Protocol and Heuristics for Synchronizing Opinion Poll Items in Vehicular Ad-hoc Networks

Osamah Dhannoon and **Rahul Vishen** and **Marius C. Silaghi**
Florida Institute of Techology

Abstract

While drivers are not expected to vote while driving, VANETs can be an excellent media for dissemination of decentralized regional polls and for pre-recorded votes/opinions on regional issues in a decentralized opinion poll. We propose and evaluate heuristics for scheduling messages in a VANET broadcasting-based dissemination of decentralized opinion polling data among self-interested participants. The goal of the heuristics is to increase dissemination of the polling data and results under the given assumptions. The self-interest of participants is assumed to be manifested by selectivity in the storage and forwarding of topics and opinions for those topics.

The report starts by describing the concepts enabling the fully decentralized organization of the polls. The underlying protocol that we implemented for fully decentralized polling of opinions over VANETs is also introduced and evaluated.

Introduction

A protocol is proposed for dissemination of decentralized opinion polling over wireless, Vehicular Ad-hoc Networks (VANETs). When regional opinion polls are organized in a decentralized fashion, vehicle to vehicle (V2V) communication can be exploited for exchanging pre-recorded votes in neighborhoods (without the drivers being required to interact while driving).

VANETS are composed of wireless devices found in moving cars. Each of these devices can communicate with other devices found in its proximity. Common devices with powerful receivers can record messages sent from emitting devices found hundreds of meters away. A fully decentralized polling can be based on a decentralized authentication and census mechanism. Each device is owned by a self-interested user and we assume that the system is open, which implies that a user has full control over her device and its software.

Since they have full control, self-interested participants can refuse to store and forward information related to polls in which they are not interested. They can also refuse to store and disseminate opinions that they do not share. The communication model assumes that each device broadcasts data it wants to disseminate and simultaneously listens and processes data broadcast by passing-by devices. A challenge is to design heuristics for selecting what to emit to maximize dissemination of polling data under the working assumptions.

We evaluate heuristics that broadcast data either with uniform randomness, or favoring certain types of items such as: new votes, personal votes, votes similar to the personal votes or the intersection between the interests of the sender and the ones of potential receivers. Some input for these heuristics may come from information about interests of peers, and potentially their GPS location and velocity (bearing and speed). For efficiency, once packed, data can be broadcast several times. A set of queues are maintained to implement these heuristics.

To enable comparison between the described heuristics, a utility model is introduced where the dissemination of each item is associated with a numerical value. For example, the utility value for disseminating personal votes and opinions can be considered to be the highest, followed by the utility value for disseminating votes with similar choices as the personal ones. The average utility value for disseminating opposing opinions is assumed smaller, but for various users it can be either positive or negative (based on whether they want their choice to succeed by any mean, or they are principled and ready to submit to the opinion of others, or they are open and willing to learn from other's justifications and to eventually change their minds). The utility for disseminating votes on which the current user abstains can be assumed in certain experiments to have an average value between the utility for similar opinions and opposing opinions. The impact of the actual numerical ranges of these utilities on results can also be evaluated.

After presenting the background and related work, we continue by introducing the concepts involved in decentralized polling and a sample data model for the storage of each node. Subsequently we present the protocol for broadcasting in terms of message components and their semantic. In section Heuristics we discuss the tested techniques and the involved data structures. After describing the preliminary experimental settings and results, we end with conclusions.

Background

The use of broadcast in high traffic areas is known to be challenging due to high rates of transmission collisions between data packets. This problem is known as the *broadcast storm problem* (Tseng *et al.* 2002). Several broadcasting protocols (such as DV-CAST) have been proposed to increase the performance of data transfer in various traffic scenarios for VANET applications (Kumar & Dave 2012; Tonguz *et al.* 2007; EUREKA 2010). A statistical study of broadcasting between mobile nodes based on requests is available in (Karlsson, Lenders, & May 2007) and implemented in Bluetella.

The Local Peer Group (LPG) clusters neighboring nodes to restrict dissemination range (Chen & Cai 2005). P2P sharing of content over VANETs based on data popularity is introduced in the Roadcast simulator (Zhang, Zhao, & Cao 2009). It simulates delivery of relevant data (such as MP3 audio files) based on peer queries by applying information retrieval mechanisms. A VANET P2P file sharing protocol called SPAWN (*gossiping*) is introduced by (Nandan *et al.* 2005). Implementing CarTorrent in a real world scenario is reported in (Lee *et al.* 2007) which describes field tests for the SPAWN protocol and exchanges file chunks based on the AODV protocol.

A set of so called *Road-Based Vehicular Traffic (RBVT)* routing protocols on city roads use current traffic data to initiate the end-to-end communication paths (Nzouonta *et al.* 2009). VANET data dissemination can provide vehicles with parking spots availability (Caliskan, Graupner, & Mauve 2006). The Traffic View (Nadeem *et al.* 2004) project uses VANET communication to share traffic information among cars moving on roads. It can disseminate road assessments (such as foggy weather) helping to find the best route to a destination. The system aggregates data in packets, to increase efficiency.

CodeTorrent (Lee *et al.* 2006) is another protocol for P2P file sharing over VANETs. It aims to decrease file downloading time.

The Segment-Oriented Data Abstraction and Dissemination (SODAD) (Wischhof, Ebner, & Rohling 2005), aims to increase the communication range between vehicles for exchanging traffic safety data, and utilities information (e.g. locating gas stations). SODAD is used in the Self-Organizing Traffic Information System SOTIS.

Concepts

In this section we introduce formally the concepts that can make possible a fully decentralized opinion polling process. They define the data items exchanged by the proposed protocols.

Polling is a process whereby one gathers the opinion of a sample from a certain population on a given question. For the statistical relevance of the result it is important that only opinions of the members of the targeted population are recorded and that each of them is recorded only once. Minor errors can still leave the results relevant in case the support of competing opinions differs by large margins. However, it should be difficult to systematically manipulate the outcomes.

Definition 1 (Peer) *The set of software agents that coordinate publicly to represent a given user is referred here as* peer*. A peer may have agents running on various devices (laptops, desktops, phone of a user) and which share the same public and secret key pair. The peer is globally identified by its public key.*

Each semantically independent type of item (vote, justification, etc.) is uniquely identified by a *Global Identifier (GID)*. To guarantee that there is no voluntary or involuntary conflict between GIDs, these are built either as public keys for a secret, or as digests of the information that they represent.

A working definition of the population eligible for a set of polls, is captured in the concept of organization.

Definition 2 (Organization) *An organization is an entity defining the mechanism whereby an authority is defined for specifying and controlling eligibility for voting on a set of issues. An organization is defined by the unchangeable set of parameters describing its governance and function. This unchangeable characteristic is captured in its global identifier.*

The fact that an item is generated for this organization is specified by tagging the item using the *global identifier* (which is included in the digitally signed data for the item). The parameters of the organization can specify the criteria for eligibility to vote on items, and the ontology for the related communication.

Decentralized governance is currently rare. Some organizations are appropriate for a decentralized governance, such as: a diaspora, a union, or a club.

Definition 3 (Grassroot Organization) *A grassroot organization is a tuple $\langle \mathcal{O}, p \rangle$. The global identifier $\mathcal{O}$ is the digest of a set of parameters p, which can never change and are referred to as its* constitution*.*

Other organizations are more appropriate for centralized governance, such as: a class of students specified by their instructor, a faculty coordinated by the department head, or a committee coordinated by its chair.

Definition 4 (Authoritarian Organization) *An authoritarian organization is a tuple $\langle \mathcal{O}, p, r, d, s \rangle$. The global identifier $\mathcal{O}$ is the public key of the authority, which can change any of its parameters, and is referred to as the* dictator *of the organization. The parameters of the organization are specified by p, the revocation status by r, and the date of this declaration by d. A signature s is used by the authority to certify this data: $s = SIGN(SK(\mathcal{O}), \langle p, r, d \rangle)$.*

The participants defined by an organization as eligible to vote on its issues, and whose votes matter with a predefined weight in decisions, are referred to as its *constituents*.

Definition 5 (Constituent) *A constituent of an organization is a person that is eligible to cast a vote with a predefined weight on the issues relevant to that organization. A constituent can be either* active *(i.e., generating data items) or* inactive *(i.e., its existence is discussed but it is not active). A constituent is defined by a tuple* $\langle \mathcal{C}, \mathcal{C}', \mathcal{O}, i, r, d, s \rangle$ *where* $\mathcal{C}$ *is the constituent GID,* $\mathcal{C}'$ *is the GID of the active constituent submitting this data,* $\mathcal{O}$ *is the GID of the organization,* i *contains the identity details (as defined by* $\mathcal{O}$*),* r *is the revocation status of the constituent,* d *is the date and time of the data creation, and* s *the signature generated by* $\mathcal{C}'$*.*

The definition of identity details is specified by the parameter p of the organization refered by $\mathcal{O}$. An *active constituent* is identified by its public key, which is used as its global identifier. An *inactive constituent* is identified by its available description, and its global identifier is a digest of this data. The revocation status is a boolean flag that blocks any further update or usage of the constituent item, other than its dissemination. Items depending on it can be discarded. $\mathcal{C}'$ is redundant for an active constituent, being the same as its own GID $\mathcal{C}$, but for the conciseness of the description we prefer to use it, to unify the notation with the one of inactive constituents.

As data structure, a constituent is described by the set of parameters that enable his classification as to weather it is eligible to vote in the given organization.

Certain organizations can define voting eligibility based on a centralized mechanism, such as an authority that issues eligibility certificates according to its criteria. Decentralized organizations addressed here employ a mechanism that defines eligibility based on witnessing.

Definition 6 (Witnessing) *An act of witnessing is a process whereby a first participant states whether the conditions for eligibility in an organization are satisfied by a second participant. A witnessing act is described by a tuple* $\langle W, \mathcal{O}, S, T, m, e, d, \sigma \rangle$ *where* W *is the witnessing stance GID,* $\mathcal{O}$ *is on organization identifier,* S *is the constituent identifier of the witnessing participant,* T *is the constituent identifier of the witnessed participant,* e *is an explanation,* d *is the time and date of the witnessing and* σ *is a signature of the data. The mode* m *of the witnessing consists of a set of semantic statements, each of them being either* positive *witnessing or* negative *witnessing about some quality of the target.*

For large decentralized organizations it is impractical to detect whether an isolated group of people witnessing for each other are really eligible or just elements of an attack. This problem is partly addressed by the concept of neighborhood.

The constituency of an organization is classified in a hierarchical tree structure where each constituent belongs to a single leaf node and each node of the tree is called a **neighborhood**.

Definition 7 (Leaf Neighborhood) *A leaf neighborhood is a sufficiently small subgroup of the constituents of an organization such that each constituent of a leaf neighborhood can verify the eligibility of the other members of the subgroup with reasonable effort, and where the existence of the subgroup as a whole can be verified by other members with reasonable effort. A leaf neighborhood is identified by its parameters and its global identifier is given by their digest. A neighborhood is a tuple* $\langle \mathcal{N}, n, t, \mathcal{P}, c, \mathcal{C}, \sigma \rangle$ *where* $\mathcal{N}$ *is the GID,* n *is the name of the neighborhood,* t *is its level (e.g. city,street),* $\mathcal{P}$ *is the GID of the parent neighborhood (empty for the root),* c *is the list of expected neighborhood levels under this neighborhood,* $\mathcal{C}$ *is the GID of a constituent supporting its existence and* σ *is her signature for the data.*

We need to formalize the concept of reasonable effort and that can be done in terms of the cost it brings to a volunteer participant.

Definition 8 (Reasonable Effort) *In this paper we say that an effort is reasonable in the context of an organization if the majority of constituents of the given organization can perform it in a predefined unit of time (e.g., one day), fixed for that organization, without consuming extra resources besides their taxed revenue (given applicable laws) for that period of time.*

The neighborhood immediately above the current neighborhood in this tree is referred to as its *parent neighborhood*. The *set of ancestor neighborhoods* for a constituent contains the leaf neighborhood where the constituent is currently declared to belong, together with all the hierarchically higher neighborhoods containing this leaf neighborhood.

By convergence of polling data we understand that, once the generation of new data stops, eventually everybody that is connected and interested in a given topic sees the same items for that topic.

Communication items

Besides the above mentioned items used for managing the organizations (organizations, constituents, neighborhoods and witnesses), agents communicate items related to polling. The question raised by a poll is captured by the concept of motion.

Definition 9 (Motion) *A motion is a formal question with a predefined set of possible responses on which each willing constituents is called to select an answer. A motion is defined by a tuple* $\langle \mathcal{M}, t, o, \mathcal{C}, \sigma \rangle$ *where* $\mathcal{C}$ *is the GID of a constituent supporting the motion,* t *is its text,* o *is a list of possible answers of the motion, and* σ *is the signature generated by* $\mathcal{C}$*. The motion is globally identified by the hash of its data,* $\mathcal{M}$*.*

Definition 10 (Justification) *A justification is an explanation that a constituent provides for his selection of an answer for a motion. It is defined by the tuple* $\langle \mathcal{J}, \mathcal{M}, t, \mathcal{C}, \sigma \rangle$ *where* $\mathcal{J}$ *is the justification GID,* $\mathcal{C}$ *is the GID of the constituent supporting the text of the justification,* $\mathcal{M}$ *is the motion GID,* t *is the text of the justification, and* σ *is its signature.* $\mathcal{J} = HASH(\mathcal{M}, t, \mathcal{C})$.

Definition 11 (Vote) *A vote is the selected answer to a motion, as submitted by a constituent. Each vote consists of a tuple* $\langle \mathcal{V}, \mathcal{M}, c, \mathcal{C}, \mathcal{J}, d, \sigma \rangle$ *where* $\mathcal{V}$ *is its GID,* $\mathcal{C}$ *is the GID of the constituent authoring the answer to the motion,* $\mathcal{M}$ *is the motion GID and* c *the selected answer.* $\mathcal{J}$ *is the GID of a cited justification, and can be empty,* d *is a date and time, and* σ *is the signature of* $\mathcal{C}$*.*

Data Model

Each self-interested software agent stores the data of related to its own interest into a local database. The agent stores the received data if it refers to organizations, neighborhoods, constituents and motions of interest. By default, received definitions of peers and definitions of organizations received from non-blocked peers are stored. This gives users an opportunity to inspect and define their interest about them.

The database schema allows for storing the following types of items that have a stand-alone semantic and that are digitally signed, individually, by the entity generating them: peer, organization, neighborhood, witnessing, motion, justification, vote.

Each item, is tagged with three user controlled flags: `blocked, broadcastable, interest`. These flags control the communication as described in the next section. Each received data item is also associated with the arrival time, which is the date of the latest registered change to the digitally signed parameters of the item. The signed parameters of each item contain the creation time, which is the data when the signature was issued. The creation time is used to compare and select the newest item among items whose parameters change over time, such as active constituent, vote, and authoritarian organization.

For the case where an attacker or mistake leads to two distinct versions of the same item claiming the same creation time, the comparison is made on the hash of the data. This is used to prove that at convergence all participants have coherent databases.

Protocol

Let us now describe the structure of the exchanged messages. Software agents found on wireless enabled devices with ad-hoc capabilities are assumed to broadcast messages continuously (potentially with short pauses).

Communication control The default settings of our current implementations assume that a self-interested receiver normally refuses to store items about unknown organizations, as well as items relating to organizations, constituents, neighborhoods or motions that are specifically blocked by the user. To refuse items about unknown organizations, newly received organizations are blocked by default. Organizations where the user registers are automatically unblocked.

By default, all the stored data about items that are not blocked is made available for broadcasting, but that behavior can be manually controlled for each item using a flag called `broadcastable`.

For example, if an organization is blocked, then we store only its parameters but any extra data associated with it (e.g., constituents, neighborhoods, motions) are discarded. Similarly we handle blocked constituents, neighborhoods, or motions.

Messages received can refer to the GID of an unknown item (constituent, neighborhood, motion, justification). If users decide to store the item referring to unknown GIDs, then *temporary items* are created for each of the unknown GIDs, to enable their control (blocking, broadcastability). The enabling of certain temporary items, such as temporary constituents, open the door for *Storage Attacks*, namely where attackers attempt to fill users databases with data that is more difficult to verify. If temporary data is enabled, then remaining data for temporary items can be advertised as *requested* in subsequent broadcast messages. Various mechanisms (such as references to source peers) can be used to mitigate these attacks.

Items of particular interest to the user, such as motions, constituents or organizations that the user is particularly involved with, can be announced as *interests* in broadcast messages. This feature can inform cooperating peers, which can thereby give priority in sending such data back to the user. To enable this feature, each stored item is associated with the `interest` flag that the user can manually set and that the system can use to generate the corresponding interest information in messages.

Messages Each broadcast message contains a self-contained information. The two most complex types of messages are the ones carrying votes and the ones carrying witness acts (since they include data about many other types of items but are not included in other types of data).

A message containing a witness act consists of a tuple $\langle p, o, c_s, N_s, c_d, N_d, w \rangle$ describing the definition of the relevant organization o, the definition p of the peer that created the organization, the definition c_s of the constituent making the witness stance, the definition c_d of the constituent for which the witness stance is made, the definition w of the witness stance. It also contains the set of definitions of ancestor neighborhoods N_s of the neighborhood of c_s and the set of definitions of ancestor neighborhoods of the neighborhood of c_d.

A message containing a vote consists of a tuple $\langle p, o, c, N, m, j, v \rangle$ describing the definition of the relevant organization o, the definition p of the peer that created the organization, the definition c of the constituent making the witness stance, the definition m of the motion, the definition j of the justification and the definition v of the vote. It also contains the set of definitions of ancestor neighborhoods N of the neighborhood of the c.

Each broadcast message is also attaching a *set of interest hints*. This set contains some of the GIDs of the

organizations, neighborhoods, constituents and motions that the user has marked with the `interest` flag.

Probabilistically, the data concerning the details of the organization, the peer or the constituent can be dropped from a vote message or a witness message to reduce some of the replication, with the risk of rendering some messages useless (as those messages may be dropped by receivers missing one of the items required for storing it: its organization, neighborhood, etc.).

Handling Here we describe reference procedures for handling received messages. In Algorithm 1 we introduce the method used by a software agent to manage the knowledge it has about interests of peers found in passing-by cars. An interest consists of the GID of an organization, neighborhood, constituent, or motion. Whenever indication of a particular interest is received from a peer, it is stored locally, tagged with the GID of the sending peer and an expiration time. The expiration time is computed based on the arrival time of the message containing this interest, the available information about the relative speed between that peer and the vehicle of the users, and an estimation of the maximal distance within which the two devices can communicate.

When the devices are not equipped with GPS (as in the experiments reported here), then the computation simply returns the estimated expiration time as the sum between the current time and a constant `life_span` (Line 1.3). In our experiments this constant is set to 1 second. Note that each time that a message is received from the same peer, the expiration time of its interests is updated, thereby accounting for devices that are reachable for a longer period of time than the selected `life_span` constant.

A variable `min_interest` stores the current time, updated on the clock (Line 1.5) and any interests with higher expiration time is removed at that moment (Line 1.6).

Algorithm 1: Management of interest without GPS

```
1.1 procedure handle_interests (Peer, interests) do
1.2     for i in interests do
1.3         set_interest-value(i, min_interest+life_span);
1.4 procedure on clock() do
1.5     min_interest++;
1.6     drop expired interests;
```

Next we describe the algorithms used to handle received **witness** and **vote** messages (Algorithms 2 and 3). Similar and simpler algorithms are used to handle messages carrying other types of items.

The algorithms for handling messages employ the procedure `handle_interests()` defined in Algorithm 1, and a procedure `verifySignature(item)` that is checks the signature of the item passed

Algorithm 2: Receiving and Handling a Witness

```
2.1  on witness(Peer, interests, (p, o, c_s, N_s, c_d, N_d, w)) do
2.2      handle_interests(Peer, interests);
2.3      if !verifySignature(p) then return;
2.4      store-or-update(p);
2.5      if (blocked(p)) then return;
2.6      if !verifySignature(o) then return;
2.7      store-or-update(o);
2.8      if (blocked(o)) then return;
2.9      for n in N_s do
2.10         if verifySignature(n) then
2.11             store-or-update(n);
2.12             if (blocked(n)) then return;
2.13     for n in N_d do
2.14         if verifySignature(n) then
             store-or-update(n);
2.15     if !verifySignature(c_s) then return;
2.16     store-or-update(c_s);
2.17     if (blocked(c_s)) then return;
2.18     if !verifySignature(c_d) then return;
2.19     store-or-update(c_d);
2.20     if verifySignature(w) then store-or-update(w);
```

in parameter, quitting on failure. The procedure `store-or-update(item)` verifies whether a previous version of the item is already available and whether its creation date is newer than the received item. On failure it store the item (if no other version was found), or updates it (if a version with earlier date or identical date but lexicographically smaller digest value is found);

Before handling any item, first the software agent checks whether the item is not `blocked` by the user (i.e., by being generated by a blocked peer, or constituent, or for a blocked organization, neighborhood, motion, justification, or choice for the motion).

The procedures to handle messages start by handling first the more basic types of items before handling the ones that are based of the first. The typical order is: peer, organization, constituent, neighborhood, motion, justification, vote. Note that there can be a circular relation between constituent and neighborhood since a constituent may reside in a neighborhood and the neighborhood is supported/created by a constituent (potentially the same). In this case the two are stored only either if they are simultaneously available, or if storage of `temporary items` is enabled (as discussed earlier).

Heuristics

To model incentives and their relation with the behavior of the users, we formalize the utility of a message. In practice each item has its own utility, and a different utility for different users.

Definition 12 (Utility of messages) *Each user*

Algorithm 3: Receiving and Handling a Vote

```
3.1  on vote(Peer, interests, (p, o, c, N, m, j, v)) do
3.2      handle_interests(Peer, interests);
3.3      if !verifySignature(p) then return;
3.4      store-or-update(p);
3.5      if (blocked(p)) then return;
3.6      if !verifySignature(o) then return;
3.7      store-or-update(o);
3.8      if (blocked(o)) then return;
3.9      for n ∈ N do
3.10         if verifySignature(n) then
3.11             store-or-update(n);
3.12             if (blocked(n)) then return;
3.13     if !verifySignature(c) then return;
3.14     store-or-update(c);
3.15     if (blocked(c)) then return;
3.16     if !verifySignature(m) then return;
3.17     store-or-update(m);
3.18     if (blocked(m)) then return;
3.19     if !verifySignature(j) then return;
3.20     store-or-update(j);
3.21     if verifySignature(v) then store-or-update(v);
```

draws a certain utility for learning an item, depending on that item. A user also gains a given utility for disseminating an item.

In the following we assume that the utility of storing items is flat for the items in an organization, while the utility of forwarding an item depends of its similarity with the items generated by the user (and therefore describing her values).

Uninformed heuristics for broadcasting correspond to an assumption that hints received from peers are not trusted, and transmission is made based on an *a priori* model of frequency for encountering vehicles with peers traveling in the two directions. With uninformed heuristics, all peers are assumed to be interested in all items that the current peer has, and to be able to store all messages that they receive from this user. Such a model assumes that a number of A reachable vehicles travel in the same direction with a relative speed v_A while a number of B reachable vehicles travel at each moment in opposite direction with relative speed v_B. The local computer is able to load new items from a local database with an efficiency of M messages a second. Messages (each with utility u_M) can be emitted at a speed of v_M messages a second from a sending queue of size B_s, the buffer of the queue being reloaded from database at a period of time:

$$P_{reload} \geq \frac{B_s}{\min(v_M, M)}. \tag{1}$$

If D is the communication range of the device then $T_A = \frac{D}{v_A}$ is the duration for which a car traveling in the same direction is reachable, and $T_B = \frac{D}{v_B}$ is the similar duration for the opposite direction. We also assume that the assumption that the queues of preloaded messages used for sending data are long enough to provide data for the whole time T_B, i.e.,

$$\frac{B_s}{v_M} \geq \frac{D}{v_B}. \tag{2}$$

Then, the utility of sending data during time T_A is:

$$U_{T_A} = u_M \cdot A \cdot B_s \cdot \lceil \frac{T_A}{P_{reload}} \rceil + u_M \cdot B \cdot \frac{T_A}{T_B} \cdot T_B \cdot v_M$$

where the first part of the right hand expression refers to the utility obtained by sending items to cars in the same direction (cars that each receive the content of $\lceil \frac{T_A}{P_{reload}} \rceil$ full buffers of messages, each of size B_s). Note that in this equation we assume that the reminder of $P_{reload} : T_A$ is larger than $\frac{B_s}{v_M}$. The second part of the expression is the utility from the items transmitted to cars driving in opposite direction. There are $\frac{T_A}{T_B}$ road segments of size D with such cars that travel in opposite direction, each holding B cars, and each of these cars receives $v_M \cdot T_B$ messages.

If one set P_{reload} to the closest (smaller) divisor of T_A, then the utility rate per unit of time that the agent gets for broadcasting from a given queue of messages in this condition is approximated to (obtained by dividing U_{T_A} by T_A):

$$\frac{\partial U}{\partial t} = u_M \left(\frac{A \cdot B_s}{P_{reload}} + B \cdot v_M \right) \tag{3}$$

The current peer has a number N_P of personal items, a number N_S of similar items, a number N_O of other items and a number N_F of opposing opinions of positive utility (opposing opinions of negative utility are not sent). An assumption is that $N_P \ll N_O$. Based on this model we search for the best policy in terms of number of times that items with high utility should be broadcast before broadcasting some items with a lower utility.

Informed heuristics assume that peers announce their interests as sets of GIDs for organizations, constituents, neighborhoods, motions or justifications for which they want to get related items, and that they drop any other messages. Senders thereby build special queues with data of interest to these peers and give these messages priority over other items. In our experiments, agents broadcast only data relevant to current peers and found in current queues.

Each message loaded in sending queues is tagged with information about contained organizations, constituents, neighborhoods, motions, justification (and potentially vote choice), to help dynamically retrieve those of interest to new detected peers.

While our experiments were run with laptops that were not provided with GPS sensors, such sensors can

provide extra information as to when the peers travel in the same direction or in opposite direction, and for how long the peer be reachable.

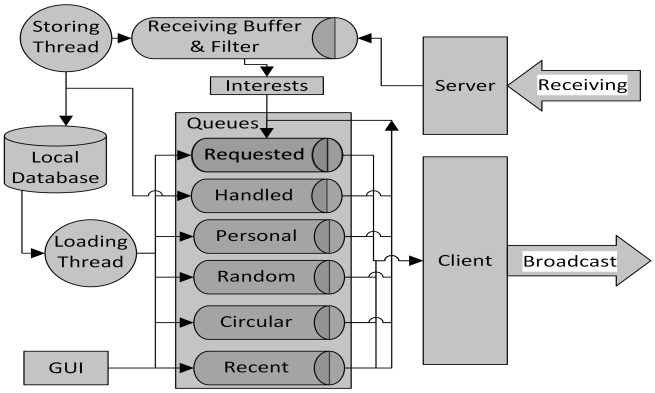

Figure 1: Architecture of the Peer

Our utility model can be combined with the statistical model of the efficiency of communication described at uninformed heuristics (as shown in the Experiments section), to decide the policy of transmission for each type of data (what percentage of each type of data should be sent at each moment of time). One can select the ratio of data of each type such as to maximize the expected utility of the sender. Rather than using the model resulting in Equation 3, one can introduce utilities in decisions based on the statistical models in (Karlsson, Lenders, & May 2007).

Agent Architecture Details

We performed experiments with our implementation of a VANET platform, based on agents running on laptops that are located in moving vehicles. We allocate an Ad-Hoc wireless cell based on the open (unencrypted) SSID `DirectDemocracy` at Frequency 2.462 GHz resulting in the cell `46:32:D1:F2:88:67`. The architecture of the server is depicted in Figure 1. Each agent starts a server bound to local port UDP/54321 and accepting broadcast messaged. Our experiments were done with the Network ID of the network card set to "10/8". The local Host ID part of the IP is set to a random value. If the random part of the IP is considered insufficient to avoid IP collisions between peers, an additional *random identifier* is also generated to uniquely detect the agent, and messages tagged with this identifier can be discarded assuming that their source is the agent of the server. When the device has more than one wireless card, the agent can be configured to only use a subset of them for this protocol. Each agent has a client that broadcasts messages on the network interfaces allocated to our protocol, sending them to the address "10.255.255.255:54321" from a set of queues prepared with preloaded messages. A small pause (e.g. 5 ms) can be introduced between the transmission of packets, as this was found to slightly improve transmission rates as well as CPU load (see the section Experiments).

Each of the queues with preloaded messages has a special policy as to the type of contained items (*personal, similar to personal, recent, random, round-robin, requested*) and its mechanisms for loading and reloading. The broadcast client picks items from the various existing queues based on a probability distribution that can be specified by the user. We experiment with various heuristics for specifying these probabilities. To maximize its dissemination efficiency, the probability of sending items of interest must grow with the number of receivers having expressed that interest (potentially serving only the items of interest to most current peers). Before broadcasting a message, the client prepends to it a header describing: *the interests of the current user*, its *random identifier*, and *available GPS data about current location and velocity*. Potentially this header can include extra information about the content of the body of the message (such as GIDs of organizations, motions, etc) to help receivers decide faster on storing or dropping messages that are not of interest. The existence of peers that drop messages not tagged with interest could push self-interested agents to provide this extra information (which otherwise reduces their bandwidth).

The servers may not be fast enough in handling and storing all the data they can receive in real time and therefore incoming data is stored in buffers. Our server has a receiving buffer of size B_r set to 20000 messages (average message size being measured to be 5kB in the current experiments). The server extracts the interests advertised by peers from the header of received messages and enqueues all the message bodies deemed new based on their size (or hash). A separate *storing thread* is used to dequeue received messages and to store their data based on the aforementioned algorithms.

If the receiving buffer is full, until the internal *storing thread* frees some entries, the server drops new incoming messages except if they are tagged in their header with information specifying that they contain items of interest to the receiver (in which case these messages are used to replace untagged messages from the buffer).

Experiments

Our implementation can run on Linux, Windows, and MacOS. The network configuration is automated on Linux and Windows while the MacOS network configuration has to be performed manually.

For the reported measurements, the databases of the agents were filled with 60000 votes for 10 organizations (O_1 to O_{10}) and 3816 motions, 9094 justifications, 629 constituents, and 4486 witness stances. These numbers were chosen based on our estimation of the ratio of the various types of items in a deployed system. To generate these items we implement a simulator that allocates each new generated vote probabilistically. First we manually generated a certain number of organizations. Then, each generated vote is allocated to a new

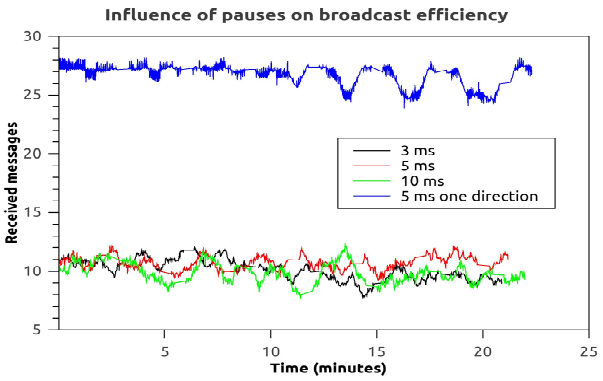

Figure 2: Experiments measuring the speed of messages transmitted (v_M). Averages for duplex communication is: 3ms pauses at 10.2 $\frac{\text{msg}}{sec}$, 5ms pauses at 10.78 $\frac{\text{msg}}{sec}$, and 10ms pauses at 9.97 $\frac{\text{msg}}{sec}$

organization with probability 10^{-5}, otherwise it is uniformly assigned to one of the existing organizations. Similarly, each vote is allocated with probability 10^{-2} to a new constituent. The size of the text of each artificially generated motion is 1000 characters and the size of each justification is 300 characters.

We performed experiments with transitive dissemination across several vehicles, validating the fact that data can be disseminated between cars that do not have direct contact. First we report numerical results about the measured characteristics of the communication between immediately connected nodes.

We measure the speed of communication v_M between two nodes in ideal conditions (when the nodes are placed far from other wireless devices). Communication is measured between an HP G62-111EE with 3GB RAM and an Acer Aspire P5WE0 with 4GB RAM running Ubuntu 12.04 on an I3 processor. Preliminary measurements were made with different pause duration (0, 3, 5, 10, 15, 250, 500, 750, 1000 ms) between transmitted packets. This pause impacts on the number of packet collisions, and therefore on the transmission efficiency. More extensive measurements were performed on the values that showed promise (3, 5, 10 ms). Measurements were taken over 25 minutes of communication for each pause duration and for each of the following two cases: when both devices transmit data. and when only one device transmits data. The results, averaged over a sliding window of size 30 seconds, are displayed in Figure 2. The maximum value of 26.7 messages per second for one direction broadcasting at 5 ms pause duration is used as reference.

We measure an estimate of the range of communication D and of the time T_B during which two devices are able to communicate. These measurements are performed with laptops found in two vehicles moving in opposite direction in several scenarios: *in a parking lot (crowded) at 15 mph, on a city street in an open area (10 wireless networks) with median strip at 40 mph, on a city street close to a school (35 wireless networks) with median strip at 35 mph, on an empty highway with median strip at 70 mph, and on the same highway (with trucks separating the communicating cars).* The measurement in the parking lot and on the city street were averaged over 10 encounters. The numbers of messages successfully transmitted in the three scenarios are shown in Table 1, as well as the duration T_B estimated from logs. It can be noticed that the speed of communication between devices is strongly influenced by the number of wireless networks in that area.

roads	speed	T_B	$M = v_M \cdot T_B$
Parking lot – crowded	15	15	158
Street – open area	40	4.3	50
Street – school area	35	2.6	15
Highway – free	70	6.3	91
Highway – trucks	70	4.5	34

Table 1: Average time of encounter (seconds) and number of exchanged messages in this time for various vehicle speeds (mph) and environments, with communication in one direction (5 ms pauses)

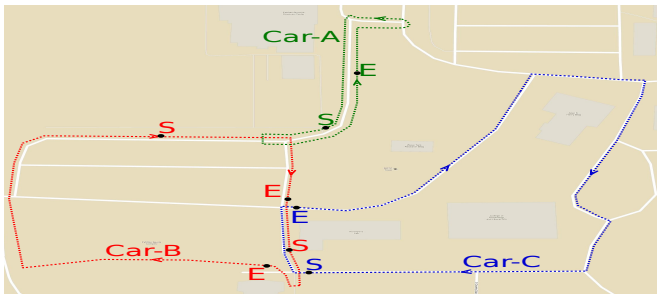

Figure 3: Trajectories in the chain topology. Areas of communication for each meeting point start at the corresponding S point and end at the corresponding E point, for each car.

To have all messages available for a peer encountered while driving in opposite direction in a crowded parking lot, the sender needs queues of size $B_s \geq \frac{D \cdot v_M}{v_B}$, which correspond to the maximum number of messages M in Table 1.

Dissemination over chains of vehicles To evaluate and confirm empirically the dissemination between vehicles that do not meet each other but communicate via other intermediary vehicles, we run experiments with three cars: A, B, and C. The car C contains a device with a preloaded database (as per the previous experiments) while the devices in the other two cars are initially empty. We evaluate two topologies of communication patterns between these vehicles: *chain* and *triangle*. For each topology the vehicles have a fix tra-

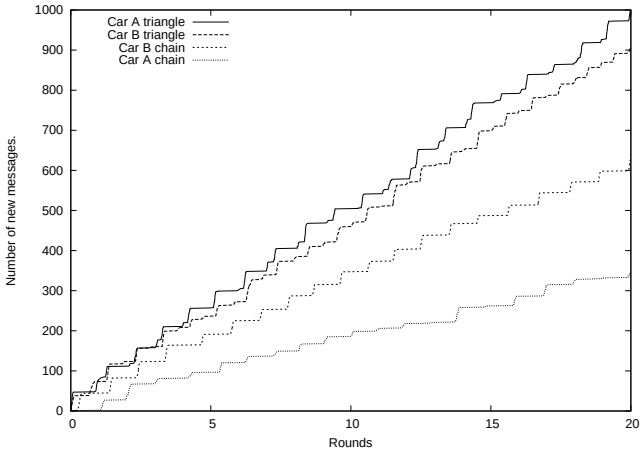

Figure 4: Received items (votes and witness stances) for cars A and B in the chain and triangle topologies. The ratio votes to witness stances is approx 2:1.

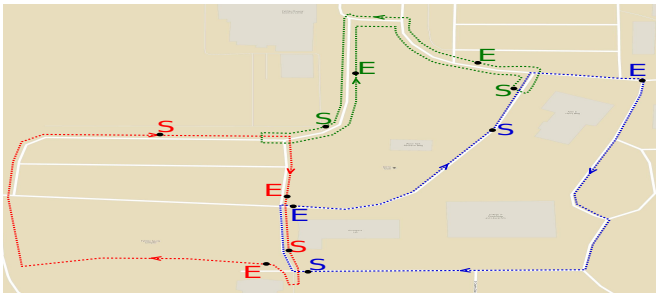

Figure 5: Trajectories in the triangle topology. Areas of communication for each meeting point start at the corresponding S point and end at the corresponding E point, for each car.

jectory that they repeat 20 times, synchronized in such a way that pairs of vehicles meet at the same location. We evaluate the impact of the studied heuristics and of the user interests on the efficiency of dissemination.

The trajectory of these cars on a map in the chain topology is shown in Figure 3. Two curves in the diagram in Figure 4 shows the number of new data items received and stored in each of the two cars during 20 rounds of encounters with this topology. We remark that the *Car A chain* curve shows that its device receives approximately 60% of what is received by the device in car B (see *Car B chain*). It is nevertheless logical to expect that the ratio would decrease with time and rounds due to the expected decrease in overlap between messages received by B from C, and messages sent by B when her database increases. The usage of queue *handled* (containing data recently received from other peers) is meant to mitigate this effect.

A comparison is made with the situation when the three cars communicate according to a triangular topology (see Figure 5). We see that the number of messages received by the car B (and car A) in this topology is approximately 50% more than the number of messages received by car B in the chain topology.

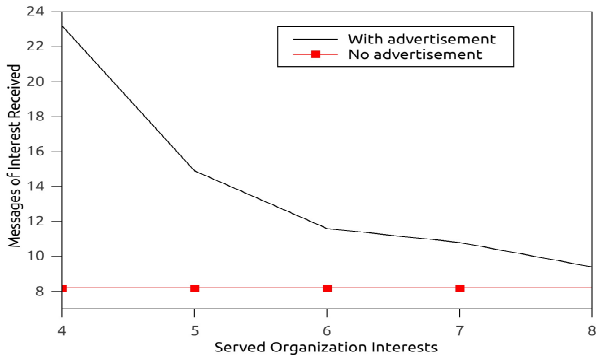

Figure 6: Comparison of efficiency with and without advertisement of interests.

Impact of Interests on Efficiency We count the number of messages of interest to the receiver, successfully transmitted to a given peer, in scenarios with the studied peer expressing interests in two organizations, while other peers also express their interests. The graph in Figure 6 shows the number of received messages given the number of different interests considered by the sender. It can be observed that the efficiency for the receiver decreases with the number of interests submitted by neighboring peers.The other straight horizontal line in the graph shows the efficiency of the receiver when no interests are advertised by anybody and the sender transmits randomly data from its 10 organizations. Note that the efficiency of the server is given by the sum of the efficiency of its receivers, being expected to grow monotonically with the number of peer vehicles receiving its data. The efficiency of the sender in disseminating its data without advertisement of interest is smaller than with advertisement, except when all the available data is of equal interest to receivers.

With the computed parameters, when we use a single sending queue with randomly picked data or with round-robin transmission, the occurrence of personally generated items has a negligible probability and the utility is practically equivalent to sending only messages of type other. Assuming that the transmission of each item has a utility of 1c for the sender and the utility of a personally generated item is 10c, the obtained utility per second with $A = 2$ vehicles driving in the same direction and $B = 2$ vehicles traveling in opposite direction on a highway is $\approx 107\frac{c}{s}$ (based on Equation 3). For the case $N_P = 10$, on a highway, the speed of sending messages with personal items has to be $v_M^P \geq \frac{B_s}{T_B} = \frac{10}{3.4} \approx 3$. Therefore the speed of sending the other types of messages (assumed to be all of type

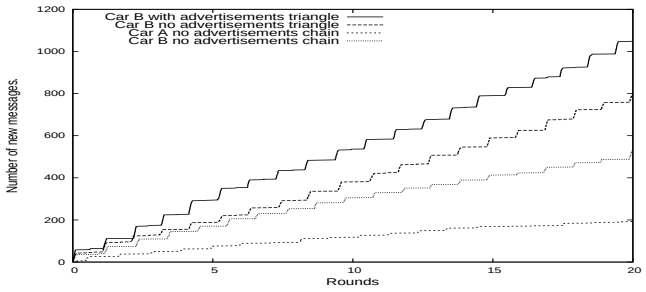

Figure 7: Impact of interest advertisement.

"other") can be $v_M^{max} - v_M^P \approx 23.7$. The total utility with this configuration is $117 + 95 = 212\frac{c}{s}$ ($117\frac{c}{s}$ for personal messages). This proves that it is useful to separate messages into queues of specialized types (gaining $212\frac{c}{s}$ rather than $107\frac{c}{s}$).

Empirical Results with Interests We ran experiments with the three cars (A, B, and C) where A is only interested in storing and forwarding the organizations O_1 to O_7, and B is only interested in storing and fowarding organizations O_4 to O_{10}. The impact of advertising their interests is shown in Figure 7, with an improvement of 28%, proportional with the ratio of interest in the available organizations. It can be seen that, when devices filter received data based on their interests, car A eventually receives a lower fraction (36%) of the data received by B than in the absence of such filtering (54%, see Figure 4). Advertisement of interests compensates for this difference.

Conclusion

A set of techniques for dissemination of data in decentralized opinion polls via a Vehicular wireless Ad-hoc Network of self-interested peers is proposed and evaluated. For comparing heuristics we compute the utility of achieved dissemination from the perspective of a given sender. The long term goal is to find the behavior at equilibrium of self-interested senders. A utility model is discussed where the highest utility is for items generated by the sender, followed by items with similar opinion, while the least utility is assigned to items of opposing opinion (potentially negative utility).

Based on a set of experiments with our VANET implementation we compute the parameters of a model for the vehicle to vehicle interaction. Strategies for broadcasting based on several queues are evaluated as well as percentages of broadcast time to allocate to different types of data items. The tested heuristics can be uninformed or informed with data received from peers such as their interests, identity, position and relative speed and bearing. Interests of peers are expressed in terms such as opinion (vote choice), issues (motions), voters (constituents), or topics (organization).

Separate outgoing queues can be maintained for data of different types (random, generated by sender, similar with sender, opposing senders, others). Cars traveling in opposite direction should get the most valuable data (generated by this sender) while cars traveling in the same direction and in contact for a long time should eventually fully synchronize with the sender on all items with positive utility and of interest to them.

References

Caliskan, M.; Graupner, D.; and Mauve, M. 2006. Decentralized discovery of free parking places. In *VANET*, 30–39.

Chen, W., and Cai, S. 2005. Ad hoc peer-to-peer network architecture for vehicle safety communications. *Communications Magazine, IEEE* 43(4):100 – 107.

EUREKA. 2010. `http://www.eurekanetwork.org/project/-/id/6252`.

Karlsson, G.; Lenders, V.; and May, M. 2007. Delay-tolerant broadcasting. *IEEE Transactions on Broadcasting* 53(1):369–381.

Kumar, R., and Dave, M. 2012. A review of various vanet data dissemination protocols. *Intl. Journal of u-and e-Service, Science and Technology* 5(3):27–44.

Lee, U.; Park, J.-S.; Yeh, J.; Pau, G.; and Gerla, M. 2006. Code torrent: content distribution using network coding in vanet. In *MobiShare*, 1–5.

Lee, K.; Lee, S.-H.; Cheung, R.; Lee, U.; and Gerla, M. 2007. First experience with cartorrent in a real vehicular ad hoc network testbed. In *2007 Mobile Networking for Vehicular Environments*, 109 –114.

Nadeem, T.; Dashtinezhad, S.; Liao, C.; and Iftode, L. 2004. Trafficview: traffic data dissemination using car-to-car communication. *SIGMOBILE Mob. Comput. Commun. Rev.* 8(3):6–19.

Nandan, A.; Das, S.; Pau, G.; Gerla, M.; and Sanadidi, M. 2005. Co-operative downloading in vehicular ad-hoc wireless networks. In *WONS*, 32 – 41.

Nzouonta, J.; Rajgure, N.; Wang, G.; and Borcea, C. 2009. VANET routing on city roads using real-time vehicular traffic information. *IEEE Trans. on Vehicular Technology* 58(7):3609 –3626.

Tonguz, O.; Wisitpongphan, N.; Bai, F.; Mudalige, P.; and Sadekar, V. 2007. Broadcasting in VANET. In *Mobile Networking for Vehicular Environments*, 7 –12.

Tseng, Y.-C.; Ni, S.-Y.; Chen, Y.-S.; and Sheu, J.-P. 2002. The broadcast storm problem in a mobile ad hoc network. *Wirel. Netw.* 8(2/3):153–167.

Wischhof, L.; Ebner, A.; and Rohling, H. 2005. Information dissemination in self-organizing intervehicle networks. *IEEE Trans. on Intel. Transportation Systems* 6(1):90 – 101.

Zhang, Y.; Zhao, J.; and Cao, G. 2009. Roadcast: A popularity aware content sharing scheme in VANETs. In *ICDCS*, 223 –230.