Enhancing Traffic Intersection Control with Intelligent Objects

Rudi Ball
Imperial College
London
United Kingdom
r.ball@imperial.ac.uk

Naranker Dulay
Imperial College
London
United Kingdom
n.dulay@imperial.ac.uk

***Abstract*—Traffic control is an old and ever growing problem in cities throughout the world. Within many cities, intersections represent bottlenecks in the flow of traffic. Evaluating intersections control is complex and difficult. Given this, intersection management is both costly and time consuming. This paper considers the potential benefits of enhancing the traffic intersection with the use of intelligent objects in vehicles. We present, compare and demonstrate a novel Vehicle Back-Off Protocol against a classical Timed Traffic Control system. Our protocol uses ad-hoc messaging, collision avoidance and shared journey plans as a means by which to reduce delay, adapt a journey and maximise the efficient usage of a traffic intersection. We use simulation to model and evaluate intersection control.**

I. Introduction

As the populations of cities continue to grow, traffic control within them becomes an ever larger problem. Thus far, methods of alleviating traffic congestion in cities have been broadly approached using systems such as Intelligent Transportation Systems (ITS), which seek to render objects within the transportation system (e.g. vehicles, roads, traffic lights, message signs, etc.) "intelligent", embedding them with microcomputers, sensors and actuators, enabling them to communicate with each other using wireless technologies. ITSs aim to monitor and manage factors such as traffic-flow and routes to improve safety and reduce vehicle wear, reduce journey times and lower fuel consumption [3][13]. A subset of these solutions attempt to improve traffic flow at intersections, although management decisions are typically centralised [16][17]. While a traffic intersection represents a local problem, its combination with other intersections and the topography of a city makes traffic control a difficult and complex problem to address.

The work presented in this paper considers a contrasting distributed approach to traffic intersection control using shared *journey plans* and *avoidance*. The work presents a method of evaluating intersection control strategies. Our Vehicle Back-Off Protocol (VBP) is compared to a classical Timed Traffic Control (TTC) system. We assume vehicles to be addressable mobile intelligent objects. Vehicles adapt their speed to avoid predicted future collisions with other vehicles, in effect repeatedly micromanaging local speed to minimise journey delay and improve vehicle flow. Vehicles cooperate using ad-hoc messaging to safely organise and travel over a shared intersection. We envision the interaction of intelligent objects as the means by which to safely and efficiently organise intersections.

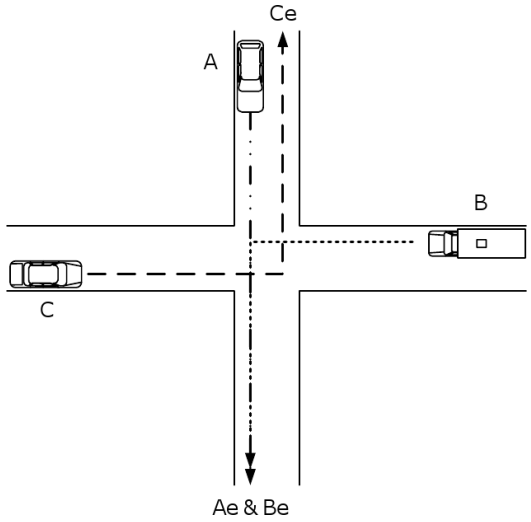

Fig. 1. Scenario: vehicles (A, B and C) plan to travel to their respective end points (Ae, Be and Ce). Vehicles must avoid colliding with one another by adapting their speed, simultaneously optimising the incurred delay during their journey. Vehicles are assumed to be intelligent objects (sensing and actuating devices).

II. Related Work

Modern traffic control has been broadly approached from both top down (large scale systems) and bottom up (localised protocols) approaches. Intelligent transport systems (ITS) serve as one such set of large scale solutions to traffic congestion [3][13]. Many ITS systems seek to manage real-time traffic conditions. They are typically expensive to both deploy and maintain and their architectures are typically centralised, requiring large networks for connectivity. Systems like WikiCity [6] propose collecting real-time data about the city using a community of mobile citizens to curb these difficulties. Using this type of city data, traffic scheduling [10] has been proposed as a method of improving traffic flow in the context of automated vehicles. The reality of deploying such a system has been motivated by initiatives like the Driverless Cars project at Google [20].

A distributed solution to traffic control provides local autonomy and modularity. However, distribution presents a number of challenges including issues of consensus, fault-tolerance and stabilisation. The Smart Cars approach [21] and work by Cahill et al. [5] represent distributed approaches to traffic control along multi-lane highways for automated vehicles. Multi-agent approaches have largely considered rule-based traffic light examples [1][8][11]. Bull et al. [4] have attempted to use machine learning techniques to learn and strategically manage traffic control at intersections. Multi-agent collision avoidance has considered swarm systems and specified behavioural approaches [7][18].

III. Scenario

The traffic intersection problem (Figure 1 and Figure 2) is a useful micro-scenario from which to understand complex global traffic systems. The layout of the intersection presents a number of intersection points where two vehicles could collide. The intersection definition used in this paper is common to previous work by Giridhar and Kumar [10] and Hirankitti et al. [11]. Two straight roads intersect one another at a perpendicular angle. There are two lanes with traffic flowing in opposite directions. The intersection represents a shared crossing where vehicles must avoid colliding while reducing total delay. Vehicles are capable of communicating using messages. Each vehicle can determine its geographic position using on-board positioning systems and techniques.

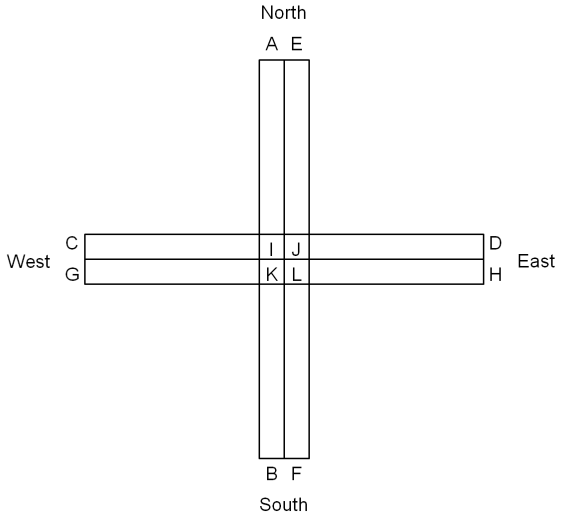

Fig. 2. Intersection scenario and context: vehicles can enter and exit from one of four compass points. We assume vehicles travel on the right of a dual carriage way road. A vehicle can travel one of 12 mobility types.

We assume that a typical driver using a GPS system will initially specify their *intended destination*. A data pair specifying *present position* and the *intended destination* is submitted to a journey planner. A set of directions are returned to the user describing the route which should be taken to reach the intended destination. The route used is represented abstractly as the *journey plan*. The journey plan is composed of route position pairs and is modifiable throughout the journey. Each vehicle holds its own *journey plan* which is shared when neighbouring vehicles enter its communication range. A vehicle is capable of measuring a variety of local data, including *position* (geographic position using latitude and longitude), *bearing* to the *next waypoint*, *speed* and the *distance* to the *next intersection*. Messages contain meta-tags which include position and last-sender information.

STRAIGHTS	TURNINGS	CROSSINGS
$A \rightarrow I \rightarrow K \rightarrow B$	$A \rightarrow I \rightarrow C$	$F \rightarrow L \rightarrow J \rightarrow I \rightarrow C$
$F \rightarrow L \rightarrow J \rightarrow E$	$F \rightarrow L \rightarrow H$	$D \rightarrow J \rightarrow I \rightarrow K \rightarrow B$
$D \rightarrow J \rightarrow I \rightarrow C$	$D \rightarrow J \rightarrow E$	$G \rightarrow K \rightarrow L \rightarrow J \rightarrow E$
$G \rightarrow K \rightarrow L \rightarrow H$	$G \rightarrow K \rightarrow B$	$A \rightarrow I \rightarrow K \rightarrow L \rightarrow H$

TABLE I
Intersection mobility patterns: the patterns represent all accepted vehicle journeys.

Using letters we can denote significant points within the intersection example (Figures 1 and 2) and define the perpendicular roadways, intersecting at the center (labelled I, J, K, L). Vehicles can travel in lanes in opposite direction. Yet, where an intersection occurs, vehicles must be organised so that crossing vehicles do not collide. Traffic travelling from perpendicular angles is required to turn or cross existing flows to correctly navigate the intersection to travel onwards to one of three other goals in the subset of originating points. We should note that the natural organisation of the intersection specifies that no more than two vehicles can collide at I, J, K or L. An example route can be defined as the set of points which a vehicle must reach, including those sub-positions, where a subset journey may be specified by the route which uses $A \rightarrow I \rightarrow K \rightarrow B$. There are twelve possible routes to negotiate the intersection, formulated in three sets (Table I). Using these routes we build traces which provide us with repeatable scenarios on which to test the performance of varying protocols.

IV. Architecture

Each vehicle is represented using a *message-based* intelligent object architecture (Figure 3) [9]. As intelligent objects, vehicles are addressable and capable of both sensing and actuating within their local space.

Multiple vehicles broadcast messages into the local space at each time step (t_B) - limited by the maximum communication range available. Messages received by neighbouring vehicles are placed in a message Inbox, where they are disassembled to reveal a list of *Input Payloads* (*Pi*). Each *Pi* is interpreted by the execution loop, which continuously executes a *behaviour algorithm* (e.g. TTC or VBP). If the time-step interval is less than or equal to 1 then we assume that a memory is not needed - operations and adaptation are real-time and memory serves no purpose for binding a previous history with the present state. A *behaviour* represents the observable action taken by a vehicle within a specific state. We show this behaviour as effecting the *local space* (communicable range of communication with intersection) within which both vehicles

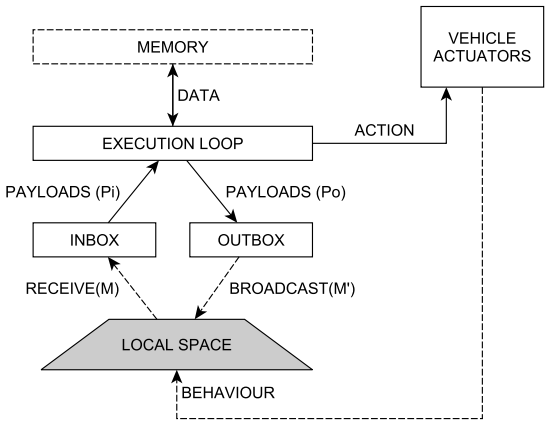

Fig. 3. Intelligent object architecture.

and messages reside. Figure 3 exposes two feedback loops existing in the architecture.

Messages are broadly generic however the *Payload* component of each message can be fashioned specifically for a given system. In this particular case, the Payload message is tailored to hold data with reference to a vehicle's *position*, intended *target*, *speed* and the sender's *journey plan* (limited to t_L positions). The journey plan can be visualised as a set of tuples holding future position-time pairs for a given sized t_L. We can write an example plan as the list sequence: $[(p_0, t_0), (p_1, t_1), ..., (p_n, t_n)]$ where $n \geq 0$, p_n represents the position of a vehicle at a moment in time (t_n). While $n = 0$ represents the present instance of time, $n \geq 1$ defines future predicted positions - where a vehicle believes it will be in a future time step. Input messages are disassembled and the *Pi* component of each message is compared and applied as a parameter of the local vehicle protocol.

V. Protocol

A traffic intersection should seek to achieve two primary goals: (i) reduce delay (within bounds, while adhering to speed limits and cautions) and (ii) avoid collisions occurring with other vehicles (maintain safety). Performance is measured by examining the *throughput* (rate at which vehicles reach their end point). We assume the following set of inputs:

- *position* using a GPS or other system;
- *input messages*, messages received from the previous time-step;
- *navigation functions*, to determine and calculate bearing, distance and speed;
- *modifiable message payloads*, data structures specific to the protocol being used.

The two protocols we present and compare (TTC and VBP) attempt to achieve the goals highlighted using these minimal inputs. A number of static variables set globally for each device. The variables include:

- Respawn - the time interval between new vehicles entering the intersection;
- Communication Range - the maximum specified meter range of communication to which a broadcast message can travel and be received;
- Broadcast Interval - the time interval between message broadcasts, specified in seconds.

A. Timed Traffic Control

The *Timed Traffic Control* (TTC) protocol emulates timed classical centralised traffic control methods (simple switched traffic lights) - green to go and red to stop. The timed traffic control algorithm was used as a control test to measure and compare algorithms. A go-stop interval was chosen as the time when traffic travelling from a particular axis was allowed to flow (East-West or North-South). Where the timer periodically elapsed the traffic direction was switched allowing waiting traffic to continue in the specified direction. While more modern alternative traffic control approaches do exist (e.g. methods using inductive loops and light sensors), TTC represents a base line performance on which to compare both VBP protocol performance and future proposed methods.

B. Vehicle Back-Off Protocol

The *Vehicle Back-Off Protocol* (VBP) uses an adjustable *ranking* mechanism for the specification of priority (Algorithm 1). Back-Off represents the yielding aspect of adaptation necessary to avert a collision. A yielding rank has thus far been tested using two methods: random seeding and ordered priority. This paper focuses on ordered priority. The message *Inbox* contains messages received from neighbouring vehicles in the previous time-step. A *Mobility* object stores the journey plan of a vehicle. In this respect the Mobility object provides an interface to position, direction, speed and predicted future track data (i.e. the position for a vehicle for t_L future steps during the journey). The orientation of a vehicle is categorised using a compass bearing and segmented regions. A 35 degree cone was used to gauge whether an object was ahead of another object. The return *Outbox* informs other neighbouring vehicles of the state of the present vehicle.

The operation of VBP is broken into three phases: (i) a *Filtering* (lines 2-7), (ii) *Collision Avoidance* (lines 8-16) and (iii) *Sharing* (lines 17-21) phases. Initially the algorithm identifies those messages from neighbouring peers which affect the present journey plan. A collision detection method determines if a collision shall occur between itself and the message sender for a discrete set of time-steps within the future, known as the look-ahead time (t_L). If a collision is detected to occur the message is placed into a secondary filtered message set. The filtered collection is then once again filtered for the closest message ahead (m_c) to the present vehicle. m_c is seen to represent the most immediate danger to a vehicle. Having identified the immediate collision to avoid, m_c is checked against the local vehicles future path and the held future path (attached by the message sender). A collision time (t_c) is computed.

The *ranking* sub-routine constitutes the most important condition within the protocol. Ranking is determined by analysis

Algorithm 1: Vehicle Back-Off containing a modifiable ranking condition.

```
Data: Inbox, Mobility, t_L
Result: Outbox
1  begin
2      List filter
3      for i ← 0 to Inbox.size do
4          Message m_i ← Inbox[i]
5          if detectCollisionIn(m_i, t_L) then
6              filter ← filter ∪ (m_i)
7      m_c ← getClosestMessageAhead(filter);
8      if exists m_c then
9          Payload p ← m_c.payload
10         t_c ← predictCollision(FuturePath(t_L),
           p.FuturePath)
11         if (exists t_c) and rankingCondition(m_c) then
12             reduceSpeed()
13         else
14             increaseSpeed()
15     else
16         increaseSpeed()
17     if clock (mod t_B) = 0 then
18         msg ← Message(position, id)
19         p ← Payload(bearing, Mobility.speed,
           Mobility.target, Mobility.futurepath)
20         msg.setPayload(p)
21         outbox.add(msg)
```

of m_c. Ranking is necessary to determine which vehicle should yield to which other vehicle - which vehicle should adapt their speed to avoid a collision. In *random seeding*, the action to yield is randomly chosen using a boolean value for each iteration, while in ordered priority we used the name identifier of a vehicle as a condition for yielding. A lower name identifier was deemed to have precedence over a higher value. In all cases the ranking condition needs to determine precedence between two vehicles based on the characteristics of the communicating vehicles. Within VBP ranking is a requirement and correct ranking is fundamental to correct operation of the collision avoidance phase.

The final phase (sharing) periodically broadcasts a new state message (*msg*) to all neighbouring vehicles such that all other neighbouring vehicles can determine their own local adaptation. Hence adaptation is repeatedly computed. Two vehicles compute from the same scenario and set states an individual adaptation. VBP is hence discrete in its capacity to find a solution. A yielding vehicle sacrifices its own optimal journey for the benefit of neighbouring vehicles.

VI. Evaluation and results

We evaluated the application of TTC and VBP for traffic intersections using the *Geographic Urban Simulator* (GUS).

	r=10s	*r*=15s	*r*=20s
cm=50m	0.02	0.03	0.01
cm=100m	0.08	0.06	0.02
cm=150m	0.11	0.13	0.04

TABLE II
TOTAL DELAY RATIO COMPARISON (VBP:TTC) EXPERIENCED DURING 600 SECONDS OF SIMULATION TIME. VBP OUTPERFORMS TTC FOR RATIOS < 1.0.

	r=10s	*r*=15s	*r*=20s
cm=50m	0.7234	1.2978	1.15
cm=100m	1.15	1.22	1.0697
cm=150m	1.2777	1.1272	1.1219

TABLE III
TOTAL THROUGHPUT RATIO COMPARISON (VBP:TTC) EXPERIENCED DURING 600 SECONDS OF SIMULATION TIME. VBP OUTPERFORMS TTC FOR RATIOS ≥ 1.0.

The GUS is a discrete event simulator built atop of the Java in Simulation Time (JiST) framework [2]. The simulation framework is efficient, out-performing existing highly optimized simulation run-times both in time and memory consumption [2]. The usage of Java and the direct integration of geographic mobility specifications makes development and deployment easier and more adaptable than alternative approaches as found in systems like NS-2 [12], the One Simulator [14] and GloMoSim [22]. The protocols experimented with using the GUS can be easily ported to devices supporting Java (e.g. Dalvik Android [19]).

We focus on experiments used to compare protocol performance for a four way traffic example (measuring delay and throughput for TTC and VBP). Each experiment used a static set maximum speed of 8 m/s (17.9 miles per hour - considered a safe intersection approach speed [15]), varying maximum communication range (*cm*) between 50 and 150 meters and varying respawn (*r*) between 10 and 20 seconds. Mobility patterns used generate synthesised traces, which in turn presented us with the capacity to rerun differing protocols on common mobility patterns.

The model has been constrained and does not yet consider the effect of pedestrians, road conditions, weather and alternative driver behaviours. All vehicles are assumed to use the same VBP protocol with an ordered ranking condition of precedence (lower numbered vehicles have priority over higher numbered vehicles).

A. Delay

Delay is measured as the cost of adaptation. Each vehicle journey plan assumes initially that no delay shall exist along the entire journey. Hence, the initial journey plan represents the optimal journey. As a vehicle experiences delays during a journey the optimal journey is modified and a delay incurred.

Figure 4 depicts a typical experiment and illustrates each total delay instance in time as an impulse. TTC is seen to produce a periodic delay effect as traffic initially slows and

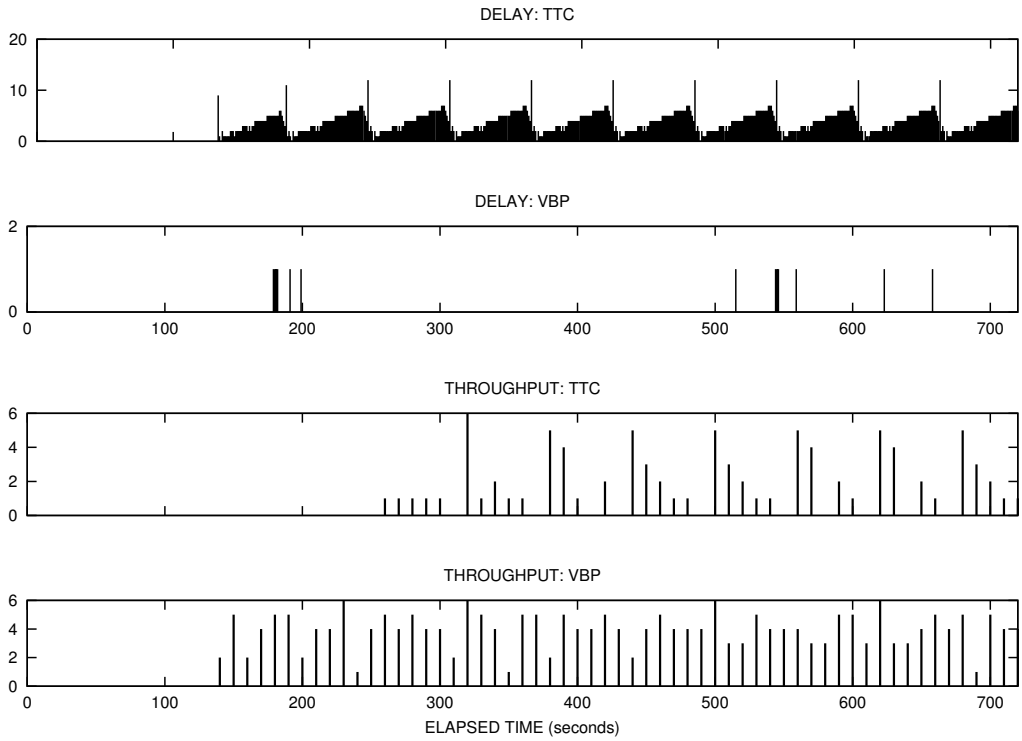

Fig. 4. Marginal delay and throughput for TTC and VBP approaches, given a communication range of 200 meters and respawn of 10 seconds. Each impulse indicates a total delay or throughput measurement taken at an instant during the experiment (elapsed time).

backs up, vehicles are seen to be waiting for a signal to change. In contrast, VBP delay is seen to be both more dispersed and more random than TTC, with a lower marginal cost.

The delay ratio performance difference between TTC and VBP was computed as VBP:TTC (Table II). Table II considers the effect of varying both communication range (*cm*) and respawn (*r*). In all experiments VBP was seen to significantly outperform TTC, with a best performance of 0.01 and a worst performance of 0.13.

B. Throughput

Throughput represents the average rate of successful vehicle journeys using the intersection. Figure 4 shows the distribution of vehicles successfully reaching their goal destination for buckets of 10 seconds. Marginal throughput spikes were higher for the timed approach as vehicles are more closely collected together after they had been required to wait at an intersection. Effectively a convoy of vehicles would reach their final destination. In contrast to VBP, TTC was seen to be favourable in scenarios where communication range was limited and vehicle density was high (Table III), while VBP throughput was significantly improved given larger communication ranges and more sparsely dispersed traffic scenarios (VBP:TTC throughput ratios $\geq$ 1.0).

C. Summary

In comparison (Tables II and III), we found VBP to reduce total system delay whilst improving the throughput of vehicles using a road intersection. However, VBP was only beneficial in cases where the following distance between vehicles was large enough to allow for a crossing on the intersection - if this was not the case, then a limited backlog of vehicles occurred at one of the four intersection stops and performance was degraded. Clear patterns can be seen in both delay and throughput comparisons, yet these patterns of performance are due to the experiment scenario, vehicles are not randomly inserted into the intersection, rather they are added periodically. Given this, the regular distribution patterns are more common in TTC than those patterns found in VBP.

VII. Discussion

Ideal traffic intersection control is difficult to define. For the purposes of this paper, we have focused on the avoidance of collision and the reduction of delay. Collision avoidance is a safety-critical concern and a primary challenge. We do not argue for or against autonomous vehicles [20]. Delay may be classified as a secondary aim. Timed traffic control systems attempt to partition and allocate time either according to a regime of “fairness” or specifically to demand (e.g. inductive loops and the employment of sensors) - most importantly they loop safety critical elements (orange lights) or periods in which the intersection can be reset to deal with a new collection of crossing vehicles. VBP attempts to maximise the usage of the intersection by focusing on collision avoidance and adapting speed to “nudge” the overall intersection into a state of minimal delay. The example given in this paper may be considered a simplified model of traffic intersection control,

yet it provides a base comparison on which further work can be applied. A more realistic approach may take into account the added complexities of such an intersection, including pedestrian mobility, weather and road conditions. However we should still be clear as to measure the system on the collision, delay and throughput metrics. The GUS has generated and compared synthesised mobility traces, as public traces for intersection usage were not available. Real intersection traces could be provided to more accurately describe and measure an intersection for presently used protocols.

VIII. Conclusion and Future Work

This paper has evaluated a new *Vehicle Back-Off Protocol* for road traffic intersections and compared the approach with classical *Timed Traffic Control*. The Vehicle Back-Off Protocol makes use of vehicles as intelligent objects at an intersection. Vehicles use sensory inputs, shared journey plans, messaging and behavioural adaptation to reduce both total delay and maintain vehicle flow. By predicting collisions and refining vehicle behaviour the Vehicle Back-Off Protocol reduces the effect of total delay. Comparing these two approaches, we present a methodology for evaluating traffic intersection control using throughput and delay metrics.

While we have shown that a distributed approach to intersection control by vehicles is feasible within bounds using specific parameters, the work presents a number of future research directions, highlighting issues including robustness, consensus, message lifetime and security.

The robustness of the protocol has not been explored, nor has the protocol been required to operate using erroneous data or in a fault tolerant environment. Performance of the system given multi-hop messaging has not been considered. The effect of scaling the system to include multiple intersections is an open problem. To what extent vehicle behaviour adaptation changes the overall system is unknown. Pedestrian behaviour may benefit from sharing journey plans with vehicles such that vehicles may adapt their behaviour to accommodate pedestrian delays and vice versa. We intend to test the Vehicle Back-Off Protocol in the context of train and airspace scenarios. Finally, the security of the system has not been investigated. It is unclear what effect inconsiderate drivers may have on the functioning of the system.

Acknowledgements

This research was undertaken as part of the Adaptable Environments for Distributed Ubiquitous Systems (AEDUS2) project. UK EPSRC research grant EP/E025188/1. We thank the Policy group for their feedback.

References

[1] M. Balmer, N. Cetin, K. Nagel, and B. Raney. Towards truly agent-based traffic and mobility simulations. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 60–67, Washington, DC, USA, 2004. IEEE Computer Society.

[2] R. Barr, Z. J. Haas, and R. van Renesse. *Scalable Wireless Ad hoc Network Simulation*, chapter 19, pages 297–311. CRC Press, August 2005.

[3] A. Biem, E. Bouillet, H. Feng, A. Ranganathan, A. Riabov, O. Verscheure, H. Koutsopoulos, and C. Moran. Ibm infosphere streams for scalable, real-time, intelligent transportation services. In *SIGMOD '10: Proceedings of the 2010 international conference on Management of data*, pages 1093–1104, New York, NY, USA, 2010. ACM.

[4] L. Bull, A. Tomlinson, J. Addison, and B. Heydecker. Towards distributed adaptive control for road traffic junction signals using learning classifier systems. In *In L. Bull*, pages 276–299. Springer, 2004.

[5] V. Cahill, A. Senart, D. C. Schmidt, S. Weber, A. Harrington, and B. Hughes. The managed motorway: real-time vehicle scheduling: a research agenda. In *HotMobile '08: Proceedings of the 9th workshop on Mobile computing systems and applications*, pages 43–48, New York, NY, USA, 2008. ACM.

[6] F. Calabrese, C. Ratti, and K. Kloeckl. Wikicity: Real-time location-sensitive tools for the city. 2008.

[7] D. Chang, S. Shadden, J. Marsden, and R. Olfati-Saber. Collision avoidance for multiple agent systems. volume 1, pages 539 – 543 Vol.1, dec. 2003.

[8] A. Doniec, R. M, S. Espi, and S. Piechowiak. S.: Dealing with multi-agent coordination by anticipation: Application to the traffic simulation at junctions. *In: EUMAS*, 2005:478–479, 2005.

[9] M. R. Genesereth and N. J. Nilsson. *Logical foundations of artificial intelligence*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987.

[10] A. Giridhar and P. Kumar. Scheduling automated traffic on a network of roads. *Vehicular Technology, IEEE Transactions on*, 55(5):1467 –1474, sep. 2006.

[11] V. Hirankitti, J. Krohkaew, and C. J. Hogger. A multi-agent approach for intelligent traffic-light control. In *World Congress on Engineering*, pages 116–121, 2007.

[12] T. Issariyakul and E. Hossain. *Introduction to Network Simulator NS2*. Springer Publishing Company, Incorporated, 2008.

[13] A. D. Joseph, A. R. Beresford, J. Bacon, D. N. Cottingham, J. J. Davies, B. D. Jones, H. Guo, W. Guan, Y. Lin, H. Song, L. Iftode, S. Fuchs, B. Lamprecht, K. Kyamakya, J. G. Fernández, J. C. Y. García, Y. S. M. García, J. de Gracia Santos, M. Nimesh, G. Pan, Z. Wu, Q. Wu, Z. Shan, J. Sun, J. Lu, G. Yang, M. K. Khan, and J. Zhang. Intelligent transportation systems. *IEEE Pervasive Computing*, 5(4):63–67, 2006.

[14] A. Keränen, J. Ott, and T. Kärkkäinen. The one simulator for dtn protocol evaluation. In *SIMUTools '09: Proceeding of the 2nd International Conference on Simulation Tools and Techniques*, New York, NY, USA, 2009. ACM.

[15] S. A. Lovegrove. Approach speeds at uncontrolled intersections with restricted sight distances. *Journal of Applied Psychology*, 63(5):635 – 643, 1978.

[16] R. Miller and Q. Huang. An adaptive peer-to-peer collision warning system. In *in IEEE Vehicular Technology Conference (VTC*, pages 317–321, 2002.

[17] C. Nowakowski and J. OConnell. Cooperative Intersection Collision Avoidance Systems - Suburban Left Turn Assistant: Interim Report on the Human Factors Data Mining Efforts. http://www.intellidriveusa.org/508/Library/Librarysections/PATH [Online; accessed October-2010].

[18] C. W. Reynolds. Flocks, herds and schools: A distributed behavioral model. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 25–34, New York, NY, USA, 1987. ACM.

[19] R. Rogers, J. Lombardo, Z. Mednieks, and B. Meike. *Android Application Development: Programming with the Google SDK*. O'Reilly Media, Inc., 2009.

[20] S. Thrun. Google Blog: What we're driving at. http://googleblog.blogspot.com/2010/10/what-were-driving-at.html, 2010. [Online; accessed October-2010].

[21] P. Varaiya. Smart cars on smart roads: Problems of control. *IEEE Transactions on Automatic Control*, 38:195–207, 1993.

[22] X. Zeng, R. Bagrodia, and M. Gerla. Glomosim: A library for parallel simulation of large-scale wireless networks. In *in Workshop on Parallel and Distributed Simulation*, pages 154–161, 1998.