cdot F(x,y).$$

The larger $P(x,y)$ is, the more likely x can upload more needed segments to y.

Bluetooth Transmission Efficiency. The construction of streaming topology should be efficiently combined with Bluetooth transmission technology, and exploits the broadcast nature of wireless transmissions for optimized streaming efficiency. In our design, we organize mobile users into an efficient Bluetooth scatternet, and explore broadcast opportunities within each piconet as much as possible.

Fig. 2 illustrates the Bluetooth network the mobile users form. Specifically, there are three different types of nodes in this Bluetooth scatternet: master nodes (MNs), master/slave nodes (MSNs), and slave nodes (SNs). The MNs own the complete video (*e.g.*, the original source of the video or mobile users who have completed the download), and serve as seeds in the streaming network. Each MSN serves as the master node in one piconet and a slave node in another piconet concurrently, corresponding to mobile users who upload cached video segments to peers while downloading further needed video segments. The SNs are mobile users who have just started downloading (*e.g.*, those just joined the system) and have not served others yet.

The Bluetooth network is dynamically constructed based on users' upload preferences and evolving with the streaming progress and dynamics of users. We make two design choices in constructing the collaborative streaming overlay over the Bluetooth network.

First, each mobile user maintains only one download connection at a time and maximally retrieves all segments the upstream peer can provide, before tearing down the connection and reconnecting to another supplier. Mapping to the Bluetooth scatternet, each node may serve as a slave in one piconet only.

This design aims to minimize the cost of establishing Bluetooth connections, as well as guaranteeing the download speed at each mobile user. In a Bluetooth network, each piconet maintains a unique pseudo-random frequency hopping sequence (FHS) and time division duplexity (TDD) is used for transmission scheduling. If a mobile user participates in more than two piconets, time is partitioned for different transmissions. The effective transmission time of each connection could be reduced, leading to degraded download speed.

Second, mobile users with similar segment requests are maximally arranged into the same piconet, while whether the master node is willing to upload to those peers considering. In this way, broadcast opportunities within each piconet are maximally explored: master node x periodically collects segment requests from slave nodes, and prioritizes the sequence of segment broadcast according to the sum of download preferences, *i.e.* $\sum_{y \in \mathcal{Y}} r_y^i W(x, y)$, for each segment i from different slaves in set $\mathcal{Y}$.

Among the slaves in a piconet, some may serve as master nodes in other piconets (*i.e.*, the case of MSNs). Since MSNs may only be active in one piconet at a time (by synchronizing to its FHS), they could miss data broadcast from the piconet where it serves as a slave. To resolve this issue, An MSN may send the same request again, if it has not received a segment for some time.

3.2 Practical Protocols

We next discuss practical protocols to implement the design in realistic dynamic environments.

Joining the system. When a mobile user first joins the streaming system, he customizes his user profile, including upload preference, social attributes, and friend list. His Bluetooth device enters the inquiry mode and discovers other Bluetooth devices within the radio range. Then it inquires service records of other devices through Service Discovery Protocol (SDP). A Bluetooth device providing service publishes its service record enclosing a number of service attributes; some service attributes are common to all (*e.g.*, ServiceClassIDList, ProviderName), but others can be defined by the service providers themselves. [20] proposes an approach for information exchange by modifying the attributes of service records. In our implementation, we make use of a similar approach to exchange information for collaborative streaming, including upload preferences, social attributes, and segment availability.

Making use of the information acquired, a newly joined user identifies the most appropriate supplying peer to connect to, based on the design in Sec. 3.1. Then the user establishes a temporary connection with the selected supplier and sends a connection request. The supplier then invites the user into its piconet. If there are seven slaves including the new user, the supplier stops publishing its service record to prevent new connection request. According to the design on segment availability, the user will join a piconet in which slaves have similar requests.

Streaming video segments. After a new slave node joins in a piconet, he will send his segment request to the master node at the first time slot assigned to him. The master node in a piconet collects segment requests from slave nodes and schedules segment transmissions. At a Bluetooth device, the frequency-hopping rate is 1600 hops per second, and each time slot for a frequency hop is 625μs. A regular Bluetooth packet can carry at most 2745 bits of data. In the streaming system, the size of a segment is typically at a few Kbytes. Therefore, multiple time slots are needed to send a video segment in a piconet. For each receiver of the segment, it remains active and follows the frequency-hopping sequence in the piconet.

A slave node in a piconet listens to each packet from the master node at the beginning of each odd time slot, and identifies the destination of a packet based on its *LT_ADDR* field. A master node can broadcast packets to all slave nodes on the *ASB-U*, *PSB-C*, or *PSB-U* logical transport. When the master node is broadcasting, the *LT_ADDR* field of the packets are set to zero.

Switching download connections. As the streaming progresses, segment availability at each mobile user changes. Such updated information will be published by the user via updated service records.

Mobile users may alter their download connections when their current suppliers can no longer serve video segments they need, in cases when the slave user has performed some VCR operations during his video playback and when the master user moves beyond the radio range of the slave. When a change of download link is necessary, the mobile user tears down the connection in its original piconet, and restarts the procedure to search for a new supplier, following the similar service discovery steps as carried out by a newly joined user.

Detailed steps of our protocols are summarized in Algorithm 1.

4 Performance Evaluation

We evaluate *SMS* by implementing our design and protocols on Android 2.1 platforms. 6 HTC Wildfire mobile phones are used in our experiments. They are equipped with 528MHz Qualcomm MSM7225 processor, 512MB ROM, 384MB RAM, and a lithium-ion battery with 1300mAh capacity.

In our experiments, a 15MB video file is distributed, with a playback time of 320 seconds. One mobile device serves as the source, with the video file preloaded.

Algorithm 1 SMS work flow

```
Initialize user profile and segment request
procedure download procedure
if (request list≠ NULL) and (connection = NULL) then
  Identify the most appropriate supplying peer to connect to
  Join in the piconet in which selected supplier acts as master
  Send request list
  while connection ≠ NULL do
    if The coming segment is in the request list then
      Receive data from master
      Update request list
      Update supply information in service record
    end if
    if (supply list of master= NULL)or(request list= NULL) then
      Disconnect with master
    end if
  end while
end if
endprocedure
procedure upload procedure
Modify service record and start service
while (Service is not stopped) and (Supply list ≠ NULL) do
  if a slave disconnects connection then
    Clear the request information about that user
    Update the request information
  end if
  if new user joins in the piconet then
    if  the number of slaves = 7 then
      Stop service
    end if
    Receive request list from new user
    Update request information
  end if
  Identify segment with the highest weight and broadcast it
  Update request information
end while
endprocedure
```

The other mobile devices join the streaming network one by one, with joining times uniformly distributed within a total duration of 9 minutes. Upon joining, a user starts requesting and viewing the video from the beginning. The users are configured with different upload preferences, friend relationships, and social attributes: the type of each user is randomly chosen among the three (strongly socially selfish, weakly socially selfish, and altruistic); the number of friends and

the number of peers with matching attributes at each user are both randomly chosen from 1, 2, and 3. Given the limited number of mobile devices, we restrict the number of slaves in each piconet to two, in order to form a Bluetooth scatternet for evaluation.

4.1 Streaming Performance

We first evaluate the streaming performance at the mobile users, by measuring the time taken for each user to complete the video download and the total waiting time during the download due to seeking suitable segment suppliers. We note that the streaming rate of the video is $15MB \times 10^3 \times 8bits/320seconds = 375Kbps$, and data rate along Bluetooth connections is around 540Kbps. Therefore, segment download should be faster than the viewing progress, which guarantees smooth playback at the users. On the other hand, the total waiting time reflects the delay overhead incurred with our streaming protocols, due to service discovery (exchanging segment availability, social attributes, etc.), tearing down and establishing new connections.

We repeat our experiment for 10 times and plot the average results among all mobile users in Fig. 3 and Fig. 4, respectively. We observe from Fig. 3 that the average video download time is around 220 seconds, which is much less than the playback duration of the video, showing smooth playback can be achieved at the mobile users. Fig. 4 shows the percentage of waiting time during the entire download process is as low as 1.2%, exhibiting low delay overhead involved in our protocols.

We have also compared *SMS* with a P2P streaming protocol over mobile networks, by which mobile users select suppliers only based on segment availability and all users are willing to upload to anyone else. The results in Fig. 3 show that when more practical considerations on users' social selfishness are included (the case of *SMS*), the download performance is only slightly worse, as compare to the performance upper-bound achieved when all users are altruistic. Fig. 4 confirms the conclusion. In most case, the difference of the total waiting time between *SMS* and a P2P streaming protocol is small, bringing little influence on the user experience.

4.2 Power Consumption

We next evaluate battery consumption on the mobile devices when running our collaborative streaming protocol. In our experiments, the mobile devices are fully charged before start.

We first select four mobile devices with representative roles in the Bluetooth network, *i.e.*, a master node with one slave, a slave node, a master/slave node with one slave, and a master/slave node with two slaves, and monitor their battery levels when running *SMS* using *PowerTutor* [4].

Fig. 5 shows that the more nodes a mobile device is uploading to, the more energy is consumed. In addition, uploading consumes more energy than downloading. Nevertheless, we observe that the battery consumption during a time

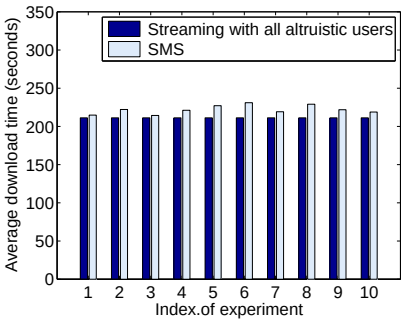

Fig. 3. Average video download time.

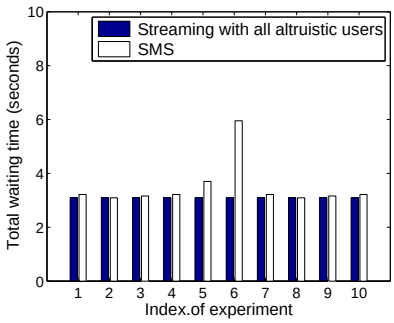

Fig. 4. Average waiting time during download.

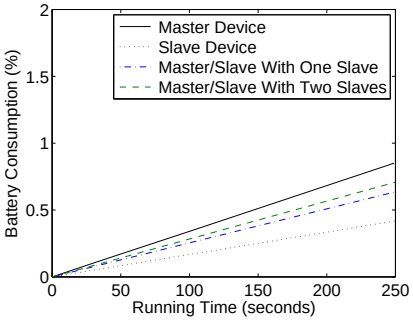

Fig. 5. Power consumption at devices of different roles.

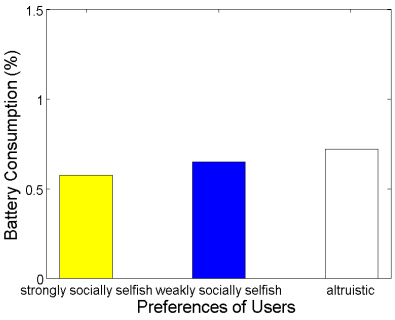

Fig. 6. Power consumption at devices of different upload preferences.

interval sufficient to download the entire video (about 250 seconds), is only less than 1% of the full battery power at all devices. This shows that *SMS* is effecient in energy consumption.

Next, we compare the battery consumption at mobile users with different upload preferences. We measure the total power consumption during a 250-second runtime at a user who wishes only to upload to friends, a user who can serve both friends and strangers with matching attributes, and a user who is willing to serve anyone. Fig. 6 shows that the altruistic user could spend 19% more battery power than a strongly socially selfish user. However, the overall energy consumption at each user is still lower than 1% of the full battery power. This shows that a mobile user can be less concerned with his battery consumption when participating in *SMS*, and should be more altruistic to boost the overall download speed in the system (the case of all altruistic users in Fig. 3).

5 Related Work

Mobile social applications that are natural extensions of online social networks have been extensively explored. In Micro-Blog [9], users generate geotagged data using their mobile phones, and share them with others via the Internet. Other

applications in this category include PeopleTones [12], Just-for-Us [11], MobiClique [16], FriendLee [5], and Social Serendipity [8].

In the other category, mobile users in physical proximity directly connect with each other. They store data (*e.g.*, profiles) locally in their mobile devices, and no central Internet server is involved. E-SmallTalker [20] is a mobile communications system facilitating social networking among people in physical proximity, and can discover and suggest topics of common interests for conversations. PeopleNet [15] presents an architecture that facilitates information seeking over a wireless virtual social network. SocialNet [14] is an interest-matching application that uses patterns of collocation to infer common interests between users over time. Besides mobile social application design, there exist other work exploiting social connections of users in their system design [10][13]. But none of them have categorized social relationships in the same way as we do.

There are some work exploiting the Bluetooth technology and protocols. [6][7][17] propose approaches to form stable Bluetooth scatternet, without taking into account the video streaming and social network scenario. Sewook *et al.* [19, 18] propose P2P streaming system designs over interconnected Bluetooth piconets. Their work focus on Bluetooth network construction without taking social relationship and selfishness of mobile users into consideration. In contrast, our work explores users' social connections and Bluetooth characteristics in an integral design, for voluntary collaborative streaming with high efficiency.

6 Concluding Remarks

This paper presents *SMS*, an architecture for VoD-type Streaming over Mobile Social networks, to achieve efficient collaborative distribution of short videos among mobile users. Our contributions in this paper are three-fold: *First*, we explore social connections among participants in the mobile social network, to facilitate voluntary collaborative streaming according to users' own upload preferences. *Second*, we construct streaming overlays by synergistically combining participants' preferences, video segment availability, and characteristics of the Bluetooth infrastructures, while effectively exploiting the broadcast nature of wireless transmissions. *Third*, we design detailed protocols to implement *SMS*, which efficiently tackles varies dynamics of the mobile users. We also implement our design on Android platforms and carefully evaluate the runtime performance of *SMS*. Preliminary results over a small-scale mobile network have shown satisfying streaming performance as well as low battery consumption with our protocols. In our ongoing work, we are implementing a practical Bluetooth emulation platform, in order to break the limitation of available hardware devices and carry out evaluations in a larger scale under realistic settings.

Acknowledgement

This research is supported in part by the University of Hong Kong Strategic Research Theme of Information Technology.

References

1. MagnetU (http://magnetucom/)
2. Cyworld (http://uscyworldcom/)
3. Tencent QQ (http://wwwimqqcom/)
4. PowerTutor (http://wwwpowertutororg/)
5. Ankolekar, A., Szabo, G., Luon, Y., Huberman, B.A., Wilkinson, D.: Friendlee: A Mobile Application for Your Social Life. In: Proc. of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services (2009)
6. Cano, J., Manzoni, P., Toh, C.K.: UbiqMuseum: A Bluetooth and Java Based Context-Aware System for Ubiquitous Computing. Wireless Personal Communications 38, 187–202 (2006)
7. Donegan, B.J., Doolan, D.C., Tabirca, S.: Mobile Message Passing Using a Scatternet Framework. International Journal of Computers, Communications & Control 3, 51–59 (2008)
8. Eagle, N., Pentland, A.: Social Serendipity: Mobilizing Social Software. IEEE Pervasive Computing, Special Issue: The Smartphone 4, 28–34 (2005)
9. Gaonkar, S., Li, J., Choudhury, R.R., Cox, L., Schmidt, A.: Micro-Blog: Sharing and Querying Content Through Mobile Phones and Social Participation. In: Proc. of ACM MOBISYS 08 (June 2008)
10. Jahanbakhsh, K., Shoja, G.C., King, V.: Social-Greedy: a socially-based greedy routing algorithm for delay tolerant networks. In: Proc. of the Second International Workshop on Mobile Opportunistic Networking (2010)
11. Kjeldskov, J., Paay, J.: Just-for-Us: A Context-Aware Mobile Information System Facilitating Sociality. In: Proc. of 7th International Conference on Human Computer Interaction with Mobile Devices & Services (September 2005)
12. Li, K., Sohn, T., Huang, S., Griswold, W.: PeopleTones: A System for the Detection and Notification of Buddy Proximity on Mobile Phones. In: Proc. of ACM MobiSys (June 2008)
13. Mei, A., Stefa, J.: Give2Get: Forwarding in Social Mobile Wireless Networks of Selfish Individuals. In: Proc. of IEEE ICDCS (2010)
14. Michael, T., D., M.E., Kathy, R., Darren, L.: Social net: using patterns of physical proximity over time to infer shared interests. In: Proc. of CHI '02 extended abstracts on Human factors in computing systems (April 2002)
15. Motani, M., Srinivasan, V., Nuggehalli, P.: PeopleNet: Engineering a Wireless Virtual Social Network. In: Proc. of ACM MobiCom (August 2005)
16. Pietiläinen, A., Oliver, E., LeBrun, J., Varghese, G., Diot, C.: MobiClique: middleware for mobile social networking. In: Proc. of the 2nd ACM workshop on Online social networks (WOSN) (2009)
17. Sewook, J., Chang, A., Gerla, M.: Performance comparison of overlaid bluetooth piconets (OBP) and bluetooth scatternet. In: Proc. of WCNC. pp. 505 –510 (apr 2006)
18. Sewook, J., Chang, A., Gerla, M.: Peer to peer video streaming in Bluetooth overlays. Multimedia Tools Application 37(3), 263–292 (2008)
19. Sewook, J., Lee, U., Chang, A., Cho, D., Gerla, M.: BlueTorrent: Cooperative Content Sharing for Bluetooth Users. In: Proc. of Fifth Annual IEEE International Conference on Pervasive Computing and Communications (2007)
20. Yang, Z., Zhang, B., Dai, J., Champion, A., Xuan, D., Li, D.: E-SmallTalker: A Distributed Mobile System for Social Networking in Physical Proximity. In: Proc. of IEEE ICDCS (2010)