Technical Report
Department of Computer Science
University of Alabama
July 28, 2018

Clustering Technique for VC-VANET

Michael Lee

***Abstract*— As small-scale devices get more and more common, more means to connect many devices into new types of networks that will allow the exchange of localized information are being developed. One such rising field of networks is VANET, a network that connects devices in or associated with vehicles in an ad-hoc fashion. One of the main goals of this sort of network is to give geographically bounded information that is of particular relevance to cars passing through the area. As such, we are working of developing a geographically bound VANET that can run cloudlet applications within a specific cluster of cars. We call this form of VANET Vehicle-Crowd based VANET (VC-VANET). One of the areas that we are focusing on enhancing with VC-VANET is behavior of the network at intersections, since these represent areas where cars will be stopped for a period of time, which we can use to create a stable network. As such, one of our areas of focus is determining how large we can expect the network to expand at each intersection at any given time. Our approach to solving this problem is through Rad-Ap, a C++ class we created to approximate the threshold distance, or radius, of the network that VC-VANET will maintain at the intersection using predictive methods based on the density of traffic at the intersection.**

I. INTRODUCTION

VANET applications attempt to connect the devices in multiple vehicles so that they can exchange location relevant to the current location or the process of driving. They may do this in order to advertise local services such as gas stops or rest areas that the driver may be interested in, or to provide warnings about traffic issues in nearby roadways. Some future versions may even pass live data on how the car is being steered so that the cars behind it can mimic the inputs, reducing the required work for the followers to drive. Regardless of their purpose, making VANET applications scalable is vital to their success, as a large number of devices will be connected in a VANET system.

One of the major ways to help increase the scalability of VANET systems is to group cars together into what is commonly called clusters or vehicle-crowds (v-crowds) [1]. These v-crowds represent a group of cars that can communicate with each other freely and will remain able to do so for some period of time, similar to a Local Area Network. They are added into the network in a hierarchical manner on top of the foundations of a traditional vehicular ad-hoc network. These v-crowds are designed to be used in our VC-VANET to introduce additional capabilities for local computation and collaborative storage systems across the vehicles inside of the v-crowd.

Applications generally try to find clusters of cars that are travelling together down the same section of the road. However, these v-crowds are difficult to identify due to the high mobility of vehicles and the lack of cohesion within groups when driving [2]. Fortunately, unlike normal mobile ad-hoc networks, systems within a VANET are generally required to follow the rules of the road, which gives us some additional knowledge that we can use to help identify clusters that we can use in our VANET applications [3].

One of the major features of VC-VANET is that the v-crowds it identifies are intended to be geographically bounded and can be used to run cloudlet applications within that area. This feature exists so that the cloudlet applications can gather and distribute information relevant only to that particular area, including congestion information, nearby points of interest, and many other types of local data. As such, our of the major areas of focus for this project was investigating how we could use intersections to identify v-crowds that we could use for our system. We chose to focus on intersections for this since they can slow down traffic, force them to form groups, and are always located at a static position. We then chose to investigate further into intersections with traffic lights, due to a couple of distinct features of these intersections. The first feature is that traffic lights have a more predictable pattern of vehicle dispersion when compared to intersections with stop signs or no signage at all. Many of them are locked to set cycles, and even the ones that arent are generally ruled by a very simple algorithm, making them very predictable. Traditional intersections without traffic lights are largely ruled by human decision making, which is far less straightforward to predict. The second feature is how traffic lights tend to be employed at intersection that have a high traffic density, increasing the ease of creating clusters as well as the stability of the clusters at that intersection. The third feature is that traffic lights in our region have vehicle traffic data collected on them and stored by the Alabama Department of Transportation. By collaborating with them, we were able to access and use data on local intersections to help guide the creation of our project.

The product we developed for use with VC-VANET we termed Rad-Ap, which is short for Radius Approximator. This product uses data on vehicles arriving at the intersection to predict the current density of vehicles at the intersection, then uses this to approximate the size of the v-crowd currently at the intersection. It can be run within a VC-VANET system as one of the applications it maintains to help the system attempt to form a v-crowd of an appropriate size without having to collect information on the vehicles currently at the intersection. It was developed with intersections that have traffic lights in mind, but may be able to be extended to other locations as well.

*** The work is supported partly by the National Science Foundation under Grant No.1719062. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.**

II. BACKGROUND AND RELATED WORKS

The architecture of VC-VANET is run on top of existing VANET system and adds in the new component, namely the v-crowd, when a group of vehicles in the same area forms. Once these v-crowd forms, each vehicle continues to have all the capabilities it had when in was just connected by the VANET system, but also gains the ability to collaborate in local computing and storage projects managed by VC-VANET. While these v-crowds may form sporadically at any location, we chose to focus on v-crowds anchored at a particular location, as they may last for a longer period of time. This may especially be the case at intersections along well-trafficked roads, so they were of particular interest to us. This upper tier of v-crowds is a logical layer built on top of the lower tier of vehicle to vehicle communications.

Vehicles within a v-crowds will collaboratively hold pieces of information and perform data processing in a distributed systems architecture. These data chunks and tasks can be passed within the v-crowd or to approaching vehicles. A v-crowd is similar to the concept of a cloudlet [ref], but VC-VANET is focused on the network aspects for forming storage and performing computations. In order to ease the communication cost across vehicles, a cluster head may be employed to coordinate systems and pass messages. When the cluster head leaves, the system can choose a new head using methods such as reelection or leader handover.

A. Rad-Ap Functionality

Rad-Ap works by recording data on when vehicles arrive at the intersection and using this data to predict the current density of vehicles at the intersection using a methodology developed for this particular purpose [4]. We then associate that density with a threshold distance, and use feedback from the system that tells us whether the network was able to be established across that distance to adjust the threshold distance to more accurately represent a maintainable distance. Each declaration of the class will represent one intersection. Optimally, the class would be pre-trained on recorded data, but since it is the first existing software we found that collects data on this subject, it is also capable of being deployed untrained. As it runs, it collects data about the intersection and how the density relates to the threshold distance, and can print this to a file that can be opened by the class to pre-train a new declaration of the class on the data that the old declaration collected.

In addition to this, the class allows the user to adjust many of the core variables of the class, so we can use the data stored by previous runs of the class to determine constants that are more appropriate to the location the class is being deployed at. The variables that can be changed include the default value that each new density starts on, how large the clusters of densities that are considered the same are, the recovery value for when a threshold distance would hit zero, and how quickly the threshold distances change after each reported success or failure. Additionally, RadAp has a system for passing and storing the current densities of local intersections. It currently does not do anything with this information, but other applications could be interested in using this data for various tasks. The size of the range for intersections to be considered local is also a variable that can be adjusted on declaration of the class.

B. Previous Work

VANETs have been adapted to provide a wide range of different services. Many of these are focused on using VANETs to improve traffic safety. These projects focus on a number of different issues, including detecting traffic accidents that have already occurred and avoiding them, avoiding collisions in intersections, and avoiding high congestion areas [5] [6] [7] [8] [9] [10]. Other applications are interested in providing internet connectivity to the users devices in the system, affording a great deal of additional utility to VANETs [11] [12] [13] [14]. Previous studies have also researched using VANETs to transport data across long distances [15].

Many previous works have laid the groundwork for processes that we will need to run VC-VANET as well as our Rad-Ap system. Among these are projects that developed broadcast techniques for VANET [16] [17] [18], routing techniques for VANET [3] [19] [20], VDTN store-and-forward techniques for dealing with intermittent connections [21] [22] [23] [24], and connectivity analysis [25] [26] [27] [28].

Many papers have focused on creating clustering techniques for VANET [29] [30] [31] [32]. Many of the clustering techniques proposed by these papers are developed for application-specific clustering techniques, so using the techniques they put forth would be sub-optimal for our application [50]. We chose to not pursue a general purpose clustering technique for our application due to the specific nature of our application.

Since the advent of VANET technologies, many different clustering algorithms have been developed for VANET implementations. These algorithms generally started off as algorithms created for MANET systems, but gradually moved to be increasingly tailored for VANET systems, as the nodes mode very differently in a VANET system from how they do in a MANET system. Many of these implementations can be classified into one of seven different types of algorithms: general purpose, routing, channel access management, security, vehicular network topology discovery, traffic safety, and combinations with cellular infrastructure [33]. General purpose algorithms are created without any application in mind and generally attempt to maintain a robust connection in the face of high vehicular mobility [34] [35] [36]. Routing algorithms attempt to build a hierarchical overlay on top of the network that can be used to pass packets across to other clusters [37] [38] [39] [40]. Channel access management algorithms have the cluster head control when a node is allowed to access the channel [41] [42]. Security algorithms feature a large amount of protection against malicious vehicles entering the cluster, allowing the cluster to function as a trusted space [43] [44]. Vehicular network topology discovery algorithms use informations on the clusters to create maps of the connectivity between vehicles, which is

important for both communication protocols as well as traffic management systems [45]. Traffic safety algorithms collect data from other cars within the cluster and uses it to help predict and prevent car collisions within the cluster [46]. Algorithms that combine with cellular infrastructure seek to provide a link between a VANET cluster and the internet that will reduce the load on the cellular network while still providing sufficient access to all the vehicles in the cluster [47] [48] [49] [50].

Our application would be best classified as a routing algorithm developed specifically for use in VC-VANET. However, it differs from other algorithms due to the focus on using data on intersections to create the clusters, which makes it have the distinct properties of being geographically bounded as well as being determined by using patterns from the vehicle density that are tuned specifically for intersections.

III. MEASUREMENT METHODS

In order to test how well our system worked, we created a program that would run our class on test data. For our test data, we used actual recent traffic data collected from 44 intersections that spanned over 84 weeks. We used this data to simulate cars entering the intersection over time. The test program then gets the threshold distance suggested by the program and fails it if it passes certain criteria. The purpose of this is to simulate when the threshold distance suggested becomes too large for the amount of cars at the intersection. In an effort to better simulate a real world environment, we made test cases that had random chances of failing regardless of size and that had some variance in the maximum size that would work for a particular density. As a result, the tests will represent a more realistic environment where not everything will be perfectly executed. We included a variety of different tests that could simulate different possible situations that the class could be operating under.

A. Natural Tests

Our first test was designed to represent the class being employed in a semi-realistic environment where we know little about the true values of the system. As such, this test featured an optimum threshold distance significantly higher that the default starting value, with the default starting value being set to 5, while the optimum value averaged around 87. Additionally, the threshold distance increases proportionally to the square root of the vehicle density at the time. Theoretically, we would expect this to also be the case in the real world, since the density represents the area contained within the circle that our cluster occupies, while the threshold distance represents the radius of our cluster. We also included a random jitter that varies the optimum threshold distance by around 5.4% in order to better represent the instability of a network connection.

Since our class is also capable of gathering data and allowing the user to analyse it or just use the the collected data to start in a pre-trained state, we also included a test that would examine how our class would perform after having been trained or pre-set correctly. In order to determine this, we created a second test that was largely similar, but had the default starting value set to be equal to the optimal threshold distance. We then created a test to determine how well our class would run in an extreme condition to investigate whether our class is sufficiently robust. As a result, our third test adjusted the default starting values to be extremely high relative to the optimal threshold distance, around 10,000 times the optimal threshold distance, with a value of 9,000,000.

B. Network Condition Tests

In addition to the previous set of tests, we created a battery of tests designed to determine how network stability affects the performance of the class. To accomplish this, we created six tests with various different characteristics. The first of these completely removed the random jitter that test 1 had in order to observe how well our program could perform on a completely stable network. This was a particular point of concern for our system, as it was designed to be able to rapidly adapt to most conditions, but as a result does not optimize as well as other implementations could on a very stable network. The next test investigated the exact opposite possibility: a network that is considerably less stable than our original tests dummy network. This test increased the random jitter to have triple the range of the original test, giving the network a random variance in range of around 16.2%. Then, since our original test did not represent how a network can randomly drop packets regardless of distance of transmission, we created a test that replaced the random jitter with a 5% failure rate on all attempted transmissions. We then also created a test that had both the tripled random jitter as well as the 5% random failure rate to see how well our program could perform in an extremely unstable environment. We then created one more test that replaced the cutoff value for the transmission distance with a random number from 0 to 100. While this test is not particularly meaningful in an attempt to model a real world system, we chose to include it in order to determine how well our class could maintain a reasonable transmission success rate in an extremely high entropy environment.

C. Scaling Rate Tests

As a means of preventing our assumptions from interfering with our results, we created one more battery of tests that examined how our class would perform if our original assumption that the radius increases proportionally to the square root of the density was incorrect. This battery of tests included three new situations: one where the density did not affect the threshold distance at all, one where the threshold distance increased proportionally to the density, and one where the threshold distance increased proportionally to the density squared.

TABLE 1
TESTS CONDUCTED

Test Name	Class's Initial Value	Random Jitter	Failure Chance	Scaling Rate
Test 1	5	5.4%	0%	$\sqrt{n}$
Test 2	80	5.4%	0%	$\sqrt{n}$
Test 3	9,000,000	5.4%	0%	$\sqrt{n}$
Net Test 1	5	0%	0%	$\sqrt{n}$
Net Test 2	5	16.2%	0%	$\sqrt{n}$
Net Test 3	5	0%	5%	$\sqrt{n}$
Net Test 4	5	16.2%	5%	$\sqrt{n}$
Net Test 5	5	100%	0%	$\sqrt{n}$
Scale Test 1	5	5.4%	0%	0
Scale Test 2	5	5.4%	0%	n
Scale Test 3	5	5.4%	0%	n^2

IV. INVESTIGATION AND RESULTS

We ran our class through each test multiple times and recorded how close the average threshold distance it suggested was to the optimal threshold distance, as well as what percentage of suggested connections were actually successful. Our goal was to reach at least a 95% success rate in the suggested connections while optimizing the distance between average threshold distance and optimal threshold distance as much as possible, so that an application using our class would be likely to establish a successful connection each time while sacrificing as little from the number of cars that would be connected as possible.

A. Natural Test Results

TABLE 2
TEST RESULTS

Test Name	Average Threshold Distance	Optimum Threshold Distance	Success Rate
Test 1	82.5120 ft	87.0114 ft	95.0779%
Test 2	82.5438 ft	87.0114 ft	95.0750%
Test 3	271.6135 ft	87.0114 ft	95.0654%

The first test achieved an average suggested threshold distance of 82.5120 feet, with the true optimum distance for that test case being 87.0114 feet, putting us within 5.18% of the optimum value. Additionally, it achieved a success rate of 95.07779%, just over our goal value of 95%. Since this test most resembles how wed expect the real world would perform, this is indicative of our classs ability to maintain a high success rate when used untrained in an actual system while keeping the suggested threshold distance reasonably close to the true optimum value.

The second test performed slightly better on average, which is expected since it represents how the class will perform once we have gathered data on the intersection and used it to adjust future versions of the class. However, the difference was surprisingly small, with the average threshold distance being only 0.0326 higher than that reported in the first test case, and the success rate was actually 0.0029% lower. Both these differences can be largely attributed to how the first test case would have had a series of lower numbers early on as it was growing to find the true mean which would have had a higher success rate, but would pull down the average suggested threshold distance. As such, we can infer that both test cases had similar performance in the long term, but the second test case had more optimal performance on the early values since it was already pre-adjusted.

The third test case produced some unexpected results, as it also reported reaching a success rate of 95.0654% on attempted connections, but it had an average suggested radius of 271.6135 feet. By examining the collected data at the end of the test, we were able to determine that the suggested radiuses at the end were largely less than the average, and if the median suggested radius were than high then with the optimal suggested radius remaining at 87.0114, the class should have had a success rate of less than 50%. As such, we can infer that the median was not as high at the mean, which indicates that the suggested values were heavily skewed towards lower values. Thus, it would seem that our average was thrown off by the values starting at 9,000,000, and is not exceptionally meaningful in this instance. However, by the success rate remaining over 95%, we can determine that our class seems to have given a large amount of suggestions that were under the optimum threshold distance, and maintained our goal success rate even with an unreasonable initial value.

Variance across multiple runs of these test cases was incredibly low, with all runs having a difference of less than 0.0248% in average suggested threshold distance and a difference of less than 0.00126% in the success rate. As such, differences between the different tests is very likely to be indicative of an actual difference in performance between the tests rather than the result of statistical variance.

B. Network Test Results

TABLE 3
NETWORK CONDITION TEST RESULTS

Test Name	Average Threshold Distance	Optimum Thresh. Distance	Success Rate
Net Test 1	82.8151 ft	92.0114 ft	95.1341%
Net Test 2	75.1491 ft	77.0114 ft	94.8629%
Net Test 3	57.1815 ft	92.0114 ft	93.9753%
Net Test 4	51.7103 ft	77.0114 ft	94.0315%
Net Test 5	5.7181 ft	-	93.7780%

We next ran the battery of tests the focused on how network performance would impact the program, using the same goals as we did for the first three tests. However, the optimal threshold distance changes slightly based on the test, and for the tests that had a 5% random failure rate it would only be possible for our class to reach a 95% success rate if it had a 100% success rate on all the transmissions that did not randomly fail, so we did not expect those tests to reach

that goal, but still wanted them to get as close to the goal as possible.

The first of the network tests achieved an average suggested threshold distance of 82.8151 feet, but since this test removed the random jitter, the true optimum value was further away than it was on previous tests, at 92.0114 feet. The success rate was also higher than the previous tests, at 95.1341%. However, in a similar way this is less impressive, as there was no randomness involved to decrease the success rate. As such, this confirmed our concerns that the class is not well optimized to performing in a very stable environment, so it may be better to use alternative solutions in such circumstances. However, we did not investigate how well the class could perform if we passed in variables that were optimized for the network environment, so it may be possible for this to be improved with better tuned variables.

The second test in this battery had the least distance between the optimum threshold distance and the average suggested threshold distance, with the average value at 75.1491 feet while the optimum value decreased to 77.0114 feet. However, this did decrease the success rate to less than our goal of 95%, at 94.8629%. Between this test, the previous, and the first three, we can see how our program has a tradeoff between success rate and proximity to the optimum distance. We can affect where this exchange is centered by changing the values of the class, but these tests give us a reasonable idea of the sort of relation between these values we can expect to see when the class is used in a real world environment.

In the third test case, which replaced random jitter for a 5% failure chance, the success rate fell to only 94.0314%, meaning that of the transmissions that didnt randomly fail, 98.98% succeeded. However, in exchange for this, the average suggested threshold distance plummeted to 57.1815 feet, while the optimum returned to the value it had in the first network test case. This is a result of the class being set up to seek a 95% success rate in a system where a 95% success rate is almost impossible to achieve. This further demonstrates the exchange rate between average suggested distance and success rate within the class.

The fourth test case, which combined the increased jitter of the second and the failure rate of the third, saw a mix of results. As in the case of the second, the average suggested threshold distance grew closer to the optimum value than it was in the third, with the average decreasing only to 51.7103 feet, while the optimum decreased to 77.0114 feet. Also similarly to the difference between the second and first, the success rate fell only slightly to 93.9753%. As such, we can see that the program is quite resilient to increases in the amount of random jitter, especially when compared to how much it suffered from having a high simulated packet drop chance in the third test.

The fifth test case is somewhat more difficult to evaluate, as there isnt a set optimum value for the threshold distance to reach. However, it did reach a success rate of 93.7782% while maintaining an average suggested threshold distance of 5.72002 feet. As such, we can see that the class will maintain a high success rate even in situations with very unstable networks, but in exchange may perform suboptimally in terms of average threshold distance as compared to the optimum value.

C. Scaling Test Results

TABLE 4
SCALING RATE TEST RESULTS

Test Name	Average Threshold Distance	Optimum Thresh. Distance	Success Rate
Scale Test 1	80.6427 ft	85 ft	95.0753%
Scale Test 2	86.3787 ft	91.1849 ft	95.0816%
Scale Test 3	135.598 ft	614.142 ft	95.1014%

We ran the last battery of tests designed to determine how different scaling rates could affect the performance of the class using the same constraints as the previous tests. Once again, the optimum value for threshold distance varied from test to test, so we made sure to re-evaluate the optimum values for each test.

The first of these tests, which had no scaling, performed comparably to how the class ran with the square root of density scaling in the first three tests. It achieved an average threshold distance of within 5.13% of the optimum value, at 80.6427 feet and 85 feet, respectively, and a success rate at 95.0753%, which was 0.0048% higher than that achieved in the first test. As such, we can see that the class actually performs slightly better when the threshold distance does not increase as the density increases. We primarily attributed this to the decrease in distance between the optimal values for each density and the initial value.

The second test case had the average suggested distance within 5.28% of the optimum value, only slightly further from it than the tests with no scaling and scaling proportional to the square root of the density. Its success rate also rose slightly to 95.0816%. We once again attributed this difference to the increased period of time where the suggested distance was lower than the optimum value, as we did between the first and second tests. Based on theses values, we determined that the class could run comparably with scaling proportional to the density when compared to our assumption of square root of density scaling.

The last of these tests faltered significantly, with the average suggested threshold distance only getting within 77.93% of the optimum value, with the average at 135.598 and the optimum at 614.142. After reviewing the data, we attributed this issue primarily to extremely high values that rarely occurred but had a very high maximum threshold distance, as they could increase the average optimum value easily and the class would not collect enough data on them to correctly determine an accurate value for the maximum threshold distance. However, since the amount of time spent under the optimum value increased, the success rate rose slightly to 95.1014%.

D. Performance

As our class is designed to be employed in a mobile network environment, it must be able to run on a system while using only a small amount of processing power. As such, when running our tests we also recorded how long it took the tests to run to determine whether a smaller computer would be able to run our class efficiently using live data. Since we used test data that covered 44 intersections across 84 weeks, our test program had to run through 3696 times the amount of data that we would expect a particular intersection to encounter within a week. With this amount of data, each of our tests managed to finish running within 30 minutes. In addition, the algorithms for processing a line of data all run in constant time, making our program run in O(n) time when processing n data. As such, we can reasonably expect a far slower computer to be able to appropriately handle live data from a single intersection.

CONCLUSION

Our investigations found that under most circumstances our class can approximate the threshold distance with a reasonable degree of accuracy while maintaining a success rate higher than 95 percent. However, our investigations also showed that it will be beneficial to correctly adjust the class for the network connection in order to optimize its performance. However, even without optimizations, the class was shown to able to perform with only a small detriment to its performance. For future work we plan on running instances of this class at multiple intersections to collect real world data, then we plan on using that data to create a better informed program that can perform more optimally for the conditions of the intersection. Additionally, we plan on developing a cluster head selection and distribution algorithm to use with this class that would allow it to be used as a full clustering algorithm.

REFERENCES

[1] V. Goswami, S. K. Verma and V. Singh, "A novel hybrid GA-ACO based clustering algorithm for VANET," 2017 3rd International Conference on Advances in Computing,Communication & Automation (ICACCA) (Fall), Dehradun, 2017, pp. 1-6.

[2] O. S. Oubbati, A. Lakas, N. Lagraa and M. B. Yagoubi, "UVAR: An intersection UAV-assisted VANET routing protocol," 2016 IEEE Wireless Communications and Networking Conference, Doha, 2016, pp. 1-6.

[3] K. Liu and K. Niu, "A hybrid relay node selection strategy for VANET routing," 2017 IEEE/CIC International Conference on Communications in China (ICCC), Qingdao, 2017, pp. 1-6.

[4] Gould, Phillip G., Anne B. Koehler, J. Keith Ord, Ralph D. Snyder, Rob J. Hyndman, and Farshid Vahid-Araghi. "Forecasting time series with multiple seasonal patterns." European Journal of Operational Research 191, no. 1 (2008): 207-222.

[5] J. Gluck, H. S. Levinson, and V. Stover, Impacts of Access Management Techniques, NCHRP Report 420, Transportation Research Board, 1999; http://tools.ietf.org/wg/manet/draft-ietf- manet-zone-zrp/draft-ietf-manet-zone-zrp-04.txt

[6] Y. Toor, P. Muhlethaler, A. Laouiti and A. D. La Fortelle, "Vehicle Ad Hoc networks: applications and related technical issues," in IEEE Communications Surveys & Tutorials, vol. 10, no. 3, pp. 74-88, Third Quarter 2008.

[7] European project REACT, www.react-project.org

[8] U.S. Department of Transportation, http://safety.fhwa.dot.gov/facts/road factsheet.htm

[9] European project PReVENT-Intersafe, http://www.prevent-ip.org/en/prevent_subprojects/intersection_safety/intersafe

[10] B. Mourllion, Extension dun Systme de Perception Embar-qu Par Communication, Application la Diminution du Risque Routier, Ph.D. diss., Universit Paris 11 (Orsay), 2006

[11] R. Wakikawa et al., Design of Vehicle Network: Mobile Gate-way for MANET and NEMO Converged Communication,Proc.2nd ACM Intl. Wksp. Vehicular Ad Hoc Networks, ACM Press, 2005, pp. 8182

[12] T. Ernst, The Information Technology Era of the Vehicular Industry, SIGCOMM Comput. Commun. Rev., vol. 36, no. 2, 2006, pp. 4952.

[13] V. Devarapalli et al., Network Mobility (NEMO), RFC 3963, Jan. 2005; http://ietf.org/rfc/rfc3963.tx

[14] V. Namboodiri, M. Agarwal, and L. Gao, A Study on the Feasibility of Mobile Gateways for Vehicular Ad-Hoc Net-works, Proc. VANET04, 2004.

[15] J. Zhao and G. Cao, "VADD: Vehicle-Assisted Data Delivery in Vehicular Ad Hoc Networks," in IEEE Transactions on Vehicular Technology, vol. 57, no. 3, pp. 1910-1922, May 2008.

[16] S. S. Alwakeel, H. A. Altwaijry and A. B. Prasetijo, "A multiple classifiers broadcast protocol for VANET," 2017 4th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE), Semarang, 2017, pp. 35-40.

[17] E. Limouchi and I. Mahgoub, "BEFLAB: Bandwidth efficient fuzzy logic-assisted broadcast for VANET," 2016 IEEE Symposium Series on Computational Intelligence (SSCI), Athens, 2016, pp. 1-8.

[18] C. Wu, S. Ohzahata, Y. Ji and T. Kato, "Joint MAC and Network Layer Control for VANET Broadcast Communications Considering End-to-End Latency," 2014 IEEE 28th International Conference on Advanced Information Networking and Applications, Victoria, BC, 2014, pp. 689-696.

[19] F. Goudarzi, H. Asgari and H. S. Al-Raweshidy, "Traffic-Aware VANET Routing for City EnvironmentsA Protocol Based on Ant Colony Optimization," in IEEE Systems Journal.

[20] A. Abuashour and M. Kadoch, "An Intersection Dynamic VANET Routing Protocol for a Grid Scenario," 2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud), Prague, 2017, pp. 25-31.

[21] A. Hamza Cherif, K. Boussetta, G. Diaz and D. Fedoua, "Improving the performances of geographic VDTN routing protocols," 2017 16th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net), Budva, 2017, pp. 1-5.

[22] S. H. Ahmed, Hyunwoo Kang and Dongkyun Kim, "Vehicular Delay Tolerant Network (VDTN): Routing perspectives," 2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC), Las Vegas, NV, 2015, pp. 898-903.

[23] J. J. P. C. Rodrigues et al., "The Vehicular Delay-Tolerant Networks (VDTN) Euro-NF joint research project," 2011 7th EURO-NGI Conference on Next Generation Internet Networks, Kaiserslautern, 2011, pp. 1-2.

[24] J. N. Isento et al., "FTP@VDTN A file transfer application for Vehicular Delay-Tolerant Networks," 2011 IEEE EUROCON - International Conference on Computer as a Tool, Lisbon, 2011, pp. 1-4.

[25] Kafsi, Mohamed, Panos Papadimitratos, Olivier Dousse, Tansu Alpcan, and J-P. Hubaux. "VANET connectivity analysis." arXiv preprint arXiv:0912.5527 (2009).

[26] Chandrasekharamenon, Neelakantan Pattathil, and Babu AnchareV. "Connectivity analysis of one-dimensional vehicular ad hoc networks in fading channels." EURASIP Journal on Wireless Communications and Networking 2012, no. 1 (2012): 1.

[27] X. Jin, W. Su and W. Yan, "Quantitative Analysis of the VANET Connectivity: Theory and Application," 2011 IEEE 73rd Vehicular Technology Conference (VTC Spring), Budapest, 2011, pp. 1-5.

[28] R. S. Tomar and M. S. Sharma, "Connectivity analysis in vehicular communication for safe transportation systems," 2017 Conference on Information and Communication Technology (CICT), Gwalior, 2017, pp. 1-5.

[29] A. Ahizoune, A. Hafid, "A new stability based clustering algorithm (SBCA) for VANETs", Proc. IEEE 37th Conf. Local Comput. Netw. Workshops (LCN Workshops), pp. 843-847, 2012.

[30] R. T. Goonewardene, F. H. Ali, E. Stipidis, "Robust mobility adaptive clustering scheme with support for geographic routing for vehicular ad hoc networks", IET Intell. Transp. Syst., vol. 3, no. 2, pp. 148-158, Jun. 2009.

[31] S.-T. Cheng, G.-J. Horng, C.-L. Chou, "Using cellular automata to

form car society in vehicular ad hoc networks", IEEE Trans. Intell. Transp. Syst., vol. 12, no. 4, pp. 1374-1384, Dec. 2011.

[32] E. Dror, C. Avin, Z. Lotker, "Fast randomized algorithm for 2-hops clustering in vehicular ad-hoc networks", Ad Hoc Netw., vol. 11, no. 7, pp. 2002-2015, 2013.

[33] C. Cooper, D. Franklin, M. Ros, F. Safaei and M. Abolhasan, "A Comparative Survey of VANET Clustering Techniques," in IEEE Communications Surveys & Tutorials, vol. 19, no. 1, pp. 657-681, Firstquarter 2017.

[34] W. Liu, C.-C. Chiang, H.-K. Wu, C. Gerla, "Routing in clustered multihop mobile Wireless networks with fading channel", Proc. IEEE SICON, pp. 197-211, Apr. 1997.

[35] S. Basagni, "Distributed clustering for ad hoc networks", Proc. 4th Int. Symp. Parallel Architectures Algorithms Netw. (ISPAN), pp. 310-315, 1999.

[36] R. Ghosh, S. Basagni, "Mitigating the impact of node mobility on ad hoc clustering", Wireless Commun. Mobile Comput., vol. 8, no. 3, pp. 295-308, 2008.

[37] R. T. Goonewardene, F. H. Ali, E. Stipidis, "Robust mobility adaptive clustering scheme with support for geographic routing for vehicular ad hoc networks", IET Intell. Transp. Syst., vol. 3, no. 2, pp. 148-158, Jun. 2009.

[38] L. Bononi, M. di Felice, "A cross layered MAC and clustering scheme for efficient broadcast in VANETs", Proc. IEEE Int. Conf. Mobile Ad hoc Sensor Syst. (MASS), pp. 1-8, Oct. 2007.

[39] R. A. Santos, R. M. Edwards, N. L. Seed, "Inter vehicular data exchange between fast moving road traffic using an ad-hoc cluster-based location routing algorithm and 802.11b direct sequence spread spectrum radio", Proc. Post Grad. Netw. Conf., Apr. 2003.

[40] T. J. Kwon, M. Gerla, V. K. Varma, M. Barton, T. R. Hsing, "Efficient flooding with passive clustering-an overhead-free selective forward mechanism for ad hoc/sensor networks", Proc. IEEE, vol. 91, no. 8, pp. 1210-1220, Aug. 2003.

[41] M. S. Almalag, S. Olariu, M. C. Weigle, "TDMA cluster-based MAC for VANETs (TC-MAC)", Proc. IEEE Int. Symp. World Wireless Mobile Multimedia Netw. (WoWMoM), pp. 1-6, Jun. 2012.

[42] Y. Gunter, B. Wiegel, H. P. Grossmann, "Cluster-based medium access scheme for VANETs", Proc. IEEE Intell. Transp. Syst. Conf. (ITSC), pp. 343-348, Sep. 2007.

[43] A. Daeinabi, A. G. P. Rahba, A. Khademzadeh, "VWCA: An efficient clustering algorithm in vehicular ad hoc networks", J. Netw. Comput. Appl., vol. 34, no. 1, pp. 207-222, 2011.

[44] N. Kumar, N. Chilamkurti, J. H. Park, "ALCA: Agent learning-based clustering algorithm in vehicular ad hoc networks", Pers. Ubiquitous Comput., vol. 17, no. 8, pp. 1683-1692, 2013.

[45] L. Zhang, H. El-Sayed, "A novel cluster-based protocol for topology discovery in vehicular ad hoc network", Proc. Comput. Sci., vol. 10, pp. 525-534, Aug. 2012.

[46] T. Taleb, A. Benslimane, K. B. Letaief, "Toward an effective risk-conscious and collaborative vehicular collision avoidance system", IEEE Trans. Veh. Technol., vol. 59, no. 3, pp. 1474-1486, Mar. 2010, [online] Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5382585.

[47] T. Taleb, A. Benslimane, "Design guidelines for a network architecture integrating VANET with 3G & beyond networks", Proc. IEEE Glob. Telecommun. Conf. (GLOBECOM), pp. 1-5, Dec. 2010.

[48] A. Benslimane, T. Taleb, R. Sivaraj, "Dynamic clustering-based adaptive mobile gateway management in integrated VANET3G heterogeneous wireless networks", IEEE J. Sel. Areas Commun., vol. 29, no. 3, pp. 559-570, Mar. 2011.

[49] G. El Mouna Zhioua, N. Tabbane, H. Labiod, S. Tabbane, "A fuzzy multi-metric QoS-balancing gateway selection algorithm in a clustered VANET to LTE advanced hybrid cellular network", IEEE Trans. Veh. Technol., vol. 64, no. 2, pp. 804-817, Feb. 2015.

[50] C. Cooper, D. Franklin, M. Ros, F. Safaei and M. Abolhasan, "A Comparative Survey of VANET Clustering Techniques," in IEEE Communications Surveys & Tutorials, vol. 19, no. 1, pp. 657-681, Firstquarter 2017.