A Decentralized Smartphone Based Traffic Information System

Marco Picone and Michele Amoretti and Francesco Zanichelli
Distributed Systems Group
Università degli Studi di Parma
picone@ce.unipr.it, {michele.amoretti, francesco.zanichelli}@unipr.it

Abstract—Location-Based Services (LBSs) are information or entertainment services where the request, the response and served contents depend on the physical position of the requesting device. LBS are frequently used to implement Traffic Information Systems (TIS), which are increasingly based on user-contributed information. In this paper we present the first prototype of our solution for a decentralized, smartphone-based TIS, called D4V, that allows each participant vehicle to efficiently discover data or services located near any chosen geographic position. The experimental evaluation has shown that D4V could be effectively used on the road to reduce the number of drivers involved in traffic jams, as well as to disseminate alert messages about potentially dangerous road stretches, thus allowing drivers to reduce risks and nuisances along their paths.

I. INTRODUCTION

Recent years have seen the relentless market explosion of mobile devices (PDAs, smart-phones, MIDs, PMPs, netbooks, etc.), whose ever increasing capabilities make them attractive to an endless number of connected applications and services in business, infotainment and intelligent vehicle domains [20], which can be fully experienced in mobility. One of the most important research field that have developed along mobile devices evolution and diffusion is related to the challenges of applications based on the geographic location of users and information. Location-Based Services (LBSs) are information or entertainment services where the request, the response and served contents depend on the physical position of the requesting device. LBS are frequently used to implement Traffic Information Systems (TIS) [19].

The vast majority of proposed architectures for real-time LBSs are based on a pure centralized or in some cases on a hierarchical infrastructure where one or more central servers have the responsibility to manage all position updates and queries from involved users, related for example to a specific point-of-interest, neighborhood discovery or path planning.

As proposed by several researchers (see section II), a partially or fully decentralized approach is able to increase the accuracy of information and the rate at which it can be retrieved by users. Moreover, it may allow to update and publish information directly, with low cost and high scalability. Last but not least, it may simplify the process of joining the virtual community and publish new services.

On other hand, one of the main reasons not to rely on a centralized system for managing location based information (such as traffic information) could be non-technical. It simply does not seem to be desirable to hand the control of this data over one single organization, potentially limiting the access to data collected jointly by all participants.

In this paper we present the first prototype of our solution for the implementation of a Traffic Information System (TIS), called D4V, that allows each participant vehicle to efficiently retrieve node and resource information (data or services) located near any chosen geographic position. D4V runs on a peer-to-peer overlay scheme called Distributed Geographic Table (DGT), based on the geographic location of involved users and information. The DGT algorithm allows to obtain a system where overlay neighbors are at the same the real geographic neighbors of a peer without the need to route additional messages. In D4V, users can participate using their smartphone to send and receive real time information about traffic conditions or potentially dangerous situations. We have implemented the DGT as a software library in Java (J2SE and Android). Then we have developed the D4V prototype, based on such library, and tested it both on PlanetLab, using several autonomous nodes, and on the field, with D4V nodes running on smartphones managed by real users driving their vehicles.

The paper is organized as follows. In section II we discuss related work on the hot topic of sensor data dissemination and aggregation within vehicular networks. In section III we summarize the main features of the DGT overlay scheme. In section IV we illustrate the results of the experimental evaluation of the D4V application on PlanetLab. Finally, in section V we propose a concluding discussion and an outline for future work.

II. RELATED WORK

An almost complete overview of existing and emerging technologies and solutions for disseminating and aggregating sensor data in vehicular networks has been proposed by Uichin and Gerla [7]. An example is the CarTel mobile sensor computing system [3], that allows to collect, process, deliver and visualize data retrieved from sensors placed on mobile units such as vehicles. MobEyes [6] proposes a strategy for harvesting, aggregating, and distributing sensed data by means of proactive urban monitoring services provided by vehicles that continuously discover, maintain, and process information about events of the urban scenario. Messages and summaries are routed to vehicles in the proximity, to achieve common goals, such as providing police cars with the trajectories of specific ”target” cars.
A fundamental issue for VSNs is connectivity: different wireless access and communication methods have been evaluated, including Dedicated Short-Range Communication (DSRC) [5], WiMax/802.16e [2], WLAN [1], as well as cellular systems [15]. The use of a cellular communication network reduces the problem of implementing a working TIS, but introduces, on the other side, the issue of collecting data and distributing them to interested users.

In [3], Hull et al. point out the major technical challenges of the client/server approach, in particular the fact that the server constitutes a bottleneck for the huge amount of simultaneous updates and queries associated with car movements. For these reasons, in recent years researcher started investigating architectures based on the P2P paradigm, to build a distributed TIS where cars are not only consumers but also producers of information. Rybicki et al., with Peers on Wheels [16] and, more recently, with PeerTIS [4], have shown P2P architectures where participating cars are peers organized in a Distributed Hash Table (DHT). Roads are divided into road segments, each with a unique ID that is used as key in the DHT. The main idea is that each node is responsible for a certain part of the ID space and, consequently, for a certain number of road segments. Up to now one of the troubling issues is the fact that obtaining full information about planned and alternative routes is expensive in terms of bandwidth consumption. The work of Santa et al. [17] shows another P2P approach based on cellular networks (CNs) and on the JXTA middleware, to enable the transmission of information among vehicles and between vehicles and infrastructure, bounding the propagation of messages. CNs are also used not only in P2P solutions but also in participatory platforms, for participatory vehicular sensing [11] [12], allowing applications such as ride quality monitoring, street-level traffic flow estimation, and proactive urban surveillance.

III. PROACTIVE LOCALIZATION WITH DISTRIBUTED GEOGRAPHIC TABLE

The Distributed Geographic Table (DGT) is an overlay scheme where each participant can efficiently retrieve node or resource information (data or services) located near any chosen geographic position. In the following we briefly recall the basic DGT concepts, data structures and protocols, with particular attention to its peer localization strategy (a more detailed presentation is available in [13]).

A. General concepts and data structures for DGT

We assume that every node knows its global position (GP), retrieved by means of a GPS receiver or other localization technologies. The same hypothesis has been made in the Mobile Millennium project [10] to implement a pilot traffic-monitoring system that uses GPS to gather traffic information, process it, and distribute it back to the phones in real time.

Each DGT node knows a set of real neighbors organized in a specific structure based on the distance with respect to the peer. The PeerDescriptor of a node contains contact information (IP address, port, proxy etc.), its geographic location and the list of message types for which it is interested. This structure is periodically updated by each single active peer with a specific discovery function described below.

Geographic position of the peer is the middle of $K$ different concentric circles (each of them), each having a different (application-specific) radius $R_i$ and thickness $r_i$, with $i$ integer $\in [1, K]$, for which $R_i$ is the sum of previous $r_i$ from 1 to $i$. In other words, each peer stores a set of lists (called GeoBuckets) of neighbors, each list being sorted according to distance. These lists are regularly updated in order to have available the latest information about node positions.

B. DGT-based node discovery

When a new peer joins into peer-to-peer network, it receives an initial list of up to $I$ references of other active nodes near its actual position. This list represents the initial knowledge of the new peer that can be used to start the first discovery procedure, in order to learn about other active users in the system. During each discovery process, a generic node $P$ picks up the closest $\alpha$ known neighbors (if available), and sends simultaneous FIND_NODE requests centered in its geographic location (while, in general, the DGT allows for discovering nodes located around any point of the map). A node that receives this type of message extracts the closest known peers near the specified location from its GeoBuckets, and returns their references to the requesting node. After $P$ receives answers from queried neighbors, it adds new received references to a temporary list and sorts it according to distance. Then $P$ picks up again the closest $\alpha$ nodes from this temporary list that were not previously contacted, and repeats the same procedure with these other peers. If at the end of each iteration no new nodes are retrieved, $P$ selects (if available) the closest not previously contacted $K$ nodes and sends them the last turn of FIND_NODE requests. When finally $P$ receives the answers from these neighbors, it adds new peers’ references to the appropriate GeoBuckets. At the end of each run of the discovery procedure, $P$ counts the number of new neighbors and upon its value it schedules the next activation of the discovery procedure. The general idea is that soon after the bootstrap or when neighbor peers show high dynamicy, the period of the discovery process may be very small (near $T_{min}$), increasing with time (towards $T_{max}$), when the knowledge becomes sufficiently distributed among active peers.

C. Consistency Preservation

Each active peer in the network may change its geographic position very frequently, periodically sending its updated GP to neighbors, in order to improve the accuracy of their knowledge base. As a measure to limit the overhead of this procedure, i.e. the message rate, each peer communicates its position updates to a neighbor only if the displacement is less or equal to $\epsilon$ (an application-specific distance). If during this message exchange a peer receives a node’s update
confirming that the new position is out of its area of interest, the neighbor’s reference is removed from the appropriate GeoBucket and a REMOVE message is sent to the peer.

Peers are allowed to build and maintain an open-ended DGT, where each node has high knowledge of geographically close neighbors and a reduced view of the outer world. Moreover, peers are able to find new connected nodes that are entering the target area, incurring in limited computational and transmission costs.

D. Content Dissemination

The distributed knowledge about geographic neighborhood, periodically maintained by peers according to the DGT protocol, enables the dissemination of traffic information messages. We recently started using an approach based on the publish/subscribe paradigm, that allows message distribution to all interested receivers in an area, by keeping messages alive in that zone for a specified period of time. Each message includes the type of notification (for example traffic events, sensor data, etc.), the location associated with the information, the range that represents the area surrounding the message source location that the notification should reach, the time to live of the event, and the payload containing - when necessary - additional and detailed information about the event. When a new message is generated, the publisher picks up from its GeoBuckets the closest known nodes within the notification’s range that are interested in the particular information type (by reading their PeerDescriptors), and sends them the new message, trying to avoid duplications. When a notification is received, the node checks if it still matches the user interests or if it does not (in case of dynamic subscription) or if it is already known. When a peer receives the reference of a new node in its area of interest, it checks if in its knowledge base there are notifications not yet expired that may be useful for that peer. If the target peer has not yet been contacted for the same reason, the node sends the message. During this dissemination process, it is necessary to check if some messages have expired, and consequently to remove them and their references from the vehicle knowledge base, thus avoiding the distribution of an obsolete notification.

IV. TIS Prototype and Evaluation

The simulative analysis of DGT we presented in previous papers, such as [13], gave us valuable insights for the development of a DGT Java Library and a first prototype of the D4V traffic information system.

The library implements the base functionalities and policies of a DGT overlay such as the discovery procedure, the management of the neighborhood and the GBs maintenance. The D4V application layer uses such features to implement the content dissemination algorithm and the user interface to get the input from the drivers related to a specific traffic event, and to show approaching dangerous situations. The development of the DGT library started from our novel peer-to-peer middleware called sip2peer [18], that is an open-source SIP-based middleware for the implementation of any peer-to-peer application or overlay without constrains on peer nature (traditional PC or mobile nodes) and specific architecture. At this moment sip2peer is available for Java SE and Android platforms, but we are working on the iOS release. The Java and Android implementations of the library are based on the Java SIP stack called MjSip [9] that allows to manage the exchange of SIP messages to control multimedia streams. Sip2Peer supports two message formats. It is possible to manage simple text message containing any kind of information like raw data or XML, and it is also possible to natively use the JSON format.

Figure 1 groups together DGT and D4V modules, to present all involved elements whose integration defines the behavior of a peer. Such modules are presented and described in the following.

The Sip2Peer Layer (SL) represents the communication module providing methods to receive and send messages from and to other active peers in the network. This layer interacts with the DMH (illustrated below) to route and forward outgoing and incoming messages, providing proper notifications when a packet has been correctly delivered or not.

The DGT Message Handler/Dispatcher (DMH) conveys and manages all DGT messages. It does notify neighbors position updates and redirect subscriptions and event messages from and to the Subscription Manager (detailed below).

The Geo-Bucket Manager (GM) manages the data structures of the peer according to its geographic location. This module interacts with the Location Manager to be notified for a position update, and thanks to DMH notifications it is able to add new discovered peers or remove nodes out of the region of interest. Since it is a base module of the DGT Library, it is configurable allowing to define the number of buckets and their thickness (target area) and the maximum number of nodes that a single GB can maintain during node life.

The Location Manager (LM) subscribes to information and updates about device location through GPS, WiFi or cellular network, trying to minimize the energy consumption according to the context application characteristics. In detail in an urban scenario where the mobile is used inside a car where the energy consumption is not a constrain the location could be detected used external localization devices such as a navigator providing that potentially could provide additional and useful information about the planned route and the target
Fig. 2: Android DGT Prototype: (a) Settings Menu; (b) Map View; (c) Traffic Message creation view; (d) Bad Surface Condition and (e) Traffic Jam Messages popup view; (f) Incoming message list.

The DGT Kernel (DK) is the core of a DGT node, implementing the routing strategy and the discovery procedure for neighborhood maintenance, as well as short/long range queries triggered by the user (through the User Interface) or periodically efficiently scheduled by the DK thread according to the application purpose and users settings. It also allows the interaction with DMH and GM to properly disseminate messages and alerts coming from UI or other external inputs.

The Subscription Manager (SM) is related to the D4V prototype and has been designed to manage the subscription system of the node, allowing to add or remove subscriptions and handling an filtering incoming events or user queries. It does interact with the UI to notify relevant incoming alerts or messages, and sets preferences about the subscriptions.

The User Interface (UI) allows to present to the user all required information and interface elements, to control DGT functionalities, like dissemination an alert messages about traffic jams, or to schedule a query concerning a region of interest. Moreover, it allows to visualize on a map (or in a dedicated list view) peer/vehicle and neighbor locations, as well as scheduled query results. Our first prototype and the associated UI has been designed and developed on the Android platform.

When he/she runs the application, the user sees a map displaying the updated vehicle location, and configures through a specific menu the IP of the bootstrapping node, used to join the DGT network (Fig.2(a)). When the first DGT discovery has been completed and the neighborhood is formed the user can see neighbor vehicles on the map view and alert messages near its geographic position (Fig.2(b)). For this first prototype we allowed the user to generate information related to four different types of events (Fig.2(c)). By clicking on the associated button, the user can generate and distribute an alert message related to Traffic Jam, Car Accident, Men at Work, or Bad Surface Condition. The message contains information about the geographic location, the event generation time, the source user and the expiration time (set by default to one hour), after which the DGT peer stops disseminating it, unless a new user refreshes the information. While the user is driving, changing its location, the application checks if one or more received traffic messages are close to the car in a range of 200 meters, and notify the alert type and its location (Fig.2(d)(e)). Furthermore, the user can review in any moment the list of received messages (Fig.2(f)) sorted by distance, visualize them on the map, renew expired content or report an abuse or false information generated by a user.

To test the designed application during the development phase, in a scenario with a relevant number of nodes, before releasing the prototype to a group of alpha tester, we have developed an additional module that emulates the behavior of vehicle moving along city streets. This module implements a FTM mobility model to evaluate the car speed, based on the switch station model that we presented in a previous paper [14], and provides a web-based tool to monitor peer movements during experiments. This module allowed us to create a complete and autonomous D4V node (called D4V-Bot), that is able to join the DGT network and generate a traffic message if required by the experiment setup.

A. Performance Evaluation of the D4V Prototype

A first evaluation phase has been conducted with a hybrid group of 50 nodes, either installed on Android devices and managed by real users, or autonomously running (as D4V-Bots) on a server of our laboratory. Peers were able to join the network, build their neighborhood and maintain it during the experiment, according to the changes of their geographic location.
location. When a node generated a new message related to a traffic event, the latter was correctly distributed and shown on smartphones of user inside the region of interest of the event.

Starting from the results of this first preliminary evaluation, it was important to properly measure the network performance in order to understand if the results of the simulation analysis are confirmed by the results of the prototype test, also with an heterogeneity in terms of access network. For this purpose we deployed our D4V-Bot experiment on PlanetLab, which is a global research network that supports the development of new network services. PlanetLab currently consists of 1089 nodes at 532 sites, and the University of Parma contributes with 2 nodes in the world and european network.

We deployed 50 D4V-Bots on 13 different PlanetLab servers in 13 different countries. Each node, every 30 seconds, logs to a file a JSON string containing all the needed information to analyze the behavior of the peer, such as geographic location, exchanged kbytes, received and sent messages. At the end of the experiment a dedicated tool parses all available log files to build a time line of the experiment made by steps of 30 seconds containing all the required statistics for the performance evaluation.

Experimental results are based on five different runs of 26 minutes. Performance metrics that have been taken in account are:

- **Bandwidth**, as average message rate sent per peer per second.
- **Coverage Percentage (CP)** of D4V messages (TrafficInformation and SensorData) at a certain time of the simulation; it is evaluated as the ratio between the number of peers that actually received a specific message, and the number of those which should have it.
- **Distance From Event (DFE)**, i.e. the average distance between the geographic location of a vehicle that did not receive a traffic jam message, and the position of that event. This metric improves the information provided by CP. Indeed, a high DFE value (compared with the message range) means that drivers who do not receive the message are far from the dangerous situation, and probably will receive the information shortly from the neighbors that have been already informed.
- **Delay** as round-trip delay time, i.e. the time that a signal needs to be sent, plus the time it takes for an acknowledgment of that signal to be received.
- **% Packet Loss** for a peer during the experiment.

Figure 4(a) shows the trend of the Coverage Percentage with the average value for the PlanetLab experiment and the same results obtained in the simulation analysis of our previous evaluation. The generation of traffic messages starts 400 seconds after the D4V-Bots have been activated, in order to give them enough time to build the DGT overlay network. Results show that the average value of CP is very high (close to 97%), and in particular significantly near the average value of our simulations (≈98%). The CP curve shows that when new messages are generated the coverage percentage goes lightly down to lower value (≈88%) but after one or two time line steps (30/60 seconds) recovers to a high coverage percentage confirming that the dissemination process and the neighborhood knowledge allow to efficiently distribute messages.

DFE values allow to better understand the behavior of the DGT protocol. For their measurement, we have considered a range of interest for disseminated messages of 4 Km. Figure 4(b) illustrates the DFE trend comparing PlanetLab and simulation average, that also in this case are really close (≈3.5 Km). The graph confirms that vehicles that did not receive the message are on average really far from the dangerous event, and have a sufficient margin to receive the message before approaching the potentially dangerous location, by changing their direction to reach their destination using a different route, or just adapting their vehicle speed for example near a portion of damaged road surface.

Figure 4(c) reports the bandwidth in terms of Kb/Sec/Peer during the experiments conducted on PlanetLab. In this case average value (≈0.3 Kb/Sec/Peer) is different from the same value obtained during the simulation (≈0.9 Kb/Sec/Peer). This difference is associated to the fact that in the real implementation there is an additional overhead due to packet header and additional exchanged information, that initially were not considered during the modeling of the communication in our simulator.

Figures 4(d) and 4(e) finally illustrate the trend and the average of delay and the percentage of packet loss measured during the experiments. The delay is reasonable for the designed application, considering that the small package size and that the experiments have been done on 13 servers located in different research institutions, with different network capabilities and load during the experiment (PlanetLab nodes are used at the same time by several application and the condition could change during a deployed experiment).

V. Conclusions

In this paper, we have presented the evaluation of the first prototype of our decentralized TIS solution, called D4V, based on the Distributed Geographic Table (DGT) overlay scheme. We have tested the D4V prototype on PlanetLab, to verify network performance in a heterogeneous environment. Simulative and real measurements appear very close, thus confirming the reliability of the simulation implementation as well the good performance of the DGT overlay. The scheme guarantees a high coverage, in terms of vehicular notification, over a wide range of system parameter values, whilst generating limited data traffic and coping reasonably well with significant packet losses. Hence, we are confident that D4V could be effectively used on the road to reduce the number of drivers involved in traffic jams, as well as to disseminate alert messages about potentially dangerous road stretches, thus allowing drivers to reduce risks and nuisances along their paths.

Starting from the encouraging results obtained with our TIS prototype, we also intend to organize and deploy a
large scale experiment across the city of Parma, in order to evaluate the application with a huge number of active users over a long period. The experiment will provide us with multiple types of data and logs at different levels. Users will comment about the User Interface usability, allowing us to understand how to improve it to reduce the number of interaction between drivers and their smartphones. Network and DGT logs will provide the required information to analyze the performance in terms of data traffic requirements, coverage percentage and distance from the event. Last but not least, an ambitious development of the DGT/D4V solution will be its integration with a smart vehicle, able to sense the environment and to automatically provide messages about external events or conditions.

REFERENCES