Cooperative Robust Forwarding Scheme in DTNs using Erasure Coding

Yong Liao†, Zhensheng Zhang‡, Bo Ryu‡, Lixin Gao†

†Dep. of Electrical & Computer Eng.
University of Massachusetts
Amherst, MA 01003, USA
email: {yliao, lgao}@ecs.umass.edu

‡San Diego Research Center
Suite A, 6696 Mesa Ridge Rd
San Diego, CA 92121, USA
email: {z.zhang, bo.ryu}@sdrcinc.net

***Abstract*—In Delay/Disruption Tolerant Networks (DTNs), network connectivity is highly dynamic due to node movements. Several fundamental assumptions of the conventional wireless networks no longer hold in DTNs, which in turn requires new architectures and protocols design. Routing in DTNs is a challenging problem as the mobility pattern and the available resources of each node, such as buffer space, link speed, remaining battery power, etc, could be dramatically different. In this paper, we propose a COoperative Robust forwarding scheme using Erasure coding (CORE) for DTNs. We first introduce a generalized expression for evaluating the capability of a node to deliver a message to its destination. The message forwarding decision is made based on the capability of the encountering nodes to relay the messages. Furthermore, instead of simply duplicating/forwarding the entire message, we adopt erasure coding to generate message redundancy. With a fixed overhead, we can generate a large number of message blocks using erasure coding and forward those message blocks to more relay nodes. Therefore, our proposed message delivery scheme takes advantage of both the cooperation and the diversity of the relay nodes. Extensive simulations show that our proposed scheme, CORE, outperforms several existing protocols.**

I. Introduction

Delay/Disruption Tolerant Networks (DTNs) [1] are usually intermittently connected mobile wireless networks, such as inter-planetary networks [1,2] and wildlife tracking/monitoring sensor networks [3,4]. In DTNs, the connectivity between nodes changes frequently due to the movement of nodes. Nodes in DTNs could move randomly or move according to certain mobility patterns. Generally, the message forwarding in DTNs works in an extended store-and-forward way. When two nodes meet with each other and a wireless link is established between them, messages can be sent over this link. When a node is not in contact with any other nodes, it stores the messages and waits for a chance to meet with other nodes and sends the messages to them. In this manner, the messages are transmitted from one node to another. Hopefully, the messages can be forwarded to their destinations eventually by some "relay" nodes carrying the message. Even though the basic idea of message forwarding is simple, making the right routing decision in DTNs is a challenging task, since the network connectivity is time variable and it is usually hard to be predicted. Besides, in reality, not only a node has limited chances to meet with other nodes, but also each node has only limited resources, such as buffer space for caching the relayed messages, remaining battery power for node operation, and outgoing bandwidth for data transmitting.

In order to create more chances to meet with other nodes, some existing routing schemes dispatch multiple identical copies of a message to different relay nodes [5,6]. We classify them as *simple replicate* schemes in this paper. But dispatching too many identical copies of one message consumes more resources of nodes in the network. Recently, another class of routing scheme based on erasure coding has been proposed [7,8]. The idea of erasure coding is to encode one message with k blocks into K blocks ($K > k$). The original message can be recovered from any k of the K blocks. Erasure coding based routing scheme benefits from the diversity of relay nodes. With the same overhead, an erasure coding based scheme makes more relay nodes involved in message forwarding than the simple replicate scheme does. Erasure coding based schemes are especially suitable for those DTNs where relay node failures are prevalent, delays are unpredictable, and minimizing the worse-case delay is important [7].

Most of the previous work assumes nodes in the network are identical and independent. This assumption, although simple, may not be realistic. Measurement studies [9,10] have demonstrated that nodes in DTNs may be significantly different in terms of mobility patterns. Besides the difference in mobility pattern, the available resources of different nodes can be quite diverse in DTNs with heterogeneous nodes. Here the resources which affect the delivery of messages can be available buffer space, remaining battery power, link bandwidth, etc. In making the next hop forwarding decision, most of the previous work considers only the probability of contacting with the destination node, which is obtained based on history information of node mobility pattern. But considering only the node contact frequency can lead to bad forwarding decisions. For example, if the available buffer space on a relay node is almost zero or the remaining power of that node is very low, even if that relay node has high probability to contact with the destination node, it may not be a good relay for new messages, since messages forwarded to it will be dropped latter.

1-4244-1513-06/07/$25.00

With the above intuition in mind, we generalize the erasure coding based routing scheme proposed in [8] and propose a COoperative Robust forwarding scheme using Erasure coding (CORE) for DTNs in this paper. In CORE, a message is first encoded into large number of small blocks and then those blocks are distributed to suitable relay nodes. CORE uses both node contact frequency information and information of node available resources to evaluate the "capability" of a relay node to successfully deliver the message. The information of node available resources includes buffer space, link speed, remaining energy level, node willingness to relay, etc. We propose to use a linear function to combine those factors into one single scale so that the capability of different nodes can be compared. How messages blocks are distributed among different nodes is determined by the capability of encountering nodes. Extensive simulations have been conducted to study the performance of our scheme. The simulation results demonstrate that the cooperative forwarding scheme with erasure coding (CORE) outperforms several existing schemes in terms of message delivery rate and achieves up to 50% reduction in message delivery time for the scenario considered in this paper.

The rest of this paper is organized as follows. We present a brief overview of related works on DTN routing in Section II. In Section III, we describe the CORE scheme in detail. The simulation results are presented in Section IV. Section V concludes this paper.

II. Related Work

Recently, there has been much research activity in the area of DTN routing [11]. Two deterministic routing schemes are proposed in [12,13]. Those schemes assume that the future (or at least for certain period of time) network connectivity dynamics can be predicted and use deterministic algorithms to compute the best path for forwarding messages between nodes.

Besides the deterministic schemes, the epidemic routing is proposed in [5], which works like flooding the messages to all nodes in the network. In the direct transmission scheme proposed in [14], the source node holds the data message until it meets with the destination node. Other routing schemes try to tradeoff between the epidemic routing and the direct transmission scheme in terms of delivery rate/delay and overhead. In [15] the authors propose to let a node duplicate a message and forward it to a contact only if the contact has a higher delivery probability for the message. Similarly, Tan et al. use expected path length as the metric in evaluating the capability of a contact to deliver the message in [16]. In [17], the authors propose to forward messages to relay nodes with similar mobility pattern as the destination node. In order to exactly bound the overhead of delivering a message, the spray-and-wait scheme is proposed in [6], in which a fixed number of copies of one message is first sprayed to relay nodes into the network, then the relay nodes wait for the chance to meet with the destinations so that the message can be delivered.

Recently, Wang et al. propose a scheme in [7] which extends the spray-and-wait scheme by using erasure coding. Erasure coding [18] transforms a message into a set of small blocks so that the original message can be recovered from a subset of those blocks. By applying erasure coding, the protocol can potentially have more relay nodes involved in forwarding messages and benefits from the diversity of relay nodes. In an extreme case, the average delay tends to converge to a constant. In [8], the authors propose an erasure coding based scheme which exploits the periodic movement patterns of nodes to estimate the chance of a relay node to deliver a message to its destination.

III. CORE: COoperative Robust forwarding scheme using Erasure coding

A. Overview of erasure coding

Erasure coding [18] is a coding technique which transforms a message of k blocks into K $(K > k)$ blocks so that the original message can be recovered from a subset of those K blocks. The ratio of k over K is called the *coding rate*, denoted by r. Under an *optimal erasure coding*, k/r blocks should be generated for a message with k blocks, where any k blocks are sufficient to recover the original message. Optimal coding is expensive in terms of memory and CPU usage when k is large. Therefore near optimal erasure coding is often used, which requires $(1 + \varepsilon)k$ blocks to recover the message. For simplicity, we ignore the ε in this paper and assume k blocks are enough to recovery the original message.

Using erasure coding in a DTN benefits the message forwarding because we can potentially generate a large number of small blocks for one message and introduce more diversity by letting more relay nodes carry the blocks of that message [7,8]. In contrast, with the same message redundancy, the simple replicate scheme can have only a few identical message copies. If some of the relay nodes carrying an entire message copy fail or those message copies are dropped because of buffer overflow, the successful delivery of that message can be greatly impacted. If we use erasure coding to encode message into small blocks and let a relatively large number of relay nodes carry those blocks, even if some of the message blocks are dropped/delayed by relay nodes, the destination node may still be able to receive enough message blocks to recover the original message.

B. The capability of a relay node to deliver a message

Using erasure coding alone may not always deliver the message quicker than simple replicate based schemes [8,18]. To deliver the message to its destination faster, we need to select the right relay nodes to carry the message blocks, which means we need to estimate the "capability" of a relay node to deliver the message. In estimating whether a relay node is a good candidate for delivering a message to its destination, the most common way is measuring how many times the relay node has encountered with the destination node of the message, since message transmission can occur only when two nodes meet with each other. If a relay node met with the destination node many times before, we can expect it has more chance to successfully deliver the messages in

the future. Therefore, most of existing works use the contact frequency between the relay node and the destination node as the metric to estimate the capability of the relay node to deliver a message.

However, considering only the historical direct contacting information can be misleading in some scenarios, since a message can be relayed for multiple hops before reaching its destination. The situation can be worse if we consider some realistic issues such as limited battery power, limited buffer space, limited communication bandwidth, etc. For instance, if node A moves faster than other nodes in the network, node A has more chance to meet with other nodes. Therefore, node A may have already used up most of its buffer space or node A may have already consumed most of its battery power. So choosing node A as the relay node could be a bad decision, because forwarding some additional messages to A would cause A to drop some of the old messages in its buffer, or A might run out of power before delivering the messages in its buffer. In order to make the right decision in message forwarding, other factors should be considered also.

Here we present a simple method to compute the capability of a relay node to deliver a message. Generally, we assume that there are a set of parameters which affect the decision of how to forward message blocks to relay nodes. Those parameters can be denoted as a vector $\overline{V} = [v_1, v_2, ..., v_n]$. We assume the capability of a relay node to successfully deliver a message is a monotonic increasing function of $v_i, i \in [1, n]$. For various network scenarios, the impact of each component in $\overline{V}$ can be different. Considering the different impact of those parameters, we assign a weight α_i to each component v_i in $\overline{V}$ and define the capability of a relay node as the linear combination of all the components in vector $\overline{V}$.

More specifically, assume that two nodes n_1 and n_2 meet with each other and their capability to deliver a message m to its destination node is denoted by $\overline{V}_1$ and $\overline{V}_2$, respectively. Note that the scales of different components in vector $\overline{V}$ can be quite different. Therefore, it is necessary to first normalize those parameters and then linearly combine these normalized components into one scale. Normalization of the components in $\overline{V}_1$ and $\overline{V}_2$ is given in (1)

$$v_{i,1}^{'} = \frac{v_{i,1}}{v_{i,1}+v_{i,2}} \quad v_{i,2}^{'} = \frac{v_{i,2}}{v_{i,1}+v_{i,2}} \tag{1}$$

where $v_{i,1}$ is the ith component in $\overline{V}_1$ and $v_{i,2}$ is the ith component in $\overline{V}_2$. The capability of n_1 or n_2 to successfully deliver message m is then given by

$$C_{j,m} = \sum_{i=1}^{n} \alpha_i v_{i,j}^{'} \quad j \in \{1, 2\} \tag{2}$$

where

$$\sum_{i=1}^{n} \alpha_i = 1$$

$C_{j,m}$ is used as the indicator for the capability of relay node n_j to deliver the blocks of message m. When two nodes meet with each other, they should exchange message blocks according to their capability parameters. For easy description, we list in TABLE I the notations used in the rest of this paper.

TABLE I
NOTATIONS

Symbol	*Description*
m	The message in concern.
k	Each message has k blocks.
K	Each message is encoded into K blocks and the original message can be recovered by any k of those K blocks.
TTL_{max}	Each message block is labelled with a time stamp on when it is generated. Message blocks older than TTL_{max} are dropped during cache replacement.
TTL_{rep}	Message blocks older than TTL_{rep} and less likely to be delivered than the newly incoming message blocks are dropped during cache replacement.
ttl_m	The "age" of blocks generated from message m.
$F_{i,j}$	The frequency of node i to contact with node j.
$Dst(m)$	The destination of message m.
$C_{i,m}$	The capability of node i to deliver message m.
$M_{i,m}$	The number of blocks of message m in node i.
Buf_i	Node i's buffer size.
$Free(Buf_i)$	The available free space of node i's relay buffer.

C. Basic operation of CORE

When a message is generated, it is first encoded into K small message blocks by the source node using erasure coding, so that the original message can be recovered from any k out of those K blocks. Each message block is labelled with a time stamp about when it is generated. The source node holds all the message blocks until it meets with another relay node and forwards some blocks to the relay node. When two nodes meet with each other, they first exchange information about the message blocks in their buffer and all other necessary information for computing each other's capability to deliver each message. After the information exchanging phase, the two encountering nodes compute their capability to deliver each message in their buffer and make the message forwarding decisions. When the buffer is full, a node first drops those message blocks who were generated more than TTL_{max} seconds ago. If the resulting buffer space is still not enough for the newly incoming message blocks, the node will drop message blocks who are older than TTL_{rep} seconds and are less likely to be delivered than the newly incoming message blocks. In Section III-D and Section III-E, we will discuss the cache replacement strategy and the message forwarding scheme in detail.

D. The cache replacement strategy

For each node, we assume it has only limited buffer space which can be used for relaying messages generated by others. When two nodes meet with each other and exchange message blocks, those nodes may have to adopt a cache replacement strategy to free some buffer space. Here the cache replacement strategy decides which message blocks to be dropped if there is not enough free buffer space to store the newly arriving message blocks.

Algorithm 1 The cache replacement algorithm

Precondition: Node B needs to receive a blocks of message m from node A, but B's buffer can store only n ($n < a$) more blocks. Node B will do cache replacement.

for each message $m^{'} \in Buf_B$ **do**
  if $n \geq a$ **then**
    break;
  end if
  if $ttl_{m^{'}} \geq TTL_{max}$ **then**
    drop min(a-n, $M_{B,m}$) blocks of $m^{'}$;
    $n = n$+min(a-n, $M_{B,m}$);
  end if
end for
while $n < a$ **do**
  find message $m^{'} \in Buf_B$ with the smallest contact frequency with the destination of message $m^{'}$;
  if $F_{B,Dst(m)} > F_{B,Dst(m^{'})}$ and $ttl_{m^{'}} \geq TTL_{rep}$ **then**
    drop min(a-n, $M_{B,m}$) blocks of $m^{'}$;
    $n = n$+min(a-n, $M_{B,m}$);
  else
    break;
  end if
end while
Inform A to forward n blocks of message m;
return n;

Selecting which message blocks to drop is a challenge here. Intuitively, those messages which were generated long time ago should be dropped first, because those messages may already be delivered to their destinations by other relay nodes or those message may be out-of-date. Second, the messages which are not likely to be delivered to their destinations should be dropped, because carrying those messages may sacrifice the delivery of messages with higher chance to be delivered. Therefore, we adopt a two-stage scheme to do cache replacement.

Suppose nodes A and B meet with each other and node A is going to forward a blocks of message m to node B after they compare their capability to deliver message m. Node B does not have enough buffer space so B will do cache replacement. First, node B drops those message blocks who are older than TTL_{max} seconds. The TTL_{max} parameter can be set according to the application requirements. If the free buffer space is still not enough, node B will drop some message blocks which are less likely to be delivered than the blocks node A is going to forward. In order to prevent B from dropping those message blocks which were generated recently, we set another parameter TTL_{rep}. For the blocks of message m in node B, B will drop the blocks of message m only when $ttl_m \geq TTL_{rep}$ and node B's frequency of meeting with the destination of m is smaller than its frequency of meeting with the destination of message blocks from A. After node B finishes dropping messages, it informs node A of its free buffer space size and A starts to transfer message blocks to B. The pseudo-code for the two-stage cache replacement algorithm is shown in Algorithm 1.

E. Message forwarding scheme

When two nodes meet with each other, they first exchange a summary of the message blocks in their buffer. For the blocks of each message, the summary includes the destination node ID, the relay node ID, the message life time, the message sequence number, number of message blocks, and the contact frequency with the message's destination node. Besides of those parameters, the two nodes should also exchange information such as each other's free buffer space, available battery power, etc. Based on the exchanged information, the two encountering nodes can compute each other's capability in delivering the message to its destination node, using (1) and (2). If the available buffer space on one node is not enough to store the message blocks which will be transferred to it by the other node, the cache replacement strategy described in Section III-D will be used to decide whether to drop some message blocks or deny the transferring of new message blocks. The pseudo-code in Algorithm 2 shows how message blocks are forwarded between two encountering nodes.

Algorithm 2 The message block forwarding process

Precondition: Node A encounters B and node A transfers message blocks to B.

for each message m in Buf_A **do**
  if $Dst(m)$ is B and B is in connection **then**
    B receives $\min(k, M_{A,m})$ blocks of msg m from A;
  end if
  if $Dst(m) \neq B$ and B is in connection **then**
    $trans = M_{A,m} \times \frac{C_{B,m}}{C_{A,m}+C_{B,m}}$;
    if $trans > Free(Buf_B)$ **then**
      call the procedure in Algorithm 1;
      set $trans$ to the return value of Algorithm 1;
    end if
    B receives $trans$ blocks of m from A;
  end if
end for

IV. Simulation Evaluations

In order to evaluate the performance of the CORE scheme, we have implemented an event driven simulator using C++ language and have conducted extensive simulation experiments. In our simulations, we compare the performance of four different erasure coding based schemes, the source forwarding scheme (SRCF), in which only the source node delivers message blocks to relay nodes and relay nodes deliver message blocks to the destination node directly; the binary spray&wait scheme (BINSW), which is similar to the scheme proposed in [6] except here small message blocks are sprayed into the network instead of the entire messages; the estimation based erasure coding forwarding scheme (EECF) [8], which uses only the contact frequency as the parameter to estimate the capability of a relay node; and the cooperative robust forwarding scheme (CORE). The randomness in our simulations comes from the nodes movement only. That is, for different simulation instances, no matter what routing schemes are used, the nodes movement is exactly the same if the same random

seed is used to run the simulations. In order to fairly compare those four schemes, we use the same set of random seeds to run the simulations of the four different schemes.

A. The mobility model

In our simulations, we use a restrict random waypoint mobility model, which is similar to the model used in [8,16]. Different from the conventional random waypoint model, each node in the restrict random waypoint model has a given set of waypoints. After the node arrives at a waypoint, it will stay at that point for a random "thinking time". Then the node randomly chooses one destination from its waypoint set and moves towards that point. Therefore, instead of having uniform visiting probability over the whole network, each node has a higher chance to visit some areas and may seldom move to some areas in the network. By carefully setting the distribution of each node's waypoints, we can simulate different types of movements. We believe this restrict random waypoint mobility model captures some important characteristics of realistic mobile networks, as the recent measurement in pocket switched networks [9] shows that nodes always have heterogeneous mobility patterns. Some nodes visit certain areas more frequently than other nodes; or a group of nodes with common interest may meet with each other more frequently. Similarly, the PeopleNet [10] also observes that people with similar interests form "bazaars", where they have more chance to meet with each other.

The network used in our simulations consists of 100 nodes moving in a 10000×10000 $unit^2$ square area. Each node has 20 waypoints. Among those waypoints, the first one is randomly chosen in the network. Then 14 waypoints are randomly chosen from a 1000×1000 $unit^2$ square area centered at the first waypoint. The rest 5 waypoints are randomly chosen in the entire 10000×10000 $unit^2$ square area. Among all those 100 mobile nodes, three fourth of them move at the speed of $V_{max} = 50$ $unit/step$ and the others move at the speed of $V_{min} = 5$ $unit/step$. In the initial state, nodes are randomly located in the entire square area. The thinking time at each waypoint is uniformly selected in $[0, 20]$ simulation steps. The communication range of a node is set to 60 $units$. At the beginning of each simulation instance, we have all the nodes moving in the network for 2000 simulation steps, so that nodes can build and stabilize their contacting frequency tables [1]. After that initial phase, we run the simulation for 5000 steps.

B. The node model

We assume each node, say node A, in the network has two kinds of buffer spaces. One is for storing relay messages. We call this buffer as "relay buffer". Message blocks in node A's relay buffer are neither generated by A nor destinated to A. The other buffer is used to store message blocks generated by A or destinated to A. This buffer is called "self buffer". We assume each node has infinite self buffer but only limited relay buffer. In order to be fair in comparing the performance of different schemes, nodes in all four different routing schemes use the process described in Section III-D to do cache replacement when they do not have enough relay buffer space. The default relay buffer space on each node is set to be $50M$ bits. Each node generates 100 messages after the initial table building phase and the destinations of those message are randomly selected from all the mobile nodes in the network. Each message is $0.5M$ bits long and it is encoded into 25 message blocks using erasure coding. The size of each message block is $0.5/5 = 0.1M$ bits. With any 5 message blocks among those 25 blocks, the original message can be recovered.

In the simulation for the CORE scheme, we consider two parameters in evaluating the capability of a relay node to forward a message m. One parameter (represented by $v_{1,i}$) is the node i's contact frequency with destination node of that message. The other parameter (represented by $v_{2,i}$) is the free buffer space in relay node i. To compute the capability of each node to deliver message m when two nodes A and B meet with each other, the following linear function is used

$$C_i = \alpha v_{1,i}^{'} + (1 - \alpha) v_{2,i}^{'} \tag{3}$$

where $v_{1,i}^{'}$ is node i's normalized contact frequency with the destination of message m, $v_{2,i}^{'}$ is node i's normalized free buffer space, α is the weight assigned to the contact frequency parameter, and $(1-\alpha)$ is the weight assigned to the free buffer space parameter. In the following experiments, the default value of α is set to 0.7 if we do not explicitly specify it.

C. The message delivery delay

In this experiment, we compare the message delivery delay of the SRCF, BINSW, EECF, and CORE schemes. For each scheme, we use ten different random seeds to run the simulation and plot the CDF (Cumulative Distribution Function) of the message delivery delay. The simulation results are shown in Fig. 1. From the figure, we can see that our CORE scheme outperforms the other three forwarding schemes. With probability 0.9, CORE can deliver messages within 1600 simulation steps. The BINSW scheme delivers message faster than the SRCF scheme because it sprays the message blocks quickly into different relay nodes. The EECF scheme is better than BINSW because the message blocks are forwarded to those relay nodes who are more likely to meet with the destination node. Our CORE scheme makes even better decision than the EECF scheme in forwarding the message blocks, because the CORE scheme not only considers whether the relay nodes have higher chances to meet the destination node, but also considers whether the relay nodes can keep the message blocks long enough instead of dropping those message blocks before forwarding them to other nodes.

[1] Although the contacting frequency table is useless for SRCF and BINSW schemes, there is still an initial table building phase in the simulations of SRCF and BINSW schemes because we want to ensure the nodes movements are exactly the same for these four schemes.

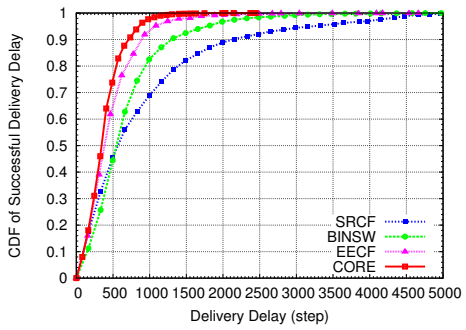


Fig. 1. The CDFs of four different message forwarding schemes

D. Impact of buffer size on the performance

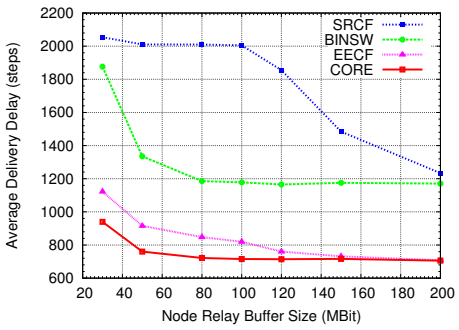
Since messages may be carried by relay nodes for considerably long time, the buffer size of each node can greatly impact the message delivery. In this experiment, we range the relay buffer size on each node from 20 Mbits to 200 Mbits and run the simulations for the four forwarding schemes. The average message delivery delay result is plotted in Fig. 2(a). From Fig. 2(a) we can observe that in general the average message delivery delay in four different schemes decreases as the node relay buffer size becomes larger. But the average message delivery delay of the CORE scheme is always smaller than the other three schemes, no matter what the node relay buffer size is. Besides, compared with the other three schemes, our CORE scheme can achieve shorter message delivery delay especially when the buffer size is small.

We also count the number of successfully delivered messages in the simulations of the four different schemes. The result is shown in Fig. 2(b). From this plot we can see that our CORE scheme always delivers more messages than the other three schemes within 5000 simulation steps. In some cases, the EECF scheme can deliver less messages than the BINSW scheme. The reason might be that in the EECF scheme, those nodes with higher chance to meet with other nodes can always receive large number of message blocks, which makes them frequently drop message blocks.

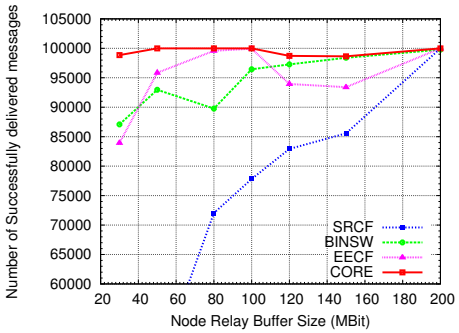
From the figures plotted in Fig. 2 we can see that our CORE is better than the other three schemes especially when the node buffer size is small. When the mobile nodes have large enough relay buffer spaces (larger than 150 Mbits), the CORE scheme retrogresses to the EECF scheme since there are always available buffer space and it does not make any difference whether to consider the buffer space in making forwarding decisions or not.

E. Impact of the capability weights on the performance

The weight assigned to each component which determines the capability of relay nodes is important for the CORE scheme to achieve good performance in terms of message delivery delay and the message delivery rate. It is difficult to analytically prove a set of weights is better than an other set of weights, since the selection of the weights should be determined by a number of factors such as nodes mobility, the



(a) Average message delivery delay



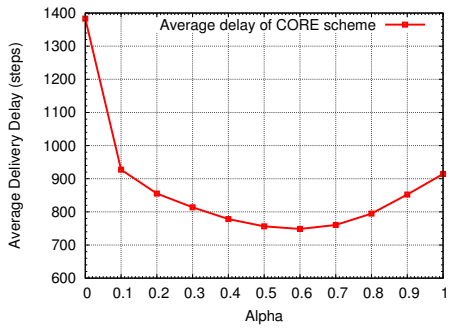
(b) Number of successfully delivered messages

Fig. 2. Comparison of the average message delivery delay and the number of successfully delivered messages of four different message forwarding schemes when the node buffer space is set to different sizes.

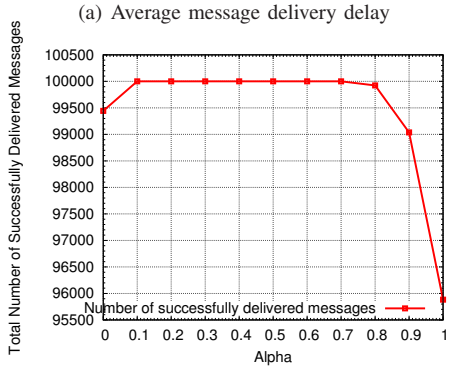
new message generating model, etc. Here we use an empirical approach to study how the weight assigned for each factor can affect the performance of our CORE scheme. As we use equation (3) to compute the node capability in our simulations, we range the value of α from 0 to 1 and run the simulations of our CORE scheme. Here, when α is close to 0, only the node buffer size information is considered and the node contact frequency information is not used, while when α is close to 1, the node buffer size information is not used and only the node contact frequency information is used.

The results are plotted in Fig. 3, in which we present both the result of the average message delivery delay and the result of the total number of successfully delivered messages. It is easy to notice from Fig. 3 that when α is too small (close to 0) or too large (close to 1), the CORE scheme suffers longer delivery delay and delivers less messages. This verifies our intuition that considering one parameter alone in making forwarding decision does not yield good performance.

It can also be noticed that the average delivery time is more sensitive to the weights assigned to different factors, while the number of successfully delivered messages is less sensitive to the weights. One reason for this phenomenon is that most messages in DTNs will eventually be delivered if nodes have large enough buffer space. Therefore, most of the messages can be delivered to their destinations even we assign "bad" weights (if the buffer size is large enough and we do not limit the delivery time). From the average delay curve plotted in Fig. 3(a) we can see that the optimal value of α is about 0.6,



(a) Average message delivery delay



(b) Number of successfully delivered messages

Fig. 3. The average message delivery delay and number of successfully delivered messages in CORE scheme when the weights assigned to each factors determining node's capability are different. The α parameter here is the weight assigned to the contact frequency factor; $(1-\alpha)$ is the weight assigned to the node free relay buffer factor.

which seems to achieve the best balance between the contact frequency factor and the available relay buffer size factor.

V. Conclusion

In this paper, we have presented CORE, a COoperative Robust forwarding scheme with Erasure coding for DTNs. CORE adopts erasure coding to encode messages into small blocks and makes the intelligent and robust decision in forwarding those message blocks. When making the message block forwarding decision, CORE takes into consideration a number of factors which determine the capability of a relay node to deliver the message, such as contact frequency with the destination node of that message, the node's available buffer size, the remaining power level, etc. Simulation results demonstrate that even considering only two factors, the contact frequency and the available buffer space, CORE outperforms three existing DTN routing schemes. By jointly considering a set of factors in evaluating the capability of a relay node, CORE can deliver message faster and can achieve higher message delivery rate. Our study indicates that in order to deliver more messages and deliver messages faster, it is essential to differentiate relay nodes when forwarding message blocks by considering the capability of those relay nodes. In the future, we will study the impact of the remaining energy level of relay nodes on the system performance, as energy efficiency becomes more and more important in protocol design.

References

[1] K. Fall, "A delay-tolerant network architecture for challenged Internets," in *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM Press, 2003, pp. 27–34.

[2] S. Burleigh, A. Hooke, L. Torgerson, K. Fall, V. Cerf, B. Durst, keith Scott, and H. Weiss, "Delay-tolerant networking: An approach to interplanetary Internet," *IEEE Communications Magazine*, June 2003.

[3] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein, "Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with zebranet," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. 5, pp. 96–107, 2002.

[4] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*. New York, NY, USA: ACM Press, 2002, pp. 88–97.

[5] D. B. Amin Vahdat, "Epidemic routing for partially connected ad hoc networks," Duke University, Technical Report CS-200006, Apr. 2000.

[6] T. Spyropoulos, K. Psounis, and C. Raghavendra, "Spray and wait: an efficient routing scheme for intermittently connected mobile networks," in *WDTN '05: Proceeding of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*. New York, NY, USA: ACM Press, 2005, pp. 252–259.

[7] Y. Wang, S. Jain, M. Martonosi, and K. Fall, "Erasure-coding based routing for opportunistic networks," in *WDTN '05: Proceeding of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*. New York, NY, USA: ACM Press, 2005, pp. 229–236.

[8] Y. Liao, K. Tan, Z. Zhang, and L. Gao, "Estimation based erasure-coding routing in delay tolerant networks," in *IWCMC' 06: International Wireless Communications and Mobile Computing Conference, Delay Tolerant Mobile Networks Symposium*, Vancouver, Canada, July 2006.

[9] P. Hui, A. Chaintreau, J. Scott, R. Gass, J. Crowcroft, and C. Diot, "Pocket switched networks and human mobility in conference environments," in *WDTN '05: Proceeding of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*. New York, NY, USA: ACM Press, 2005, pp. 244–251.

[10] M. Motani, V. Srinivasan, and P. S. Nuggehalli, "Peoplenet: engineering a wireless virtual social network," in *MobiCom'05*. New York, NY, USA: ACM Press, 2005, pp. 243–257.

[11] Z. Zhang, "Routing in intermittently connected mobile ad hoc networks and delay tolerant networks: Overview and challenges," 2006, IEEE Communication Survey and Tutorial.

[12] S. Jain, K. Fall, and R. Patra, "Routing in a delay tolerant network," in *Proceeding of SIGCOMM'04*, Aug. 2004.

[13] S. Merugu, M. Ammar, and E. Zegura, "Routing in space and time in networks with predictable mobility," Georgia Institute of Technology, Technical Report GIT-CC-04-7, 2004.

[14] T. Spyropoulos, K. Psounis, and C. Raghavendra, "Single-copy routing in intermittently connected mobile networks," in *Secon ' 05: The First IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*, Oct. 2004.

[15] A. Lindgren, A. Doria, and O. Schelen, "Probabilistic routing in intermittently connected networks," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 7, no. 3, pp. 19–20, 2003.

[16] K. Tan, Q. Zhang, and W. Zhu, "Shortest path routing in partially connected ad hoc networks," in *Globecom*, 2003.

[17] J. Leguay, T. Friedman, and V. Conan, "DTN routing in a mobility pattern space," in *WDTN '05: Proceeding of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*. New York, NY, USA: ACM Press, 2005, pp. 276–283.

[18] H. Weatherspoon and J. Kubiatowicz, "Erasure coding vs. replication: A quantitative comparison," in *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*. London, UK: Springer-Verlag, 2002, pp. 328–338.