RoadSpeak: Enabling Voice Chat on Roadways using Vehicular Social Networks

Stephen Smaldone, Lu Han, Pravin Shankar, and Liviu Iftode
Department of Computer Science, Rutgers University, Piscataway, NJ
{smaldone,luhan,spravin,iftode}@cs.rutgers.edu

ABSTRACT

A great number of people spend one or more hours each day driving between home and the office. These daily roadway commutes are highly predictable and regular, and provide a great opportunity to form virtual mobile communities. However, even though these commuters are already physically present in the same location, they are limited in their ability to communicate with each other. This paper presents a framework for building such communities, which we call Vehicular Social Networks (VSNs), to facilitate better communication between commuters driving on highways. As a proof of concept, we present the design of RoadSpeak, a VSN-based system which allows drivers to automatically join VSNs along popular highways and roadways, and communicate with each other by means of voice chat messages.

1. INTRODUCTION

Social Networks allow people with common interests to come together and form virtual communities. Mobile Social Networks connect people who are already together (sharing the same locality), enabling them to take advantage of their close proximity to form more tightly-coupled, possibly even ad-hoc, virtual communities. In this socially-driven virtual world, people can form "instant villages" that mirror and facilitate real-world interaction.

Today, services such as Dodgeball [5] utilize mobile phones to exchange information between users regarding their locality to identify opportunities to meet. Friends can find each other when they happen to be close by, and people can find others they have never met who share their interests, as well as, their location. Mobile social networking services are successful primarily because they directly take advantage of the, often periodic, physical locality patterns of the users to improve their knowledge about opportunities to socialize.

A heretofore unexplored source of physical locality, which can be taken advantage of for mobile social networking, is the daily roadway commute. A great number of people, especially in densely populated locales, spend one or more hours each day driving between home and the office. Since these commutes typically take people over highways and other frequented corridors, they are highly predicable and regular. Day after day, the same people travel along the same roadways at the same time. Therefore, the opportunity exists to form periodic virtual mobile communities. We call these virtual communities *Vehicular Social Networks*.

This paper presents a framework for building Vehicular Social Networks (VSNs) and, as a proof of concept, a VSN-based system, called RoadSpeak. RoadSpeak allows drivers to automatically join VSNs called *voice chat groups* along popular highways and roadways.

In our framework, a VSN such as a Voice Chat Group (VCG) is defined by a profile consisting of the triplet $\{time, location, interests\}$. Aside from interests, on which traditional social networks are formed, VSNs also include time and location in their definition, which may limit membership to specified roads and intervals of time. In RoadSpeak, the VCGs are maintained by a central group manager, although decentralized approaches can also be envisaged. The VCG profile is defined by the group owner (which can be the manager, an organization, or simply a user), when the group is created. Users are admitted to groups based on matching their user profiles to the group's profile. Admission can be free or it can be restricted by certain requirements, which the user must prove, such as a location frequency threshold for a specified roadway segment, during a specified time slot. By limiting group membership based on the users' roadway recurrence frequency, VSN-based groups may be limited to commuters, the best target for this system, rather including than transient travelers. Additionally, time and location properties can be used to provide regularity in a group's membership, i.e., the same group of people are in contact with each other on a daily basis with new members only being admitted once they have met the frequency requirement.

The primary goal of RoadSpeak has been to facilitate better communication between people who are already physically present in the same location, but who are limited in their ability to communicate with each other. In most social scenarios, for example, a cafe, nightclub, etc., people can easily communicate with each other. They are not restricted by their activity, as they are when they are driving on highways or other roadways. Today, if people wish to talk to each other while occupying a vehicle, they must either be collocated in the same vehicle, join a radio talk show, or initiate a direct connection to each other (e.g., make a cell phone call.) RoadSpeak takes advantage of the common situation drivers on the same road segments share, to allow opportunities for these people to socialize and communicate in an easy and natural manner.

We believe that the most likely target for VSNs in general and for RoadSpeak in particular are daily commuters. These people would, over time, establish virtual communities along the various roadways that constitute their individual routes to and from work. It is the regularity and frequency of commuter travel that enables these communities. For example, groups might be

formed for drivers to discuss recent social and political issues, to discuss sports, or any other interesting topics. There might also be groups established to coordinate carpooling, to communicate roadway events (e.g., accidents, congestion, etc). Moreover, there is also value in accommodating transient travelers, and we envision "public" chat channels to exist to accommodate them. For example, local towns or states might establish groups to discuss interesting local events or points of interest along popular highways, which transient travelers might use to query for advice in selecting a local restaurant or hotel. Finally, the VSNs can become particularly useful in case of emergency for drivers to help each other, and can constitute trusted network overlays for any other distributed application.

To the best of our knowledge, this paper is the first one to introduce the idea of Vehicular Social Networking. We present the design of a centralized system for automatically forming VSNs based on time, location, and interests. We describe a particular VSN system called RoadSpeak, a voice chat system for drivers on the road, which we preliminarily prototyped using 3G-based cellular communication.

The remainder of this paper is organized as follows. Section 2 presents the RoadSpeak scenario to illustrate the benefits of the application and VSNs in general. Section 3 presents the RoadSpeak system design and Section 4 describes the RoadSpeak prototype implementation and evaluation. Related work is presented in Section 5 and the paper is concluded in Section 6.

2. ROADSPEAK SCENARIO

To illustrate our VSN vision, in what follows, we describe a usage scenario for our RoadSpeak project. Joe would like to start participating in RoadSpeak during his daily commute. From his home computer, he goes to the RoadSpeak portal to create an account and to download the RoadSpeak client to his smart phone. While he is logged into the system from home, he browses through the various groups available along the route he takes between home and work. He chooses to join NJ Turnpike's 6 AM - 7 AM and 7 AM - 8 AM groups, since he is usually on the highway from 6:30 AM - 7:15 AM. He also joins a 7 AM - 8 AM group on Route 1, since he drives on Route 1 after exiting the highway. Then, he does something similar for his return home in the evening.

The next morning, Joe gets into his car, sets his smart phone in its cradle and puts on his hands-free kit, for safe driving. As he travels to work, the RoadSpeak client running on his smart phone tracks his location via the built-in GPS receiver, and when he enters NJ Turnpike at 6:32 AM, the client contacts a RoadSpeak server and adds Joe to the group. Joe receives an audible alert from his phone, notifying him that he is joining the chat group. Joe introduces himself to the group and begins listening to the current voice chat messages that his RoadSpeak client is downloading from the server.

After Joe listens to the group, for a while, he intervenes and speaks to the group. The RoadSpeak client captures and transmits his voice message to the server. As the server continues to send out messages in the sequence it received them in, it reaches Joe's message and distributes it to the other users currently participating in the voice chat group. Two people, Fran and Harry, hear Joe's question to the group and respond, about the same time. Their responses are ordered by the server based on the arrival time, and scheduled to be delivered in this order to everyone in the group, including Joe. Just prior to 7 AM, the RoadSpeak client annnounces, though an audible alert, that Joe's current group is about to end. At 7 AM, the client removes Joe from his current group and adds him to the second group in which he has chosen to participate. Finally, when Joe leaves the NJ Turnpike, the RoadSpeak client running on his smart phone, detects his departure and notifies him that he will leave the chat group, shortly. Joe transmits a farewell message to the group, and continues along his route to work on Route 1, where he joins a new group.

3. DESIGN

In this section, we describe the RoadSpeak design. First, we describe our general model for Vehicular Social Networks (VSNs) and then the specifics of the RoadSpeak system design.

3.1 Vehicular Social Network Model

A Vehicular Social Network (VSN) is a social network of users who traverse popular roadways. This section defines our model of VSNs and describes the issues related to VSN creation, enforcement, privacy, and infrastructure requirements.

VSN Creation: VSNs are created by specifying its profile: *(i)* the time interval, *(ii)* location/road, *(iii)* and group's interests. For example, a user might create a VSN of users for the time slot starting at 4 PM and ending at 6 PM. This VSN would be composed of users who are connected to the system between 4 and 6 each day. Similarly, a user could define a VSN for a specific section of roadway, and this VSN would include any user who is currently located on that section of roadway. Location properties could be specified using either coordinates (i.e., GPS values) or some logical description of the location (e.g., miles 501-576 on Route 1). Finally, VSNs can be specified using group's interests. These are user-defined keywords that provide a meaningful definition of the purpose or collective interests of the users included in the VSN. For example, a user may create a VSN to include users who are interested in political discussion.

Once a Vehicular Social Network has been specified, it is generated based upon the properties defined in the specification. VSNs can be generated and advertised centrally or in a distributed fashion. In the centralized method, specifications are submitted to a central system that generates the VSN and advertises it to all participating users. In the decentralized method, VSNs are created local to users and advertised in a peer-to-peer fashion. There is a trade-off between these methods, which are beyond the scope of this paper.

Location Validation: Certain VSNs may enforce membership to satisfy time, location, or other elements specified in the VSN profile. Time constraints are straight-forward to handle, however, location properties are substantially more difficult to enforce. A user that provides false location information would be able to participate in a VSN, contrary to the VSN specification properties. Therefore, location information must be validated. This can be performed through direct validation, for example, using road-side infrastructure to periodically sample user positions. Alternatively cross validation could be performed between users. For example, users might report the positions of other users local to them, along with their own position. Then, by comparing the various reports for a specific location, users

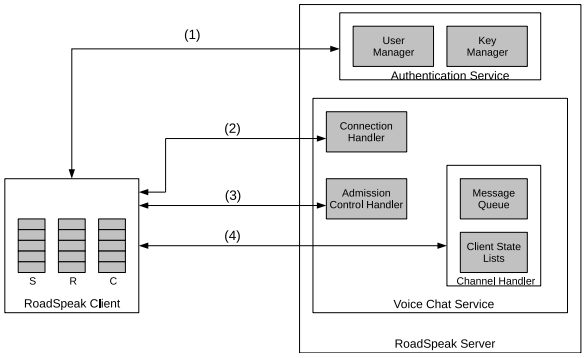

Figure 1: RoadSpeak vehicular voice chat application overview.

could validate each other.

Privacy: Since a VSN may use per-driver time-location information, there is an obvious privacy issue beyond what traditional social networks must handle. The privacy requirements for VSNs will differ based on their specification properties. A VSN model that includes cross-validation will directly expose user location information to users around them. Therefore it may be important to include mechanisms to anonymize the data with respect to the individual users, while only exposing location information to the central authority [16]. Alternatively, a model that does not include cross-validation would only require users to expose location information to a centralized, trusted, authority. In either case, this data must be handled as highly sensitive to its users and must not directly leak this information between users or allow indirect inferences based on the information that is exposed.

Infrastructure requirements: There is a broad range of infrastructure that can be assumed to support Vehicular Social Networks. First, we assume that vehicles are equipped with some form of wireless communication. This may include wireless-enabled smart phones (e.g., 3G data access, WiFi, WiMAX, etc.), or vehicles may be equipped with embedded vehicular computing systems, which support car-to-car communication (C2C). External to the vehicles, we can either assume a fully deployed road-side infrastructure, which supports car-to-infrastructure (C2I) communication, no infrastructure, or some deployment model in between. The infrastructure assumptions we make will have a substantial impact on the design of any VSN-based system.

3.2 RoadSpeak Voice Chat

Figure 1 shows an overview of the RoadSpeak vehicular voice chat application. The following subsections describe the key application modules and how they form, and operate as Vehicular Social Networks.

Central to RoadSpeak is the concept of a voice chat group. We define a voice chat group to be a VSN composed of users who wish to communicate with each other in near real time over a voice communication system similar to a combination of voice over IP (VOIP) and Internet chat rooms. The closest applications similar to this are Internet voice chat systems, such as Ventrilo [20] or TeamSpeak [18].

3.2.1 RoadSpeak Overview

User creation and authentication: Users download a client to their smart phones (or embedded vehicular PCs) and use the client to create an account in the system (see (1) in Figure 1). To access groups users login through the client. Both user account creation and login authentication are handled over a secure web protocol (i.e., HTTPS) by the User Manager. At the time of account creation, the client generates an asymmetric cryptographic key pair (using PKI) and registers its public key (K_{PUB}) with the User Manager.

Once a user has been authenticated to the system, they are free to create new groups, to invite other users to the groups they own, or to request membership to groups owned by other users. Group invitations and join requests are forwarded to other users (potential participants and group owners, respectively) to either accept or decline. In the case where a group is owned by multiple users, then the owners vote to either allow or deny new users into the group.

Group membership: Voice chat groups can be defined as public or private. Anyone is free to join a public group so long as they satisfy the group properties, while private groups require the explicit permission of the group owner. Groups can also be defined as open or moderated. In an open group, anyone can freely communicate with members of the group, whereas in a moderated group, permission to communicate is explicitly granted by the group owner and can be revoked at any time. Finally, a group can be hidden or advertised. Hidden groups are only advertised to those users who are allowed to participate in the groups.

Admission to a group is handled by the Group Admission Manager. To be admitted to a group, a user submits his profile. This data allows the Group Admission Manager to verify that the user attempting to join the group, actually satisfies the group's properties. If the Group Admission Manager validates the user, and allows him to join the group, the group key (K_G) is transmitted back to the client to be used by the client in the future for that group.

Connection handler and admission control: A user chooses which group he wishes to participate in, once he has logged into the RoadSpeak system (through the local client on his smart phone). A client may only enter one RoadSpeak chat group at a time. When a RoadSpeak client connects to a RoadSpeak chat application server (see (2) in Figure 1), the Connection Handler spawns a new Admission Control Handler (ACH) and hands the connection to the ACH to perform admission control (see (3) in Figure 1). The Connection Handler may also choose to deny connection for a client, in the event that the maximum number of clients have already connected. Additionally, should a single client attempt to open two concurrent connections to the RoadSpeak chat server, the Connection Handler would detect this and unconditionally terminate any connection beyond the first.

To initiate Admission Control, a RoadSpeak chat client transmits the the group name, the user's name, profile, and the SHA1 hash of the group key ($sha1(K_G)$) to a vehicular chat server. This information is encrypted by the client, using its private key (K_{PRIV}), prior to being sent to the server. The server validates the user based on the user's information and the hash of the group key submitted by the user. If validation is successful, the server sends back an acknowledgment to the client. The ACH then hands the client connection off to the Channel Handler for the requested VSN group (see (4) in Figure 1). If validation fails, then the server transmits a "request denied" message to the client, and terminates the connection.

API Function	Function Description
$createUser(< username >, < cprofile >)$	Create user named $< username >$ in the VSN and set its initial profile to $< cprofile >$.
$getUserProfile(< username >)$	Fetches the user profile data for $< username >$.
$setUserProfile(< username >, < cprofile >)$	Updates the user profile data for $< username >$ to $< cprofile >$.
$createGroup(< groupname >, < gprofile >)$	Create a VSN group named $< groupname >$ in the VSN and set it's initial profile to $< gprofile >$.
$getGroupProfile(< groupname >)$	Fetches the group profile data for $< groupname >$.
$setGroupProfile(< groupname >, < gprofile >)$	Updates the group profile data for $< groupname >$ to $< gprofile >$.
$enumGroups()$	Enumerates the current list of existing groups.
$enumUsers(< groupname >)$	Enumerates the current list of users connected to the group $< groupname >$.
$lookupGroupKey(< groupname >)$	Fetches the current public group key for $< groupname >$ group.
$generateGroupKey(< groupname >)$	Generates a new group key pair for group $< groupname >$.
$getModList(< groupname >)$	Fetches the current list of moderated users connected to the group $< groupname >$.
$setModList(< groupname >, < modlist >)$	Updates the current list of moderated users for group $< groupname >$.

Figure 2: RoadSpeak API.

Channel handler: The Channel Handler spawns one thread per client connection. All of the Channel Handler threads share a global data structure, called the Message Queue. The Message Queue maintains a list of message chunk rows ($< message\ \#, chunk\ \#, chunk\ data, username, statusflags >$). Each row represents a chunk of a voice chat message that has been submitted to the server by a user participating in the group chat channel. Messages in this list are maintained in the order in which they were received, such that they are sent out to each client, one message at a time in the same order.

Each Channel Handler thread also maintains a local data structure called the Client State List, which maintains a set of client state rows ($< message\ \#, chunk\ \# >$). There is one row per message being sent to the client. Each row tracks the last message chunk sent to and acknowledged by the client.

Message handling: Each RoadSpeak client maintains a set of three message queues: *(i)* send queue, *(ii)* receive queue, and *(iii)* control queue. These queues are used to buffer outgoing and incoming voice chat messages, and for flow control.

Send handling. Voice messages are captured by the client's Voice Capture thread (e.g., using the smart phone microphone). Captured voice messages are then broken into fixed sized message chunks, encrypted with the group key (K_G), and placed into the client send queue. The client's Send Handler thread streams message chunks to the server, and the server responds with acknowledgments. Once the client receives an acknowledgment from the server for a particular message chunk, that chunk is removed from the send queue. Note that similar to TCP flow control, when a client receives an acknowledgment for a message chunk, this acknowledgment also applies to any chunks sent earlier but not yet acknowledged.

Receive handling. Once the server receives a message chunk, it places the message chunk on the global Message Queue and sends a message chunk acknowledgment back to the sending client. Each Channel Handler thread accesses this queue and streams chunks to its associated client. The client's Receive Handler thread receives the message chunk, places it into the client's receive queue, and sends an acknowledgment back to the server. Once the server's Channel Handler receives the acknowledgment, it updates the $statusflags$ for the message chunk acknowledged, signifying that it was received by the Channel Handler's associated client. It also checks to see if all clients have received this message chunk (by inspecting the $statusflags$) and will remove the chunk from the Message Queue, in that case.

Message sequencing and flow control. The client's Voice Playback thread is notified by the receive thread whenever all chunks for the lowest numbered message have been received and the message is complete. The Voice Playback thread then removes chunks, one at a time, from the receive queue, decrypts them using the group key (K_G), and streams them in order to the user (e.g., using the smart phone's speaker). In this way, messages are played back to all users in the same sequence that they were received by the server.

Flow control is handled in a similar fashion to the TCP protocol. When a client connects to the server, it exchanges window sizes (in terms of total number of message chunks) with the Channel Handler. Additionally, window updates are exchanged with acknowledgments. Even though the Channel Handler can choose to send message chunks from different messages to the client out of order, the Channel Handler knows the remaining number of chunks to be sent to its associated client for the next message in the Global Message sequence, and it will only send chunks from other messages when it has ample client window space to complete the next message. Otherwise, it would be possible for a deadlock to occur between the client and server.

The server Channel Handler threads can also send $pause$ and $resume$ messages to their associated clients. These higher-priority messages are placed on the clients' control queue when received and are handled by the receive thread. When a $pause$ is received from the server, the client receive thread notifies the client send thread, to pause all chunk streaming from the client to the server. Upon receiving a $resume$, the client receive thread notifies the client send thread to begin chunk streaming, once again.

User event notification: There are a number of conditions that must be reported to the user as feedback, during a voice chat session. For each of these, the client provides audible and visual feedback to the user. For example, when the send queue has been filled the client notifies the user to wait until the client makes progress in sending, such that everything spoken by the user is properly captured and sent by the client. Due to the size constraints of this paper, we do not enumerate or further describe these types of conditions.

Group moderation: RoadSpeak chat groups can be created with moderation privileges for the group owner. For the voice chat system, enabling moderation on a group gives the owner the ability to silence other members of the group. When this is enabled, the server sends a $pause$ message to the silenced client. Additionally, should the client ignore the $pause$ message, the server can either discard messages from the client or chose to terminate the client session. When a client is silenced through moderation, the user is notified via a visual and audio alert in a similar fashion to other user event notifications.

Application Programming Interface: Figure 2 lists the API that the RoadSpeak server exports for use by clients. This API can also be used to extend RoadSpeak clients to provide enhanced functionality. The figure provides a description for all of the functions available, at this time.

3.2.2 Time-Location Traces

In certain scenarios groups may require members to submit location proofs, and this can be done by having clients perform discovery of each other over WiFi, and submit location proofs to the server for cross-validation. This method assumes both the existence of WiFi at each client, and a large enough concentration of clients close to each other such that enough discovery can occur to perform the cross validation.

Time-location trace collection: Time-location traces are collected by individual clients for drivers as they travel along roadway segments included in the system. Location is determined through the use of GPS. For any such roadway segment, a RoadSpeak client polls for the presence of other participating drivers in the same location, utilizing inter-vehicular wireless communication. The client keeps a list of wireless identifiers (e.g., MAC addresses) per road segment/time slot, in a local state store. For each entry in the list, the client maintains a counter. This counter is incremented each time the client discovers the same identifier during the same time slot while traveling on the same road segment, once per day. So, for example, if two participating drivers happen to be within the same location/time slot for two days, the counter each client maintains about the other would be 2.

In addition to the information gathered about others, each client keeps the same information for itself. Each time a driver enters a known roadway segment, the client increments the appropriate counter, based on the time and location. Finally, whenever a client leaves a known road segment, it transmits updated traces for the roadway segment it has just left.

Trace submission and cross-validation: Once a client has joined a RoadSpeak group, it may be required to submit time-location traces in order to be admitted to that group. This depends on the properties specified for the group, which also defines the frequency thresholds, in terms of time-location that must be met. For example, a group may require that its members travel on the relevant roadway segment(s) for the group 4 out of 5 weekdays, each week, in order to be admitted to the chat group.

Once a trace has been submitted to a RoadSpeak server by a client, the server performs three checks on the trace. First, the server decrypts the trace using the client's registered public key (K_{PUB}). Next, the server check that the trace satisfies the matching criteria for the group: *(i)* the trace is for the correct roadway segment and time slot, and *(ii)* the client's counter meets the group threshold requirement. Finally, the server performs cross-validation on the traces using historical data stored from previous trace submissions from other clients to the server.

Cross-validation is utilized to verify that the traces provided by any client are authentic. To do so a server searches its historical trace data set to verify that other clients who were physically located on the same roadway as the submitting client, discovered the submitting client. In this way, each client provides proof for the clients it discovers.

There are two issues in cross-validation that must be handled. First, it is possible that a client submits traces that cannot be completely cross validated. This may occur due to issues in wireless communication, where Client A discovers Client B, but Client B does not discover Client A. Therefore, we only require a client to meet a minimum threshold in terms of the number of clients that validates the submitting client's presence. Second, colluding clients could cross-validate each other, thereby circumventing the admission control policy. At the present, we consider this to be outside the scope of this work, but acknowledge the issue, nonetheless.

4. IMPLEMENTATION AND PRELIMINARY RESULTS

We have implemented the RoadSpeak system using Java and tested it using laptops with Verizon EVDO PC cards which provide 3G connectivity. We deployed 2 clients that periodically send their location and neighborhood traces to the server over 3G. One of the clients forms a RoadSpeak group, and the other client shared its route, and hence becomes a part of the group, and the two clients can send messages to each other.

Our preliminary results show that the 3G network bandwidth is able to successfully transfer voice messages at a sampling rate of 32 Kbps. However, owing to the unstable 3G connection, at times, we noticed large spikes in propagation delay of these voice messages. Since RoadSpeak is less sensitive to packet loss, but typically very sensitive to delays, we are working on an RTP/RTCP based implementation of RoadSpeak for exchanging streaming voice. We are also working on more detailed scalability experiments to study the effect of a large number of clients on the 3G network, as well as, the server and the client queuing delay.

5. RELATED WORK

Traditional online social networks such as Facebook [7] and LinkedIn [11] are essentially a virtual view of physical communities, however they are agnostic to location and proximity. Recently there has been an increasing trend towards mobile social networks that can help people connect with their physical communities and surroundings. Social Net [19] infers shared interests between people by storing the IDs of the nearby devices and analyzing this co-location data collected over a long period of time. Social Serendipity [6] is another similar mobile phone-based application using Bluetooth and a database of user profiles to recommend face to face interactions between nearby users who share common preferences. MobiSoc [2] is a middleware that enables formation of mobile social networks and provides a platform for managing the social state of physical communities.

Vehicular networks have become increasingly popular in the recent past, due to the interest from government, as well as, auto manufacturers. Various types of applications that make use of vehicular networks have been proposed, ranging from improving safety [13], traffic management [22], emergency management [17], urban sensing [9], multimedia sharing [15], gaming [14], to disseminating advertisements to passengers [10]. All existing work in the field of vehicular networks consider links between vehicles to be spontaneously formed as a result of the vehicles being in wireless range of each other. To the best of our knowledge, ours is the first work that defines the idea of vehicular social networks based on the social links that can exist

between passengers in different vehicles.

Vehicular networks are a special case of mobile adhoc networks (MANETs), as well as, delay tolerant networks (DTNs). In the field of MANETs and DTNs, various works have proposed applying knowledge about social networks in order to improve networking protocols performance. In [4], the authors propose a routing algorithm for MANETs that, based on the small world phenomenon seen in social networks, identifies bridge nodes for routing based on their centrality characteristics. [12] shows how knowledge of nodes' social interactions can help improve performance of mobile systems by means of three example systems, a DTN routing protocol, a firewall, and a mobile P2P file sharing system. [1] studies the impact of users' social relationships and movement patterns on the performance of routing protocols in opportunistic networks. Similarly, a social network based overlay for publish-subscribe communication in DTNs has been studied in [21]. In this work, the authors study human contact patterns using human connectivity traces based on bluetooth and cellular phones. None of the existing works, however, consider vehicular networks as the underlying network that can make use of a social network overlay.

A number of projects have described ways of spontaneously creating mobile social networks in different network settings. [3] describes a middleware for managing location-dependent relations in mobile social networks and for propagating the social networks' visibility up to the application layer. The Haggle project has performed [8] extensive studies of human mobility traces in opportunistic networks and extracted information related to clustering, and community structure.

Our work is distinct from all the earlier work in that RoadSpeak is the first application built on a vehicular network with an underlying social network overlay. We believe that our work will inspire existing vehicular applications to leverage the underlying social connections that exist in vehicular networks and form vehicular social networks.

6. CONCLUSION AND FUTURE WORK

We have proposed a novel framework called vehicular social network, using which different people traveling along the same roadways at the same time can form virtual mobile communities. We presented the design of RoadSpeak, an inter-vehicle voice chat system that allows users to automatically join voice chat groups along these roadways. We presented mechanisms to discover and form these groups, which utilize both time and locality, along with other user preferences. As future work, we plan to evaluate our RoadSpeak prototype on the roads with a larger number of cars.

7. REFERENCES

[1] C. Boldrini, M. Conti, and A. Passarella. Impact of social mobility on routing protocols for opportunistic networks. In *Proc. of The First AOC Workshop*, 2007.

[2] C. Borcea, A. Gupta, A. Kalra, Q. Jones, and L. Iftode. The mobisoc middleware for mobile social computing: Challenges, design, and early experiences. In *Proceeding of Mobilware*, 2008.

[3] A. S. N. Context-Aware Middleware for Anytime. Dario bottazzi and rebecca montanari and alessandra toninelli. In *IEEE Intelligent Systems Volume 22, Issue 5, Pages 23-32*, 2007.

[4] E. Daly and M. Haahr. Social network analysis for routing in disconnected delay-tolerant manets. In *Proceedings of MobiHoc*, 2007.

[5] Dodgeball. `http://www.dodgeball.com/`.

[6] N. Eagle and A. Pentland. Social serendipity: mobilizing social software. In *In Pervasive Computing, IEEE, volume 4(2), pages 28-34*, 2005.

[7] Facebook. `http://www.facebook.com/`.

[8] P. Hui, E. Yoneki, S. Chan, and J. Crowcroft. Distributed community detection in delay tolerant networks. In *ACM SIGCOMM Workshop MOBIARCH*, 2007.

[9] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. K. Miu, E. Shih, H. Balakrishnan, and S. Madden. CarTel: A Distributed Mobile Sensor Computing System. In *4th ACM SenSys*, Boulder, CO, November 2006.

[10] S.-B. Lee, G. Pan, J.-S. Park, and M. Gerla. Secure incentives for commercial ad dissemination in vehicular networks. In *Proceedings of MobiHoc*, 2007.

[11] Linkedin. `http://www.linkedin.com/`.

[12] A. G. Miklas, K. K. Gollu, K. K. W. Chan, S. Saroiu, K. P. Gummadi, and E. de Lara. Exploiting social interactions in mobile systems. In *Proceedings of the Ninth International Conference on Ubiquitous Computing (Ubicomp)*, 2007.

[13] T. Nadeem, S. Dashtinezhad, C. Liao, and L. Iftode. Trafficview: A scalable traffic monitoring system. In *Proceedings of 2004 IEEE International Conference on Mobile Data Management (MDM)*, 2004.

[14] C. E. Palazzi, M. Roccetti, S. Ferretti, G. Pau, and M. Gerla. Online games on wheels: Fast game event delivery in vehicular ad-hoc networks. In *Proceedings of V2VCOM*, 2007.

[15] J.-S. Park, J.-S. Park, J. Yeh, G. Pau, and M. Gerla. Codetorrent: Content distribution using network coding in vanets. In *Proc. of MobiShare*, 2006.

[16] N. Ravi, M. Gruteser, and L. Iftode. Non-inference: An information flow control model for location-based services. In *In the Proceedings MOBIQUITOUS*, 2006.

[17] M. Roccetti, M. Gerla, C. E. Palazzi, S. Ferretti, and G. Pau. First responders' crystal ball: How to scry the emergency from a remote vehicle. In *IPCCC 2007 / NetCri07 IEEE International Workshop on Research Challenges in Next Generation Networks for First Responders and Critical Infrastructures*, 2007.

[18] Teamspeak. `http://www.goteamspeak.com/`.

[19] M. Terry, E. D. Mynatt, K. Ryall, and D. Leigh. Social net: Using patterns of physical proximity over time to infer shared interests. In *Proc. Human Factors in Computing Systems (CHI 2002), pages 816-817*, 2002.

[20] Ventrilo. `http://www.ventrilo.com//`.

[21] E. Yoneki, P. Hui, S. Chan, and J. Crowcroft. A socio-aware overlay for publish/subscribe communication in delay tolerant networks. In *10th ACM/IEEE MSWiM*, 2007.

[22] J. Yoon, B. Noble, and M. Liu. Surface street traffic estimation. In *Proceedings of MobiSys*, 2007.