Restore Partitioning in MANETs with Dynamically Deployed Helping Hosts — Message Cabs (MCabs)

Ting Wang, Chor Ping Low
School of Electrical and Electronic Engineering,
Nanyang Technological University, Singapore,
wang0235@e.ntu.edu.sg, icplow@ntu.edu.sg

***Abstract*—Helping hosts are intensively used in various schemes to restore partitioned *Mobile Ad Hoc Networks (MANETs)*. Most of the existing schemes offers only deterministic helping host deployment, and are thus not able to deal with fluctuating network traffic, which is a practical condition in many MANET applications. In this paper, we argue that dynamic helping hosts, with deployment that response to the changes in the traffic demand of the network, may overcome this drawback and reduce the message delay in the networks. To demonstrate the effectiveness of this observation, we propose a new helping host scheme namely the *Message Cab (MCab)* scheme for partition restoration in MANETs, and validate the performance through simulations.**

***Index Terms*—Mobile Ad Hoc Networks (MANETs), Helping Host, Message Cab, Adaptive Route Design, Dynamic Deployment**

I. Introduction

A *Mobile Ad Hoc Network* (MANET), as described in [1], is a kind of mobile wireless networks which is comprised of a collection of mobile hosts connected through wireless channels. The direct connections between the hosts are referred to as *links*. While the mobility of hosts enables the network to span over a large area, it also causes a highly dynamic topology, which is a major challenge to the applications of MANETs. When a host moves out of another's communication range, the link between them breaks, and the entire message routing path may be destroyed by this broken link. This possibility of link breakage may split hosts into different parts between which there is no possible path. This in turn may result in packets not being able to reach their destinations. We refer to this phenomenon as *network partitioning*. Each isolated part of a network is referred to as a *partition* of the network. The partitioning problem makes a critical strike on ad hoc routing because most protocols typically assume that the network is always connected. To enhance the reliability and conserve energy, partitioning should be restored in MANETs.

Various approaches have been proposed for MANETs survivability and restoration. In particular, *helping hosts* are deployed to reconnect the network partitions, and we refer to such action as *partition restoration*. The helping hosts are able to reconnect the network connectivity by moving from one partition to another in a MANET despite the fact that the *normal hosts* (*i.e.* non-helping hosts) may be partitioned. This idea is also extensively discussed in *Delay Tolerant Networks (DTN)* and *Wireless Sensor Networks (WSN)* in order to collect data from disconnected parts of the networks.

The helping hosts are addressed by different names in various schemes. In [2], [3], [4], [5], they are referred to as *ferries*, while in [6], [7], [8], they are called *helping nodes* or *forwarding nodes*. *Data MULEs* in [9], [10], [11] are also known as a kind of helping hosts, and in the DakNet project [12] buses are used as helping nodes to connect broadband network to rural villages.

In most of these existing schemes, such as [2], [8], [11], [12], the helping hosts are different from the normal hosts and are assigned as helping hosts prior to the commencement of network operations. The deployment is thus *static*, or *deterministic*. Since the number of helping hosts is fixed, it may turns out that there is too many, or too few helping hosts to meet the traffic demand in the network. Thus static deployment may not be adaptive to network traffic demand. To overcome this drawback, *dynamic deployment* is used in [6], where normal hosts with suitable moving direction and speed are chosen as helping hosts (*a.k.a helping nodes*). However, it is a centralized scheme and thus not scalable with network size. More importantly, under the assumption that the network is partitioned, it is infeasible to make all the hosts' movement information available to a central server to perform the selection. Similar problem can also be observed in [7].

Therefore, we propose a localized algorithm, namely the *Dynamic Cab Deployment (DCD)* algorithm to deploy helping hosts — *Message Cabs (MCabs)* — in this paper. The DCD algorithm allows the number of helping hosts to change dynamically with the volume of traffic, and thereby reduces the weighted average delay of messages in the network, and enhances the scalability of the deployment process. We will demonstrate that the MCab scheme overcomes the drawbacks of existing schemes by incurring lower average message delay and the results are validated through simulations.

In the following parts of this paper, we introduce the backgrounds of our work in Section II, and the model together with our objectives in Section III. The main part of the *Message Cab (MCab)* scheme, namely the *Dynamic Cab Deployment (DCD)* algorithm is presented in Section IV. Simulation and results are discussed in Section V, while Section VI concludes this paper.

II. Assumptions and Notations

While many existing schemes (*e.g.* [2], [5], [9]) focus on stationary hosts with known locations and predictable network

 ISBN: 978-1-61208-140-3

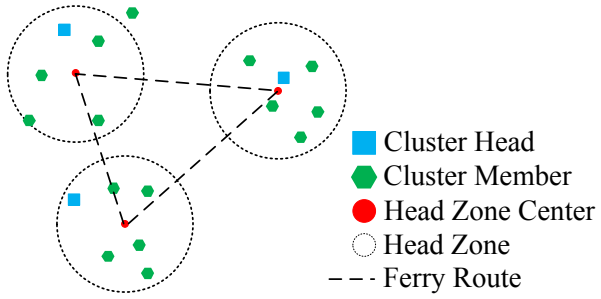


Fig. 1: Network Topology

traffic, we try to deal with a more practical scenario, where:

- the hosts are mobile;
- the traffic in the network is randomly variant and is thus not predictable.

As discussed in [13], movements of hosts in MANETs can be considered as *Levy Flights*, in which the hosts tend to stay in a certain area for a long time and occasionally make long distance flights to places far away. We could thus assume that a host stays in the same cluster for a certain period of time, and will also sometimes decides to move to another cluster. Practically, this could be due to the assignment of tasks to the host, such as in a wireless sensor network, a sensor may need to reallocate itself to new positions to collect new data.

Let's assume that the MANET of our interest has n_h mobile hosts. We assume the network is partitioned to n_c components, each of which forms a *cluster* with a chosen *cluster head*. The hosts in the same cluster are always connected. In order to simplify the problem, we restrict the movement of a cluster head to be inside a circular area, which is referred to as the *head zone*. The radius of head zone is equal to the communication range of the cluster head, denoted as r, as shown in Fig. 1. The center of head zone of cluster s is denoted as point C_s, $1 \leq s \leq n_c$. We use ϵ_{sd} to denote the time taken by a host to travel from C_s to C_d, $1 \leq s, d \leq n_c$. We note that ϵ_{sd} is a positive real number. We say cluster d is a *neighbor* of cluster s if a host is able to move directly to cluster d from cluster s in one hop.

Using proper clustering schemes, such as [14], the cluster head will always be aware of the changes in its cluster members, such as the movements, arrivals of new members and departures of existing members. It also knows the locations of other clusters, so when a host decides to move to a new place, the cluster head knows which is the next cluster the host is going to join. In addition, the cluster head works as a proxy between other normal hosts in the cluster and the helping host as well. If a helping host stops at C_s, it is able to communicate with the head of cluster s and deliver the messages to other hosts in cluster s via the cluster head. The route of a helping host will be a sequence of head zone centers (C_s's).

III. System Model and Objectives

To demonstrate our methodology of the MCab scheme, we describe an analogy between the MANET and a simple transportation system which serves several small towns. It will also explain the reason why we name our helping hosts as *cabs* and how they are different from *ferries*.

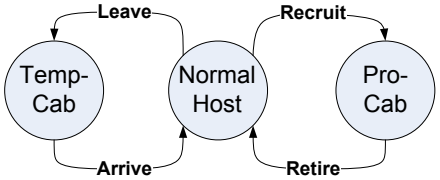


Fig. 2: States of a Host

The towns represent the clusters, each of which is distant away from the others. The cars are like the hosts — they usually move within a town, and when it is necessary, they are also able to move from one town to another. The passengers are akin to messages, which by themselves cannot move from town to town.

Buses are provided as an existing solution to the inter-town transportation problem. They are similar to the other statically deployed helping hosts (such as message ferries) that we have discussed in the previous section. The static deployment and fixed route design of buses perform poorly with varying volume of passengers. Hence, it would be preferable if we have a flexible solution.

Besides taking buses, the passengers could consider hiring a *cab* to travel to another town. The route of a cab will depend on the passengers' demands, and is thus more flexible than the bus routes. Practically, it is also more convenient in terms of saving time to hire a cab. The number of cabs is determined by how many passengers there are. When there are more passengers, more cabs can be recruited from the private cars (normal hosts) available; when the number of passengers drops, some cabs could retire and become private cars again.

Moreover, since it is possible that some cars will move to another town by their drivers' own decisions, a passenger may also take a ride from a private car which is also moving to the his/her destination town. This will utilize the mobility of private cars in the transportation system to serve the passengers' needs and save their traveling time. A car offering ride to passengers is like a one-time temporary cab that only works for a single trip to a particular town. We refer to such cars as *temp-cabs*. To distinguish from them, we refer to those cabs that works for multiple trips in a longer period of time as professional cabs, or *pro-cabs*.

We imitate the method of hiring pro-cabs and taking rides from temp-cabs in the MANETs and propose the *Message Cab (MCab)* scheme. As shown in Fig. 2, the state of a normal host, which is not a cab nor a cluster head, can transform to a *pro-cab* or a *temp-cab* under the control of the *Dynamic Cab Deployment (DCD)* algorithm.

The key difference between message cabs and the other types of helping hosts in the existing works (such as ferries, helping nodes *etc.*) is that they are not selected before the network starts. Therefore the number of cabs and cab routes can dynamically change with the traffic. Although this kind of deployment does not allow the helping hosts to have higher

 ISBN: 978-1-61208-140-3

capability in moving speed or storage space (since message cabs are just selected normal hosts), we found that it can still effectively reduce the message delay in the network.

We note that since hosts in the same cluster are connected, the delay incurred by the message transmission within a cluster (*intra-cluster communication*) is much smaller than the delivery between different clusters (*inter-cluster communication*), and is less relevant to the cab deployment route design. As a consequence, intra-cluster communication will be omitted in our following discussion.

We should note that before a message is collected by a cab, it needs to wait at some cluster head for a certain time duration. We refer to this amount of time spent on waiting as the *waiting delay (ω)* of the message. After it is collected by a cab, the cab will travel from its source cluster to its destination cluster, and thus incurs a *traveling delay (τ)* for the message. Therefore, the *overall delay* δ of a message is $\delta = \omega + \tau$.

Assume within time duration $(0, t]$, there are n_m messages that have been transmitted by the cabs. The size of message i ($1 \leq i \leq n_m$) is μ_i. The waiting, traveling and overall delay of message i are denoted as ω_i, τ_i and δ_i respectively. Similar to the objectives in [2], we are interested in reducing the *weighted average overall delay (Δ)* of the messages:

$$\Delta = \frac{\sum_{i=1}^{n_m} \mu_i \delta_i}{\sum_{i=1}^{n_m} \mu_i} = \frac{\sum_{i=1}^{n_m} \mu_i \left(\omega_i + \tau_i\right)}{\sum_{i=1}^{n_m} \mu_i} = \Delta_\omega + \Delta_\tau, \qquad (1)$$

where the *weighted average waiting delay (Δ_ω)* and the *weighted average traveling delay (Δ_τ)* of the messages are given by

$$\Delta_\omega = \frac{\sum_{i=1}^{n_m} \mu_i \omega_i}{\sum_{i=1}^{n_m} \mu_i}, \text{ and } \Delta_\tau = \frac{\sum_{i=1}^{n_m} \mu_i \tau_i}{\sum_{i=1}^{n_m} \mu_i}. \qquad (2)$$

We can see that if there are more cabs in the network, each cluster could be visited more frequently, and the waiting delay of messages reduces. It shows that the waiting delay (Δ_ω) is closely related to the cab deployment plan. On the other hand, optimization of cab routes results in a shorter traveling delay Δ_τ.

IV. Dynamic Deployment of Message Cabs

In the ideal scenario, once a inter-cluster message is received by the cluster head, there is always a cab ready to carry it towards its destination, and Δ_ω will be 0. However, in practise, with unpredictable and fluctuating traffic, it is impossible to have a cab ready for message delivery as soon as a message arrives at the cluster head. It would also be inefficient if a cab carries only one message and does not fully utilize its storage space. To solve this problem, we try to make use of the normal hosts mobility as a *temp-cab* to give a ride to the messages as well as to recruit *pro-cabs* from normal hosts in the DCD algorithm. We show that by doing so, we are able to bound the weighted waiting delay of messages from above by a predetermined value, denoted as Ω seconds.

Since the change of cluster membership is handled by the cluster head, a host (say host a) needs to report to the head of its current cluster, say cluster s (as source) before it leaves for another cluster (say cluster d, as destination). It becomes possible that the head of cluster s lets host a to carry some messages which have cluster d as their destinations. Host a will then deliver these messages to the cluster head d when it arrives there and registers itself as a new cluster member. Therefore, when a host is leaving a cluster, its state changes from normal host to temp-cab, and it is used to deliver as many messages as possible across the partitioned clusters. Upon its arrival, it delivers the messages to the head of the new cluster and its state changes back to that of a normal host. This procedure of deploying a temp-cab is depicted in Fig. 3a.

On the other hand, a timer is used to control the time when a *pro-cab* should be selected. Assuming there are n'_m inter-cluster messages stored in cluster head s, of each the size is μ_i. The *current waiting delay* of a message is the length of the time duration since the message is received by the cluster head, and is denoted as ω'_i. The timer is set to

$$t = \Omega - \frac{\sum_{i=1}^{n'_m} \mu_i \omega'_i}{\sum_{i=1}^{n'_m} \mu_i},$$

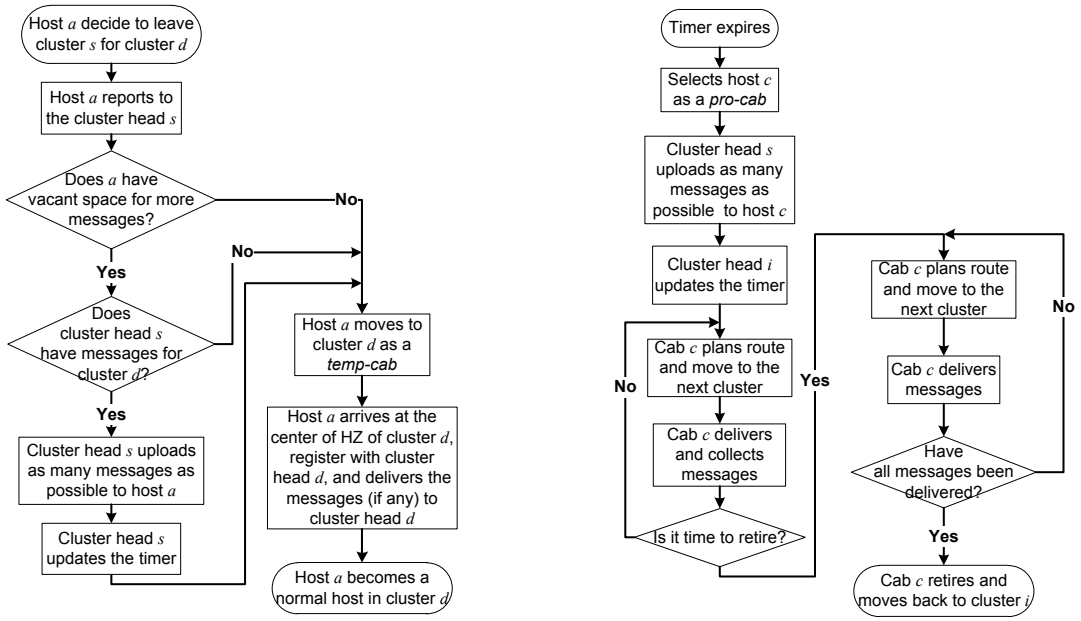
so that if a pro-cab is recruited right before the timer expires, the average weighted delay of the messages stored in cluster head s will be less than Ω, which is the required upper bound. We note that the value of t needs to be updated every time there is a change in the number of inter-cluster messages which are stored in cluster head s, such as when a new inter-cluster message is received and stored, or when some messages have been uploaded to a cab (either temp or pro) by cluster head s.

We let the time duration between the recruitment and retirement of a cab be denoted as Θ. After having served for Θ seconds as a pro-cab, it retires. Upon its retirement, a pro-cab does not stop immediately. It continues moving among the clusters to deliver the remaining messages it has already collected, but without collecting new messages. When all the messages are delivered, it moves back to the cluster where it has been recruited, and register with the head as a normal host again.

Fig. 3 depicts the flowcharts for the DCD algorithm. To avoid going into too much details, we make some simplifications to the algorithm:

- The pro-cabs are randomly selected from the normal hosts in the cluster;
- The messages are dropped according to the *hot-potato* rule.

By the hot-potato rule we mean that we deal with the elements that have heaviest impact to the system by either processing them or dropping them first. By dropping those messages with



(a) Temp-Cab (b) Pro-Cab

Fig. 3: Flowcharts of the *Dynamic Cab Deployment (DCD)* Algorithm

heavier weighted delay, the average weighted delay of the successfully delivered messages can be reduced.

With these simplifications, the DCD algorithm is a fully distributed scheme which does not require any global information. Moreover, it is not difficult to see that the worst case complexity of the DCD algorithm is $O(n_m)$, where n_m is the number of messages that have been generated during the period of network operation.

V. Simulation Results

To validate the effectiveness of the MCab scheme, extensive simulations are carried out. We randomly generate the network topology and traffic in C++ programs and compare the results with existing schemes.

We model a MANET constructed in a $500\mathrm{m}$ by $500\mathrm{m}$ area. The number of hosts in each cluster is set to 20 at the beginning of the simulations. The communication range of the hosts is $5\mathrm{m}$, and the storage size is $10\mathrm{Mb}$. The hosts move at a speed of $10\mathrm{m/s}$.

The traffic in the MANET can be specified by the number of messages per unit time, but as the size of messages also varies, it is more convenient to describe it as the size of data transmitted per unit time, *i.e.* kilobits per second (kb/s). We refer it as the *volume* of the traffic.

The number of pro-cabs may vary in the simulation. We define the *effective number of pro-cabs* as the average number of pro-cabs over time to describe how many pro-cabs are deployed in the MANET.

A. Number of Pro-Cabs

We start with our study on the performance of the DCD algorithm. The main objective of the DCD algorithm is to dynamically recruit cabs in the MANET to deliver inter-cluster messages. The algorithm is controlled by two parameters, namely Ω (the upper bound of the weighted waiting delay) and Θ (the time duration that a host serves as a pro-cab). To deploy temp-cabs, the performance of the DCD algorithm is also closely related to the frequency in which the hosts move from one cluster to another. A variable ρ is defined as the average length of period (in seconds) that a host stays in a cluster before it moves to another one.

TABLE I: Effective Number of Pro-Cabs

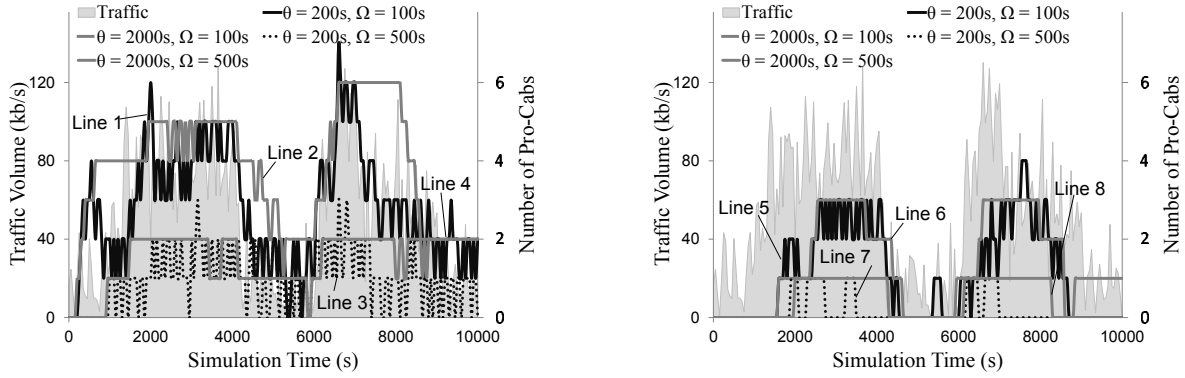
	$\rho = 5000\mathrm{s}$		$\rho = 200\mathrm{s}$	
	$\Theta = 200\mathrm{s}$	$\Theta = 2000\mathrm{s}$	$\Theta = 200\mathrm{s}$	$\Theta = 2000\mathrm{s}$
$\Omega = 100\mathrm{s}$	2.81	3.78	1.13	1.35
$\Omega = 500\mathrm{s}$	0.98	1.49	0.15	0.44

Fig. 4 displays how the number of recruited cabs changes with the volume of inter-cluster traffic. The left vertical axis corresponds to the traffic volume while the number of pro-cabs are indicated on the right axis. To study the impact of Ω, Θ and ρ to the performance of the DCD algorithm, we choose two (high and low) values for each of these variables. Each of the figures corresponds to a particular combination of the values of Ω and ρ, and plots the two groups of results which correspond to the two values of Θ. As shown in Table I, the values of Ω are $500\mathrm{s}$ and $100\mathrm{s}$; the values of Θ are $2000\mathrm{s}$ and $200\mathrm{s}$; the values of ρ are $5000\mathrm{s}$ and $200\mathrm{s}$. The effective number of pro-cabs are also shown in Table I.

1) The value of ρ: The value of ρ significantly changes the effective number of pro-cabs.

With a high value of ρ ($\rho = 5000\mathrm{s}$), the hosts tend to stay in the same cluster for a long time, and thus there are

 ISBN: 978-1-61208-140-3



(a) $\rho = 5000$s (b) $\rho = 200$s

Fig. 4: Performance of the DCD Algorithm

less chances that messages could be delivered by a temp-cab. Therefore, more pro-cabs have to be recruited to keep the waiting delay below Ω.

If the value of ρ is low ($\rho = 200$s), it implies that the hosts frequently move among the clusters. Therefore a lot of inter-cluster messages could be delivered by temp-cabs, and there are fewer messages that are left on the cluster head. Therefore, fewer pro-cabs need to be recruited.

We can observe this phenomenon in Fig. 4a and Fig. 4b by comparing line 1 with line 5 For the same values of Θ and Ω, more pro-cabs will be recruited when $\rho = 2000$s, as depicted by line 1. When ρ decreases to 200s, more temp-cabs could be used, and thus fewer pro-cabs are needed to deliver the messages, as shown by line 5. Similar fact can be observed by comparing line 2 with line 6, line 3 with line 7, and line 4 with line 8 in Fig. 4. In Table I, we can also see the effective number of pro-cabs is much smaller when ρ is low ($\rho = 200$s).

2) The value of Θ: Θ controls the length of duration for which a host serves as a pro-cab. It controls the frequency for triggering the pro-cab recruiting procedure, and thus affects how fast the system could respond to the change in traffic volume.

When the value of Θ is high ($\Theta = 2000$s), the pro-cabs have a long service time and a slow retirement. Comparing line 1 with line 2 in Fig. 4a on the time interval $[7000, 8000]$, we can see that the number of cabs stays as high as 6 when $\Theta = 2000$s (as depicted by line 2) even though the traffic volume has already dropped to a lower level, where only about 3 pro-cabs will be deployed if $\Theta = 200$s (as depicted by line 1), meaning that several cabs may be unnecessary for the purpose of keeping the weighted waiting delay low. Similar phenomena can also be observed in the time intervals $[4000, 5000]$ and $[9000, 10000]$ in Fig. 4a, and $[8500, 10000]$ in Fig. 4b, resulting in the effective numbers of cabs being much larger when $\Theta = 2000$s (as shown in Table I).

A low value of Θ ($\Theta = 200$s) causes the pro-cabs to only serve for a short period of time, and change back to the state of normal host sooner. However, messages are still being generated and stored in the cluster heads, causing new pro-cabs being recruited. The change in the cab number is thus more rapid and frequent than when $\Theta = 2000$s. Since cabs are frequently recruited and retired, and lines 1, 3, 5 and 7 appear to be more "spiky" than line 2, 4, 6 and 8 respectively in Fig. 4.

3) The value of Ω: The value of Ω influences the number of cabs more directly.

When the value of Ω is high ($\Omega = 500$s), the cluster heads are able to tolerate larger weighted waiting delay before recruiting new pro-cabs, and less pro-cabs will be hired in the MANETs. This is the reason why in Table I, the effective numbers of pro-cabs are much smaller when $\Omega = 500$s.

On the other hand, if the value Ω is low ($\Omega = 100$s), the cluster heads have to frequently hire new pro-cabs to keep the waiting delay of the messages low. From Fig. 4 it is easy to observe that the number of pro-cabs significantly increases when the value of Ω is low.

B. Delay of Messages

One of the advantages of the MCab scheme is that it can be used with different types of routes, since route design and helping host deployment are often solved as two separate problem in the existing schemes, although their performance can be inter-related, as we will show in this section. Two route design algorithms for helping host are considered, namely the TSP route (used in [2]) and the *Adaptive Message Ferry Route (aMFR)* (proposed in [15]). We compare the values of message delay caused by these types of route with and without the DCD algorithm. A single helping host is deployed in these two cases.

We demonstrate this result in Fig. 5 by showing how the MCab scheme perform when there are different number of clusters (n_c). In each figure, two values of ρ (5000s and 200s) are used in the simulation. Two different implementations of the DCD algorithm — $\Omega = 100\text{s}, \Theta = 2000\text{s}$ (denoted as DCD-a and $\Omega = 500\text{s}, \Theta = 200\text{s}$ (denoted as DCD-b) — with $\rho = 5000$s are demonstrated. The results for the other two cases ($\Omega = 500\text{s}, \Theta = 2000\text{s}$ and $\Omega = 100\text{s}, \Theta = 200\text{s}$)

 ISBN: 978-1-61208-140-3

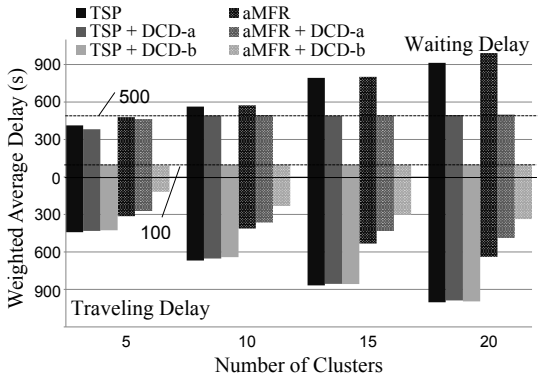


Fig. 5: Delay of Messages

and other values of ρ exhibit similar results, and are thus omitted in this section.

In Fig. 5 the simulation results are shown as a group of columns. The upper part (above 0) of each column shows the duration of average waiting delay Δ_ω, and the lower part (below 0) shows the duration of average traveling delay Δ_τ. As a result, the entire bar length shows the overall average delay Δ of the corresponding scheme.

It can be seen from Fig. 5 that the total delay is much lower with the DCD algorithm for both routes. For the TSP routes, since every helping host follow exactly the same route, the traveling delay of the messages delivered by the pro-cabs will be the same, independent to the number of pro-cabs. However, temp-cabs deliver messages in a single hop, and therefore causes less delay than the pro-cabs that follows the TSP route, and thus we can still observe from the lower parts of the solid columns that the message traveling delay is also slightly reduced by the DCD algorithm. The waiting delay of the messages is reduced by recruiting more pro-cabs when the current average waiting delay is about to exceed the threshold Ω. As shown in the graph, the values of waiting delay with DCD-a and DCD-b are bounded from above by the values of 100s and 500s, respectively.

The reduced message waiting delay can also be observed from the results for aMFR routes. Adaptive routes are used in aMFR, where a helping host's route depends on the messages carried by it. When more helping hosts are deployed, each of them will carry less messages on average. Thus a message can be delivered with less hops, and the traveling delay can also be reduced as shown in [15]. This is also shown by the dotted columns in Fig. 5.

VI. Conclusion

In this paper, we propose a scheme with flexible helping hosts, namely *Message Cab (MCab)* for message delivery in partitioned MANETs. It uses the *Dynamic Cab Deployment (DCD)* algorithm to select helping hosts (MCabs) according to the traffic volume in the network. Comparing with the existing schemes with helping hosts, we have shown that the MCab scheme effectively shorten the delay of the messages with the simulations and is adaptive to the varying traffic in the network, which has not been discussed in the existing works.

References

[1] S. Corson and J. Macker, "Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations," United States, 1999.

[2] W. Zhao and M. Ammar, "Message Ferrying: Proactive Routing in Highly-Partitioned Wireless Ad Hoc Networks," in *Proc. of the Ninth IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS'03)*, Washington, DC, USA, 2003, pp. 308–314.

[3] M. M. B. Tariq, M. Ammar, and E. Zegura, "Message Ferry Route Design for Sparse Ad Hoc Networks with Mobile Nodes," in *Proceedings of the 7th ACM international symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '06)*. New York, NY, USA: ACM, 2006, pp. 37–48.

[4] M. Ye, X. Tang, and D. L. Lee, "Fair Delay Tolerant Mobile Data Ferrying," in *Proce. 10th International Conference on Mobile Data Management: Systems, Services and Middleware (MDM'09)*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 182–191.

[5] M. H. Ammar, D. Chakrabarty, A. D. Sarma, S. Kalyanasundaram, and R. J. Lipton, "Algorithms for Message Ferrying on Mobile Ad Hoc Networks," in *Proc. the IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'09)*, 2009, pp. 13–24.

[6] Q. Li and D. Rus, "Communication in Disconnected Ad Hoc Networks Using Message Relay," *J. Parallel Distrib. Comput.*, vol. 63, no. 1, pp. 75–86, 2003.

[7] S.-H. Chung, K.-F. Ssu, C.-H. Chou, and H. C. Jiau, "Improving Data Transmission with Helping Nodes for Geographical Ad Hoc Routing," *Computer Networks*, vol. 51, pp. 4997 – 5010, 2007.

[8] C.-H. Ou, K.-F. Ssu, and H. C. Jiau, "Connecting Network Partitions with Location-Assisted Forwarding Nodes in Mobile Ad Hoc Environments," in *Proc. 10th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC'04)*, Washington, DC, USA, 2004, pp. 239–247.

[9] D. Borsetti, C. Casetti, C.-F. Chiasserini, M. Fiore, and J. M. Barceló-Ordinas, "Virtual Data Mules for Data Collection in Road-Side Sensor Networks," in *Proc. 2nd International Workshop on Mobile Opportunistic Networking (MobiOpp'10)*. New York, NY, USA: ACM, 2010, pp. 32–40.

[10] R. C. Shah, S. Roy, S. Jain, and W. Brunett, "Data MULEs: Modeling a Three-tier Architecture for Sparse Sensor Networks," Intel Research, Tech. Rep. IRS-TR-03-001, 2003.

[11] D. Jea, A. Somasundara, and M. Srivastava, "Multiple Controlled Mobile Elements (Data Mules) for Data Collection in Sensor Networks," *Lecture Notes in Computer Science*, vol. 3560, pp. 244–257, 2005.

[12] A. A. Hasson, R. Fletcher, and A. Pentland, "DakNet: A Road to Universal Broadband Connectivity," First Mile Solutions, Tech. Rep., 2003.

[13] I. Rhee, M. Shin, S. Hong, K. Lee, and S. Chong, "On the Levy-walk Nature of Human Mobility: Do Humans Walk like Monkeys?" in *Proc. of the 27th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '08)*, 2008.

[14] Y. Zhang, C. P. Low, J. M. Ng, and T. Wang;, "An Efficient Group Partition Prediction Scheme for MANETs," in *Proceedings of the 2009 IEEE Wireless Communications and Networking Conference (WCNC '09)*, 2009, pp. 1–6.

[15] T. Wang and C. P. Low, "Adaptive Message Ferry Route (aMFR) for Partitioned MANETs," in *Proc. of the 2nd International Conference on Mobile Lightweight Wireless Systems (MOBILIGHT '10)*, 2010, p. TBA.

 ISBN: 978-1-61208-140-3