Trading Accuracy and Resource Usage in Highly Dynamic Vehicular Networks

Fitzgerald, Emma; Landfeldt, Björn

Published in:
International IEEE Conference on Intelligent Transportation Systems

2013

Link to publication

Citation for published version (APA):
Fitzgerald, E., & Landfeldt, B. (2013). Trading Accuracy and Resource Usage in Highly Dynamic Vehicular Networks. In A. Hegyi (Ed.), *International IEEE Conference on Intelligent Transportation Systems* IEEE - Institute of Electrical and Electronics Engineers Inc..

Total number of authors:
2

Trading Accuracy and Resource Usage in Highly Dynamic Vehicular Networks

Emma Fitzgerald*†
Email: emma@it.usyd.edu.au
*School of Information Technologies
The University of Sydney
NSW 2006
Australia
†National ICT Australia
Australian Technology Park
Level 5, 13 Garden Street
Eveleigh NSW 2015
Australia

Bjorn Landfeldt‡
Email: bjorn.landfeldt@sydney.edu.au
‡School of Electrical and Information Technologies
Lund University
SE-221 00 Lund
Sweden

Abstract— **Vehicular networks bring new ways of viewing road traffic management and safety applications. For the first time, it will be possible for vehicles to exchange information and build fine-grained knowledge about the current situation, estimating risks and adapting their driving. Central to these applications is the need to exchange information in a highly dynamic environment, building a view of the current situation before the conditions change. This is turn requires the distributed algorithms used to converge on low error margins quickly.**

In this paper, we investigate the performance of such a distributed algorithm which aims to build a common assessment of the risk level among vehicles to trade off accident risks with road throughput. In particular, we examine how the convergence rate is affected by network size and node density, and also how the error in the algorithm's output is affected by the rate at which nodes send out update beacons. We develop a variable-rate beaconing scheme in order to find a trade-off between accuracy of outputs and network resource usage. We then formulate this as a more general optimisation problem applicable to other applications or distributed algorithmic problems in highly dynamic distributed systems such as VANETs.

I. Introduction

As mobile systems become more widely used and more pervasive in different areas of our lives, they are required to operate in more uncontrolled and variable environments. One particular example of this is in vehicular ad-hoc networks (VANETs), where applications must function under the sometimes turbulent and always changing conditions of road traffic. In this paper, we will examine the performance of applications in such highly dynamic networks, in which there is a large degree of node mobility and where network topology may change frequently.

In particular, we will focus on distributed applications for VANETs in which data is processed and used in-network and where the outputs of the application are time-critical and need to be kept accurate, even when input data may change rapidly and unpredictably. We develop a methodology for analysing such algorithms and adapting them to changing network conditions whilst minimising usage of network resources.

We take as an example an accident risk estimation algorithm for a VANET and examine how the convergence rate of this algorithm is affected by varying the network size and node density, which may be caused by changing traffic conditions. We will further examine how the error in the algorithm's output varies with the rate at which nodes send out update beacons by taking a mobile scenario in which there is a large and sudden change in input values. We will then develop a variable-rate update scheme in order to balance the concerns of low error with low usage of network resources, and finally we will extend this idea to a general optimisation problem for finding the best trade-off between these two goals.

The rest of this paper is organised as follows. In Section II, we give an overview of the characteristics of the type of networks and applications we are dealing with and why it is important to understand the relationships between convergence rate, network size and node density, beaconing rate and error. In Section III we describe in more detail the particular algorithm we are working with and then our aims for this work in Section IV. Sections V–VIII describe our experimental work, first on how convergence rate is affected by the number and spacing of nodes, then on error and beaconing rate, and finally on our variable beaconing rate scheme. In Section IX we discuss how this work can be generalised to other applications and scenarios and finally we conclude in Section X.

National ICT Australia (NICTA) is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

II. Motivation

Networks which have rapidly changing topologies and high node mobility present unique challenges in managing error in distributed applications. Even where an algorithm will converge to a steady state, because of the frequent changes in the network, it may often not be possible to actually reach this state, or, once reached, to stay there for any length of time. In this paper, we investigate how network conditions affect convergence rates and levels of error, and how these can be managed effectively.

We examine the relationship between convergence rate, error and update rate for a distributed application using sensed data in a VANET. In particular, we consider applications where it is critical that data is accurate and up-to-date (such as vehicular safety applications), and where inputs may change rapidly, unexpectedly and non-continuously. We discuss each of these characteristics further below using an accident risk estimation algorithm for VANETs and as such, we consider each of these characteristics in this context. However, our work is applicable to any system that satisfies the following criteria.

The network has high mobility and dynamic topology. In a VANET, because each node is a vehicle, nodes are constantly moving at high relative speeds to each other. Additionally, the node density and configuration varies greatly — consider, for example, the difference between a busy urban intersection in peak hour and a rural highway late at night — and can change quickly. VANETs often also suffer from significant problems with shadowing, particularly in urban environments, which introduces even more topology changes [1].

Data is used in-network. The aim here is not to collect the data and send it elsewhere but rather to directly process it and use it within the network itself. This scenario is common in VANETs where sensed data is used for vehicle control, route planning and other tasks without ever leaving the network itself. In our studied case, we wish to use information about accident risk in the current situation to inform vehicles' decision making.

The application is distributed. Processing of data must be done in an entirely distributed manner; there is no central controller to send the data to or receive instructions from. Each node must be able to make decisions on its own in a timely manner using available information. Again, this is a typical scenario in a VANET, where each vehicle is responsible for its own operation, and is particularly true for time-critical safety applications where there is not enough time to consult an external controller — decisions must be made locally and swiftly.

Inputs to the application can change rapidly and non-continuously. Accident risk depends on factors involving the driver, the vehicle itself and the surrounding environment (including other vehicles) and has both long and short-term components [2], [3], [4]. As such, risk levels are constantly changing and sudden events can occur that dramatically change risk levels in a short period. This problem is further compounded by the highly dynamic topology of VANETs: a new node may appear in the network, or an existing node may leave, at any time, resulting in changed risk levels. This means that we cannot make assumptions about how the inputs to the application — and thus the desired outputs — will change over time.

Data must be accurate and up-to-date. We consider applications where it is important that nodes make decisions based on data that is current and accurate. In some cases, inaccurate data can be compensated for — if the magnitude of the error is known — by behaving more conservatively. However, this will often result in reduced utility, sometimes drastically so, for instance in a case where a vehicle has a false positive for a dangerous situation and comes to a stop unnecessarily. In our risk-estimation application, inaccurate or stale data can result in either a reduction in safety, road network utility, or both.

We thus want to ensure that the outputs actually produced by the application are sufficiently close to the correct values at all times, in spite of the challenges presented above, or that they become so within a short enough time of the inputs changing. We will examine further what is meant by the "correct values" in the following sections; briefly, however, this can be considered as the limit of convergence of the algorithm for processing the data — that is, the steady state the algorithm would eventually reach, if it had enough time to do so without the inputs or network topology changing. We also wish to know how far from the correct values we are or might be, that is, the absolute error at any given time, so as to tailor nodes' responses not only to the actual values produced by the application, but also to the level of confidence we can have in those values.

Given an application that fits the above characteristics, we first need to know its convergence rate and how this changes with network size and node density. Once we have an understanding of the convergence rate, we are then able to investigate the error in the algorithm's outputs in a network with mobile nodes and dynamic topology. This then allows us to vary the beaconing rate in response to changing network conditions so that the error remains acceptable for the particular application, even with rapid and discrete changes in inputs. In the following sections, we go through this process for our risk estimation algorithm.

III. Coupled Risk Estimation Algorithm

We will explore these issues using the coupled risk estimation algorithm from [5]. A full description of this algorithm can be found in [5], and in the following we will cover a few key points relevant to this work. The aim of this algorithm is for vehicles to calculate their current accident risk level based on their own sensed information as well as information received from neighbouring nodes in the network. Nodes do not reach a consensus value but rather each vehicle determines its own risk value based on the particular constellation of risk factors that pertain to it. These include factors relating to each vehicle and driver, but there are also common risk factors relating to which other vehicles are present and to the surrounding environment that affect all vehicles in an area. Accordingly, each vehicle will determine a weighting for each of the risk values it receives from its neighbours based on how reliable and relevant that value is. This weighting is affected by a number of factors including the distance and direction between the nodes, the number of neighbours each has in the network, and the time since the last update from that neighbour was received.

Each vehicle then takes a weighted average over the values received from its neighbours as well as its own internal risk value based on its own sensors and information relating only to itself, i.e. each vehicle calculates

$$r = \frac{w_0 r_0 + \sum_{v \in N} w_v r_v}{w_0 + \sum_{v \in N} w_v} \qquad (1)$$

where r is the vehicle's risk estimate, r_0 is the internal risk value, w_0 is the weight given to the internal risk value, N is the set of (single-hop) neighbouring vehicles — those vehicles from which a message has been received, r_v is the risk value received from vehicle v and w_v is the weight given to vehicle v.

In order to calculate this average, nodes take an initial estimate (their internal risk value) and then update their estimate each time they receive a new estimate from a neighbouring vehicle. The entire system of nodes is thus collectively solving a set of linear equations, in which each row corresponds to one node, in a distributed fashion by performing an asynchronous Jacobi with local Gauss-Seidel method, as in [6].

IV. Aims

We first want to determine the convergence rate, in terms of both time and update iteration count, for the coupled risk estimation algorithm, and investigate how this is affected by different numbers of nodes (i.e. the size of the network) and node densities. We will do this using a static scenario in which nodes do not move and inputs to the algorithm do not change, as this allows the algorithm to actually converge.

We will then look at the case where nodes are mobile. In this case, the solution to the set of linear equations will be constantly changing, since node weightings depend on the distance and direction between nodes. Additionally, discrete events such as topology changes in the network will also affect the solution — nodes which are not neighbours will receive a weighting of zero, so when a new neighbour is identified or an existing neighbour leaves, the weight matrix will change accordingly.

This means that the algorithm can never actually converge as nodes are constantly moving (and thus their weightings are constantly changing). However, it is still important to know what the error is in this scenario, i.e. how far are the values produced by the algorithm from the actual solution to the set of linear equations at any given point in time? Our second set of experiments determine this and also examine how error is affected by the chosen beaconing rate, that is, the rate at which updates are transmitted.

Finally, we will look at the trade-off between error minimisation and consumption of network resources and a strategy for achieving a balance between the two in our scenario. In Section IX, we will look at this question in more detail and discuss how it might be framed as an optimisation problem in the general case.

V. Convergence rate

The first set of experiments we conducted investigated the relationship between the convergence rate of the coupled risk estimation algorithm and the number and spacing of nodes in the network. We used two measures of convergence rate: time to convergence and iteration count. Iteration count was the number of updates a node received from its neighbours before it converged, and convergence time was simply the time in seconds before this occurred.

A. Simulation Parameters

For our simulations we used ns-3 [7], a discrete-event network simulator, with the `YansWifi` [8] physical layer model, and the `NqosWifi` MAC layer model. While a VANET introduces additional shadowing effects over traditional wireless networks, these models give similar results for the inter-node distances used in our work [9]. Nodes were laid out in a grid of four columns, representing lanes of traffic, with a 3 m spacing between lanes. The longitudinal spacing between nodes was varied as an independent variable. The distribution of initial risk values given to nodes was log-normal (mean = 1.0, stddev = 0.5) as this distribution fits the definition of risk in [5]. For these experiments, beacons containing updated risk estimates were sent out by each node at a rate of 100 Hz. Nodes were considered to have converged when their risk value did not change by more than 0.05 for at least 10 iterations, and nodes were required to wait a minimum number of iterations (10) before starting to determine whether they had converged, in order to allow other nodes time to transmit their values, as nodes did not all start transmission at the same time in order to avoid interference.

B. Results and Discussion

We tested average convergence times and iteration counts to convergence for longitudinal node spacings of 1 m, 10 m, 20 m and 50 m. For each node spacing value, the number of nodes was varied from 2 to 400. Figures 1 and 2 show the convergence times and iteration counts to convergence at a node spacing of 20m, with 95% confidence intervals. Figures for the other node spacings are omitted for brevity as they produced very similar results.

From these figures, we see that convergence occurs rapidly. The iteration count required for the algorithm to converge is consistent across differing numbers of nodes, however, this means that for small networks, convergence time is slower as nodes must wait longer before receiving the number of updates required to converge. Nonetheless, the convergence time is an order of magnitude faster than the time required for a driver to react to a hazardous situation in

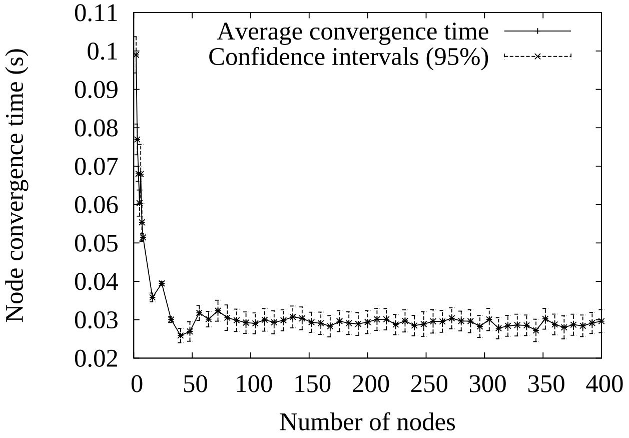

Fig. 1. Convergence time vs. number of nodes at 20 m spacing

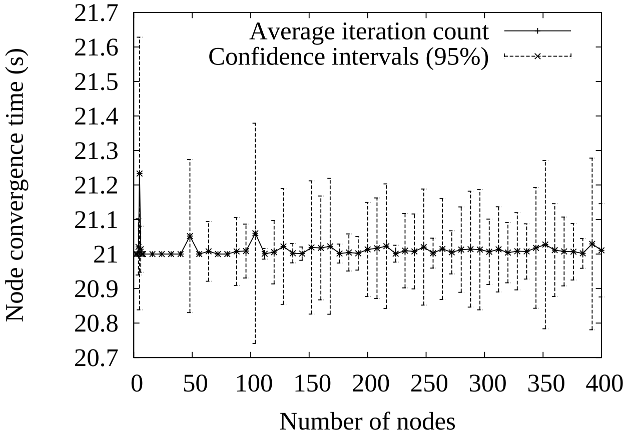

Fig. 2. Iteration count before convergence vs. number of nodes at 20 m spacing

all cases. Driver perception-brake reaction times — the total time for a driver to perceive a hazard and begin to apply the brakes — vary from about 0.7s to 1.5s depending on various factors including driver expectation, characteristics of the driver, cognitive load and the urgency of the situation [10].

Node spacing does not appear to have a significant effect on the iteration count required for convergence but convergence time is slower for larger node spacings as each node has fewer neighbours, resulting in fewer updates received per unit time and thus longer times to receive a requisite number of updates for convergence. However, again, convergence times remain at acceptable levels for all node spacing conditions tested.

VI. Error and beaconing rate

Next, we looked at how the error in the system was affected by the beaconing rate. Here, we consider error to be the distance (in N-space, where N is the number of nodes) of the vector of current risk values as produced by the coupled risk estimation algorithm from the actual solution to the system of linear equations it is solving, i.e. the limit to which the algorithm converges.

In order to examine how error changes with beaconing rate, we chose a scenario in which error can be expected to be high but in which the source of error is clear and consistent. We take a group of nodes which has already converged — in fact, we give all these nodes the same risk value — and which are all travelling in the same direction at the same speed, so that their weightings for each other will remain consistent over time. We then introduce a node with a very different risk value in front of and at a close distance to the group of nodes. This ensures that the new node will be given a high weighting by the others. Additionally, we have the nodes moving at high relative speeds: the new node is travelling in the opposite direction to the initial group of nodes. Thus we have a situation in which the error will increase suddenly when the new node is introduced and we can then observe the change in error over time as the algorithm attempts to correct for it, whilst the nodes are moving.

This scenario is somewhat contrived in order for us to be able to investigate how error is affected by the beaconing rate, however, it is conceivable that such a situation might occur in a real traffic scenario. For instance, shadowing caused by large buildings or heavy vehicles may prevent a node from having line of sight to others until it is quite close, and if this node has a very different risk estimate, this will cause a sudden spike in error for the oncoming nodes. It would always be possible to create a new, worse scenario in which errors are higher, however here we are not so much concerned with the size of the error but rather how it changes over time, that is, how quickly the algorithm recognises the discrepancy in values and corrects for it.

A. Simulation Parameters

Our scenario consists of a group of 10 nodes travelling in the one direction in a single lane, all with risk values of 1.0. A new node is then introduced travelling in the opposite direction and 20 m ahead of the lead vehicle in the group, with a much higher risk value (4.0). Vehicle speeds are 70 km/hr for all vehicles and the beaconing rate was varied from 100 Hz to 1 Hz. The simulation time was 10 s in total.

To determine the error, weights were determined and the solution to the system of linear equations was found each millisecond of simulation time. This was done using the Eigen library [11], with a Householder rank-revealing QR decomposition with column pivoting (`ColPivHouseholderQR`). The vector of risk values produced was then compared with the risk values the nodes had actually calculated from the coupled risk estimation algorithm. We then took the pointwise absolute difference to obtain an error vector, and used the norm of this vector as our measure of error. Note that here we only used the error from the group of ten nodes, not from the oncoming node, as this node's purpose in the simulation was to act as a source of error. In reality, sudden changes in risk values may occur not only from new nodes appearing, but also from environmental changes, thus we exclude this node in order to make the results more general to either situation.

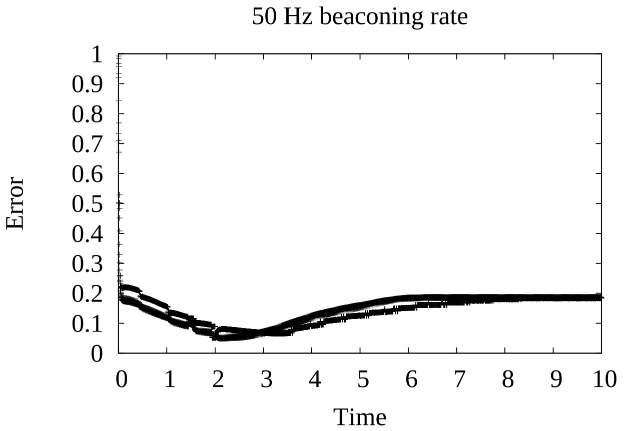

Fig. 3. Error over time for a beaconing rate of 50 Hz: beaconing rates of 100 Hz, 20 Hz and 10 Hz produced similar results

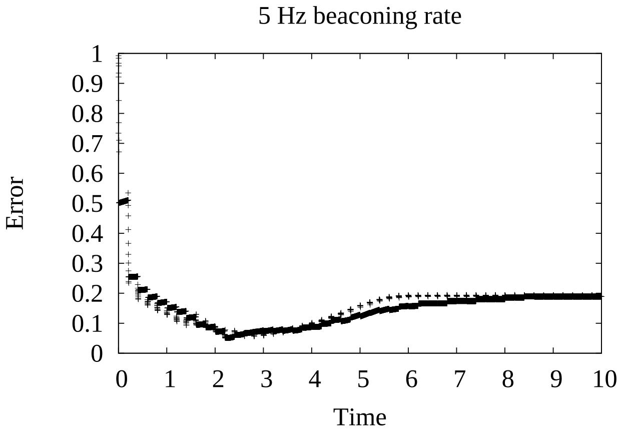

Fig. 4. Error over time for a beaconing rate of 5 Hz

B. Results and Discussion

Figures 3–6 show the error vector norm over time for beaconing rates of 50 Hz and 5–1 Hz. The results for faster beaconing rates (100 Hz to 20 Hz) were similar to the 10 Hz case and are thus omitted in the figures. Figures 7–10 show an expanded view of just the first two seconds of the simulation for each rate tested (50 Hz and 5–1 Hz). We can see that the error is initially high, as expected, and then drops as the nodes adjust their risk estimates. The error rises somewhat again later as the high-risk node starts to move behind the group of nodes, causing its weightings from the other nodes to decrease as its distance to them increases, and changes in the network topology as it gets out of range.

Higher beaconing rates meant that the high initial error caused by the sudden appearance of the new node dropped off more quickly. Thus, we see that a higher beaconing rate allows us to adjust sooner to a sudden change in risk values, keeping us closer to an accurate representation of risk for more of the time. We therefore conclude that a variable beaconing technique should be considered for this class of applications.

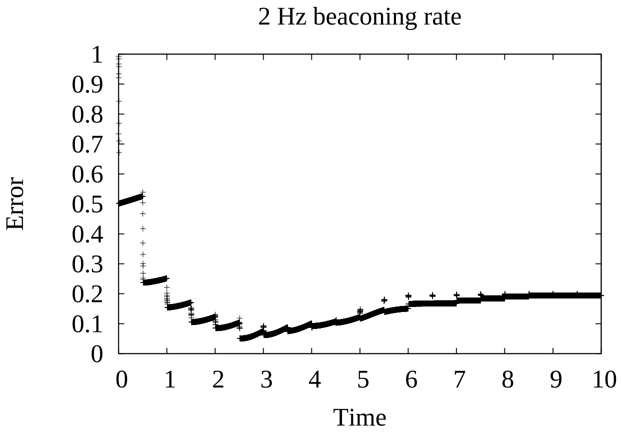

Fig. 5. Error over time for a beaconing rate of 2 Hz

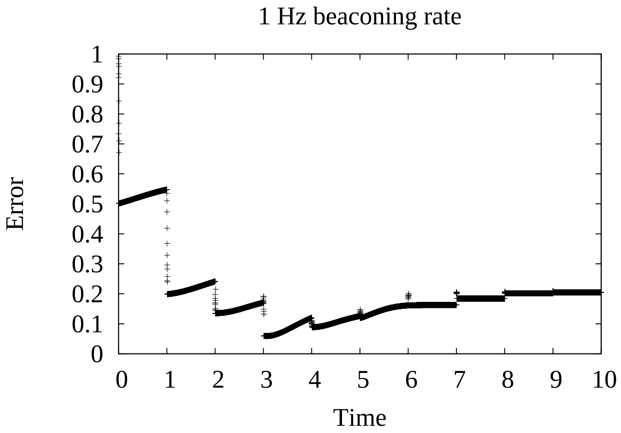

Fig. 6. Error over time for a beaconing rate of 1 Hz

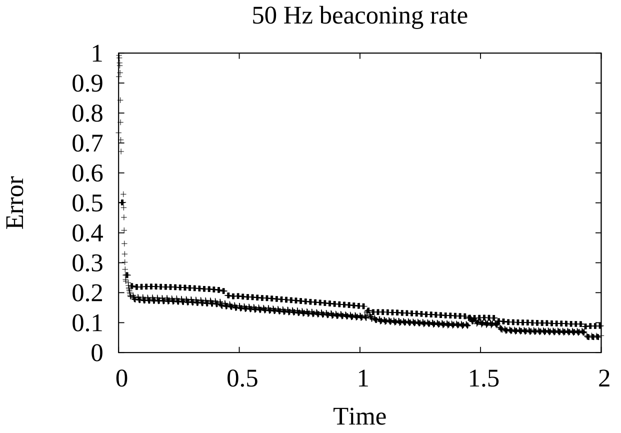

Fig. 7. Error over time for a beaconing rate of 50 Hz: first 2s of simulation: beaconing rates of 100 Hz, 20 Hz and 10 Hz produced similar results

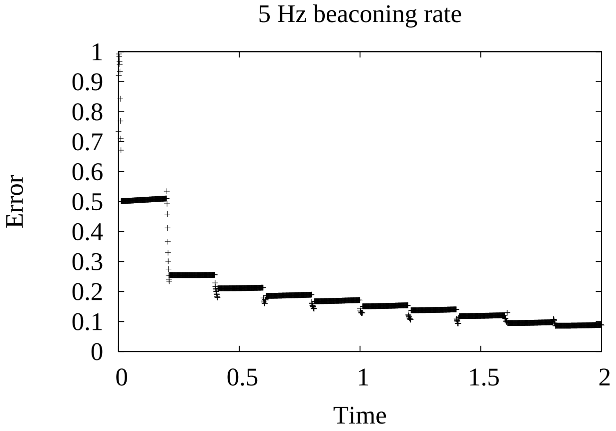

Fig. 8. Error over time for a beaconing rate of 5 Hz: first 2s of simulation

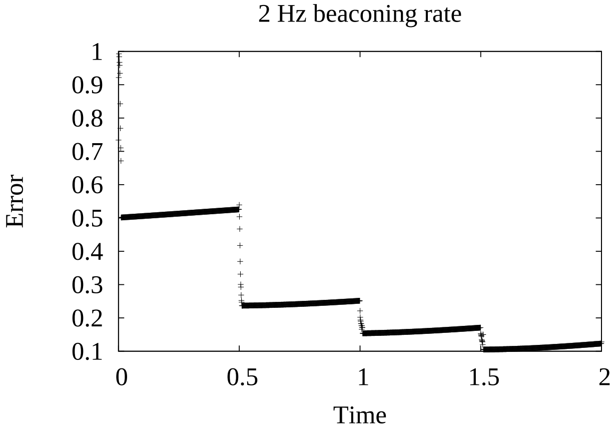

Fig. 9. Error over time for a beaconing rate of 2 Hz: first 2s of simulation

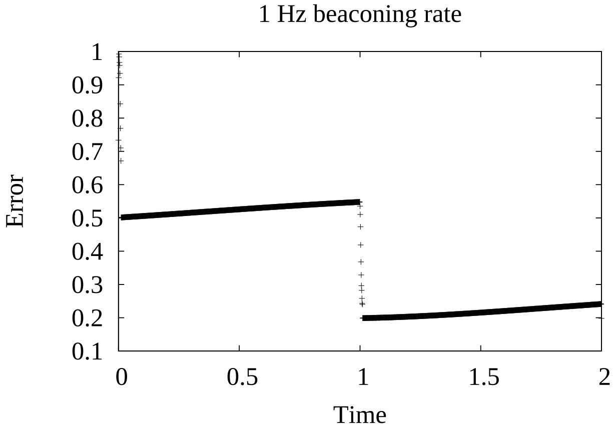

Fig. 10. Error over time for a beaconing rate of 1 Hz: first 2s of simulation

TABLE I
PACKETS SENT FOR DIFFERENT BEACONING RATES

Beaconing rate (Hz)	Packets sent
100	19990
50	9990
20	3990
10	1990
5	990
2	390
1	190
Variable 50/5	1046

VII. VARIABLE BEACONING RATE

While reducing error is an important goal, we cannot simply use a very high beaconing rate all the time. In situations where there is high node density, this will cause contention in the network, potentially leading to interference and packet loss. Even in cases where contention is not an issue, there may be other applications needing to use the network and we should thus try to minimise the bandwidth taken up by sending out our updates. Additionally, in some networks energy usage may be a consideration (though this does not typically apply to VANETs).

We thus have a trade-off between keeping error low — and adapting quickly to changes in input values — and usage of network resources such as bandwidth. To deal with this, we investigated using a variable beaconing rate scheme. We used a simple threshold scheme in which the beaconing rate could be one of two values: one fast and one slow. When a beacon was received by a node that caused a change in its risk estimate greater than the threshold, the faster beaconing rate would be used. However, once a received beacon only caused a small change in the risk estimate — below the threshold — the node would drop back to the slower beaconing rate.

A. Simulation parameters

For these experiments, we used the same scenario as in Section VI. The two beaconing rates used were 5 Hz for the slow rate and 50 Hz for the fast rate. 50 Hz was chosen for the fast rate as our results in Section VI-B indicate that error does not improve significantly at faster rates than this. Driver perception-brake reaction times — the total time for a driver to perceive a hazard and begin to apply the brakes — vary from about 0.7s to 1.5s [10]. This means that a slow rate of 5 Hz will still provide time to respond to a large change in risk in time for this information to be relevant to the driver in control of the vehicle. While these two beaconing rates provide good results, they may be optimised further, as discussed in Section IX. The threshold for change in risk value to switch rates was the same as the convergence threshold in our previous experiments: 0.05.

B. Results and Discussion

Figure 11 shows the error for the variable beaconing rate, and Figure 12 shows a larger view of the first two seconds of the simulation. Table I gives the total number of packets sent during the simulation for all beaconing rates.

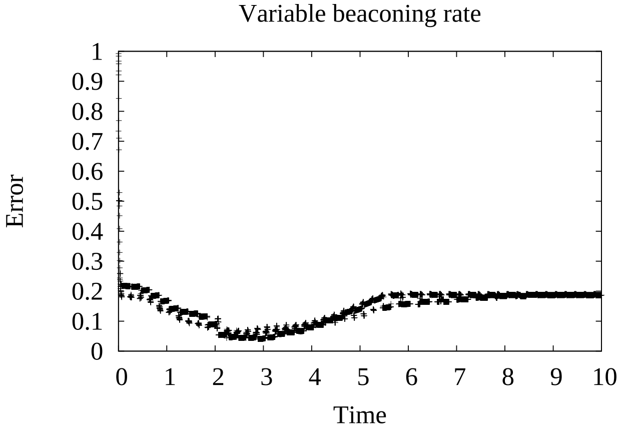

Fig. 11. Error over time using variable beaconing rate

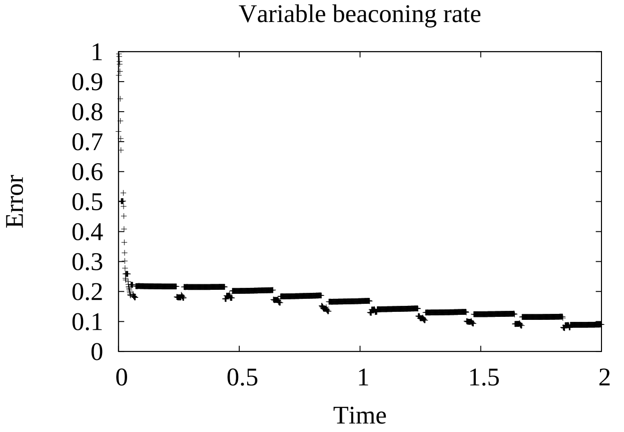

Fig. 12. Error over time using variable beaconing rate: first 2 s of simulation

If we compare Figures 11 and 12 with the corresponding figures for beaconing rates of 5 Hz and 50 Hz from the previous section (Figures 3, 4, 7 and 8), we can see that the variable rate gives error levels in between the two other rates. At the beginning of the simulation, when the error is highest and risk values are changing the most rapidly, the variable rate reduces the error to a low level in a similar amount of time as the fast rate of 50 Hz. However, when error is low and the risk values are changing slowly, the variable rate behaves more like the slow rate of 5 Hz.

In terms of packets sent, the variable rate gives a value only slightly higher than the slow rate, thus consuming much less bandwidth than the fast rate. Hence even a simple variable scheme such as used here immediately yields significant benefits, with error levels and response to changing inputs similar to a fast beaconing rate but bandwidth usage similar to a slow rate.

VIII. Discussion

With the variable rate beaconing scheme, we have a reasonable trade-off between error and network resource usage for the coupled risk estimation algorithm. The values used here are particular to this algorithm — the error level that is acceptable is relative to the accuracy and time scale for decision making required for this application. However, this process can be applied to other VANET applications, taking into account the needed characteristics of their outputs. In order to do so, the distributed algorithm's convergence rate must be determined, as we have done in Section V, and the effect on this of the wide range of traffic conditions likely to occur in a VANET needs to be ascertained. In our case, node density and network size did not have a large effect relative to the convergence rate needed for determining accident risk levels.

In a VANET, it is not only the changing traffic conditions that can affect the performance and accuracy of such an application, however. The dynamic topology, shadowing problems and high node mobility mean that an application must also respond to both discrete and continuous changes in input values. The speed with which an application can respond to these depends not only on its convergence rate, but also on the beaconing rate used as this will determine how often nodes receive updates. Thus the next step in analysing an application's performance is to investigate how the beaconing rate affects error under these conditions, as in Section VI.

Lastly, the data gathered during this process can be used to inform a strategy for managing both network resource usage and error. We have employed a simple threshold scheme and in some applications this may be sufficient. However, in other cases. where applications have more stringent requirements, a more complex strategy may be called for. Additionally, even with a simple scheme, it is likely not enough to look at the performance under one scenario as it is very difficult to envisage any "worst-case" scenario — one can always create a more problematic set of traffic conditions that would affect the algorithm performance more severely. Because of this, in the following section we will formulate the problem of finding the right variable-rate beaconing scheme as a more general optimisation problem that can then be applied to any given application and situation.

IX. Future work

We now have an optimisation problem in determining the best beaconing rate(s) to use and how to change the beaconing rate to adapt to different circumstances. Essentially, we have a point in N-dimensional space which represents the desired values for the nodes and which moves constantly as the nodes move, the network topology changes, and the input values themselves change. Moreover, this target point does not move in a predictable or even continuous way and the rate of change of its position is highly variable, and in fact, because of discontinuities caused by sudden events or network topology changes, unbounded. The times when the target point moves the fastest, or jumps discontinuously, are also when it is most important that the actual output remains close to it, as these typically represent critical situations such as a new vehicle arriving or a hazard appearing suddenly.

We then want the vector of the actual node values to be as close as possible to the target point at all times, while also not wasting network resources unnecessarily. We can consider this problem in two ways: we can either optimise for minimal error given a constraint of available or desired network resource usage, or we can optimise for minimal resource usage given a constraint of allowable error. The best approach to take here will depend on the particular application. For a safety-critical application, it is likely to be necessary to put a hard limit on the error and then try to reduce resource usage as much as possible. However, for other applications, such as navigation, entertainment, traffic information services, etc. it may make more sense to limit resource usage as occasional breakdowns in the functioning of the application due to high errors are likely to have less severe consequences than using too much bandwidth and thus causing contention or preventing other applications from functioning.

In both these cases, we also have a question as to how to define the minimum error or resource usage that we are optimising for. In the previous sections, we have taken the norm of the error vector as our measure of error, however, in some cases it may be better to instead minimise the maximum error across the nodes to ensure all are treated fairly and have reasonably accurate values. Similar arguments apply to resource usage, but here we are instead considering fairness in resource allocation to prevent high bandwidth usage in some parts of the network, even if the overall usage is minimised. The parameters used may be either the two beaconing rates as used above, i.e. a fast rate and a slow rate, along with the threshold for switching rates, or else a more complex scheme with more levels of beaconing rates could be used.

Given a method to solve this optimisation problem for a given set of constraints and utility function, it would then be possible to apply it to different traffic scenarios and applications as needed, or, for example, to use vehicle traces or traffic simulations to determine the best balance for a particular region, network architecture, etc.

X. Conclusion

We have examined the convergence rate for the coupled risk estimation algorithm and measured how it is affected by node density and spacing. The update iteration count to convergence was stable under different conditions, resulting in longer convergence times for small and widely-spaced networks, given a constant beaconing rate. However, in all conditions tested, the convergence rate was sufficiently fast, relative to the reaction time of a human driver, for an accident risk estimation algorithm in a VANET.

We also investigated how the beaconing rate affects the error in the algorithm's output in a scenario with mobile nodes and a large change in input in the form of a new node with a high risk value. Our results show that higher beaconing rates reduce error, particularly when the risk values change rapidly. However, higher beaconing rates also consume more network resources. To counteract this problem, we developed a variable beaconing rate scheme in which nodes test their change in risk value against a threshold to determine which of two different beaconing rates to use. We found that using this scheme we achieved good performance both in terms of error and resource usage. In future, this work can be extended to a more general optimisation problem to determine how best to adapt the beaconing rate given a particular rate of change of the risk value.

This methodology can be applied to any distributed algorithm matching the same network and data model characteristics. These are a highly mobile network with dynamic topology, and a distributed application in which data is used in-network, accuracy of results is critical and must be assured within a certain amount of time, and the solution to the data processing task changes unpredictably and at a highly variable rate. Vehicular networks fit these characteristics and these kinds of applications are appearing more and more for them. Many applications in VANETs are safety-critical and thus require rapid and accurate responses to changing data, and data is often processed and used directly by vehicles in a distributed fashion without leaving the network. The process we have described can thus be followed to inform the parameters used for such applications in order to balance error levels with network resource usage.

References

[1] J. Otto, F. Bustamante, and R. Berry, "Down the block and around the corner the impact of radio propagation on inter-vehicle wireless communication," in *Distributed Computing Systems, 2009. ICDCS'09. 29th IEEE International Conference on*. IEEE, 2009, pp. 605–614.

[2] M. Peden *et al.*, "World report on road traffic injury prevention," 2004.

[3] J. Treat and Indiana University, *Tri-level study of the causes of traffic accidents: Final report*. Dept. of Transportation, National Highway Traffic Safety Administration; Available through the National Technical Information Service, 1979.

[4] L. Vogel and C. Bester, "A relationship between accident types and causes," in *24th Annual Southern African Transport Conference, South Africa*. SATC, 2005, pp. 11–13.

[5] E. Fitzgerald and B. Landfeldt, "A system for coupled road traffic utility maximisation and risk management using vanet," in *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*, 2012, pp. 1880–1887.

[6] J. Bull and T. Freeman, "Numerical performance of an asynchronous Jacobi iteration," *Parallel Processing: CONPAR 92VAPP V*, pp. 361–366, 1992.

[7] "ns-3," http://www.nsnam.org/.

[8] M. Lacage and T. Henderson, "Yet another network simulator," in *Proceeding from the 2006 workshop on ns-2: the IP network simulator*. ACM, 2006, p. 12.

[9] T. Abbas, F. Tufvesson, and J. Karedal, "Measurement based shadow fading model for vehicle-to-vehicle network simulations," *arXiv preprint arXiv:1203.3370*, 2012.

[10] M. Green, ""How long does it take to stop?" Methodological analysis of driver perception-brake times," *Transportation Human Factors*, vol. 2, no. 3, pp. 195–216, 2000.

[11] G. Guennebaud, B. Jacob, *et al.*, "Eigen v3," http://eigen.tuxfamily.org, 2010.