Adaptive Message Ferry Route (aMFR) for Partitioned MANETs

Ting Wang and Chor Ping Low

School of Electrical and Electronic Engineering,
Nanyang Technological University, Singapore
{wang0235,icplow}@ntu.edu.sg

Abstract. *Message ferrying (MF)* scheme is a scheme to restore a *Mobile Ad Hoc Network (MANET)* from partitioning. Despite of effectiveness, it suffers from long delay caused by the poor design of ferry route. The *Message Ferry Route (MFR)* problem has been studied to minimize this drawback of MF schemes. In this paper, we argue that an adaptive way of constructing the ferry route would be more practical and effective. We thus define the *Adaptive MFR (aMFR)* problem with more general and realistic assumptions, and propose a new scheme for adaptive message ferrying. The proposed scheme adopts the *Shortest Process Time First (SPTF)* rule for the *Job Sequencing Problem (JSP)*, to construct ferry routes and is referred to as *AMFeR*. From simulations, we show that AMFeR can shorten the average message delay. With its simplicity and distributed nature, AMFeR is a suitable solution for partition restoration of MANETs.

Keywords: Mobile Ad Hoc Network (MANET), Message Ferry, Ferry Route, Message Delay.

1 Introduction

A *Mobile Ad Hoc Network* (MANET) is a kind of mobile wireless network, which is a collection of mobile hosts connected through wireless channels. The hosts of a MANET are usually called *nodes* while direct connections between the nodes are referred to as *links*. A node in a MANET exchanges messages with other nodes as a terminal and forwards the messages as an intermediate routers at the same time. The routing path of a message in a MANET is formed by a collection of mobile nodes. Messages are forwarded hop by hop.

While mobility of nodes enables the network to span over a large area, it also causes a highly dynamic topology, which is a major challenge in the applications of MANETs. When a node moves out of another's communication range, the link between them breaks. The entire message routing path may be destroyed by this broken link. This possibility of link breakage may split nodes into parts (components) between which no possible path exists. This in turn may result in packets not being able to reach their destinations. We refer this phenomenon as *network partitioning*. Each isolated component of a network is referred to as a

P. Chatzimisios et al. (Eds.): MobiLight 2010, LNICST 45, pp. 99–111, 2010.

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 prevented in MANETs.

Various approaches have been proposed in recent years for the MANET survivability and restoration. One of them, namely the *Message Ferrying (MF)* scheme [10], uses special nodes as *ferries* to deliver messages across partitions of a MANET. With MF scheme, despite the fact that the nodes in a MANET are partitioned, the ferries restore the connection by moving from one partition to another. However, ferries' movement consumes considerable amount of energy and time, causing long message delay in the network. These drawbacks of MF scheme can be minimized by optimizing the route on which each ferry moves. In [10] Zhao *et al.* formulated the *Massage Ferry Route (MFR)* problem and adapted solutions from the well studied *Traveling Salesman Problem (TSP)*. In [8], inter arrival times of areas are used to construct the ferry routes. In all these works, the ferry does not change its route when it moves in the network, making its route static. We will show in this paper why such a static solution of MFR will not be optimal and an adaptive scheme would be preferred.

Some other existing works such as [5] do consider the destination of each message when constructing the ferry routes. However, the location and movement information, such as coordinates, speed and directions of other nodes are needed to construct the ferry routes. This need of collaboration between nodes and ferries makes these schemes costly in energy consumption and bandwidth usage. Moreover, under the assumption that the network is partitioned, such information may not be available to the ferries. Thus, the route could not be determined effectively. Hence, localized schemes would be preferable.

In this paper we redefine the MFR problem in an adaptive context, as a new problem referred to as the *Adaptive Message Ferry Route (aMFR)* problem. A new scheme, namely *AMFeR* is also proposed for this problem. As opposed to the existing schemes, AMFeR is adaptive in the sense that it only decides the next segment of the route based on the traffic in the network. We adpot the *Shortest Process Time First (SPTF)* rule from the *Job Sequencing Problem (JSP)* and define a selection factor to construct the ferry route. This allows a ferry to change its route adaptively in respond to the traffic. Moreover, AMFeR is a fully localized scheme that does not incur any additional communication among nodes in the network. Our simulation shows that AMFeR can effectively shorten the average message delay in the network as compared to existing schemes.

The remaining parts of this paper is organized as follows. Section 2 states the assumptions we make in this work and defines the aMFR problem. In Section 3, we describe our observations concerning the aMFR problem. These observations lead us to our solution to the problem, namely the AMFeR scheme, which will be introduced in Section 4. Our simulation results are discussed in Section 5. Finally, we conclude this paper with Section 6.

2 Problem Formulation

The route of a ferry in MF scheme is usually defined by a series of points referred to as *waypoints*. The ferry stops on each of these waypoints and moves towards the next one according to a given sequence. The line segment connecting each pair of consecutive waypoints is referred to as a *leg* of the route. In most of the existing works, the sequence of waypoints does not change its order once being constructed, and is thus referred to as a *static route*. Moreover, since the solution of TSP is usually adapted for route construction, the route is a *cycle* in the topology graph, where no waypoint repeats. The time needed to finish one cycle of the static route is the *length* of the route.

In this paper, we aim to construct an *adaptive route* for the ferry. It is adaptive because the sequence of waypoints changes over time according to the messages being transmitted in the network. Also, we allow the route to be a *walk* instead of a cycle, where waypoints may repeat. In this section, we discuss the assumptions and notations used in this paper, and then define the *adaptive Message Ferry Route (aMFR)* problem.

2.1 Assumptions and Notations

The original *Message Ferry Route (MFR)* problem is defined in [10] with two fundamental assumptions:

- the nodes are stationary and their positions are known to the ferry;
- the traffic between two given nodes can be estimated prior to constructing the ferry route.

These assumptions can hardly be met in a typical MANET, where nodes move around and the traffic can hardly be predicted. Therefore, in this paper, we target to consider a more practical scenario, where:

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

The MANET of our interest has n mobile nodes and f ferries. We assume the network is partitioned to k components, each of which forms a *cluster* with a chosen *Cluster Head (CH)*. In order to simplify the problem, we restrict the movement of a *CH* inside a circular area, namely the *Head Zone (HZ)*. The radius of *HZ* equals to the communication range of the nodes, denoted as r, as shown in Fig. 1. The center of *HZ* of cluster i is denoted as point C_i, $1 \leq i \leq k$. We use ϵ_{ij} to denote the time taken by a ferry to travel from C_i to C_j, $1 \leq i, j \leq k$. We note that ϵ_{ij} is a positive real number. We say cluster i and j are *neighbors* unless $\epsilon_{ij} = \infty$, which means the ferry cannot move directly from cluster i to cluster j.

We assume the movements, splitting and combinations of the clusters are all handled by some clustering scheme so that the *CH* is aware of any change of the cluster members, *i.e. CH* works as a proxy between the cluster members and the

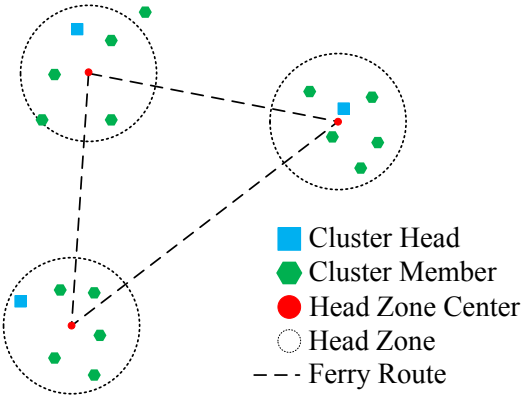


Fig. 1. Network Topology

ferry. There are many existing clustering schemes which can be adopted, such as [9], for this purpose.

By the above definitions and assumptions, if a ferry stops at C_i, it is able to communicate with the *CH* of cluster i. The ferry only need to communicate with *CH*s to deliver the messages to other nodes. The route of ferry, denoted as $\mathbb{R}$, will be a sequence of *HZ* centers (C_i's), each of which is equivalent as a waypoint.

Moreover, since nodes in the same cluster are connected, we can use normal MANET routing protocols such as AODV or DSR for the communication between a CH and its members. We note that the delay incurred by the message transmission within a cluster (*intra-cluster communication*) is much smaller than between different clusters (*inter-cluster communication*), and is also less relevant to the ferry route. As a consequence, intra-cluster communication will be omitted in our following discussion.

2.2 Objective

In [10], the objective function of the original MFR problem is defined to minimize:

$$\Delta^{\mathbb{R}} = \frac{\sum\limits_{1 \le i,j \le k} b_{ij}\delta_{ij}^{\mathbb{R}}}{\sum\limits_{1 \le i,j \le k} b_{ij}}, \tag{1}$$

where b_{ij} is the average traffic from i to j, and $\delta_{i,j}^{\mathbb{R}}$ is the time spent by the ferry to travel from i to j on the static route $\mathbb{R}$. $\Delta^{\mathbb{R}}$ is thus the weighted average of delay incurred by route $\mathbb{R}$. We note that $\delta_{i,j}^{\mathbb{R}}$ is the time spent on traveling on the route, and thus $\Delta^{\mathbb{R}}$ is referred to as the *traveling delay* on route $\mathbb{R}$.

We should also note that before the messages are collected by the ferry, they need to wait at their sources for certain amount of time. This kind of delay is not

counted in the traveling delay. We refer to this amount of time spent on waiting at source as the *waiting delay*.

The waiting delay is not taken into account in the objective function of MFR. This is because for a static route, waiting delay is always bounded from above by the length of route $\mathbb{R}$, denoted as $\delta^{\mathbb{R}}$. If the traffic is uniform, the expected value of waiting delay is fixed at $\delta^{\mathbb{R}}/2$.

However, since we have assumed that the traffic is not uniform and unpredictable, b_{ij} could not be determined beforehand. We could not use Eqn. 1 to measure the delay under our new assumptions to the aMFR problem. In addition, because we use an adaptive route, the length of the route is not fixed. This in turn imply that the upper bound and the average value of the waiting delay for a static route is not applicable in this case. Due to these facts, we need to measure the delay for each message to compute the overall delay.

Assume within time duration $(0, t]$, there are m messages to be transmitted by the ferries. The size of message h $(1 \leq h \leq m)$ is μ_h, and its source and destination cluster are denoted as s_h and d_h respectively. The *delay* (δ_h) of the message h is comprised of two parts, namely *waiting delay* (ω_h) and *traveling delay* (τ_h).

After the *CH* receives messages from other cluster members, it needs to firstly wait for a ferry to come in contact with it. This amount of time spent for waiting the ferry is ω_h. The *weighted average waiting delay* (Δ_ω) of all the messages is given by

$$\Delta_\omega = \frac{\sum_{h=1}^{m} \mu_h \omega_h}{\sum_{h=1}^{m} \mu_h}. \tag{2}$$

Next, the ferry will move from C_{s_h} to C_{d_h} on its chosen route, incurring the traveling delay τ_h. However, since the route adapts to traffic and changes dynamically, τ_h may not be equal to τ_g, for any two given messages h and g which have the same source and same destination. The *weighted average traveling delay* (Δ_τ) of all the messages is taken as

$$\Delta_\tau = \frac{\sum_{h=1}^{m} \mu_h \tau_h}{\sum_{h=1}^{m} \mu_h}. \tag{3}$$

The overall delay of message h, $\delta_h = \omega_h + \tau_h$. The *weighted average overall delay* of the messages is thus

$$\Delta = \frac{\sum_{h=1}^{m} \mu_h \delta_h}{\sum_{h=1}^{m} \mu_h} = \frac{\sum_{h=1}^{m} \mu_h \left(\omega_h + \tau_h\right)}{\sum_{h=1}^{m} \mu_h} = \Delta_\omega + \Delta_\tau. \tag{4}$$

The objective of aMFR problem is to find an adaptive route with C_i as waypoints to minimize Δ.

3 Observations

We could make the following observations from the aMFR problem:

- **It's infeasible to find the optimal adaptive ferry route in the aMFR problem.** The complexity of the aMFR problem can be reduced when we only consider Δ_ω or Δ_τ. When we neglect the waiting delay and assume that only one cluster transmits messages, the aMFR problem can be reduced to the weighted *Minimum Latency Problem (MLP)*, which is known to be a MAX-SNP-hard problem [4]. Generally, there is no polynomial time approximation scheme for MAX-SNP-hard problems [1]. Even for a more simplified scenario, where μ_i is constant and the traffic is uniform among the clusters, the problem reduces to the *Traveling Salesman Problem (TSP)*, which is still NP-hard. On the other hand, if we neglect the traveling delay and consider only the waiting delay, the problem is in fact still a NP-hard variation of TSP, namely *Prize Collecting TSP (PCTSP)* [3]. Therefore, it is highly unlikely that we could construct an optimal adaptive ferry route in polynomial time.
- **A static route can not be the optimal solution.** This is because in the network, traffic may not be constant and uniform all the time. If some cluster i receives messages more frequently than others, it will be a waste of time to visit all the other clusters before returning to cluster i, which is what will take place with a static route. Moreover, due to the nature of the MANET tasks, such as area exploration or surveillance, the frequency and sizes of messages may vary over time. A static route could not adapt itself to such changes.
- **The key to finding a suitable route is to determine the *next leg*.** The ferry stops to deliver and collect messages to/from a CH after each leg. Therefore, we do not need to plan the route beyond the next leg, as the future legs should be dynamically determined according to the messages that a ferry will collect in the future and are not unveiled to the ferry yet. Therefore, if a scheme can produce the next leg effectively, it will offer a good solution to the aMFR problem.

Based on the above observations, we propose to use a new scheme namely AMFeR to choose the "best"[1] *CH* among all the neighbors as the next waypoint of the ferry to form the dynamic ferry route.

4 The AMFeR Scheme

4.1 JSP and the SPTF Rule

In the *Job Sequencing Problem (JSP)* [2] of a single machine, we need to find the optimal sequence of a series of jobs j with weight[2] l_j and process time p_j to minimize the average delay of these jobs given by:

[1] As the most suitable waypoint in a adaptive ferry route to minimize Δ.
[2] Larger l_j represents higher importance and vice versa.

$$\Delta_{\text{jsp}} = \frac{\sum_j l_j \delta_j}{\sum_j l_j} \tag{5}$$

Smith proved in [7] that the optimal solution to JSP can be produced by following the *Shortest Process Time First (SPTF)* rule:

– **SPTF Rule:** Sequencing the jobs in order of non-decreasing ratio p_j/l_j produces an optimal schedule to minimize Δ_{jsp}.

Comparing Eqn. 5 with Eqns. 2, 3 and 4, we can observe that the objective functions in JSP and aMFR are similar (details will be discussd in the next Section). Hence SPTF would be a useful tool in constructing adaptive ferry route. In this paper, to ensure that the equations that we derive in the next Section are well defined, we define an *inverse SPTF (iSPTF)* rule:

– **iSPTF Rule:** To optimally sequence the jobs in JSP, the job with *highest* l_j/p_j value should be chosen first.

It is easy to see that iSPTF is logically equivalent to SPTF. They are both able to solve JSP optimally.

4.2 AMFeR

In the aMFR problem, we may interpret the event that the ferry goes to a destination cluster j as job j. Assuming the ferry is currently at cluster i, we could say that the required processing time of job j is the traveling time from cluster i to cluster j, *i.e.* ϵ_{ij}, which is similar to p_j in the JSP.

In order to apply the iSPTF rule, we need to determine the weight l_j of these jobs. We note in JSP, the weight of a job indicates the how much each job contributes to the objective function Δ_{jsp}. In contrast, job j contributes in both Δ_ω and Δ_τ in the aMFR problem.

The contribution of job j to Δ_ω is based on the size of the messages that a ferry collects when it visit cluster j. We define the total size of these messages as $M_{s=j}$. On the other hand, job j also contributes to Δ_τ by delivering the messages with destinations in cluster j. We denote the total size of messages delivered by the ferry in job j is denoted as $M_{d=j}$. Therefore, the overall contribution of job j to the objective function Δ is $M_{s=j} + M_{d=j}$. We use this value as the weight of job j (like l_j in JSP). Then we can choose the cluster with maximal ratio of $(M_{s=j} + M_{d=j})/\epsilon_{ij}$ as the next waypoint of the adaptive route according to the iSPTF rule.

We also note that the aMFR problem and JSP are still two different problems. This is due to the fact that in the aMFR problem, ϵ_{ij} varies with the ferry's location (cluster i), while in JSP p_j is constant. Therefore iSPTF may not provide the exact optimal solution to the aMFR problem. However, since we are not interested in the entire route but the next leg, we believe the iSPTF rule still provides a good indication of which cluster should be the next waypoint.

Table 1. Algorithm for AMFeR

```
arrive(cluster i, time Φ){
  initialize F_j, M_{s=j} and M_τ to 0 for all j
  deliver messages to cluster i
  for each message h collected from cluster i do        %Step 1%
    store message h
    compute M'_{s=i} += μ_h
  for each undelivered message h do                     %Step 2%
    compute M_{d=d_h} += μ_h
  for each cluster j do                                 %Step 3%
    compute M_{s=j} = M'_{s=j}/φ_j (Φ + ε_ij − φ_j)
    compute F_j = (M_{s=j} + M_{d=j})/ε_ij
  for all the clusters do                               %Step 4%
    find j_max of which F_{j_max} is the maximum
  arbitrarily break ties
  set φ_i = Φ
  move to the next waypoint C_{j_max}
}
```

We still need to compute $M_{s=j}$ and $M_{d=j}$. $M_{d=j}$ depends only on the messages collected by the ferry and not yet be delivered, *i.e.* the stored messages. The ferry can find the exact value of $M_{d=j}$ by going through its storage to calculate

$$M_{d=j} = \sum_{h|d_h=j} \mu_h, \tag{6}$$

where h is the id of the message.

However, $M_{s=j}$ is unveiled only after the ferry has arrived at cluster j. Hence we have to project its value. We use Φ to denote the current time and ϕ_j as the time of the most recent arrival of the ferry at cluster j ($\phi_j < \Phi$). The total size of messages previously sent from cluster j is denoted as $M'_{s=j}$. The average data rate during $(0, \phi_j]$ is thus $\frac{M'_{s=j}}{\phi_j}$. If the ferry goes from cluster i to cluster j in the next leg, it will arrive at C_j at time $\Phi + \epsilon_{ij}$. The messages waiting at cluster j are accumulated from time ϕ_j to $(\Phi + \epsilon_{ij})$. Their total size is predicted as

$$M_{s=j} = \frac{M'_{s=j}}{\phi_j} \left(\Phi + \epsilon_{ij} - \phi_j\right). \tag{7}$$

Using Eqns. 6 and 7, we can define a *selection factor* $\mathcal{F}$ as

$$\mathcal{F}_j = \frac{M_{s=j} + M_{d=j}}{\epsilon_{ij}} = \frac{\sum\limits_{h|d_h=j} \mu_h + \frac{M'_{s=j}}{\phi_j} \left(\Phi + \epsilon_{ij} - \phi_j\right)}{\epsilon_{ij}} \tag{8}$$

for each cluster j. Cluster with maximal $\mathcal{F}_j$ will be chosen as the next waypoint of the route by iSPTF rule. Ties can be broken arbitrarily.

Lemma 1. *The complexity of algorithm* `arrive` *is* $O(k+m)$.

Proof. Step 1 in `arrive` updates the collected messages size $M'_{s=i}$ of current cluster (cluster i). Since the total number of messages is m, Step 1 takes $O(m)$ time in the worst case. In Step 2, the algorithm goes through the ferry's storage to calculate the accumulated size of messages $M_{d=d_h}$ that will be delivered to cluster d_h. This also takes at most $O(m)$ time. In Step 3, the projected size of messages $M_{s=j}$ waiting at each cluster is calculated. It will takes $O(k)$ time. To find the maximal selection factor $\mathcal{F}_j$ in Step 4, another $O(k)$ time is required. The overall complexity for algorithm `arrive` is thus $O(k+m)$. □

We note the complexity of algorithm `arrive` is much lower than almost all the known TSP schemes. Moreover, being a fully localized scheme, AMFeR is very suitable to the MANETs where the resources are limited.

5 Simulations Results

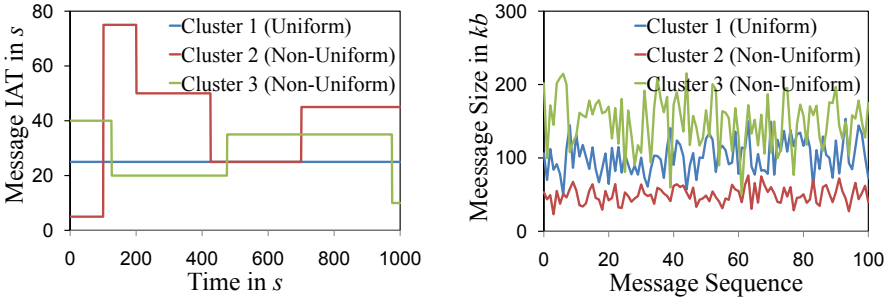
In the simulations, we randomly generate the network topology and traffic to test the effectiveness of AMFeR and compare with existing MFR solutions, namely TSP route and approximate TSP route.

We model k clusters distributed in a 200m by 200m area. The number of nodes in each cluster is a uniform random integer in the interval $[5, 20]$. A group of f ferries are used in the MANET. The communication range of the nodes and the ferries is 5m. We require each pair of *HZ* centers be at least 20m apart to ensure that the network is partitioned. The ferries move at a speed of 10m/s. Ferry routes are constructed using AMFeR and TSP (with *HZ* centers as waypoints).

It is well-known that TSP is a NP-hard problem, where optimal solution can not be found within polynomial time. We use brute force to find the optimal TSP route in the simulation. However it would only be feasible to use approximate TSP route for the ferries in practise. To reflect this fact, we also use the *Improving Search Algorithm* [6] to find an approximate solution to TSP as a feasible route. Both optimal and approximate TSP routes are static and are used as benchmarks to compare with AMFeR.

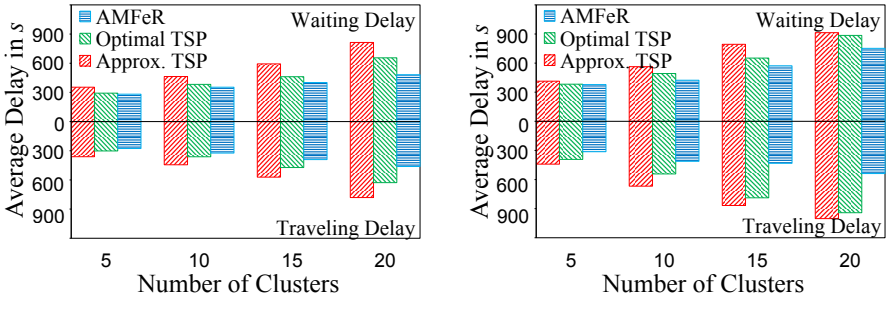
The network traffic can be described by the average size of the messages and the expected time duration between the generation of two consecutive messages at a source. The latter parameter is referred to as the *message inter arrival time*. Since the intra-cluster communication is omitted, we consider each *CH* as a source of the traffic. We use NS-2 to simulate the traffic in the network. Two traffic patterns are considered, namely *uniform traffic* and *non-uniform traffic*.

In uniform traffic pattern, the message inter arrival time fixed at 25sec and and the average size of messages is 100kb for all clusters. However, as we stated in Section 3, uniform traffic may not be realistic for most applications in MANETs. Therefore we use random numbers to control the expected inter arrival time and average size of the messages in non-uniform traffic pattern, as shown in Fig. 2.



(a) Message Inter Arrival Time (b) Message Size

Fig. 2. Traffic Patterns



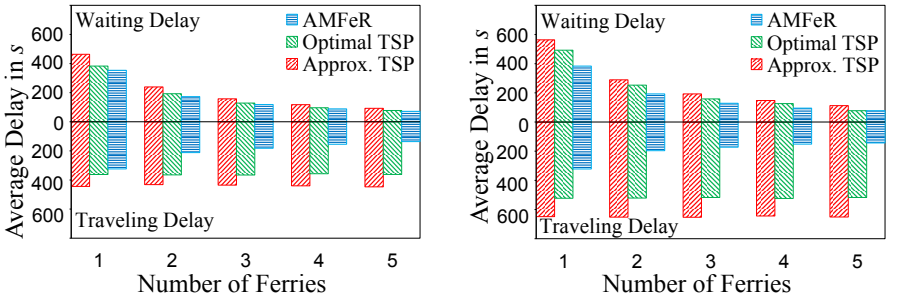
(a) Uniform Traffic (b) Non-Uniform Traffic

Fig. 3. Performance with Different Number of Clusters (k)

Fig. 2(a) depicts the expected message inter arrival time of 3 clusters, one of which has uniform traffic, another two has non-uniform traffic which changes its expected inter arrival time every 25sec with a probability of 0.2.

The values of message size in both traffic patterns are plotted in Fig. 2(b). Cluster 1 represents a cluster with uniform traffic pattern while cluster 2 and 3 are with non-uniform traffic pattern, in which the size of a message is generated using a Gaussian distribution. For each cluster, the mean value of the Gaussian distribution is a random number drawn from the interval [10kb, 300kb].

To show the performance of the MFR schemes in different scenarios, we assign different values to k and f. For each scenario, we generate a total number of 10000 messages to calculate their weighted average delay according to Eqns. 2, 3 and 4. The results are plotted in Fig. 3 and 4. In each figure, the results are shown as a group of vertical bars. The upper part of each bar shows the length of average waiting delay Δ_ω, and the lower part corresponds to the traveling delay Δ_τ. As a result, the entire bar length shows the overall average delay Δ of the scheme.



(a) Uniform Traffic (b) Non-Uniform Traffic

Fig. 4. Performance with Different Number of Ferries (f)

Fig. 3 shows the performance of the three ferry route schemes with different numbers of clusters, which increases from 5 to 20. A single ferry is used in these scenarios. For both uniform (Fig. 3(a)) and non-uniform (Fig. 3(b)) traffic, AMFeR outperforms the other two schemes by having much lower average delays. With increasing number of clusters, the ferry may need to stop at more clusters before reaching the destination, which results in longer waiting delay and traveling delay for all the schemes with both traffic patterns. The difference in traveling delay of the schemes are much larger with non-uniform traffic pattern, as shown in Fig. 3(b). This is because in AMFeR the route is generated dynamically according to the traffic and thus shorten the traveling delay. However, the waiting delay of AMFeR also increases to a higher value in the non-uniform traffic scenario as compared to the uniform traffic case. This is because we projected the value of $M_{s=j}$ to construct the route. With non-uniform traffic pattern, this projection will become less accurate, causing a higher waiting delay.

We also vary the number of ferries to obtain Fig. 4. Ferries are randomly allocated at one of the total 10 clusters initially. We note that AMFeR is a fully distributed scheme; so no collaboration is needed among the ferries. Each ferry can decide its route locally. The increase in ferry number does not affect the complexity of AMFeR scheme at all.

From Fig. 4, we can see the ferry number has different impacts on the waiting and traveling delays. The traveling delay drops to a lower value with increasing ferry number only when AMFeR is used. For TSP and approximate TSP routes, no significant change in Δ_τ can be observed. This is because when the same static route is used, the delay of a message only depends on the distance between its source and destination on the route, and is not dependent on the number of ferries. Therefore once the static route is constructed, the traveling delay of messages will not change due to the change in ferry number. On the other hand, when AMFeR is used, the ferries defines their own route based on the messages they are carrying. The traveling delay of a message can be shortened if there are fewer messages carried by the same ferry. Time can be thus saved. As the

number of ferries increases, fewer messages will be carried by each of them, and the average traveling delay is thus further decreased.

On the other hand, the waiting delay drops for all 3 schemes when more ferries are deployed in the MANET. This is because even when a static ferry route (TSP or approximate TSP) is used, a cluster can be visited more frequently by the ferries when we use more ferries in the network. The time for which the messages have to wait at the *CH* is thus shortened. More specifically, if the value of waiting delay is Δ_ω when a single ferry is used, the value decreases to approximately Δ_ω / f when f ferries are deployed. Similar result can be observed with AMFeR.

In all the scenarios, the overall delay in AMFeR is significantly lower than other schemes. The results show that ferries using adaptive ferry route deliver the messages faster than those using static route. AMFeR is an effective ferry route scheme in partition restoration in MANETs.

6 Conclusion

In this paper, a problem of constructing message ferry route is addressed. We study the existing Message Ferry Route (MFR) problem and define an adaptive version of the problem (aMFR), which is believed to be more practical to real life applications as compared to the original definition. We propose AMFeR as a solution to the aMFR problem. In AMFeR, we used a similar concept from the Job Sequencing Problem (JSP) and defined a selection factor to construct the ferry route adaptively. From our simulations, we show that AMFeR effectively reduce the overall delay as compared to the existing solutions. Moreover, as AMFeR is a fully distributed scheme with little computation and storage complexity, it is a suitable solution for partition restoration in MANETs.

References

1. Arora, S., Lund, C., Motwani, R., Sudan, M., Szegedy, M.: Proof verification and the hardness of approximation problems. J. ACM 45(3), 501–555 (1998)
2. Baker, K.R.: Introduction to Sequencing and Scheduling. John Wiley & Sons, Inc., Chichester (1974)
3. Balas, E.: The Prize Collecting Traveling Salesman Problem. Networks 16(6), 621–636 (2006)
4. Blum, A., Chalasani, P., Coppersmith, D., Pulleyblank, B., Raghavan, P., Sudan, M.: The Minimum Latency Problem. In: Proceedings of the 26th Annual ACM Symposium on Theory of Computing (STOC 1994), pp. 163–171. ACM, New York (1994)
5. Li, Q., Rus, D.: Communication in Disconnected Ad Hoc Networks Using Message Relay. J. Parallel Distrib. Comput. 63(1), 75–86 (2003)
6. Merz, P., Freisleben, B.: Genetic Local Search for the TSP: New Results. In: Proceedings of the 1997 IEEE International Conference on Evolutionary Computation, pp. 159–164 (April 1997)
7. Smith, W.E.: Various Optimizers for Single-Stage Production. Naval Research Logistics Quarterly 3, 59–66 (1956)

8. Tariq, M.M.B., Ammar, M., Zegura, E.: 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 2006), pp. 37–48. ACM, New York (2006)
9. Zhang, Y., Low, C.P., Ng, J.M., Wang, T.: An Efficient Group Partition Prediction Scheme for MANETs. In: Proceedings of the 2009 IEEE Wireless Communications and Networking Conference (WCNC 2009), pp. 1–6 (2009)
10. Zhao, W., Ammar, M.: 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 2003), Washington, DC, USA, pp. 308–314 (2003)