

Automated Ride Share Selection using Vehicular Area Networks

Amrinder Arora¹, Mira Yun², Timothy Kim², Yu Zhou², and Hyeong-Ah Choi²

¹Department of Computer Science, Bowie State University, Bowie, MD

²Department of Computer Science, George Washington University, Washington, DC

Email: aarora@bowiestate.edu, {mirayun, timothyk, yuzhou, hchoi}@gwu.edu

Abstract—This paper considers the possible application of ride share services in Vehicular Area Networks. We propose a system architecture and protocol for this specific application using the Vehicle to Vehicle (V2V) and Vehicle to Road (V2R) communications. We present how the system would integrate with the current VANET infrastructure and the On-Board Unit, and present algorithms for selection of ride share options. We also present simulation scenarios and theoretical analysis of the likelihood of finding an acceptable ride share option using the proposed system, and analyze the expected network overhead caused by the system.

Index Terms—vehicular networks, intelligent vehicles, ride share, carpool

I. INTRODUCTION

Automation of vehicles has been an important goal for the past few years, and various applications have been proposed using the vehicular area networks [1], [2]. Electronic Control Units (ECUs) and sensor units are being introduced into the cars by various automakers, that allow communication in a vehicular area network. Advanced sensing, communications and computing promise to allow vehicles to operate automatically, to avoid collisions, and to provide other helpful applications for passenger convenience. There have been demonstrations of automated vehicles in San Diego, Phoenix, and the Netherlands. Some of the building blocks of an eventual automated vehicle are already in use: adaptive cruise control, in-vehicle navigation, etc. The California Department of Transportation is testing the use of magnets in the road to guide snowplow operators in poor visibility. Collision avoidance systems for trucks have been developed and marketed in Japan and the US. Auto makers currently offer systems that track the car and call for assistance in the case of a crash.

Thus, there is a clearly defined market for individual intelligent vehicle components that increase safety, comfort, or convenience. One possible application for the Vehicle to Vehicle (V2V) and Vehicle to Road (V2R) networks is the automated/assisted ride share scheduling. The goal of this application is to allow different vehicle operators to be able to arrive at any ride share point without any prior registration or planning, and evaluate and schedule a ride share in real time from the pool of other drivers at that ride share point. Currently, there are many ride share applications that exist that require human coordination, and do not operate in real time. These typically operate via web registration wherein the users specify their usual itineraries and the ride share matching

service finds the suitable ride share partners. An “Intelligent Travel Assistant” was proposed in [3] which focused largely on the database aspects of the problem.

A. Advantages of Integrating Ride Share services using V2V/V2R Communications

There are significant advantages of using a ride share selection service using V2V/V2R communications. It can be done in real time and requires no prior knowledge of the trip. Even if the trip plans change slightly (or a lot), it does not impact either the vehicle operator or other operators in terms of ride share planning. This also allows finding the most suitable ride share partners from the current pool of participants.

B. Environmental and Economic Impact

The most important impact of any ride share services is environmental. The benefit of an instance of a ride share is not solely in terms of the savings that are obtained by the vehicle that is off the road. A more substantial share of savings comes from a decreased congestion, which allows the other vehicles to move uninterrupted, thus having a cascading positive impact.

A very significant positive impact of ride share services is economic. The decrease in congestion that results from ride share services is not just good for the environment, it also decreases actual cost in terms of labor and fuel wasted due to congestion. A report in 2001 [4] showed that congestion costs 78 billion USD to the US economy, which was a 39 percent increase relative to 1990. A more recent report from Canada [5] measured the total cost at approximately 3 billion Canadian dollars for each of the 9 metropolitan areas in Canada.

Thus, not surprisingly, traffic congestion is an active research area across multiple disciplines, and only recently, the potential of using V2V communications as a part of the solution for this problem is being explored.

II. SYSTEM MODEL AND PROBLEM STATEMENT

We consider Vehicle to Vehicle communication using the 75 MHz band in the 5.85 - 5.925 GHz range (commonly referred to as 5.9 GHz) licensed for Dedicated Short Range Communications (DSRC). The 5.9 GHz DSRC is in nascent stage, and is currently intended to replace the existing 915 MHz DSRC in US and the 5.8 GHz DSRC in Europe. While

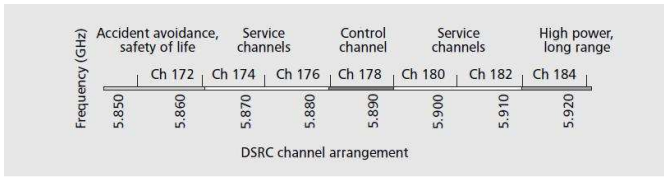


Fig. 1. Channels in 5.9 GHz DSRC

the proposed application layer protocol does not depend upon the specific band, the physical layer details, etc, the simulation analysis and physical data analysis has been done assuming 5.9 GHz DSRC.

We separate our discussion of the system model as per the following components of the system.

A. DSRC Channels

The formal definition of the relevant IEEE standard is given in [6]. As shown in Figure 1, 5.9 GHz DSRC spectrum is divided into seven 10 MHz channels, the central of them (channel 178) is the control channel and the others are service channels. Channels 172 and 184 are dedicated for safety of life issues and public safety issues respectively. Channels 174, 176, 180 and 182 are available for both safety and non-safety applications, and thus can be used for the proposed application. The advertisement for the service (for example by a road side unit near a ride share coordination area) is permitted on channel 178, but its use is limited to “occasional”. More details on the various DSRC channels and their roles can be found in [7], [8], [9], etc.

B. System Infrastructure Components

We assume the following system infrastructure components to be available in the system:

- Road side units (RSU) - also referred to as Road Side Equipment (RSE) in the IEEE standards
- A certain fraction of vehicles are VANET enabled, and have an On Board Unit (OBU), that communicates with other OBUs and the RSUs. OBUs have GPS and trip planner capabilities, and interface with the display unit, that doubles as the vehicle’s TV panel

C. Problem Statement

Before presenting the problem statement, we present a cost model that is used to objectify ride share options. Given a trip itinerary r_j for a ride share program participant u_i , the total cost of the trip itinerary is denoted by $x(r_j)$ and is given by: $c(r_j) + w_i \times t(r_j)$, where $c(r_j)$ is the cost of the itinerary (fuel plus any public transportation cost), $t(r_j)$ is the total (one-way) transit time for the itinerary, and w_i is the time-to-cost weightage ratio for the participant u_i . This simple linear model allows for each of the ride share participants to have different weightages (importance) of the time and cost components of the itinerary.

We make the following assumptions in our system model:

- We consider a linear cost model involving economic cost and transit time and ignore any environmental cost, as the environmental cost is proportional to the economic cost.
- We only consider one ride share component per trip. Therefore, a ride share that ends before the trip destination must end at a place that is accessible to public transportation.
- Participants can have their own time-to-cost weightage (w_i) and their own security policies with respect to who to ride share with. This is a strong motivation for a distributed algorithm as each user has her own perspective on potential ride share partners.
- We assume that multiple equi-distant paths to one destination may exist, but all of the options are acceptable to participants.

Given this system model, the ride share scheduling RSS problem is defined as follows.

Ride Share Scheduling Problem (RSSP)

- **GIVEN:** A collection of n ride share participants: u_1, u_2, \dots, u_n that are currently in the vicinity of the ride share area, and their respective trip destinations d_1, d_2, \dots, d_n .
- **TO DO:** (i) Analyze the ride share options in real time using V2V/V2R communications based on 5.9 GHz DSRC, (ii) present the ride share options using OBU and display unit inside the vehicle, and (iii) schedule the ride shares using V2V communications.
- **SUCH THAT:** Total cost savings for all the ride share participants are maximized.

III. PROPOSED SOLUTION

Before presenting the main algorithms, we present the system architecture, which consists of communications architecture and electronic component architecture for the On Board Unit.

A. Communications Architecture

The communications architecture is shown in Figure 2 and consists of a road side unit at the ride share service area, and optional road side units about 200 m approaching the ride share service area. The on board units on the participating vehicles connect with the nearest road side unit to listen to service broadcasts and to communicate their service parameters.

We propose the use of channel 174 for this application. This channel has the power limit of 33dBm, and a coverage radius of at least 200 m [10].

B. Electronic Component Architecture

A sample electronic component architecture that supports the ride share application is shown in Figure 3. The on board unit (OBU) consists of GPS receiver and DSRC transceiver, that retrieve the real time traffic information, including the ride share traffic data. The multiprocessor unit (MPU) runs the

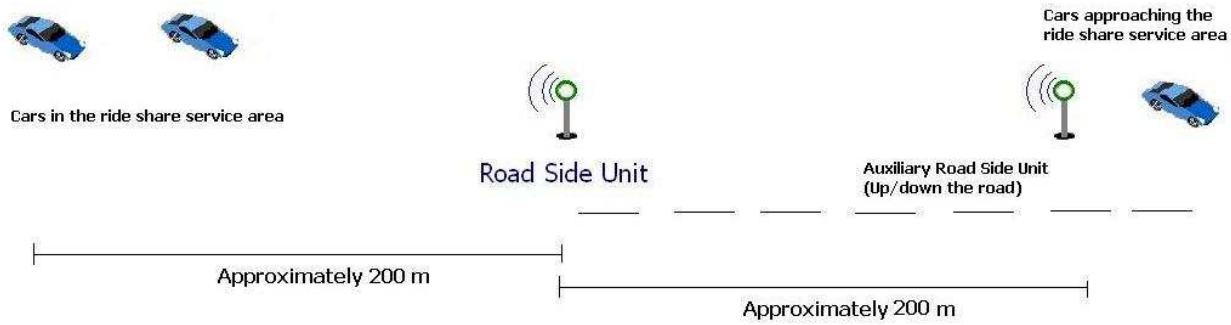


Fig. 2. Communications Architecture

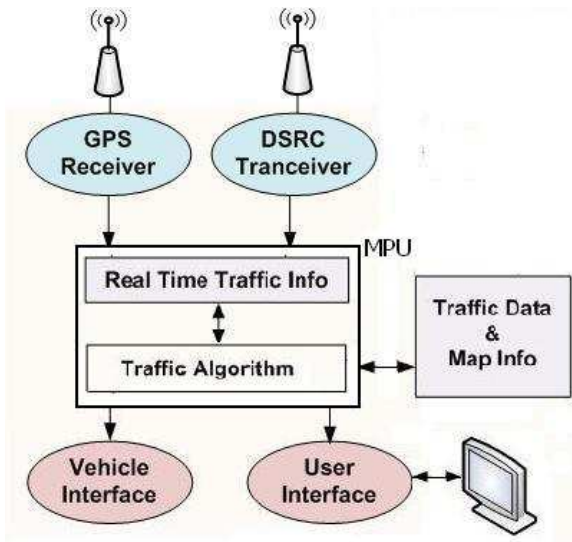


Fig. 3. Electronic Component Architecture

traffic, safety, ride share and other algorithms, and interfaces with the human user through the display unit. The MPU also interfaces with the vehicle directly, possibly for applications other than ride share application.

C. Algorithms

Before presenting the algorithms, we present a total cost function X , which takes a trip origin, trip destination and travel mode as arguments and returns the total cost of the trip. The travel mode can be: 1: drive without a ride share, 2: drive with a ride share, 3: ride as a ride share, and 4: public transportation. That is:

$X(s, d, 1)$: returns the total cost to drive without any ride share from origin s to destination d

$X(s, d, 2)$: returns the total cost to drive (and take a ride share partner along) from origin s to destination d

$X(s, d, 3)$: returns the total cost to ride with a ride share partner from origin s to destination d , including the cost to park a vehicle at s ,

$X(s, d, 4)$: returns the total cost of public transportation from origin s to destination d

We remind the reader that the total cost is a linear function of economic cost and time, as explained in Section II-C. We also note that the cost to drive without any ride share and the cost to drive with a ride share can be different from each other due to *high occupancy vehicle* policies: the decrease in travel time can very well contribute to a decrease in cost when driving with a ride share. That is, $X(s, d, 2) \leq X(s, d, 1)$.

The proposed system uses two core algorithms:

- **Match Two Trips:** Given trips trip1, trip2, it returns two options from the perspective of the owner of the trip 1: (one for driving, and other for sharing the ride with the owner of the trip 2). The Match Two Trips algorithm is summarized in Algorithm 1.

We point out that the constant Δ in Algorithm 1 is an artificial cost component to encourage helping other users in case extending help does not directly benefit the user himself.

- **Find Ride Share Options:** This is a distributed algorithm, that creates ride share options by evaluating trips of other ride share participants. It uses the Match Two Trips algorithm to quantify cost values for ride share options. The Find Ride Share Options algorithm is summarized in Algorithm 2.

Dynamic Programming variation of Match Two Trips algorithm: We note that a dynamic programming variation of the trip matching algorithm is easily possible, by storing each calculated value of $X(s, p, i)$ and $X(p, d, i)$. The dynamic programming variation shows significant improvements in practice, even though it does not yield any provably better upper bound on execution time, as the set of the points of interest on a shortest path route may be disjoint when compared to points of interest on a different shortest path route.

We observe that in the Find Ride Share Options algorithm, the security check is delayed until after the ride compatibility has been met and the cost of the ride option has been determined to be within the threshold. The reason for this is because the security check requires a request message and response from the RSU.

There are two interesting optimization options for the security check:

Algorithm 1 Match Two Trips starting at origin s

Require: d_1, d_2 two trip destinations
Initialize $c_1 \leftarrow X(s, d_1, 1) + \Delta$ and $p_1 \leftarrow nil$
Initialize $c_2 \leftarrow \infty$ and $p_2 \leftarrow nil$
Initialize $\mathcal{R}_1 \leftarrow$ set of shortest path routes to d_1
Initialize $\mathcal{R}_2 \leftarrow$ set of shortest path routes to d_2
for each route $r_{1i} \in \mathcal{R}_1$ **do**
 for each route $r_{2j} \in \mathcal{R}_2$ **do**
 for $p \in$ common points of interest on r_{1i} and r_{2j} **do**
 if $X(s, p, 2) + X(p, d_1, 1) < c_1$ **then**
 Update c_1 and p_1
 end if
 if $X(s, p, 3) + X(p, d_1, 4) < c_2$ **then**
 Update c_2 and p_2
 end if
 end for
 end for
end for
Return (c_1, p_1) and (c_2, p_2)

Algorithm 2 Find Ride Share Options for Participant u_i

Require: d_i - the trip destination for u_i
Obtain in real time, the list of active ride share participants U from the RSU
Initialize Options $\leftarrow \phi$
for $u \in U \setminus u_i$ **do**
 if rideCompatibility(u_i, u) not OK **then**
 Skip to next possible participant
 end if
 $(c_1, p_1), (c_2, p_2) \leftarrow$ Match Two Trips(u_i, u)
 if $c_1 \leq$ THRESHOLD **then**
 if securityCheck(vehicle of u , driver of u) OK **then**
 Add (c_1, p_1, u_i) to the Options.
 if $c_2 \leq$ THRESHOLD **then**
 Add (c_2, p_2, u_i) to the Options.
 end if
 end if
 end if
end for
Return Options

- (i) security check can be delayed by first collecting all low cost options, and then using a single security check request message and response to check for all ride share options (this however, does not allow showing the options to the user until all options have been collected);
(ii) security check can be done by RSU even before broadcasting the list of ride share participants, in which case, it is not required to do a security check within the *Find Ride Share Options* algorithm. We use this option in our simulation setup.

IV. RIDE SHARE PROTOCOL

In this section we describe the ride share protocol.

Msg Type	Vehicle Credentials (32 bytes)	Driver Credentials (30 bytes)
Occupancy Details (8 bytes)	Trip Details / Dest Vehicle (32 bytes)	Preferences / Result Value (24 bytes)

Fig. 4. RSS Application Service Data Unit Structure

A. Packet Structure

The application's service data unit has the following components:

- 1) Message Type: 2 bytes, can be (i) rss active, (ii) rss active vehicles, (iii) itp, (iv) itp ack, (v) ack, (vi) security check request, or (vii) security check response.
- 2) Credentials of vehicle: 32 bytes, including the vehicle identification number, and other government agency registration number
- 3) Credentials of driver/operator: 30 bytes, including license number, issuing authority and issuing authority number
- 4) Vehicle occupancy details: 8 bytes, first 4 bytes for total number of seats, and next 4 bytes for available number of seats. If the vehicle is not available for ride share, then the available number of seats can be set to 0.
- 5) Trip itinerary: 32 bytes, 8 bytes for location code and 24 bytes for location name. This field also doubles for destination vehicle when the message type is itp, itp ack or ack.
- 6) Preferences: 24 bytes. This field is also used for value of the security check request, that is, when the message type is security check response.

Packet Structure for Ride Share scheduling application is shown in Figure 4. It may be noted that all application service data units are 128 bytes, except the list of rss active vehicles, which is a multiple of 128 bytes.

B. State Transition Diagram

All entities in the ride share participation start in the "Initial" state. When an appropriate ride share partner is found, the entity enters into a negotiating state, which can be "Prop Sent", or "Prop Recd" depending upon who initiates the proposal. The main difference between these two states is that a vehicle in "Prop Sent" state could have sent the proposal to multiple other vehicles, and may continue to send more proposals. A vehicle in the "Prop Recd" state has however responded to exactly one proposal and does not respond to any other proposals in this state. Once the negotiations are completed, the entity enters into the final "Established" state. If the negotiations do not succeed (for example, if the prospective ride share partner concludes negotiations with a different partner), then the entity returns to the initial state. If the entity abandons the ride share, it moves to the final "Abandoned" state. Transitions are presented in Figure 5.

C. Protocol Description

The RSS protocol allows the vehicles to find other ride share partners and to establish a connection once suitable ride share

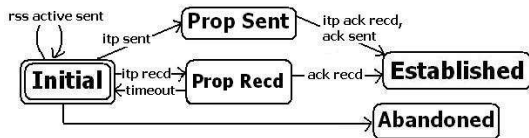


Fig. 5. State Transition Diagram

partner has been found. It uses a regular 3-way hand shake based on the familiar TCP handshake. It is important NOT to overemphasize the automation of the handshake protocol, since the ride share may involve the human user’s interaction. Nonetheless, the protocol is presented to be complete in itself, and if running on automatic mode, should be able to make the decisions on its own.

- 1) While the vehicle is in “initial” state, it sends an **rss active** message every T_a seconds.
- 2) RSU broadcasts **rss active vehicles** list every T_b seconds that includes the application packet data units from all active vehicles.
- 3) In “initial” state, each vehicle receives the list of **rss active vehicles** message and runs the “Find Ride Share” algorithm to obtain a list of candidate ride share partners. After allowing the driver to review the list of candidate ride share partners (or automatically, if so configured), the vehicle enters the “prop sent” state and sends **itp (intent to partner)** messages to the other vehicles.
- 4) In “initial” state, if a vehicle receives an **itp** message from a suitable vehicle, it allows the driver to review the candidate ride share partner. After human user’s acceptance (or automatically, if so configured), vehicle accepts the intent to partner, and sends an **itp ack** back to the potential ride share partner and enters “prop recd” state.
- 5) In “prop sent”, if a vehicle receives an **itp ack** message, it sends an **ack** message and enters the “established” state.
- 6) In “prop recd” state, if a vehicle receives an **ack** message, it enters the “established” state.

V. RESULTS

In this section, we present both analytical and simulation results for the proposed solution of the ride share scheduling problem. We focus the results on following main parameters:

- (i) Is the solution presented above feasible, in terms of 5.9 GHz DSRC specification?
- (ii) Assuming reasonable probability of success of matching ride share partners, how long does it take for vehicles to leave the ride share area?

A. Analytical Results

Consider n vehicles active at a given time on average in the vicinity of the ride share area. Consider W as time spent inside the ride share area by each vehicle on average, prior to reaching a final state. This translates to vehicles arriving and

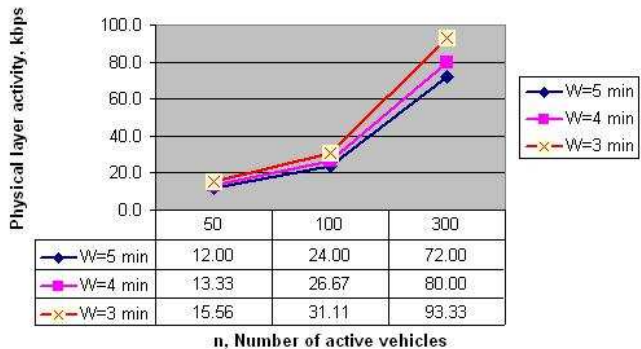


Fig. 6. Analytical Results for Used Throughput

departing at a rate of every W/n seconds on average. This means n **rss active** application service data units (ASDUs) of 128 bytes each every T_a seconds, and one ASDU of $n * 128$ bytes every T_b seconds. Assuming an average of k proposals per successfully completed negotiation, there are $4 * k$ ASDUs of 128 bytes each every W/n seconds on average. Thus, total byte size of ASDUs per second can be derived as follows.

$$\text{ASDU bytes/second} \approx \frac{n * 128}{T_a} + \frac{n * 128}{T_b} + \frac{4 * k * 128 * n}{W}$$

Using proposed numbers of $T_a = 30$, $T_b = 30$ and hypothetical values $k = 4$, and using a factor of 2 to translate ASDU size to actual physical layer transmission, we can obtain the analytical results for different values of n and W , which are presented in Figure 6.

We note that even in the case of 300 active vehicles in the vicinity of ride share area, and an average inter-arrival time of 0.6 seconds ($W = 3$ minutes), the application uses a throughput of less than 100kbps, thus, taking only about 1/60-th portion of estimated throughput of one channel.

B. Simulation Results

For the purpose of the simulation results only, we use the following model.

Geographic model:

- (i) The vehicles in the ride share area are going in the same direction on the current road (referred to as Road 1)
- (ii) Road 1 crosses Roads 2, 4, 6, which run perpendicular to road 1.
- (iii) Points of interest are randomly distributed on these roads with approximate distance of 2 km between them, which have public transportation options.
- (iv) Each vehicle is destined to some internal point which is a random distance up to 2 km away from a point of interest.

Cost model:

- (v) Driving costs vary randomly between cars from 15¢ to 60¢ per mile.
- (vi) Public transportation costs 1.5\$ up to 5km, 2.5\$ up to 10km, and 3.5\$ for more than 10km.
- (vii) The time-to-cost weightage ratio (see Section II-C) for each participant varies randomly between 1 and 2.

Vehicle arrival model:

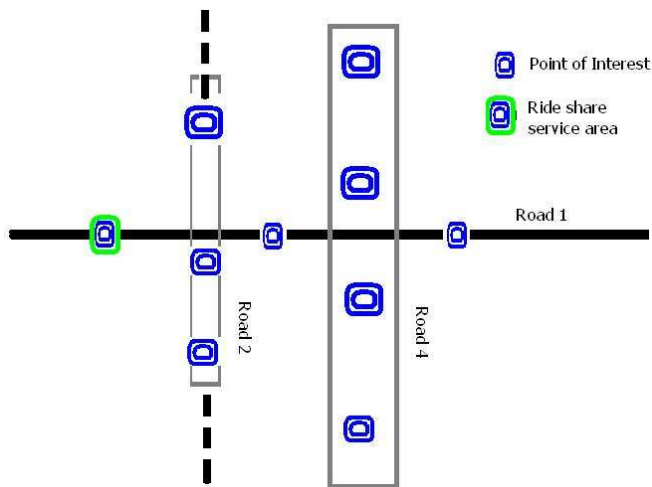


Fig. 7. Geographical model used in simulations

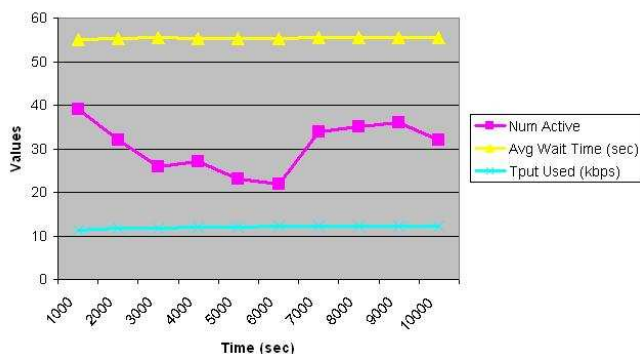


Fig. 8. Simulation results for number of active vehicles, average time to wait, and total throughput used, for an average interarrival time of 2 seconds

(viii) We assume a Poisson arrival process for the vehicles, and the inter-arrival time varies across different simulation runs. The geographical model is shown in Figure 7.

For the purpose of simulation, we use Vehicle Area Network Simulation Toolkit (VAST). Figure 8 shows the simulation results for interarrival time of 2 seconds. Figure 9 shows the simulation results for interarrival time of 8 seconds. For calculating the average number of active vehicles, we ignore the first 1000 seconds (the “build up” phase).

VI. CONCLUSIONS AND FUTURE LOOK

In this paper, we show how the Vehicular Area Networks can be used to solve a practical problem that currently requires significant human coordination and still only reaches partial results. Our results show that while it is possible that the proposed approach can solve the practical problem, it depends upon a critical mass of the users using the proposed system to be successful. Like any other collaborative application, it shows the signs of a chicken and egg problem - without sufficient collaboration, it does not yield results, and it needs good results to encourage collaboration.

The good news however, is that the system shows high

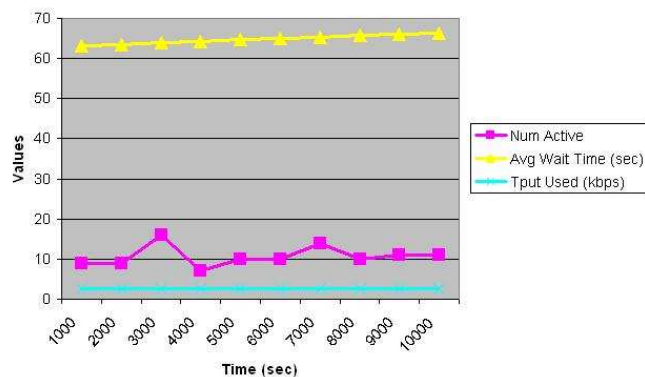


Fig. 9. Simulation results for number of active vehicles, average time to wait, and total throughput used, for an average interarrival time of 8 seconds

success ratio precisely where it is needed the most - in the areas of high congestion. As the traffic and the congestion increases, the likelihood of finding a convenient ride share increases, even assuming minimal market penetration by the proposed system.

The future work can consist of three separate tasks: (i) more simulation and analysis for other geographical models, (ii) system development including integration of OBU and the display unit using the technologies and the APIs already available, and (iii) extension of solution to walkers in the ride share area.

REFERENCES

- [1] IEEE Intelligent Transportation Systems Society, “<http://www.ieee.org/its/>,” Retrieved on November 1, 2008.
- [2] CAR 2 CAR Communication Consortium, “Car 2 car communication consortium manifesto,” August 2007, Version 1.1.
- [3] J.F. Dillenburg, O. Wolfson, and P.C. Nelson, “The intelligent travel assistant,” *Intelligent Transportation Systems, 2002. Proceedings. The IEEE 5th International Conference on*, pp. 691–696, 2002.
- [4] Phillip J. Longman, “American gridlock,” http://www.usnews.com/usnews/news/articles/010528/archive_000087.htm, 2001, U.S. News and World Report.
- [5] Environmental Affairs, “The cost of urban congestion in canada,” <http://adec-inc.ca/pdf/02-rapport/cog-canada-ang.pdf>, Retrieved on November 1, 2008.
- [6] IEEE Vehicular Technology Society Intelligent Transportation Systems Committee, “IEEE trial-use standard for wireless access in vehicular environments (wave) - networking services,” 2007, IEEE Std 1609.3.
- [7] ASTM International, “Astm e2213-03 standard specification for telecommunications and information exchange between roadside and vehicle systems - 5.9 ghz band wireless access in vehicular environments (wave) / dedicated short range communications (dsrc) medium access control (mac) and physical layer (phy) specifications,” <http://www.astm.org>, 2003.
- [8] D. Jiang, V. Taliwal, A. Meier, W. Holfelder, and R. Herrtwich, “Design of 5.9 ghz dsrc-based vehicular safety communication,” *IEEE Wireless Communications*, vol. 13, no. 5, pp. 36–43, 2006.
- [9] Qi Chen, Daniel Jiang, Vikas Taliwal, and Luca Delgrossi, “IEEE 802.11 based vehicular communication simulation design for ns-2,” in *VANET '06: Proceedings of the 3rd international workshop on Vehicular ad hoc networks*, New York, NY, USA, 2006, pp. 50–56, ACM.
- [10] Jing Zhu and S. Roy, “Mac for dedicated short range communications in intelligent transport system,” *Communications Magazine, IEEE*, vol. 41, no. 12, pp. 60–67, Dec. 2003.