Trafficopter: A Distributed Collection System for Traffic Information

Alexandros Moukas[1], Konstantinos Chandrinos[2] and Pattie Maes[1]

[1] Software Agents Group, MIT Media Laboratory, Cambridge, MA 02139
moux,pattie@media.mit.edu

[2] Institute for Computer Science, FORTH, Heraclion, Crete, Greece
kostel@ics.forth.gr

Abstract. We describe Trafficopter, a multi-agent system that collects and propagates traffic information in an urban setting using distributed methods. Agents into the vehicles themselves collect and propagate traffic-related information in a decentralized, self-organizing fashion with no single point of failure. The ideas in this system are the use of the vehicles/agents themselves as a way of collecting traffic data and the way those data are distributed to the interested vehicles. The tools used are a traffic simulator and a set of PDAs / WindowsCE-based terminals equipped with GPS and wireless transceivers. The simulator is used for the investigation of the validity of traffic control and information propagation algorithms in a distributed environment and the WindowsCE terminals for applying the above ideas into the real world.

1 Introduction

Trafficopter is a multiagent system that provides a fast and reliable way of assisting the driver of a vehicle in deciding the most convenient route to his/her destination without using any special road infrastructure. These suggestions change in regular time intervals according to real-time traffic conditions and do not require particular infrastructure for collection and dissemination of the relevant information. Our approach stems from the simple observation that the only reliable witness of the traffic conditions that lay ahead are the cars that are there now, or have been there recently. We attempt to exploit this by creating a virtual distributed network of intercommunicating agent nodes that pass along information (messages) that is enough for drawing conclusions on local traffic conditions. These traffic conditions are depicted on dashboard mounted map-viewing facilities. Extreme accuracy in the quantity of vehicles is not of paramount importance and the perfornance of the system gracefully degrades as the number of vehicles equipped with the system drops.

When driving, in order to find the fastest way to go somewhere one needs two things: a computer map with one's position on it (usually derived by a GPS) and the traffic conditions along the possible paths from the origin to the destination. The former already exists in a set of high-end vehicles. Current approaches to have a system that does the latter are stuck because of the huge costs associated with building a smart road infrastructure (digging up the road and putting magnetic coils beneath the

pavement, sending all signals to data collection centers and then on to a centralized traffic management system which, in turn, sends everything back to the cars). In our proposal the vehicles themselves collect and distribute information about the traffic conditions they are experiencing to other interested vehicles.

To illustrate the potential use of the Trafficopter we include the following example: Let's say someone travels from Boston towards New York. His best source of information about traffic in front of him is the vehicle that is travelling in the opposite direction, from New York to Boston. It knows exactly what is happening further up the road on the opposite lanes. The first vehicle queries about traffic conditions near New York. If the other vehicle doesn't know how to answer the query, it propagates it to other vehicles until somebody responds or the query dies out.

An important feature of the architecture is the use of a *communication network built from nodes with dynamic topology*, which is itself not stationary but mobile (the vehicles are moving). On a layer above this virtual network, message objects are moving from node to node. These objects are unique in that they know their physical as well as their network position unlike conventional computer networks where the IP address is not related to the physical location. Moreover, instead of the nodes of the network (vehicle) “routing” the message objects, the message objects do that on their own, when they instantiate themselves in a node, behaving as “active packets”. When a node broadcasts a message object, copies of this message object get instantiated into a number of different nodes in close proximity with the broadcaster. Unlike an IP packet, the different message objects decide for themselves whether to propagate further, when to expire and they try to minimize the number of hops between origin and destination (thus maximizing network efficiency).

As mentioned above each car is equipped with a commercial GPS receiver for acquiring position, speed and direction and a wireless transceiver for receiving and transmitting data to other cars. All the nodes (we are using the terms vehicle, network node and agent to describe the Trafficopter device that is installed inside each vehicle.) that comprise our system are equivalent and each broadcasts its information on a one-to-many basis. The main advantages of the proposed architecture are the following:

- Very cheap technology: the cost of the car-mounted device is not more that $200 in mass production.
- Car manufacturers can incorporate the system into their new car models without the need of an expensive supporting infrastructure.
- Not all carmakers need to have the system. Our experiments show that even if 25% of the vehicles are equipped with the system, the architecture performance is high.

This paper is organized as follows: in the next section we introduce our motivation and discuss related work. Section 3 analyses our implementation and discusses our experiment, and section 4 summarizes the paper and provides future research directions.

2 Motivation and Related Work

In recent years there have been a number of attempts to offer on-line traffic related information by combining cellular telephony with the Global Positioning System (GPS) technology. Most eminent among them is perhaps General Motors OnStar [13], a system that constantly communicates with several satellites orbiting the earth. The system triangulates the car's position, retains the information and, at the driver's request, transmits the vehicle's location to the OnStar Center via a cellular phone. An advisor at the OnStar center can send assistance if it is called for or use the cellular phone to guide the driver to his or her destination.

In late November 1997, a consortium of hi-tech companies in (IBM, DELCO, Netscape and Sun) have announced the Network Vehicle [11], which according to the respective press release during the COMDEX where a prototype was exhibited, aims at offering Web access to drivers and passengers inside a car through specially constructed panels and head-up displays. A similar prototype presented in October from the same initiative with the collaboration of Daimler-Benz Research and Technology Center comprised a set of computers in the trunk and was connecting to the Internet via an AT&T wireless digital cellular phone and a Metricom wireless modem. The system is briefly described to promise integration of voice-recognition for extra safety, ability to allow car mechanics to diagnose faults from afar, access to travel, weather and business information along with regular net-surfing and provision of games for the kids. Customized traffic updates and navigation help are also mentioned but the details given at the time of writing are geared towards the display of the data through custom-made browsing facilities and not on the actual collection of the traffic data.

Unlike on-board computer systems like Onstar, or similar products marketed in Europe by Philips, which link drivers to proprietary navigation wireless networks, the system we propose does not depend on any central authority, guaranteeing privacy. Also, it does not depend on external or internal infrastructure, other than the existence of a palmtop system to host the software and support the display. Although we use Microsoft Windows CE, during development, all our software is deployed in Java so as to achieve platform independence. Ideally, one should be able to use our navigation software whether he/she is using Windows CE, or any other personal-digital assistant (PDA). On the other hand, systems like the one proposed by the aforementioned consortium involve extensive interconnection with the mechanics of the actual vehicle a development that will require a great deal of time for standards to be established by automobile manufacturers. In our proposal a possible production cost, will be so low that will allow the construction and sale of add-on modules irrespective of the car make.

The majority of the research efforts in Traffic Coordination Systems evolve around efficient and optimize timing of traffic lights in order to maximize the average throughput of cars at major intersections in a given city setting using a centralized traffic control center. The work we are describing herein complements and augments the above-mentioned goal in the sense that it provides the centralized traffic control systems with a cheap way of collecting information. On top of that it provides the individ-

ual car drivers with information on the traffic condition of the road in front of them. It is important to realize that once we have a given number of vehicles operating on the road infrastructure, essentially the vehicles *are* the infrastructure and can be used as data collection mechanisms. A lot of research has been conducted on the area of automated traffic management and centralized traffic data acquisition. Traffic prediction patterns [17] and network flow theory [1] are recent examples. On the data acquisition frontier, sensors developed include sound-based systems [3],[5], laser-based systems [12] and vision-based systems [7]. Different intelligent transportation vehicle applications include data transmission models through cellular channels [19] but unlike this work have to rely on channels.

A very popular system in the DAI community DVMT [10] has a similar scope, and was used for conducting a series of experiments into a variety of DAI-related subjects, like approximation [4] and scheduling [6]. DVMT provides a collection of identical blackboard systems and its domain is the monitoring and interpretation of data from a set of distributed sensors.

In terms of network routing, the most common approaches to routing include link-state routing and source path routing. A typical example of link-state routing networks is the Internet that is based on regular IP routing [16]. In source-path routing the header of the message contains information of all the nodes through which it has to go through. A typical source-path routing protocol is UUCP.

3 Implementation and Experiments

3.1 The Simulator

The MIcroscopic Traffic SIMulator (MITSIM) has been developed by Qi Yang [18] for modeling traffic flows in networks involving advanced traffic control and route guidance systems. MITSIM represents networks at the lane level and simulates movements of individual vehicles using car-following, lane changing, and traffic signal response logic. Probabilistic route choice models are used to capture drivers' route choice decisions in the presence of real time traffic information provided by route guidance systems. The major elements of MITSIM are: network, traffic controls and surveillance sensors, incidents, travel demand, vehicle routing, and vehicle movements. The traffic simulation model is iterations of functions at specified frequencies (time-based) or when certain events occur (event-based). We are running MITSIM on an 8-processor SGI Onyx with 1GB of RAM. We simulate part of Boston area's new Central Artery and Amsterdam's major highway map and traffic conditions with around 4,000 vehicles.

3.2 Traffic data

Our objective is to have all the vehicles' agents know what the traffic conditions are in a metropolitan area by using data collected by other vehicles' agents. Each vehicle agent broadcasts a message object with its local data information (position, speed, direction). A set of other agents in nearby vehicles receives that message object. If all

of them propagated each object they received our network would be very quickly saturated due to the exponential explosion of message objects. If we take into account the same-frequency communication model we are using (all vehicles share the same radio frequency: just one can broadcast at any given time in a given radius), it is evident that in the case of full propagation of all message objects, the network congestion would be eminent.

As stressed above, effort must be put into limiting what information each individual vehicle is going to transmit. We have different types of message objects:

- Beacon-like individual message objects, which are transmitted at regular time intervals by each vehicle. These are the data intrinsic to the vehicle itself (GPS_Position_X, GPS_Position_Y, GPS_Speed, GPS_Direction)
- Centroid message objects, which are again transmitted in regular intervals but contain information about a group of vehicles moving on the same direction with the vehicle that transmits them. Any given vehicle receives a number of individual message position objects (like the one described directly above) from a number of different vehicles. It filters out those that are not travelling in the same general direction that it is, and uses the rest to generate average data of "clusters" of vehicles (Centroid_Position_X, Centroid_Position_Y, Centroid_Speed, Centroid_Direction)
- Query message objects, which are requests for information about traffic conditions at specific points. Unlike the above two, the agents inside each vehicle decide when and where to send query message objects in order to fill-in spatial traffic knowledge gaps and find optimal vehicle paths from the point of origin to the point of destination.

3.3 Map: Grid-layered data

We divide the whole space of our map into a grid. Our vehicle is positioned in the center of the grid. The resolution of the grid is high near the vehicle and drops as it moves away from it. There are certain areas away from the vehicle with high resolution and these are in general areas critical between the point of origin and the point of destination. The data from the grid are superimposed on a map. The vehicles store each message object they receive into their cache and place it on the proper point on the grid. If they receive a more recent packet, they update that grid point. A half-time decay function of the traffic data on the grid is essential for refreshing the grid data.

3.4 Automatic Data Collection: Query Message Objects

In order to sample faster the traffic conditions on coordinates along its path, a vehicle in addition to *passively* listening to other vehicles' transmissions is able to *actively* request the support of other vehicles. Here is a scenario of how the querying system works:

In Figure 1, the vehicle in position A wants to go to its destination, Z. In order to find the fastest way to get there, the vehicle needs to know the traffic conditions in a bunch of points throughout the city (marked by asterisks) and evaluate the optimal path

for the vehicle. Let's say that our vehicle wants to find out about position E in the map. It sends out a query request in the following form:

- (0) Unique ID
- (a) Originator Vehicle Position, Speed, Direction
- (b) Originator's Timestamp
- (c) Requested Position, Direction
- (d) Last Propagator Position
- (e) Last Propagator Timestamp
- (f) Hops Counter
- (g) Total Number of Hops

where (0), (a), (b), (c) and (g) are the initial data. (d), (e) and (f) are filled-in by other vehicles that propagate the information.

Figure 1 Figure 1A: Traffic data superimposed on a Boston Area Map.

Let's say that the vehicle at position A sends out a request. Vehicles south of A receive the request but ignore it since the vehicle at A is closer to E than they are. A vehicle a bit north of A listens to the request, realizes that it is closer to E and since it

doesn't know the traffic conditions at position E, propagates it. The same happens with the vehicle in position C and D. A vehicle near position E receives the request and sends back a reply (i.e. at position E and direction North the average speed is 45 km/h.) A vehicle south of position E receives the request and propagates it south until it reaches the general area of where vehicle A was, adjusted for its new position. The vehicle that originated the query sends out a “received” message to indicate that it has its answer and that there should be no more propagation.

In case a vehicle out of range doesn't receive the cancellation notice and continues to propagate the query or the answer, there are two safeguard measures: the request dies off after a certain number of hops or after a specific time period. Note that the reply has been propagated through the main artery to position A and that all the vehicles on that highway know the traffic conditions at point E. Not all the vehicles that receive a request have to reply (more than one replies for redundancy purposes), but all of them saved the information into their grid and can use it later on, in case another query about E arises.

The query system is based on a stack of messages (a simple mailbox) present at each vehicle. When a vehicle receives a query, it appends the message to its stack and waits for a period of time. The waiting period before propagating the query is a function of its distance between itself and the last propagator of the message: the larger the distance, the shorter the delay. The aim is to minimize the amount of hops between origin and destination (i.e. increase the distance between two hops) as well as the “collisions” of transmitted queries propagated at the same time by everybody that received them.

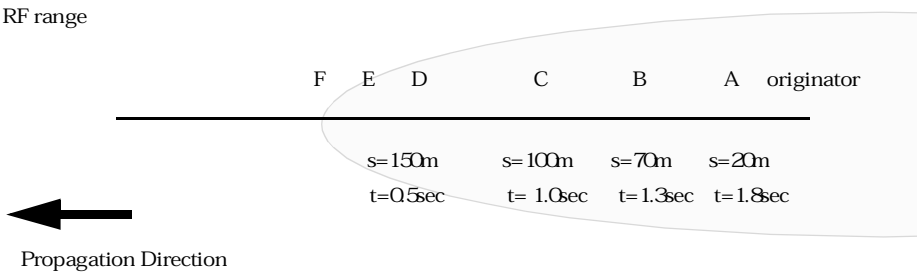

Figure 2 Hop minimization in a query object using positioning data

We described above that we don't want all vehicles to reply to a specific query. On the other hand, we don't want just one vehicle to reply/propagate a query because the message might be lost. In the future the number of the vehicles that re-propagate a query might be adaptive depending on the congestion of the network, but in the current implementation we chose to have each message propagated by at most three vehicles. Let's take the case where vehicle B originates a query on the traffic conditions ahead of F and vehicles A,C,D,E and F receive it (see Figure 2).

- Vehicle A ignores it because as mentioned above is farther away than the originator vehicle (B).
- Vehicles C,D,E and F all receive the same query and append a message to their stack.
- Vehicle F goes off and propagates the message first because it has the greater distance from the last propagator (which happens to be the originator) of the query
- Vehicles C,D and E receive the propagated query and now they all have the original query and the propagated query messages on their stack.
- Before vehicle F replied, the other three vehicles were planning on replying by each waiting for an amount of seconds (let's say t_C,t_D and t_E) inversely related to their distance from the last propagator.
- Now that vehicle F replied, C,D and E register the fact that somebody propagated the query they received (because both messages in their stack have the same ID.)
- Vehicle E goes off and re-propagates the query, and vehicles C and D now have three messages in their stack, all for the same query.
- After vehicle D propagates the query as well, vehicle C has four messages about the same query (the original and three propagations), so it realizes that already three vehicles propagated the query and it should not bother: it proceeds by deleting all four messages from its stack without taking any action.

3.5 Cached Data

If we review the process above, we would notice that a lot of bandwidth could have been saved if there was an efficient caching mechanism [8]. In our specific example, a car traveling from E towards A would know exactly what was happening on the highway ahead of A, on northbound lane. So A's request wouldn't have to be propagated all that far if there was one vehicle coming up from the opposite direction.

If a vehicle receives a more recent packet, it must certainly update the relevant map point. However, a weighting system tracks for the importance of that information to the particular route, otherwise we might had constant updating of irrelevant information. This leads to an architecture that resembles a multilevel hierarchical priority queue with the addition of information expiration along a half-time decay function. In other words, the system decides how up-to-date a piece of traffic information is based on factors like the importance of the particular route to the vehicle’s path from origin to destination, the time that the information is kept in the cache, etc. [15],[9],[2]

3.6 RF Radio Communication

We are using short-range (~200m), low-power RF transceivers for communication between the vehicles’ agents. In an ideal scenario, the effective range of the transceivers would be a perfect circle around the vehicle. However, due to practical limitations (namely buildings on the sides of the majority of urban streets), the real range area looks more like an ellipsis. In special cases, like major cross-sections the

range area looks like a cross, with the remote edges extending along the crossing streets. The major characteristics of our network are the following:

- Low-power and limited range: Since the nodes are transmitting at the same frequency and the network is quite dense, we cannot afford to have long-range broadcast transceivers
- Position-based routing: The system uses only positioning information to forward its packets. The packets know their physical location and decide where to go next based solely on that. Since any node communicates only with its immediate neighbors, there can be no central controlling hub, and the burden of network controls falls upon individual nodes. Each node must play an equal part in routing messages and maintaining network state.

3.7 Experimental Results

We conducted a number of experiments to assess the validity of our architecture. We tried to address issues of overall feasibility and investigate the performance of agents sub-components like the cache and the hop minimizer mechanisms.

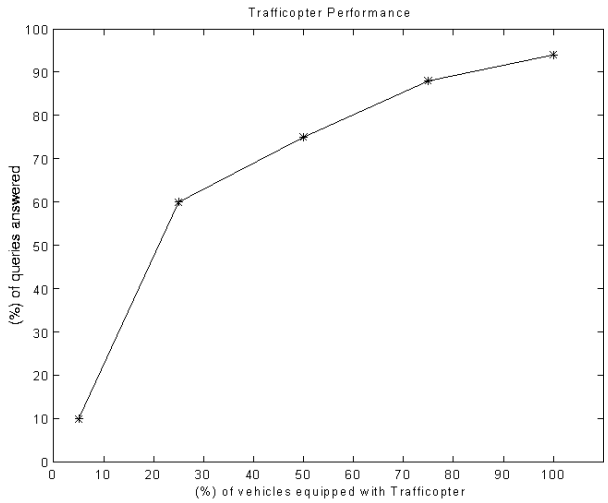

Figure 3 Trafficopter Query Performance

Figure 3 shows how the performance of the system improves when the percentage of vehicles equipped with the system increases. Trafficopter is able to perform quite well even in the case where just a quarter of the vehicles were Trafficopter-aware (60% of queries were answered). In our current implementation, when all the vehicles are equipped with Trafficopter, 94% of all the queries posed are answered. The above-mentioned experiment doesn't take into account the density of the vehicles in the region. In an urban setting the density is quite high and this enhances the performance

of the system. In a low-traffic highway setting we believe that the performance curve would fall off considerably. We are currently conducting experiments where we correlate the percentage of queries answered, the percentage of the vehicles equipped with trafficopter and the density of the vehicles (3D chart). In parallel, we are trying to obtain a finer analysis of the system performance curve, especially in the small-percentage boundary cases. Figure 4a shows the performance of the traffic information caching mechanism. The vast majority of the vehicle queries are answered by other vehicles, without the query message object to have to reach its destination. We are currently performing experiments to better understand spatial patterns of cache hits and assess partial query propagation effects.

Figure 4b reports results from experiments on the validity of the objects "hop-minimizing" mechanism. When this technique is applied, the number of hops an object needs to reach its destination is substantially lower. This increases the bandwidth of our communications network. This is a representative result from our experiments pool. The actual number of hops shown in the figure varies depending on the size of the network, the average distance between each vehicle's origin and destination and other simulation factors.

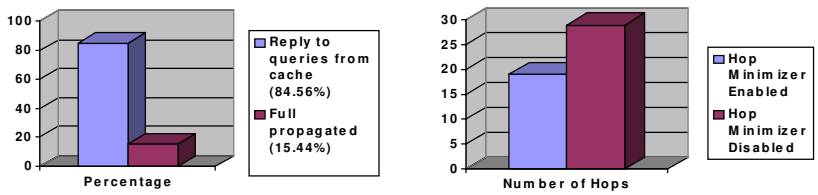

Figure 4 Initial experiments of vehicle cache performance (left) and Hop minimizing technique (right) in Trafficopter simulations

4 Future Work and Summary

In terms of future simulator development we would like to equip traffic lights with passive trafficopter receivers so we can assess the effect on traffic flow. At the same time, we are trying to use the same testbed for different types of architectures, where message objects are going to be the focus of the system. If we view each message object as a virus that tries to infect all the "un-infected" vehicle agents with its traffic information on a specific location, we could reach the same goal (traffic awareness) but from a completely different standpoint. We would like to implement and evaluate both approaches and try to find their strong and weak points.

In another set of experiments we are currently performing, the RF tranceiver range of the tranceivers is variable depending on the number of the Trafficopter-aware vehicles and the density of the vehicles. If the number of density of vehicles increases, the range goes down, in order to increase bandwidth. If on the other hand there are not many vehicles on the road, the tranceiver range increases.

We described Trafficopter, a multiagent system that collects, processes and distributes traffic conditions information in a completely decentralized and distributed fashion. The proposed architecture enables the agents to use message objects in order to request and propagate information about traffic. Results show that our hypotheses are valid and that such a distributed solution is feasible

5 References

[1] Ahuja R., Magnanti T, and Orlin J. (1993). *Network Flows: Theory, Algorithms, and Applications.* Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[2] Bodin F., Seznec A. (1994), *Cache organization influence on loop blocking*, IRISA Internal Report, PI-803, 1994

[3] Chem S., Sun Z. and Bridge B (1997). Automatic traffic monitoringby intelligent sound detection. In *Proceedings of the 1997 IEEE Conference on Intelligent Transportation Systems.*

[4] Decker K., Lesser, V. and Whitehair R. (1990). Extending a blackboard architecture for approcimate processing. *The Journal of Real-Time Systems*, 2(1/2):47-79.

[5] Forren J. and Jaarsma D. (1997). Traffic monitoring by tire noise. In *Proceedings of the 1997 IEEE Conference on Intelligent Transportation Systems.*

[6] Garvey, A. and Lesser, V. (1993). Design-to-time real-time schediling. *IEEE Transactions on Systems, Man and Cybernetics.*

[7] Hancock T., Judd S., Novak C., Ricard S. (1997). Automatic vehicle location using cameras. In *Proceedings of the 1997 IEEE Conference on Intelligent Transportation Systems.*

[8] Handy, J. (1993). *The Cache Memory*, Academic Press.

[9] Hossain, A., Marudarajan A., and Manzoul M. (1991). *Fuzzy Replacement Algorithm for Cache Memory, Cybernetics and Systems*, v 22, pp. 733-746

[10] Lesser V. and Corkill D.(1983). The distributed vehicle monitoring testbed. *AI Magazine*, 4(3):63-109.

[11] NetVeh, 1997, The Network Vehicle, http://www.alphaworks.ibm.com/networkvehicle.

[12] Nishizawa S., Cheok K., Young W. and Zhao W. (1997) Traffic monitor using two multiple-beam laser radars. In *Proceedings of the 1997 IEEE Conference on Intelligent Transportation Systems.*

[13] Onstar (1997) General Motors Smart Car http://www.onstar.com

[14] Poor, R. (1997) Hyphos: A self-organizing, wireless network. Master's thesis, Massachusetts Institute of Technology.

[15] Przybylski, S. A., (1990). *Cache and Memory Hierarchy Design*, Morgan Kaufmann.

[16] Tanenbaum, A. (1996). *Computer Networks.* Prentice-Hall, NJ, 1996

[17] von Toorenburg J, der Linden R., van Velzen B.(1996). Predictive control in traffic management. Final Report AV-2468, Ministry of Transport, Public Works and Water Management, The Netherlands, 1996.

[18] Yang and Koutsopoulos (1996). *A microscopic traffic simulator for evaluation of dynamic traffic management systems.* Transportation Research C, 4 (3):-129, 1996.

[19] Zhao, Y. (1997) Efficient and reliable data transmission for cellular and GPS-based mayday systems, In *Proceedings of the 1997 IEEE Conference on Intelligent Transportation Systems.*