Look-Ahead Routing and Message Scheduling in Delay-Tolerant Networks

Yi Xian, Chin-Tser Huang
Dept. of Computer Science and Engineering
University of South Carolina
{xian, huangct}@cse.sc.edu

Jorge Cobb
Department of Computer Science
The University of Texas at Dallas
cobb@utdallas.edu

Abstract— **Routing is one of the most challenging development issues in delay-tolerant networks (DTNs) because of lack of continuous connection. Existing routing schemes for DTNs provide best effort service, but are unable to optimize QoS and support message priority. In this paper, we present a Look-Ahead Routing and Message Scheduling approach (ALARMS) which exploits more accurate knowledge about various parameters regarding routing to achieve better QoS in the DTN. We assume a variation of the well-known ferry model, in which there are ferry nodes moving along pre-defined routes to exchange messages with the gateway node of each region on the route and also pass to the gateway nodes look-ahead routing information about when it will arrive at each gateway node on the route in the next two rounds and how long it will stay. The gateway nodes use this information to estimate the delivery delay of each message when being delivered by different ferries, and schedule the message to be delivered by the ferry which arrives earliest at the destination. Simulation results show that ALARMS outperforms three existing routing protocols: epidemic routing, spray-and-wait, and spray-and-focus, in terms of delay time, delivery ratio, and overhead. We also discuss three enhancement strategies on ALARMS and how ALARMS can support message priority.**

Index Terms—**Delay-Tolerant Networks, Quality of Service, Look-Ahead Routing, Message Scheduling.**

I. INTRODUCTION

Delay Tolerant Networks (DTN) is one emerging type of networks whose distinguishing characteristic is that communications between two nodes in a DTN take prolonged period of time because of lack of continuous connection. DTNs were initially designed to operate over extreme distances encountered in space communications where the long latency in communications inevitably causes TCP connections in the network to break down. As the TCP communication breaks down in a DTN, a Bundle Protocol [15] can be used to handle the underlying communication with the remote party using node-to-node custody transfers. This protocol also allows the DTN to span multiple heterogeneous networks and allows the DTN protocols to be separated from the naming and addressing schemes used in the underlying networks.

However, the discontinuous connection problem due to the cosmic distances can also occur over shorter terrestrial distances due to lack of infrastructure. For example, many people living in remote rural areas and in developing countries do not have access to the Internet [8]. To realize data communication in such cases, it requires hardware to store large amounts of data, and also requires special media to transfer data between the nodes. Therefore, in a terrestrial context, a DTN can be appropriately regarded as a group of highly disconnected networks/regions where the communication environment is characterized by high error rates, long delays and sporadic connectivity with the outside world.

An approach to transfer the data between the disconnected network regions is by making use of message ferry [20, 21] which travels on arbitrary paths between the regions and has a data storage buffer which can be used to keep messages. Each region has a fixed gateway node which handles the communication needs of incoming and outgoing traffic for the hosts within the region. When a ferry arrives at a region, it contacts the gateway node, and during the contact time it will unload to the gateway the messages that are destined to this region, and load from the gateway the messages that this region wants to send to other regions. To reduce the latency, multiple ferries can be deployed in the network. Figure 1 illustrates the use of multiple message ferries.

However, this message ferry approach has two main constraints. First, there is no guarantee of the performance of the network if the disconnected regions have no information of the ferry route, schedule and capacity. Second, this straightforward approach does not support message prioritization [5], which is characterized by different importance levels of messages and is usually handled with differentiated scheduling and delivery of messages. Although the communication quality of DTNs is far from ideal, QoS and message prioritization are still desirable in many cases, especially when the DTN is used in critical missions such as rescue tasks after natural disasters and emergency response in face of terrorist attacks.

In this paper, we propose to use a Look-Ahead Routing and Message Scheduling (ALARMS) approach to optimize the performance in a DTN. The main motivation of our work is to improve the DTN performance via smart forwarding of routing information to the disconnected regions and informed scheduling of data packets by the gateway nodes in the network. In our model we assume that the DTN has multiple ferries operating on predefined routes within the network. Each time when a ferry arrives at a gateway node on its route, the ferry will pass to the gateway node the look-ahead routing information about when it is estimated to arrive at each gateway

node on the route in the *next two rounds* and how long it will stay at each gateway node. Thus, a gateway node can use the look-ahead routing information collected from the ferries that have this gateway node on their route to decide for each message which ferry is the best choice to deliver it and schedule the messages in the queue according to their destination, time-to-live and priority. Note that our design injects some determinism into the DTN in that the ferries operate on predefined routes and try to stay with the estimated schedule. However, we believe this is desirable and beneficial when the DTN is used in critical missions and needs to have optimized QoS. One of our objectives in this research is to assess and demonstrate the benefits of this partially deterministic design. To achieve this goal, we use a simulation model based on a variation of a DTN simulation tool called ONE [22] to evaluate and compare ALARMS and other existing routing policies for DTNs.

The remainder of this paper is organized as follows. In Section II, we give an overview on related works. In Section III, we formulate the problem and introduce our scheme of look-ahead routing and message scheduling (ALARMS). In Section IV, we discuss our simulation setup and experiment results. In Section V, we discuss three potential enhancement strategies to provide better QoS. In Section VI, we discuss how ALARMS can support message prioritization. Finally, we conclude and discuss future works in Section VII.

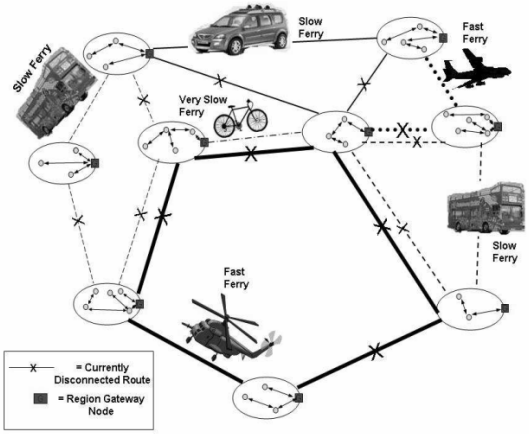

Fig. 1: A simple DTN with different types of message ferries in fixed routes between disconnected network regions. A link with X represents that there is currently no ferry on that link to transfer data.

II. Related Works

The research and development of DTNs has attracted significant attention in recent years. With the development and work regarding architectural and design principles related to the DTNs being now maintained by the DTN research group (DTNRG), there are publications addressing almost every aspect of DTNs.

Routing has been a major problem in DTNs due to unstable end-to-end paths, high latency, low data rate, long queuing times, limited resources and other problems inherent in DTNs. Routing protocols for traditional ad hoc networks, such as Optimized Link State Routing (OLSR) [7] and Ad hoc On-Demand Distance Vector (AODV) [14], fail in DTNs because end-to-end connectivity is not always available [12]. Therefore, the first issue to be concerned about is what medium can be used for carrying and routing messages. The exploration began with Fall et al. [9], in which the authors discuss node-to-node custody transfers using store-and-forward methods at intermediate gateways. Then in [20, 21], the authors propose to use message ferries to transfer the data between disconnected network regions.

Afterward, several solutions on routing in DTNs have been proposed. In Jain et al. [10], the authors discuss a scheme of oracle-based routing. Although the scheme is not practical because it is very hard to realize the proposed oracle in reality, the work provides a theoretical framework for developing DTN routing schemes. Indeed, our scheme can be viewed in one way as providing a feasible and up-to-date version of "oracle" to the gateway nodes. In Burns et al. [4], a routing algorithm with robust delivery rate is described. In Leguay et al. [12], the authors define a generic routing scheme for DTNs using a high-dimensional Euclidean space constructed upon nodes' mobility patterns. In Lindgren et al. [13], the authors propose a probabilistic routing protocol (PROPHET) which uses the probability of non-random movement of nodes for routing the messages using history of node encounters and then compare it to the epidemic method of routing. In Burgess et al. [3], the authors proposed a scheme called MaxProp, which is a routing protocol based on delivery likelihood to peers, packet hop counts, and list of intermediaries in data, but the method also uses network-wide acknowledgments which wastes resources. In Spyropoulos et al. [16], the authors propose a scheme called Spray-and-Wait, which bounds the number of copies replicated per message in order to reduce the overhead caused by arbitrary replication. Later in [17, 18], the same authors propose another controlled replication scheme, called Spray-and-Focus, which has increased intelligence compared to existing Spray-and-Wait schemes in that it can recognize and take advantage of potential opportunities to forward a message "closer" to its destination, according to an appropriately designed utility function. To sum up, each of the above schemes solves the problem of routing in DTNs partially, but none of them addresses the issue of QoS guarantees in DTNs.

In our previous work [6], we propose a scheme for performance improvement in DTNs using forward routing information and usage of Dijkstra's shortest path algorithm. This paper is an extension to [6] where we run simulations to test the various parameters in the DTN and to evaluate the effectiveness of our scheme against other proposed routing schemes. To conserve resources, our scheme does not use network-wide acknowledgements, and does not maintain any list of intermediaries.

It is instructive to note that in [1, 11], the authors classify the routing schemes for DTN into two categories: replication and forwarding. Replication routing protocols replicate a message and send out multiple copies of it, whereas forwarding routing protocols forward only a single copy of the message. Apparently, our scheme belongs to the forwarding category because a gateway node will forward only a single copy of each message to the best ferry chosen for the message. Although in

one enhancement strategy we propose to let the gateway node forward one more copy to the second best ferry, it is still consistent with the spirit of the forwarding category.

III. Look-Ahead Routing and Message Scheduling

In this section, we present our Look-Ahead Routing and Message Scheduling (ALARMS) scheme. We first introduce how a ferry provides look-ahead routing information to a gateway node and exchanges data messages with a gateway node. Then, we explain how a gateway node makes use of the look-ahead routing information received from the ferries to schedule its outgoing messages.

The interaction between a ferry and a gateway node during their contact time is illustrated in Figure 2. First, the ferry will pass look-ahead routing information to the gateway node. The look-ahead routing information includes a list of the gateway nodes it will visit on its route, and the time when the ferry arrives at each gateway node in the next two rounds. Note that we let the ferry give out its schedule for the next two rounds instead of the next round so that the gateway nodes can calculate and decide the best ferry for each message in advance, and not give out its schedule beyond the next two rounds so that the ferry can adjust its schedule according to the recent traffic demand at different gateway nodes (this is an enhancement strategy discussed in Section V.C). Next, the ferry will deliver the messages of which the gateway node is the destination, and the gateway node will transfer to the ferry the messages which the ferry is chosen as the best ferry to deliver. The ferry leaves the gateway node after the contact time is over.

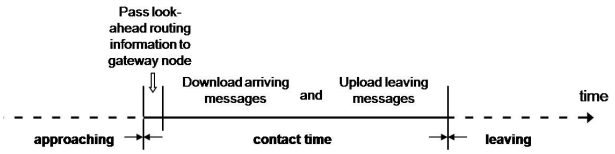

Fig. 2: The interaction between a ferry and a gateway node during their contact time.

We assume that the ferries have ample storage buffers since memory is relatively cheap nowadays. We also assume that the ferries and gateway nodes have multi radios (e.g., [2]) to allow simultaneous download and upload. However, the contact time between a ferry and a gateway node is set to be limited because the ferry has to follow its schedule to travel to the next region. Therefore, when a ferry visits a gateway node, the total throughput is constrained by the contact time, not by the size of the ferry's storage buffer.

A gateway node can schedule its outgoing messages according to the look-ahead routing information received from the ferries. When a new message is generated at a region, the gateway node at the region will pick the best ferry for delivering this message and schedule the message in the queue corresponding to the chosen ferry according to the following factors: the time-to-live of the message, the destination of the message, each eligible ferry's arrival time at the destination in the next round, and the creation time of the message. A message created earlier is considered to have higher priority than a message created later and will be scheduled prior to the later message in the queue. Since the processing of a queue is first in, first out, messages will be transferred from the gateway node to the ferry and be delivered from the ferry to the destination node according to the order of their creation time.

The algorithms used by ALARMS can be presented with the following pseudocode:

Global setup:
m Gateway nodes $G_1, G_2, \ldots, G_m$
n Ferry nodes $F_1, F_2, \ldots, F_n$

Data structure in each gateway G_i:
An array of queues, one for each ferry that has G_i on its route

Data structure in each ferry F_j:
An array of queues, one for each gateway on F_j's route

Procedure Gateway(i)_message_scheduling
// executed when a new message is generated at G_i
Input: route and schedule (for the next two rounds) of each ferry that has G_i on its route; a new message, with its size, destination, and TTL
Output: the ferry chosen to deliver the message; the queue corresponding to the chosen ferry is updated with the message added into the queue
Begin
1: From the input, find all the eligible ferries that have the message's destination on their route and will arrive at the destination before the message's deadline; if no eligible ferry is found, return null
2: Choose the ferry that will arrive earliest at the destination
3: Add the message into the queue corresponding to the chosen ferry according to the message's creation time
End

Procedure Gateway(i)_contact_with_ferry(j)_in_round(t)
Input: gateway G_i's queue corresponding to ferry F_j
Output: the queue updated with the transferred messages removed from the queue
Begin
1: while contact time is not over:
receive from ferry F_j messages destined to G_i
2: transfer the messages to ferry F_j according to their order in the queue for F_j
End

Procedure Ferry(j)_contact_with_Gateway(i)_in_round(t)
Input: ferry F_j's array of queues
Output: the queue corresponding to gateway G_i updated with all delivered messages removed from the queue; the queues corresponding to other gateways updated with the messages transferred by gateway i inserted
Begin
1: F_j sends its schedule for rounds t+1 and t+2 to G_i
2: while contact time is not over:
transfer the messages to G_i according to their order in the queue

3: receive from G_i the messages to be delivered to a destination on F_j's route; schedule each message into the queue corresponding to the destination gateway according to the message's creation time

4: calculate the ratio $dr[i, t]$ of the number of delivered messages to the number of all messages for G_i in this round t

End

IV. Simulation And Evaluation

In order to evaluate the performance and benefits of ALARMS, we have modified the Opportunistic Network Environment (ONE) simulator [21] to simulate an initial model of our proposed schemes. In this section, we present the details about our simulation model and environment setup, and discuss the meaning of the experimental results.

A. Simulation Model and Evaluation Metrics

Our simulation is based on the ONE simulator with necessary extensions. The ONE simulator is a powerful tool for generating mobility traces, running DTN messaging simulations with different routing protocols, and visualizing the simulations. In order to simulate the gateway nodes and ferry nodes in our ALARMS scheme, we extended the ONE simulator by adding two new routing classes: a GatewayRouter class and a FerryRouter class. The GatewayRouter class models the gateway node at each region, whose responsibility is to determine which ferry to choose for a particular message by estimating ferries' schedules in advance, and transfer a single copy of the message to the chosen ferry, instead of sending multiple copies of the message to all the ferries which visit this gateway node. Moreover, one main characteristic of the GatewayRouter class is that its speed is set to 0, which means it is stationary and never move.

The estimation of ferries' schedules is implemented as follows. The GatewayRouter has one queue for each ferry whose route traverses this GatewayRouter. When a new message is created, the GatewayRouter uses the schedule information it receives from all ferries that have it on their route to choose the ferry that arrives earliest at the message's destination as the best ferry for the message. Then, the GatewayRouter appends this new message to the end of the queue corresponding to the chosen ferry. When one ferry arrives at the GatewayRouter, the GatewayRouter checks the queue corresponding to this ferry and transfers the messages in the queue to this ferry.

The FerryRouter class is extended from the PassiveRouter class in the ONE simulator. The FerryRouter and the PassiveRouter are similar in that their main function is transferring messages to final recipients. However, the FerryRouter is more advanced in that it supports exchanging and interpretation of informational messages. When a new connection with gateway nodes is up, the FerryRouter of this ferry node calculates its schedule of the next two rounds according to its route, current speed and contact time. The FerryRouter then generates a schedule message containing its schedule information and send it to each gateway node on its route. Besides, the FerryRouter can have all the messages sorted in the order of their creation time before forwarding them to the final destination nodes, because each message is inserted to the queue corresponding to the destination gateway according to its creation time.

We also add a new movement class for FerryRouter, called the FerryMovement class, which extends from the MapRouteMovement class. The ferry nodes follow the pre-defined map to travel around. Besides, using FerryMovement model, the ferry nodes can dynamically adapt their activeness status (i.e., active or inactive, which is used in one enhancement strategy discussed in Section V.B), the contact time and the speed on demand.

We compare our scheme with three other routing models that are provided in the ONE simulator and are widely used in DTN research: Epidemic [19], Spray-and-Wait [16], and Spray-and-Focus [17, 18]. Epidemic routing is a flooding-based routing protocol, which continuously replicates and transfers each unexpired message to every newly encountered node that does not have a copy of the message. Spray-and-Wait attempts to gain the delivery ratio benefits of replication-based routing as well as the low resource utilization of forwarding-based routing by transferring a copy of the message to the first n encountered nodes only. Spray-and-Focus, an extended version of Spray-and-Wait, uses last encounter timers to recognize and take advantage of potential opportunities to forward a message "closer" to its destination. We choose the number of copies (n) to be 5 for Spray-and-Wait and 3 for Spray-and-Focus. These values are calculated according to their original papers in order to get best performance.

In all scenarios considered in the paper, we use three measurement metrics of the ONE simulator for evaluation purpose: *delivery ratio*, *overall latency* and *overhead ratio*. The delivery ratio measures the ratio of the number of delivered messages to the number of all messages. The overall latency measures the latency from the creation time to the delivery time. The overhead ratio is the average number of forwarded copies per message.

B. Environment Setup and Simulation Results

Next, we introduce the environment setup of our simulations. In the simulations, we start with 13 gateway nodes and 8 ferry nodes as shown in the screenshot in Fig. 3. Each ferry travels on a different route and each route contains 6 to 8 gateway nodes such that each gateway node has at least 4 ferry nodes that can be considered as a candidate for delivering a message to any other gateway node. This setup is designed to test the performance of the look-ahead routing feature of ALARMS. Every simulation runs for 12,000 seconds of simulation time but has different total number of messages, including 500, 1000, 1500, 2000, 3000, 5000, 10000, 15000, 20000, 25000, and 30000. This is aimed to compare the performance of the four routing schemes under different traffic loads. For each number of messages tested in the simulations, we generate a message creation schedule that is followed by the gateway nodes to create messages. The size of the messages is randomized but ranges from 100K to 1M bytes. Each ferry node travels along a

pre-defined route to exchange messages with the gateway nodes on the route. To test the effectiveness of our message scheduling scheme, we define a different contact time and movement speed for every ferry.

Figure 4 shows the delivery ratio of the four routing schemes under different traffic loads. When the traffic load is low (less than 1000 messages), the delivery ratio of ALARMS is 100%, and the delivery ratios of all three routing schemes are all above 90%. As the amount of messages increases, the delivery ratios of all four models drop due to the limited contact time and bandwidth. However, the delivery ratio of ALARMS is always higher than the delivery ratios of the other three schemes. When the message number is greater than 5,000, the delivery ratio of ALARMS excels the other three schemes by 5-15%. ALARMS's superiority on delivery ratio can be attributed to the fact that ALARMS forwards only one copy of each message and thus saves bandwidth to forward more messages.

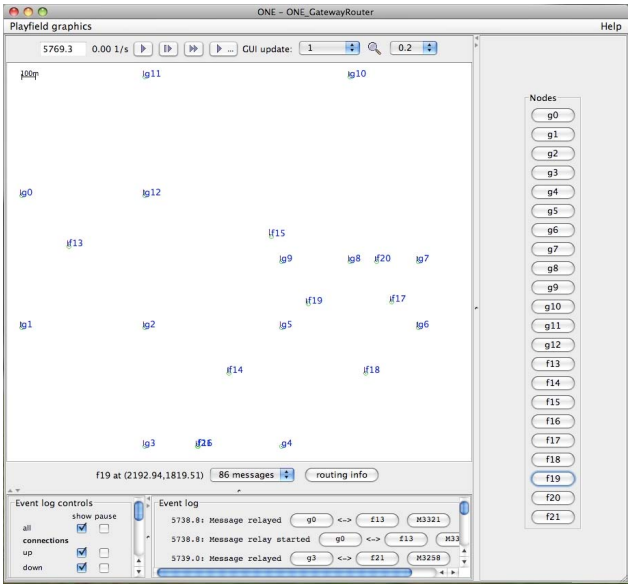

Fig. 3: A screenshot of the simulation setup.

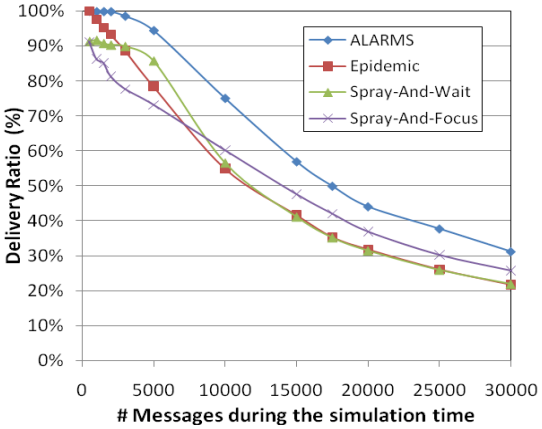

Fig. 4: Delivery ratio for different total number of messages.

Another important metric is the overall latency. For those messages that cannot be delivered to their destination before their TTL expires, we set the latency to be the message TTL. Figure 5 displays the overall latency under different traffic loads. As we can see, ALARMS always has the shortest latency since the gateway routers choose the best ferry for every message in advance.

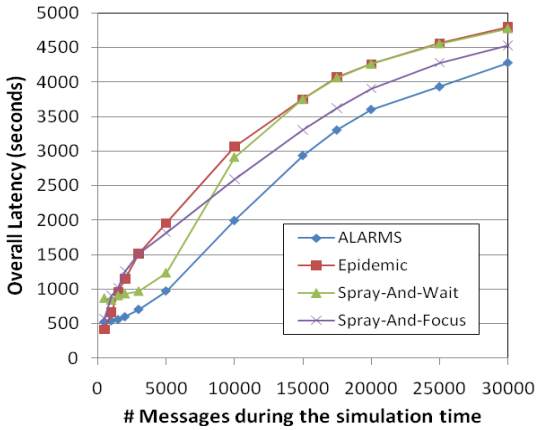

Fig. 5: Overall latency for different total number of messages.

Then, we compare the overhead ratio of these routing schemes. Figure 6 shows that ALARMS has the least overhead ratio and the smoothest curve among the four schemes, because ALARMS always forwards only one copy of each message. The overhead ratio of Epidemic, Spray-and-Wait, and Spray-and-Focus also stabilizes when the total message number is beyond 15,000 because they cannot transfer any more messages during the limited contact time. However, their overhead ratios remain higher than ALARMS's.

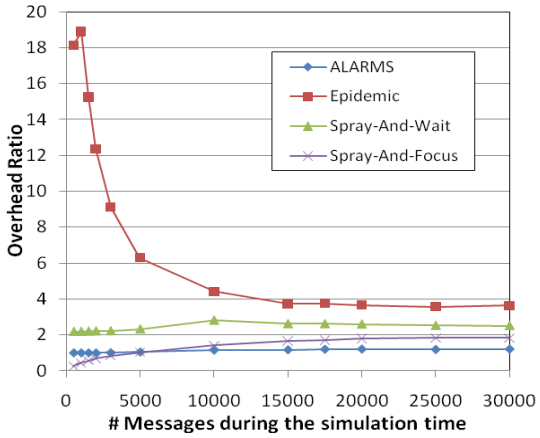

Fig. 6: Overhead ratio for different total number of messages.

V. Enhancement Strategies For Better QoS

As shown in Section IV, when the traffic load of the DTN is too high, the delivery ratio drops fast due to the limited bandwidth and contact time. In this section, we propose three potential enhancement strategies for ALARMS and use simulations to evaluate their benefits. Due to page limit, we will only analyze their delivery ratios.

A. *Forwarding Each Message to More Than One Ferry*

In ALARMS, a gateway node forwards only one copy of every message to the chosen best ferry. The advantage of this approach is to save bandwidth and lower the overhead ratio. However, if the traffic load is high, some ferries may not be able to forward all the messages to their destination gateway nodes during the limited contact time. Those messages that have not been forwarded to the destination gateway node during the contact time must wait until next time the ferry node visits this gateway node again, and by that time some messages may have expired already and will be discarded. In this case, forwarding every message to more than one best ferry (i.e., to the second best ferry, the third best ferry, and so on) may allow more messages to be delivered to their destination within their TTL, at the price of higher overhead. Note that this strategy is different from Spray-and-Wait in that Spray-and-Wait will forward a copy of the message to the first *n* encountered ferries, which may not be the best ferries for the message.

We tested forwarding one, two, three and four copies of every message (to the best one, best two, best three and best four ferry nodes respectively) using different number of messages, from 5,000 to 30,000. Figure 7 displays the average delivery ratio. The results indicate that for all different total message numbers, forwarding to two best ferries increase the delivery ratio by around 2%, but forwarding to three or four ferries cannot improve the delivery ratio any further. This shows us that for the messages that cannot be delivered by the best ferry during the first contact time with the destination gateway, forwarding multiple copies of these messages have the potential to increase the possibility of allowing them to be delivered by the second, the third or the fourth best ferry node before their TTL expires. However, these messages only account for a low percentage (~2%) of all messages, and many other messages are still undelivered due to the constraint of limited contact time.

On the other hand, forwarding multiple copies of every message will cost more transmission time, which in turn will prevent more new messages from being forwarded, especially when the traffic load is very high. Therefore this strategy is appropriate only when the overhead it causes is still within tolerable range.

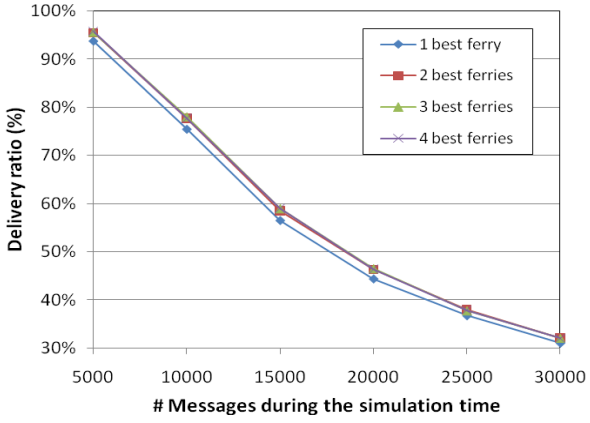

Fig. 7: Delivery ratio for forwarding different number of copies.

B. *Adding Extra Ferry Nodes*

The second strategy is to add to the DTN extra ferries in the middle of simulation if the delivery ratio is lower than a predefined threshold, which aims to provide more connectivity and bandwidth to the gateway nodes. To simulate this strategy in the ONE simulator, we create a FerryActivenessHandler class to manage the activeness of ferry nodes. Extra ferries are added to the DTN but are set to be inactive at the beginning. During the simulation, the active ferries can check current delivery ratio at specified time to see if it meets the threshold. If the delivery ratio is lower than the threshold, extra ferry nodes can be set active. While this strategy seems to be straightforward, adding extra ferry nodes will introduce extra message scheduling overhead to the gateway nodes.

In order to test whether extra ferry nodes can improve the delivery ratio, we run the simulations with different settings of ferry nodes: we add up to 3 extra ferry nodes in the middle of the simulation if the delivery ratio is lower than the threshold. The routes of these extra ferries are different from the routes of the original 8 ferries in order to provide more connection opportunities to the gateway nodes. We choose a high value 90% as the threshold so that more extra ferries will be enabled in the simulation. The results in Figure 8 show that adding 1, 2, and 3 extra ferries can improve the delivery ratio by about 3%, 5%, and 6%, respectively. We tried to add more extra ferries, but the amount of improvement is no longer noticeable. We think this is due to the extra message scheduling overhead on the gateway nodes when more extra ferries are added.

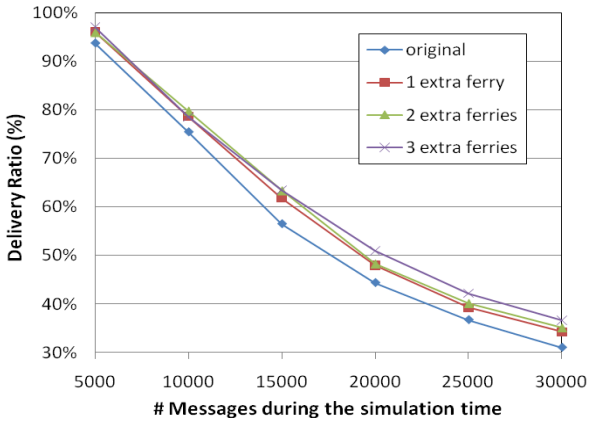

Fig. 8: Delivery ratio when extra ferries are added.

C. *Adjusting Ferries' Contact Time at Every Gateway According to Individual Delivery Ratio*

The third strategy is to adapt the ferries' contact time at each gateway according to the traffic demand at that gateway. The motivation of this strategy is to provide more contact time to high-traffic gateway nodes so that more messages can be delivered to them, while reducing some contact time with low-traffic gateway nodes if they have some spare contact time. However, the price that this strategy has to pay is that if the contact time at some gateway nodes is increased, the time for a ferry to travel a round of its route is prolonged due to the extra contact time, and the ferry will only be able to complete less rounds of its route during the same amount of simulation time.

To implement this strategy, the ferries need to be able to identify which gateway nodes are high-traffic nodes, and which gateway nodes are low-traffic ones. This can be achieved by making use of individual delivery ratio at each gateway node, which can be calculated by

$$\frac{\#messages\ delivered\ to\ the\ gateway\ in\ current\ round}{\#messages\ it\ carries\ to\ deliver\ to\ the\ gateway\ in\ current\ round}$$

To identify high-traffic nodes, the ferry will check if the delivery ratio at a gateway node is lower than a predefined threshold, *thr*. The contact time of a high-traffic node will be adjusted by taking the minimum of the contact time in this round divided by the delivery ratio in this round and a predefined maximal contact time ct_{max}; that is, the contact time will be increased more if the delivery ratio is lower until it reaches ct_{max}, because we cannot let the contact time increase forever. To identify low-traffic nodes, the ferry will check if the delivery ratio at a gateway node is 100% for *k* consecutive rounds, where *k* is a constant. The contact time of a low-traffic node will be adjusted by multiplying α, where $0.9 \le \alpha \le 1$. This strategy can be realized by adding the following procedure to the algorithm presented in Section III.

Procedure Ferry(j)_adjust_schedule_after_round(t)
// executed when ferry F_j finishes one round
// initially, in rounds 0 and 1 F_j follows the default schedule, in which the contact time at each gateway node is the same
Input: array *dr* of delivery ratio at each gateway node (*dr*[*i, t*] denotes the delivery ratio at gateway G_i in round *t*); array *ct* of contact time with each gateway node (*ct*[*i, t*] denotes the contact time with gateway G_i in round *t*)
Output: updated schedule for round *t*+2
Begin
1: For each *i*:
If *dr*[*i, t*] is below a predefined threshold *thr*, then
ct[*i, t*+1] := min(*ct*[*i, t*] / *dr* [*i, t*], ct_{max});
If *dr*[*i, t*], *dr*[*i, t*-1], …, *dr*[*i, t*-*k*+1] are all 100%, then
ct[*i, t*+1]:= *ct*[*i, t*] * α ($0.9 \le \alpha \le 1$)
2: calculate the schedule for round *t*+2 according to the array *ct*
End

We use a different, simpler scenario to test the performance of this strategy, because for this strategy to be useful the simulation scenario needs to contain some gateway nodes with much higher traffic demands compared to other gateway nodes. In this scenario, there are 13 gateway nodes, with 80% of all messages set to be between 2 selected gateway nodes (the high-traffic nodes), and the remaining 20% uniformly distributed between all 13 gateway nodes. There are only 2 ferries; both of them traverse all 13 gateway nodes but travel in reverse direction. The parameters are set as follows: *thr* is set to 90%, α is set to 0.9, *k* is set to 3, and ct_{max} is set to 100. We run simulations on different total number of messages, from 5000 to 50,000. The results in Figure 9 show that this strategy can improve the overall delivery ratio by more than 10% when the total message number is less than 20,000. But when the total number of messages is very high, the improvement becomes quite small. This is because for those nodes other than the two high-traffic nodes, the number of messages they have is large enough to use up their contact time with the ferry.

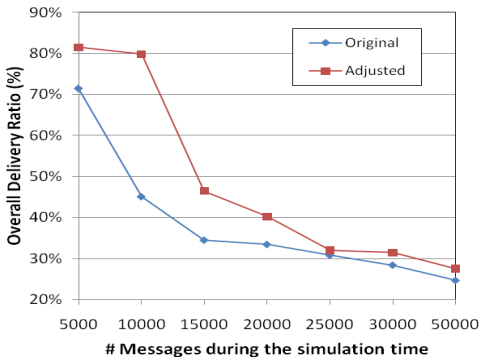

Fig. 9: Delivery ratio when the contact time is adjusted according to delivery ratio at each gateway node.

VI. Support for Message Prioritization

In Section IV we only discuss and apply one basic priority assignment scheme, which assigns higher priority to messages with earlier creation time. However, for ALARMS it is quite easy and natural to implement more advanced priority assignment schemes, such as assigning messages a priority class of *bulk*, *normal*, or *expedited* defined in [5].

To support message prioritization in ALARMS, we can define a tuple (*priority class*, *creation time*) for every message and sort on the priority class first to get a total order of the message set. When a new message is created at a gateway node and needs to be scheduled, all messages of a higher priority class will be scheduled in front of all messages of a lower priority class in the queue. For messages in the same priority class, a message with an earlier creation time is scheduled in front of a message with a later creation time. Therefore, messages with higher priority will be transferred to the chosen ferry prior to messages with lower priority.

On the other hand, during the contact time with a gateway node G_i, a ferry F_j receives outgoing messages from G_i, and inserts these outgoing messages into the queues corresponding to their destinations according to the priority and creation time of these outgoing messages. The ferry F_j also delivers messages destined to G_i according to the order of messages in the queue corresponding to G_i. Therefore, messages with higher priority will be delivered to the destination prior to messages with lower priority.

To evaluate the performance of ALARMS's support for message prioritization, we conduct simulations which classify messages into two priority classes, High and Low. Messages in High priority class are assigned a shorter TTL of 10 minutes, while message in Low priority class are assigned a longer TTL of 30 minutes. We use the same environment setup as in Section IV to run simulations to compare the performance of ALARMS and Spray-and-Focus. In each simulation, a total of 30,000 messages are generated based on different combinations of High priority and Low priority messages, from 10% of all messages are High priority and the rest 90% are Low priority, to 90% are High priority and 10% are Low priority. Figures 10 and 11 show the delivery ratio of all messages and the delivery

ratio of High priority messages only. Both figures show that the delivery ratio of ALARMS is considerably higher than the delivery ratio of Spray-and-Focus. This is because ALARMS will always deliver High priority messages first, but this feature is not supported by Spray-and-Focus.

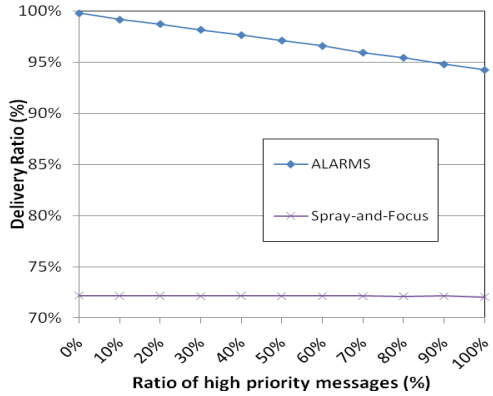

Fig. 10: Delivery ratio when messages are categorized into High priority class and Low priority class.

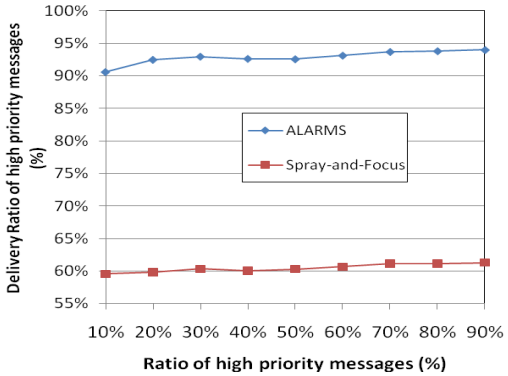

Fig. 11: Delivery ratio of only messages in High priority class when messages are categorized into High priority class and Low priority class.

VII. Concluding Remarks

In this paper, we first pointed out the need for improving the performance and providing QoS guarantees in a DTN network. Then, we proposed a novel Look-Ahead Routing and Message Scheduling (ALARMS) scheme in which the ferry nodes inform the gateway nodes about their travel schedule so that the gateway nodes can decide which ferry to use for each message in advance and schedule the message in the queue accordingly. We conducted simulations using an extended version of ONE simulator to evaluate the performance of our scheme and compare it with three other existing routing schemes, and the results show that ALARMS achieves better delivery ratio and shorter delivery delay with much lower overhead. We also discussed three potential enhancement schemes for optimizing QoS and demonstrated ALARMS's capability to support message prioritization.

In the future work, we will investigate how to allow the ferry nodes to adaptively adjust its dispatch frequency, route, and contact time according to the traffic demands and patterns of the gateway nodes it observes in the past few rounds. This leads to the possibility of designing an autonomous DTN that self-stabilizes according to the communication needs in the network.

Acknowledgment

This work is partially supported by an NSF grant (CNS-0916857).

References

[1] A. Balasubramanian, B. Levine, A. Venkataramani, "DTN routing as a resource allocation problem," ACM SIGCOMM Computer Communication Review, Vol. 37, No. 4, October 2007.

[2] N. Banerjee, M. Corner, B. Levine, "An Energy-Efficient Architecture for DTN Throwboxes," Proceedings of IEEE INFOCOM 2007, May 2007.

[3] J. Burgess, B. Gallagher, D. Jensen, B. Levine, "MaxProp: Routing for Vehicle-Based Disruption-Tolerant Networks", Proceedings of IEEE INFOCOM 2006, April 2006.

[4] B. Burns, O. Brock, B. Levine, "MV routing and capacity building in disruption tolerant networks," Proceedings of IEEE INFOCOM'05, March 2005.

[5] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, H. Weiss, "Delay-Tolerant Networking Architecture," RFC 4838, April 2007.

[6] S. Chaturvedi, P. Kalakota, C.-T. Huang, "Improving delay-tolerant network performance using forward routing information", Poster, 14th IEEE International Conference on Network Protocols, November 2006.

[7] T. Clausen and P. Jacquet, "Optimized link state routing protocol (OLSR)," RFC 3626, October 2003.

[8] B. DeRenzi, Y. Anokwa, T. Parikh, G. Borriello, "Reliable data collection in highly disconnected environments using mobile phones," Proceedings of ACM SIGCOMM Workshop on Networked Systems for Developing Regions (NSDR'07), August 2007.

[9] K. Fall, W. Hong, S. Madden, "Custody Transfer for Reliable Delivery in Delay Tolerant Networks," IRB-TR-03-030, July 2003.

[10] S. Jain, K. Fall, R. Patra, "Routing in a Delay Tolerant Network," Proceedings of ACM SIGCOMM 2004, August 2004.

[11] E. P. Jones and P. A. Ward, "Routing strategies for delay-tolerant networks," Proceedings of ACM SIGCOMM'04, August 2004.

[12] J. Leguay, T. Friedman, V. Conan, "DTN routing in a mobility pattern space," Proceeding of 2005 ACM SIGCOMM Workshop on Delay-tolerant networking, August 2005.

[13] A. Lindgren and A. Doria, "Probabilistic Routing Protocol for Intermittently Connected Networks", draft-lindgren-dtnrg-prophet-02.txt, August 2006.

[14] C. Perkins, E. Belding-Royer, S. Das, "Ad hoc on-demand distance vector (AODV) routing," RFC 3561, July 2003.

[15] K. Scott, S. Burleigh, "Bundle Protocol Specification," RFC 5050, November 2007.

[16] T. Spyropoulos, K. Psounis, C. Raghavendra, "Spray and Wait: An Efficient Routing Scheme for Intermittently Connected Mobile Networks," Proceedings of ACM SIGCOMM'05 Workshops, August 2005.

[17] T. Spyropoulos, K. Psounis, C. Raghavendra, "Spray and Focus: Efficient Mobility-Assisted Routing for Heterogeneous and Correlated Mobility," Proceedings of Fifth Annual IEEE International Conference on Pervasive Computing and Communications Workshops (PerComW'07), March 2007.

[18] T. Spyropoulos, K. Psounis, C. Raghavendra, "Efficient Routing in Intermittently Connected Mobile Networks: The Multiple-copy Case," IEEE/ACM Transactions on Networking, Vol. 16, Issue 1, pp. 77-90, February 2008.

[19] A. Vahdat, D. Becker, "Epidemic routing for partially connected ad hoc networks," Technical Report CS-2000-06, Department of Computer Science, Duke University, April 2000.

[20] R. Viswanathan, J. Li, M. C. Chuah, "Message Ferrying for Constrained Scenarios," Proceedings of Sixth IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks (WoWMoM'05).

[21] W. Zhao, M. Ammar, E. Zegura, "Controlling the Mobility of Multiple Data Transport Ferries in a Delay-Tolerant Network," Proceedings of IEEE INFOCOM 2005, Miami, Florida, 2005.

[22] The Opportunistic Network Environment (ONE) Simulator, available at http://www.netlab.tkk.fi/tutkimus/dtn/theone/