

PAPER TITLE:

Ferry Replacement Protocols in Sparse MANET Message Ferrying Systems

ABSTRACT:

The Message Ferrying (MF) scheme has been proposed as a strategy for providing connectivity in sparse partitioned ad hoc networks. A set of nodes called *ferries* are responsible for carrying messages for all nodes in the networks. A ferry periodically moves around the deployed area along a pre-determined route and relays messages between mobile nodes which otherwise cannot communicate with each other directly. The MF scheme relies on the ferry to provide connectivity and is vulnerable to a single point of failure. Also, mobile nodes might take turns to be a ferry since they normally have limited resources. Therefore, ferry replacement is important for robustness in the MF deployment. In this paper, we propose two ferry replacement protocols. The ferry designates its successor in the designation scheme while nodes collaborate and elect one node among themselves as the ferry replacement in the distributed election approach. We evaluate the performance of these two replacement protocols through comprehensive simulation.

KEYWORDS:

Sparse MANET, store-carry-forward routing, Message Ferry (MF), failure, ferry replacement, ferry designation, ferry election

AUTHOR NAMES:

Jeonghwa Yang, Yang Chen, Mostafa Ammar, and Chungkee Lee

AFFILIATIONS AND ADDRESSES:

Networking and Telecommunication Group
Georgia Institute of Technology
Atlanta, Georgia 30332

TELEPHONE NUMBERS:

404-894-6737

E-MAIL ADDRESSES:

{jeonghwa, yangchen, ammar, cklee}@cc.gatech.edu

Ferry Replacement Protocols in Sparse MANET Message Ferrying Systems

Jeonghwa Yang, Yang Chen, Mostafa Ammar, and Chungkee Lee
Networking and Telecommunication Group
Georgia Institute of Technology
Atlanta, Georgia 30332
Email: {jeonghwa, yangchen, ammar, cklee}@cc.gatech.edu

Abstract—The Message Ferrying (MF) scheme has been proposed as a strategy for providing connectivity in sparse partitioned ad hoc networks. A set of nodes called *ferries* are responsible for carrying messages for all nodes in the networks. A ferry periodically moves around the deployed area along a pre-determined route and relays messages between mobile nodes which otherwise cannot communicate with each other directly. The MF scheme relies on the ferry to provide connectivity and is vulnerable to a single point of failure. Also, mobile nodes might take turns to be a ferry since they normally have limited resources. Therefore, ferry replacement is important for robustness in the MF deployment. In this paper, we propose two ferry replacement protocols. The ferry designates its successor in the designation scheme while nodes collaborate and elect one node among themselves as the ferry replacement in the distributed election approach. We evaluate the performance of these two replacement protocols through comprehensive simulation.

I. INTRODUCTION

Mobile Ad hoc Networks (MANET) are considered a promising communication paradigm in situations where rapid deployment and self-configuration are essential. They allow nodes to communicate with each other without any existing infrastructure or centralized administration and have applications in a variety of environments such as rescue operations and battlefield communication. In some applications, MANETs may be partitioned because nodes are sparsely deployed in a wide area [1] or because of small radio coverage, fluctuating signal strength, node failure and physical obstacles [2]. In partitioned networks, multi-hop paths between node pairs may not exist at any point in time. This makes traditional MANET routing protocols not applicable [3].

Node mobility can be exploited to help data delivery in a partitioned MANET. Nodes buffer packets during network partition, carry them and forward them when they meet other nodes or destinations. This store-carry-forward routing paradigm has been used in [4], [5], [6], [7], [8], [9].

Message Ferrying (MF) is one approach that has been proposed by our group for routing data in a partitioned MANET based on the store-carry-forward paradigm. It is a proactive mobility-assisted approach which utilizes a set of special mobile nodes called *ferries* to provide communication services for nodes in the network. The ferries move in the

deployed area along the predefined routes, collect messages from source nodes, carry those messages and deliver them to corresponding destinations. The ferry routes can be optimized in terms of delay and bandwidth for stationary nodes [8]. On the other hand, if the deployed nodes are mobile, they can proactively adapt their trajectories to meet and exchange messages with a ferry [10]. In both scenarios, the MF scheme can provide intermittent yet regular connectivity for otherwise disconnected nodes.

The MF scheme can be adapted to many applications. In sensor networks, the low-cost and low-power sensor nodes are usually sparsely distributed without any means of direct communication. Ferries can be used to collect data, assign tasks and do regular maintenance. In crisis-driven cases such as battlefield/rescue operations, direct communication could be lost due to hostile attack or harsh environments. A ferry, e.g., an Unmanned-Aerial-Vehicle (UAV), can provide regular yet intermittent data transmission which is essential for the success of those operations. For cost-efficiency reasons, buses or other vehicles can act as ferries to provide low cost network services to remote areas [11].

In this paper, we enhance the MF scheme to make it robust. The MF system relies only on the ferry to provide connectivity in the whole network. This is essentially subject to a single point of failure. Furthermore, the wireless communication environment is harsher than that for wired networks and conventional survivability provisioning cannot be applied in MANET [12]. It helps communication protection to provide multiple ferries covering the same area simultaneously. Even in this case, a failure degrades the system performance. Therefore, a low overhead yet robust *ferry replacement scheme* is important for the operation of the MF system.

Being a ferry involves extra movement and much more frequent message exchanges. In some scenarios, the ferrying duties are assigned to one of the nodes in the system (in contrast to having the ferry being a specifically provisioned node). Since regular nodes usually have limited resources, they should perform the ferry role only for a fixed duration and then rotate the role with others.

There are two scenarios for ferry replacement: the *graceful ferry stop* and the *abrupt ferry failure*. By the graceful stop we mean that the ferry informs one or more nodes of its departure before it stops functioning as a ferry. The abrupt ferry failure

*This work was supported by the NSF Grant ITR-0313062 and AFOSR MURI Grant F49620-00-1-0327.

means that the ferry suddenly crashes due to some unexpected reasons such as device failure or external attacks so that it does not have the opportunity to inform nodes of its departure. In this paper, we focus our discussion on the abrupt ferry failure problem as it is the most challenging.

We propose two different classes of ferry replacement protocols, the *successor designation scheme* and the *distributed election approach*. The former requires the current ferry on duty to designate one node as its successor. The successor regularly checks the ferry status. On detecting ferry failure, the successor takes over the role of the failed ferry. The distributed election protocol, on the other hand, requires nodes to elect one among them as a ferry replacement without any centralized coordination. Once nodes detect the ferry failure, they collaborate with each other to elect one. The distributed election protocol is needed when the successor designation scheme fails as will be discussed later. We evaluate the performance of these two protocols through comprehensive simulations.

The rest of the paper is organized as follows. The system model is given in Section II along with the problem definition. Section III describes the successor designation scheme. Section IV provides details of the ferry election protocol followed by correctness proof. This paper concludes in Section V.

II. SYSTEM UNDER CONSIDERATION

A. MF Overview

In this paper, we focus on a ferrying scheme described in [10]. Ferries move around a deployed area according to specific pre-determined routes and provide communication service for disconnected nodes. Ferry routes are known to nodes, e.g., periodically broadcast by the ferry via its long range radio or conveyed by other out-of-band means. With knowledge of ferry routes, nodes take proactive movement to meet up with a ferry when they have messages to be sent or received. Figure 1 illustrates how the MF scheme works.

A ferry and a node communicate via a short range wireless radio when they are within the transmission range of each other. To communicate, the ferry and the node must be able to detect each other. Moving along the ferry route, the ferry broadcasts HELLO messages periodically. Nodes simply listen to the broadcast channel. When a node receives a HELLO message, it replies with an ECHO message to the ferry. After identifying each other, they exchange messages until they are out of the transmission range or message exchange is finished.

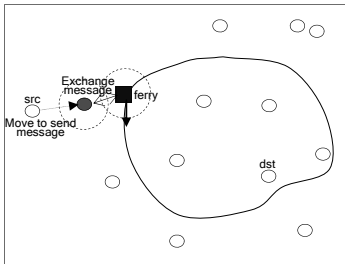


Fig. 1. The MF scheme

B. System Model

In this paper, we consider a MF system where a single ferry and a set of regular mobile nodes are deployed. A single ferry moves within a covered area over a well-known route but with possibly variable speed. Regular nodes are assumed to have assigned tasks that involve movement in the deployment area. They adjust their trajectory to meet up with the ferry only when they need to send or receive messages. Each node maintains a local message buffer which stores messages to be sent to other nodes. When the number of messages stored in the message buffer reaches a certain threshold or a node wants to check messages destined for it, the node adapts its trajectory to approach the ferry.

Nodes are associated with different amount of resources. The amount of resources which we call "capability level" is represented by an integer. The capability level decision factors can include battery life, memory, processing power, and movement speed. Different weights can be put on each factor in computing capability level.

The ferry periodically broadcasts its route, movement speed, and current location information via a long range radio. A node uses this information to compute the optimal meeting point on the ferry route so that it meets up with the ferry in the shortest time. A node not having this information moves to the nearest point on the ferry route from its current location and waits for the ferry at that point. The maximum waiting time of the node is bounded to the ferry cycle time T , i.e., L/v_{min} where L and v_{min} are the ferry route length and minimum movement speed, respectively.

C. Problem Definition

1) *Problem Statement*: The MF system relies only on the ferry to provide connectivity in the whole network. This is essentially subject to a single point of failure. In addition, regular nodes can coordinate with each other for message delivery by having one of themselves, rather than a specifically provisioned node, act as a ferry. Being a ferry involves extra movement and much more frequent message exchanges. Since regular nodes are usually assumed to have limited resources, they should perform the ferry role only for a fixed duration. Therefore, when the ferry fails or finishes its duty, a new ferry should replace the current one.

2) *Design Considerations*: We consider the following metrics in designing ferry replacement schemes.

- **Recovery latency** Recovery latency is the delay from when the current ferry stops to the time a new ferry takes the place of the failed ferry.
- **Overhead** The burden on the ferry or regular nodes required for ferry replacement should be moderate.
- **Failure probability** The probability that the ferry replacement will not succeed or last indefinitely should be minimized.

D. Overview of Proposed Ferry Replacement Schemes

In this paper, we propose two classes of ferry replacement protocols. The first approach is the successor designation

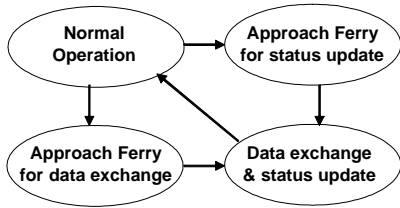


Fig. 2. State transition diagram of successor

scheme. In this scheme, the responsibility to select a replacement ferry is on the current ferry which designates one node as its successor against its failure. The successor periodically checks the ferry status. On detecting ferry failure, the successor takes over the failed ferry. The other approach is the distributed election scheme. In this scheme, the burden to select a ferry replacement is distributed among the regular nodes. They are potentially responsible for detecting ferry failure. If the ferry fails, nodes elect one among themselves as a ferry replacement without any centralized coordination.

Recall that we focus on abrupt (non-graceful) ferry failure scenarios. These two schemes can be used in a separate manner according to various ferry failure situations¹. It is also possible to use a hybrid scheme in which the distributed ferry election scheme is used as a backup in case the successor designation scheme fails (for example, because both the ferry and its designated successor failed simultaneously).

III. SUCCESSOR DESIGNATION PROTOCOL

A. Protocol Overview

The main idea of the designation replacement scheme is to have the current ferry select one node as a successor against its failure. Nodes that satisfy the minimum capability level to take the role of a ferry are potential ferry successors. The current ferry designates one successor from the set of potential successors before its failure. The designated successor is responsible for detecting ferry failure and becoming a new ferry after failure detection. In the designation scheme, only one node, i.e. the successor, is fully responsible for ferry failure detection and replacement. The recovery latency is the ferry failure detection time of the successor.

B. Successor Designation

The ferry designates one successor among regular nodes that it meets while moving along the ferry route. When the ferry first meets a node having the capability level higher than a certain threshold, it designates the node as its successor. Once a node is selected as the ferry successor, it should replace the current ferry once a ferry failure is detected. The ferry also periodically checks the status of its successor and designates a new successor if needed as will be discussed later.

C. Ferry Failure Detection

If all the nodes including the successor are stationary, the ferry's failure will be detected within T . However, if the

¹For graceful ferry stops, the successor designation scheme is the only scheme that makes sense and is relatively easy to deploy

successor is a mobile node and has a low message generation rate, it might not approach the ferry for a long time period since it does not collect enough messages in its buffer. If the ferry fails during this period, the failure detection latency may be long. For example, suppose the successor approaches the ferry route every $10T$, the average failure detection latency is $6T$. Furthermore, this latency is unbounded if there is no guarantee on the time interval between successor's visits to the ferry route.

A straightforward constraint on the successor's movement for faster or bounded ferry failure detection is to force the successor to be stationary around the ferry route. However, this will stop the normal operation of the successor. To reduce this overhead on the successor, we introduce a regulated status check movement.

Instead of always staying around the ferry route, the successor checks the ferry status regularly. After the successor meets the ferry, it goes back to normal operation with a timer set to be t_p . Once the timer expires, the successor moves towards the ferry for a status check. Besides the meeting with the ferry for status check, the successor may meet the ferry for data delivery as a normal node. After this type of data exchange meeting, the timer will be reset as in the status update ferry-successor meeting. The state diagram of the successor is illustrated in Figure 2. Theorem 1 shows that recovery latency is bounded to be the same magnitude of T after this constraint is introduced to the successor's movement.

Theorem 1: The designation scheme with periodic status check guarantees that the failure detection delay, t_d is bounded by $2t_p + T$.

Proof: Suppose the successor and the ferry meet at time t before the ferry fails. Then the ferry failure will be detected after the successor's next movement onto the route plus the T . t_d consists of three parts, the time t_p of normal operation, the time T on waiting to confirm the failure of the ferry as well as the time for the successor to move back to the route. At the time $t + t_p$, the successor must be within a circle with a radius of $v * t_p$ and centered at the last meeting point. In the worst case, the node will move back to the last meeting point within time $v * t_p / v = t_p$. Therefore, $t_d \leq t_p + t_p + T = 2t_p + T$. ■

D. Successor Redesignation

The successor may also stop working or fail due to resource exhaustion or abnormal reasons. The successor informs the ferry that it can not take its role during their meeting. Then, the ferry designates the first capable node to be a new successor in the same way presented in the Section III-B. The successor might stop working before informing the ferry. The ferry should detect this and redesignate another node. In other words, if the ferry does not see the successor for $(2t_p + T)$ since the last meeting, it considers this as an ungraceful successor departure and starts looking for another successor.

E. Performance Evaluation

1) *Simulation Environment:* In this section, we observe the performance of the designation scheme in terms of recovery

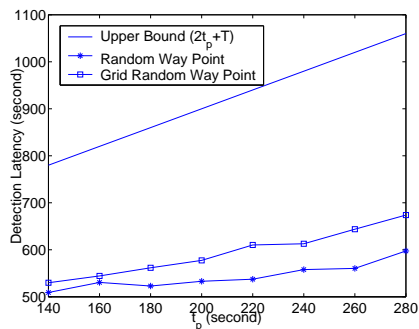


Fig. 3. Ferry failure detection latency

time, overhead, and failure probability. A single ferry and 40 nodes are deployed on a 5000m \times 5000m area. The ferry moves along the ferry route at the default speed of 20m/s. The ferry route follows a rectangle with (1250,1250) and (3750,3750) as diagonal points. Nodes move in the area according to the *random way point model* [13] with a default speed of 10m/s and pause time of 20s. Given the non-uniform distribution of this model in simulation [14], i.e., nodes are concentrated at the center of the area, we also use a *grid random way point model* in which the area is divided into four equal but smaller squares, nodes are limited in each square with random way point movement. By using this modification, we try to reduce the non-uniformity of nodes' distribution in the simulation.

The local message buffer size of a node is 20. Inter-generation time of a new message is exponentially distributed with an average of 100s. If the number of messages stored in the message buffer exceeds 7, the node adapts its trajectory toward the ferry. We set the ferry's long radio range to 1000m. Therefore, the successor is not able to detect the failure of ferry simply by "heart-beat" signals. The short range radio transmission range is 100m. Each node is randomly assigned capability level between 1 to 10 and only nodes with a capability higher than 3 can be chosen as successor.

2) *Detection Latency and Overhead*: We first investigate the effect of t_p over the detection latency and overhead. Here, we define the overhead as the percentage of time a node spends on approaching the ferry for either data exchange or status update. When the grid random way point mobility model is used, the nodes, including the successor, have less chance to appear around the center and statistically longer distance to the ferry route. The ferry failure detection latency is increased but still follows the upper bound.

Figure 3 shows the effect of t_p on failure detection latency, which is always bounded by $2t_p + T$. Although smaller t_p results in the shorter ferry failure detection latency, the overhead increases correspondingly. While normal nodes spend 15% of their operation time moving towards the ferry for message exchange, the successor has 2 to 3 times the overhead for the t_p from 140 sec to 280 sec due to its extra movement for status update as illustrated in Figure 4. Note that this overhead can be shared by having the ferry change its designated successor periodically.

Note that in an environment where the membership of nodes

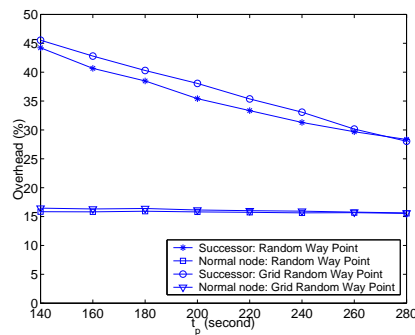


Fig. 4. Overhead comparison between the successor and regular nodes

changes often, i.e., nodes join in and depart from a MF system frequently, to guarantee a successful ferry replacement, we will use a distributed ferry election scheme as the backup which is to be described in the following section.

IV. DISTRIBUTED ELECTION PROTOCOL

A. Protocol Overview

The distributed election scheme requires nodes to collaborate and elect one node among themselves as a new ferry without any centralized coordination. Nodes that detect ferry failure make a local decision on how and when to participate in ferry election. They participate in ferry election as either a *ferry candidate* or an *elector*. The former can become a ferry and the latter votes for the candidate having the highest capability level.

To prohibit unqualified nodes from being a candidate and to keep the number of redundant ferry candidates low, we adopt a backoff delay scheme. Once ferry failure is detected, each node derives a backoff delay from its capability level such that the higher the capability level the shorter the backoff delay. Before becoming a candidate a node waits the backoff delay duration. While waiting, if it sees any other candidates, it stops waiting and becomes an elector. Instead of relying on random movement of mobile nodes, we exploit the non-random movement of the ferry. Candidates move along the pre-determined known ferry route, advertising themselves to electors that they meet. Electors elect one node among candidates as a ferry. Once they participate in a ferry election process, they should keep remain stationary around the ferry route until the end of the ferry election process.

B. Ferry Failure Detection and Election

The state transition diagram of a node and election algorithm are shown in Figure 5 and Figure 6, respectively. The election operations at a node is initiated by detecting ferry failure. We use HELLO messages as a means to determine the ferry status. When a node approaches the ferry route with the knowledge of the ferry location, the node should hear HELLO messages at their expected time. Otherwise, the node concludes that the ferry may have failed. Then it becomes a failure detector and enters an election mode. If a node goes to an arbitrary point on the ferry route without knowing the ferry location, the node should meet the ferry within the ferry cycle time. Otherwise, it enters the ferry election mode.

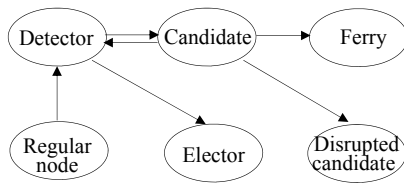


Fig. 5. State transition diagram of node

Once a node becomes a detector, it decides on whether it will participate in the ferry election as either a candidate or an elector. The detector calculates its backoff delay and waits the backoff delay before deciding its role. Backoff delay is a function of capability level. It is inversely proportional to capability level such that the higher capability level the detector has the shorter backoff delay it should wait. The maximum backoff delay is defined as the delay which a node having the lowest capability level should wait. The backoff delay of a node having the highest capability level is assumed to be 0. Then backoff delay linearly increases from 0 up to the maximum delay as capability level decreases. Once it calculates the backoff delay, the detector schedules the backoff timer. When the timer expires, it changes its state into a candidate. Then it moves along the ferry route advertising itself to detectors or electors that it meets. On the other hand, if a detector sees any other candidates while waiting for its backoff delay to expire, it stops the backoff timer and changes its state into an elector.

If the candidate meets a detector waiting for its backoff delay to expire, it sends the detector its *advertisement* message. Then the detector becomes an elector. When it meets an elector which is already changed from a detector by another candidate, it also sends its *advertisement* message. An *advertisement* message contains $\langle AD, candidate_id, capability_level, failure_detection_time \rangle$.

All electors maintain a local variable *highest_candidate*. This variable keeps the *advertisement* of the candidate having the highest capability level. The election criteria can be chosen according to an election policy. Each time an elector meets a new candidate, it updates its *highest_candidate* according to the result of a comparison between the *highest_candidate* and the newly received *advertisement*. If the newly met candidate has a higher capability level than the candidate of the *highest_candidate*, the elector sends a *WIN* message to the candidate and updates the *highest_candidate* with the newly received *advertisement*. Otherwise it sends a *LOSE* message with the advertisement which is stored in its *highest_candidate*. A *WIN* and a *LOSE* message contain $\langle WIN, elector_id \rangle$ and $\langle LOSE, elector_id, highest_candidate \rangle$, respectively.

On hearing a *WIN* message, a candidate continues to move along the ferry route. On hearing a *LOSE* message, it resigns as a candidate and becomes a disrupted candidate. It sets its *highest_candidate* to the value which was contained in the *LOSE* message. Then it goes back to its original location. This disrupted candidate participates in ferry election like any elector nodes. When a candidate encounters another candidate during touring, the two candidates directly exchange and

```

state: which state node is in;
round: which round candidate is in;
highest_candidate: highest_candidate of elector;

advertisement: advertisement message;
WIN: WIN message;
LOSE: LOSE message;

REGULAR NODE:
On detecting failure,
state:=DETECTOR;

DETECTOR:
Calculate backoff delay based on capability level;
while backoff delay not expired do
  On receiving advertisement from candidate,
    state:=ELECTOR;
end while
state:=CANDIDATE;

CANDIDATE:
round:=first;
Start to move along ferry route;
while TRUE do
  On meeting elector,
    if Receive WIN then
      if round:=second then
        state:=FERRY;
      endif
    else if receive LOSE then
      highest_candidate:=highest_candidate of
        elector;
      state:=DISRUPTED CANDIDATE;
    end if
  On returning to initial position,
    if meet no elector then
      state:=DETECTOR;
    endif
    round:=second;
end while

ELECTOR:
highest_candidate:=null;
while ferry not announced do
  On receiving advertisement from candidate
    compare (advertisement, highest_candidate);
  if candidate WINS then
    highest_candidate:=advertisement;
    Send WIN;
  else
    Send LOSE;
  end if
end while

DISRUPTED CANDIDATE:
Go to original location;
Participate in election as elector;

FERRY:
Declare as ferry and take ferry role;
  
```

Fig. 6. Election operations at node

compare their advertisements. When the candidate sees a disrupted candidate, it compares its advertisement with the the *highest_candidate* of the disrupted candidate.

A candidate finishes its first round when it returns to its initial location. Only candidates that defeat other candidates survive at the end of the first round. However, the fact that a candidate survives in the first round does not guarantee the candidate is elected as a ferry. The candidate might not have had a chance to contend with candidates that visit all electors at a later time than it did. This is because candidates basically rely on advertisements received from electors to resolve contention unless they encounter each other directly. A survived candidate in the first round goes to the second round by revisiting electors. When a candidate visits the first elector for the second time, the candidate can decide whether it is a ferry or not. This is because the elector has seen advertisements from all candidates. If the first elector's *highest_candidate* still stores its advertisement, the candidate considers itself a ferry. The elected node revisits the rest of the electors announcing that election is terminated by its being a

ferry. At the same time, it carries the role of a ferry. Electors change their state to a regular node when they hear the election termination announcement from the new ferry.

To give a bound on election participation, nodes which detect failure after one ferry cycle time since the first detector detects failure can not be a candidate. A node resign a candidate if it hears from an elector that other candidate having earlier failure detection time than that of itself passed by the elector. It can happen that all detectors are running for a ferry, thus, they have no chance to meet any electors. Without getting advertisements from electors, candidates can not compare with each other. Therefore, if a candidate meets no elector during its first round, it goes back to a detector and waits for its backoff delay before making the first round again.

C. Correctness of the Protocol

Theorem 2: The protocol ensures that exactly one node enters a ferry state after no more than two rounds from the time the first node detects ferry failure.

In the proof process, we assume that participating nodes do not fail while election is on-going and that there is at least one elector. We will discuss the case that there is no elector forever in the next section. The proof of this theorem is straightforward when there is only one candidate. We use induction to finish the proof when there are multiple candidates. For space limitation, a more detail proof can be found in the full version of the paper [15].

D. Performance Evaluation

We evaluate the performance of the election protocol through simulation.

1) *Metrics and Environment:* The first metric is recovery latency. There are two measures of recovery latency in the election scheme. One is defined as the elapsed time from the time the ferry stops to the time one node is elected and starts to take the place of the stopped ferry (hereafter, *election latency*). The other is defined as the elapsed time from the time the ferry stops to the time all participating nodes hear the election termination announcement and become aware in the newly elected ferry (hereafter, *termination latency*).

To quantify the overhead consumed for election, we use the number of disrupted nodes and the average disruption time per disrupted node. A disrupted node is one which once was a ferry candidate but was not elected. The number of such candidates should be minimized from the viewpoint of resource consumption. As our overhead metric, we use the ratio of the number of disrupted nodes to the total number of participating nodes. The disrupted time of a disrupted node is the elapsed time from when the node becomes a candidate to when it is disrupted.

The final metric is the election failure probability. Our election protocol is a distributed and randomized one where nodes independently detect ferry failure and make local decisions on how and when to participate in election.

We use the same simulation environment as in Section III-E through all simulations unless specified otherwise. Through

all simulations, we use only the random way point model since mobility model does not make a big difference on our replacement schemes as we learned from the simulation result of the previous section. Each simulation starts at 0s and the ferry stops at 1000 sec. Each one was run for 10000s.

2) *Effect of the Number of Deployed Nodes:* In this experiment, we observe the effect of the number of deployed nodes on the election performance. The larger number of deployed nodes allows more nodes to detect ferry failure. This will consequently cause earlier ferry failure detection and more electors as well as more candidates. In our protocol, densely located electors reduce the time it takes candidates to meet the first elector for the second time, which is the final decision time.

Figure 7 shows the simulation result with the different number of deployed nodes. As expected, election latency decreases as the number of deployed nodes increases. Meanwhile termination latency is almost the same regardless of the number of deployed nodes as two times of L/v_n where L and v_n are the ferry route length and the movement speed of an elected node, respectively. The ratio of the number of disrupted nodes to the number of deployed nodes decreases as the number of deployed nodes increases. The average disruption time per disrupted node also decreases since more densely located electors allow candidates to know comparison results with each other earlier.

3) *Effect of Backoff Delay:* Backoff delay performs a primary role to control the number of candidates. To see the effect of backoff delay on the election performance, we run simulations with different maximum backoff delays. A maximum backoff delay is defined as the delay for which a node having the lowest capability level, i.e. 1 in our simulations, should wait before becoming a candidate. We set the backoff delay of the highest capability level, here 10, to 0. Backoff delay linearly increases from 0 up to the maximum delay as capability level decreases. We use the ferry cycle time as the basic unit to sample the maximum backoff delay.

As shown in Figure 8, backoff delay does not have a big effect on election and termination latency although both slightly increase. This is due to the fact that election and termination latency are dominated by the failure detection time and movement speed of an elected node. In our simulation, we let a node having the highest capability level among candidates be elected as a ferry. However, the backoff delay of a node having a higher capability level is less affected by the maximum backoff delay than one of a node having a relatively lower capability level. Of course, the longer backoff delay incurs the smaller number of disrupted nodes although the average disruption time per disrupted node increases.

4) *Election Failure Probability:* Our protocol requires nodes to locally decide on how and when to participate in ferry election. The worst scenario is where all nodes are running for a ferry and there are no electors. In this case, we never succeed to elect a ferry. The condition under which this will happen is (a) all nodes have the same backoff delay and movement speed, and (b) before a candidate reaches the next detector

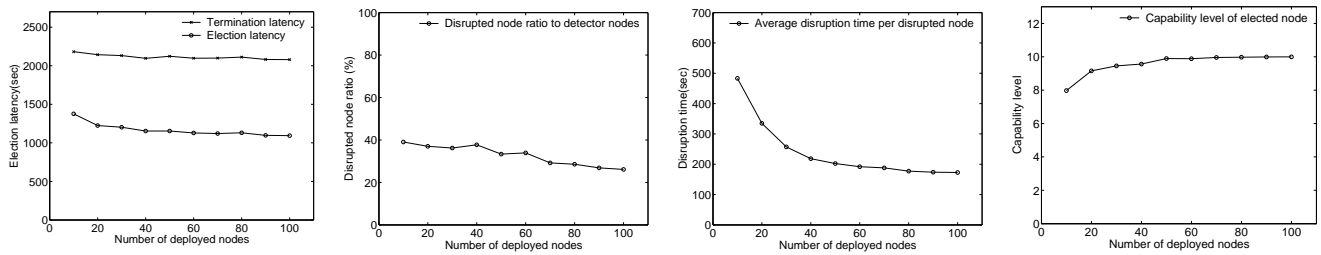


Fig. 7. Election performance as a function of the number of deployed nodes

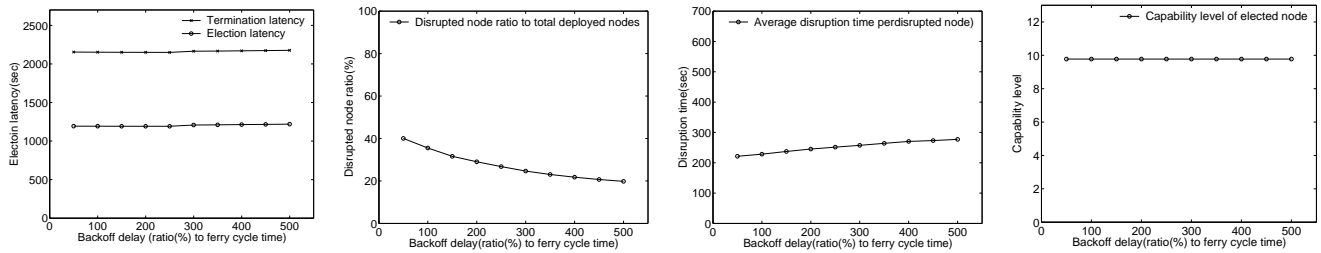


Fig. 8. Election performance as a function of the maximum backoff delay

node, the detector becomes a candidate.

With our simulation settings, in all simulations performed in the previous subsections, only in simulations with less than 40 deployed nodes and the maximum backoff delay of 100% the of ferry cycle time, less than 5% of failure probability was shown. This is because the capability level and movement speed of nodes are randomly assigned. If capability level and movement speed happen to be highly correlated among nodes, failure probability will increase. Failure probability can be reduced by adjusting backoff delay range or by using exponential backoff delaying scheme used for collision resolution in the Ethernet.

V. CONCLUDING REMARKS

In this work, we discuss ferry replacement protocols for a message ferrying system. In the MF system, a set of special nodes called ferries are responsible for carrying messages for all nodes in the networks. This is vulnerable to a single point of failure. To make the MF system robust, we proposed two classes of ferry replacement protocols. In the designation scheme, the ferry designates its successor as a ferry replacement. The successor is responsible for periodically checking the ferry status and takes the place of the failed ferry in case of ferry failure. In this scheme, there is a tradeoff between recovery latency and overhead on a successor, which can be adjusted by the status check period. We also propose a distributed election approach in which nodes collaborate and elect one node among them as a ferry replacement without any centralized coordination. The overhead for ferry replacement in this scheme is distributed among regular nodes. There is also a tradeoff between recovery latency and overhead on regular nodes. Recovery latency can be reduced by the larger number of participating nodes and a shorter backoff delay. However, this makes the overhead on regular nodes heavier.

We envision that the distributed scheme can be used as a

backup if the designation scheme fails e.g., when the successor membership changes frequently.

REFERENCES

- [1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Computer Networks (Elsevier)*, vol. 38, pp. 393–422, 2002.
- [2] K. Fall, "A delay-tolerant network architecture for challenged internets," in *ACM SIGCOMM*, August 2003, pp. 27–34.
- [3] E. M. Royer and C.-K. Toh, "A review of current routing protocols for ad-hoc mobile wireless networks," *IEEE Personal Communications*, pp. 46–55, 1999.
- [4] J. Davis, A. Fagg, and B. Levin, "Wearable computers as packet transport mechanisms in highly-partitioned ad-hoc networks," in *International Symposium on Wearable Computing*, October 2001, pp. 141–148.
- [5] Q. Li and D. Rus, "Sending messages to mobile users in disconnected ad-hoc wireless networks," in *MobiCom*, August 2000, pp. 44–55.
- [6] D. Nain, N. Petigara, and H. Balakrishnan, "Integrated routing and storage for messaging applications in mobile ad hoc networks," in *WiOpt: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, 2003.
- [7] A. Vahdat and D. Becker, "Epidemic routing for partially-connected ad hoc networks," Duke University, Tech. Rep. CS-200006, 2000.
- [8] W. Zhao and M. Ammar, "Message ferrying: Proactive routing in highly-partitioned wireless ad hoc networks," in *IEEE Workshop on Future Trends in Distributed Computing Systems*, May 2003.
- [9] S. Jain, K. Fall, and R. Patra, "Routing in a delay tolerant network," to appear at Sigcomm 2004.
- [10] W. Zhao, M. Ammar, and E. Zegura, "A message ferrying approach for data delivery in sparse mobile ad hoc networks," in *Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*, May 2004.
- [11] <http://web.media.mit.edu/~amir/daknet/>.
- [12] J. Sterbenz, R. Krishnan, R. Hain, A. W. Jackson, D. Levin, R. Ramnathan, and J. Zao, "Survivable mobile wireless networks: Issues, challenges, and research directions," in *ACM Workshop on Wireless Security (WiSe)*, September 2002.
- [13] D. Johnson and D. Maltz, "Dynamic source routing in ad-hoc wireless networks," in *ACM SIGCOMM*, August 1996.
- [14] C. Bettstetter, "Mobility modeling in wireless networks: Categorization, smooth movement, and border effects," *ACM Mobile Computing and Communications Review*, vol. 5, pp. 55–67, 2001.
- [15] J. Yang, Y. Chen, M. Ammar, and C.-K. Lee, "Ferry replacement protocols in sparse manet message ferrying systems," College of Computing, Georgia Tech, Tech. Rep., August 2004, (http://www.cc.gatech.edu/~jeonghwa/ferry_replacement.pdf).