Using Known Vectors to Improve Data Dissemination in Opportunistic Networks

Jyh-How Huang[1], Ying-Yu Chen[2], Li-Ping Tung[3], and Ling-Jyh Chen[2]

[1]Sport Information and Communication Department, National Taiwan University of Physical Education and Sport

[2]Institute of Information Science, Academia Sinica

[3]Intelligent Information and Communications Research Center, National Chiao Tung University, Taiwan

Abstract

An opportunistic network is a type of challenged network in which contacts are intermittent, an end-to-end path rarely exists between the source and the destination, disconnection and reconnection are common occurrences, and link performance is highly variable or extreme. Conventional methods of data dissemination in opportunistic networks rely on a meta-message exchange scheme, called the Summary Vector (SV), to prevent redundant transmission of data bundles that already exist on receivers. We consider that the SV scheme is costly in terms of the data transmission overhead, which is unaffordable in opportunistic networks. Hence, we propose an alternative scheme, called the Known Vector (KV), to improve the efficiency of meta-message exchanges for data transmission in opportunistic networks. Using a comprehensive set of simulations with realistic network scenarios, we demonstrate that the KV scheme can be easily integrated into existing opportunistic network routing protocols (e.g., Epidemic and PRoPHET routing). Moreover, it can reduce the communication overhead significantly, thereby improving energy efficiency for data transmission in opportunistic networks.

I. Introduction

An opportunistic network is a type of challenged network that has the following characteristics: (1) network contacts (i.e., communication opportunities) are intermittent; (2) there is rarely an end-to-end path between the source and the destination; (3) disconnection and reconnection are common events; and (4) the link performance is highly variable or extreme. Emerging applications of opportunistic networks are wide ranging. For instance, it would be advantageous if mobile search and rescue nodes could be interconnected in disaster areas where communication infrastructures have been disabled by earthquakes, hurricanes, wildfires, or floods. Other applications include facilitating message exchange in underdeveloped

areas (remote towns and villages interconnected by wireless networks, but not guaranteed an always-on Internet connection) and scientific monitoring of wilderness areas (remote monitoring of various forms of wildlife).

Data transmission in an opportunistic network is challenging and completely different to routing in a conventional network. Ideally, a routing scheme for opportunistic networks should provide reliable data delivery, even when the network connectivity is intermittent or an end-to-end path is temporarily unavailable. Proper handling of mobility is the key to the success of mobile networking applications. Grossglauser et al. [15] showed that network capacity can be increased significantly by exploiting node mobility as a type of multi-user diversity. In [33], Small and Hass proposed a communication paradigm called *store-carry-and-forward* to facilitate data dissemination in opportunistic networks. Specifically, a node stores a data bundle before moving. Then, when the node connects with another node, it either forwards the data bundle to, or replicates it for, the other node. The process is repeated in the hope that the data bundle will eventually reach its destination.

Several data dissemination protocols have been proposed in recent years [5, 8–11, 13, 14, 16, 19, 19–21, 23, 28–30, 34, 35]. They can be classified into two categories: 1) *forwarding-based* protocols, which only have one copy of a data bundle in the network; and *replication-based* protocols, which have multiple copies. The latter are more suitable for opportunistic networks because multiple copies of a data bundle have a higher probability of reaching the destination node under the intermittent and opportunistic connectivity constraint.

In an opportunistic network, establishing a connection between two nodes involves three phases: the *probing phase*, the *meta-message exchange phase*, and the *data exchange phase*. In the probing phase, each node broadcasts a beacon periodically and listens for the beacons of its neighboring nodes. After the probing phase, two nodes establish a connection based on the beacons they receive from each other. Before exchanging data bundles, the nodes exchange meta-messages, such as details of the data bundles they have, the direction they are heading, and how much space they have in their storages. The meta-messages help each node determine which data bundles it should send, how many data bundles it can send, and the priority of each bundle. If the meta-messages are accurate, a node can make a good decision about the transmission; thus, the delivery rate may increase and the transmission overhead may reduced. Finally, in the data exchange phase, one node sends data to the other node or they exchange data bundles.

Because multiple copies of a data bundle may exist in replication-based protocols, it is necessary to exchange meta-messages before sending data bundles in order to avoid duplicating data that the receiving node already has in its storage. On receipt of a meta-message, a node selects the data bundles that should be replicated and sent to the other node, which means that the time and energy spent in the data exchange phase depends to a large extent on the accuracy of the meta-message. If the meta-message is accurate, only essential data bundles will be sent to the other node; thus, nodes can save the energy used to send/receive redundant data bundles. Intuitively, sending a summary, called a *Summary Vector* (SV), of data bundles seems reasonable. However, under the SV scheme, the size of meta-messages increases with the number of data bundles. As a result, meta-message exchange incurs a large overhead in terms of the transmission energy as well as the transmission time, which is usually short in opportunistic networks. To address these shortcoming, we propose an efficient meta-message exchange mechanism called *Known Vector* (KV) scheme. By minimizing the size of meta-messages, the KV scheme reduces the amount of energy required for meta-message exchange and allows more time for transmitting data bundles. It also prolongs the lifespan of nodes and increases the probability of successful delivery.

Using a comprehensive set of simulations, we compare the proposed KV scheme with the SV scheme under two replication-based protocols, namely, the Epidemic routing protocol and the PRoPHET (Probabilistic Routing Protocol using History of Encounters and Transitivity) routing protocol. The simulation results demonstrate that the KV scheme is a suitable meta-message exchange mechanism for both protocols. Moreover, it outperforms the SV scheme in terms of delivery and overhead performance when the buffer for nodes is limited.

The contribution of this study is three-fold:

1) To reduce the overhead of control message, we propose a new meta-message exchange mechanism called the *Known Vector* scheme.
2) Via simulations, we show that the proposed scheme can reduce the overhead of meta-message exchange and improve the delivery performance.
3) The proposed mechanism is suitable for replication-based protocols that exchange meta-messages between nodes to determine which data bundles should be transmitted.

The remainder of this paper is organized as follows. Section II provides background information on routing protocols for opportunistic networks. In Section III, we describe the proposed Known Vector

scheme; and in Section IV, we discuss the simulations performed to compare the performance of the KV with that of the SV scheme. Section V discusses the energy-memory trade-off, and VI contains some concluding remarks.

II. Background

Replication is the most popular design for opportunistic routing schemes. For instance, the *Epidemic Routing* scheme [34] sends identical copies of a message simultaneously over multiple paths to mitigate the effect of a single path failure; thus, it increases the likelihood of successful message delivery. However, flooding a network with duplicate data tends to be very costly in terms of traffic overhead and energy consumption.

To address the problem of excess traffic overhead caused by flooding, Harras *et al.* proposed a *Controlled Flooding* scheme to reduce the flooding cost while maintaining reliable message delivery [16]. The scheme controls flooding via three parameters, namely, *willingness probability*, *time-to-live*, and *kill time*. Additionally, once a message has been delivered to the receiver, a *Passive Cure* is generated to "heal" the network nodes that have been "infected" by the message. Therefore, by removing the excess traffic overhead problem and maintaining reliable data delivery, controlled flooding can reduce the network overhead substantially. Erramilli and Crovella [14] presented another approach called *Delegation Forwarding*, which controls the number of message replicas. The approach considers three metrics: the cost (i.e., the number of replicas per message), the success rate, and the average delay. The authors show that the expected cost of delegation forwarding declines to $O(\sqrt{N})$ compared to $O(N)$ under naive forwarding to high contact rate nodes. In [13], Erramilli and Crovella considered different message prioritization schemes based on delegation forwarding and assessed their performance in terms of the success rate, delay and cost.

Node mobility also affects opportunistic routing schemes. Some studies have shown that if the network mobility deviates from that of well-known random way-point mobility models (e.g., the Pursue Mobility Model [7] or the Reference Point Group Mobility Model [17]), the overhead incurred by epidemic routing and flooding-based routing schemes can be reduced by considering node mobility. For instance, the *Probabilistic Routing* scheme [23] calculates the *delivery predictability* from a node to the destination node based on the observed contact history, and forwards a message to a neighboring node if and only if that node has a higher delivery predictability value. Leguay *et al.* [19] modified the scheme by taking

the mobility pattern into account, i.e., a message is only forwarded to a neighboring node if the latter's mobility pattern is more similar to that of the destination node. In [19, 20], Leguay *et al.* showed that the modified scheme is more effective than previous schemes. Shih *et al.* [32] exploited the Levy walk mobility pattern to devise routing strategies and demonstrate that the proposed strategies are effective when information about destination is limited. In contrast, Liu and Wu [25] proposed an optimal probabilistic forwarding (OPF) protocol, which is based on the assumption that the mean inter-meeting time can be estimated from the contact history because of regular long-term mobility. The objective of OPF is to maximize the delivery performance while satisfying a certain constant on the number of forwarding per message. Using similar concept, Nelson *et al.* [27] proposed an encounter-based routing scheme, which assumes the future rate of node encounters can be roughly predicted by considering previous data.

Another class of opportunistic network routing schemes are based on encoding techniques that convert a message into a different format prior to transmission. For instance, Widmer and Boudec [36] proposed integrating *network coding* and epidemic routing techniques to reduce the number of transmissions required in a network. Lin *et al.* [22] compared DTN routing protocols with low-density erasure codes, and proposed E-NCP, which is based on network coding, to reduce the number of transmissions; however, the transmission delay increases slightly. In addition, Wang *et al.* [35] proposed combining *erasure coding* and a simple replication-based routing method to improve data delivery for the *worst delay performance* cases in opportunistic networks.

Based on the concept of erasure coding-based data forwarding discussed in [35], Liao *et al.* [21] designed the Estimation Based Erasure-Coding routing scheme (EBEC), which uses the Average Contact Frequency (ACF) estimate to adapt the delivery of erasure coded blocks. Meanwhile, Chen *et al.* [10] proposed HEC, a hybrid scheme that combines the advantages of erasure coding and *Aggressive Forwarding*. The HEC scheme has been enhanced by employing the techniques of sequential forwarding (HEC-SF), fully interleaving (HEC-FI), and block-based interleaving (HEC-BI) [11], as well as probabilistic forwarding (HEC-PF) [9].

III. Meta-Message Exchange Approaches

When two nodes encounter one another, they usually need to exchange some meta-messages to avoid sending duplicate data bundles. Many protocols apply the *Summary Vector* scheme. A summary vector is comprised of the identifiers of all the messages in a node's buffer. Each message's identifier is unique

throughout the network. In a replication-based protocol, nodes are normally required to exchange summary vectors in order to avoid sending/receiving redundant data bundles; however, exchanging the vectors usually incurs a large overhead. To reduce the overhead, we propose a scheme called *Known Vector*, which can be regarded as a pre-processor that eliminates unnecessary message identifiers in a summary vector. Every protocol that uses the Summary Vector scheme can apply the proposed Known Vector scheme to improve network performance. We apply the proposed scheme to the Epidemic Routing protocol and compare its performance with that of the Summary Vector scheme on the same protocol. In Section III-A, we provide an overview of the Epidemic Routing protocol and examine how the Summary Vector scheme works under the protocol. Then, we discuss the proposed Known Vector scheme in Section III-B.

A. Epidemic Routing

In Epidemic Routing, each node's buffer contains messages initiated by the node as well as messages relayed on behalf of other nodes. Each node also carries a fixed length summary vector that the mth bit will be flagged if the node carries the message with ID m. When two nodes encounter one another, they exchange their summary vectors. Each node then compares the received vector with its own vector and requests the messages that are not in its buffer. In this paper, we call a request for messages a *request vector*. The buffer of node x is denoted as BUF_x; and a summary vector and a request vector sent from node x to node y are denoted as $SV_{x,y}$ and $RV_{x,y}$ respectively. The following is an example of an encounter between two nodes. Suppose that, before the encounter, node i has messages M_1, M_2, and M_3 ($BUF_i = \{M_1, M_2, M_3\}$), and node j has messages M_3, M_4, and M_5 ($BUF_j = \{M_3, M_4, M_5\}$). When they meet, node i generates a summary vector $SV_{i,j} = \{M_1, M_2, M_3\}$ for node j, and node j generates a summary vector $SV_{j,i} = \{M_3, M_4, M_5\}$ for node i. After exchanging summary vectors, node i requests messages M_4 and M_5 by computing a request vector $RV_{i,j} = SV_{j,i} - BUF_i$, and node j request messages M_1 and M_2 by computing a request vector $RV_{j,i} = SV_{i,j} - BUF_j$. If the nodes have sufficient time to transmit all the requested messages, they will both have 5 messages (M_1 to M_5) after their encounter.

A potential problem of summary vector is that it is not very space efficient. E.g., in a network that uses 2 bytes as message ID, the summary vector will be 2^{16} = 65536 bits long, even if there is only one message in the buffer, while saving a message ID will take only 16 bits in memory. For the summary vector to be space efficient, there has to be more than $(2^{16})/16$ messages in this example, or $(2^n)/n$ messages in the buffer when n bits is used for message ID. The authors of epidemic routing are aware of the inefficiency

and propose using Bloom filter [6] to cut down the size of summary vector. Epidemic routing with Bloom filter provides a possible solution for DTNs routing. However, Bloom filter comes with an error rate, and exchanging vectors with an error rate is not always tolerable since two nodes do not necessarily have time to retransmit messages that are missed due to wrong hashing, e.g., in mission-critical DTNs like a search and rescue system, one specific packet arrives at base station or not may be a matter of life or death [12].

In this paper, we present another possibility to route in DTNs, Known Vector, which is error free and messages based. To have a fair comparison, we modify epidemic routing to an error free version by storing one message ID for each message saved in the buffer instead of flipping a bit and hashing the summary vector. Unlike the original epidemic routing, the modified epidemic routing protocol simulated in our paper has a varied length of summary vector. The summary vector mentioned thereafter in this paper denotes our modified version.

B. The Proposed Known Vector Approach

The Summary Vector can be viewed as a node-based approach, while the Known Vector can be viewed as a message-based approach. In a known vector, each message on a node keeps a vector, which is a list of node IDs that already have the message. Note that the vector is not an accurate record of global network knowledge. It can be regarded as a message's meta-data, which is duplicated and sent with the message. The known vectors of copies of the message on different nodes might not be the same due to different relaying paths. The copy of a message M_k that resides on node x is denoted as $M_{k,x}$, and the copy's known vector is denoted as $KV_{M_{k,x}}$. A message M_k initiated by node x does not have any record in its known vector $M_{k,x}$, i.e., $KV_{M_{k,x}} = \{\}$ at the outset. When two nodes, i and j, meet the state of node i is $BUF_i = \{M_1, M_2\}$, $KV_{M_{1,i}} = \{m, j, n\}$, $KV_{M_{2,i}} = \{a, b\}$, and the state of node j is $BUF_j = \{M_3, M_1\}$, $KV_{M_{3,j}} = \{a\}$, $KV_{M_{1,j}} = \{m\}$. The nodes then implement the following steps:

1) Generate and exchange summary vectors:

   Node i generates $SV_{i,j}$, which contains the IDs of messages whose known vectors do not include j. The algorithm used to generate a summary vector is shown in Algorithm 1. Specifically, the summary vector $SV_{i,j}$ only contains M_2, but $SV_{j,i} = \{M_3, M_1\}$.

2) Send a request:

   Node i generates $RV_{i,j} = SV_{j,i} - BUF_i$. e.g., $RV_{i,j} = \{M_3\}$, and $RV_{j,i} = \{M_2\}$.

Algorithm 1 Summary vector generation of node i when it encounters node j in the Known Vector scheme

Require: $i \neq j$
Ensure: $SV_{i,j}$
  $SV_{i,j} \leftarrow \{\}$
  for all messages $M_{k,i}$ in the buffer of node i **do**
    if j *not* in $KV_{M_{k,i}}$ **then**
      $SV_{i,j} \leftarrow SV_{i,j} \cup \{j\}$
    end if
  end for

3) Transmit requested messages:

   For every message M_k that node i transmits to node j, it first duplicates $M_{k,i}$ as $M'_{k,i}$ including $KV_{M_{k,i}}$. After transmitting $M'_{k,i}$ to node j, node i adds j to $KV_{M_{k,i}}$. When node j receives $M'_{k,i}$, it saves it in the buffer as $M_{k,j}$ with $KV_{M_{k,j}} = KV_{M'_{k,i}} \cup \{i\}$. Thus, after node i transmits $M'_{2,i}$ to node j, $KV_{M_{2,i}}$ becomes $\{a, b, j\}$, and node j saves $M'_{2,i}$ as $M_{2,j}$ with $KV_{M_{2,j}} = \{a, b, i\}$. In addition, after node j transmits $M'_{3,j}$, $KV_{M_{3,j}}$ becomes $\{a, i\}$, and node i saves $M'_{3,j}$ as $M_{3,i}$ with $KV_{M_{3,i}} = \{a, j\}$.

4) Update known vectors:

   Node i can infer that node j already has the messages in the set $SV_{i,j} - RV_{j,i}$ because node j is supposed to request every message in $SV_{i,j}$ that it does not have in its storage. Therefore, node i adds j to the known vector of every message in the set $SV_{i,j} - RV_{j,i}$. For example, node j can infer that node i already has M_1, and then $KV_{M_{1,j}}$ becomes $\{m, i\}$.

Table I shows all possible scenarios for a message M_k residing on node i when the latter encounters node j. The second column shows if the known vector of the copy of message M_k on node i (denoted as $K_{M_{k,i}}$) contains j. The third column shows if node j also has message M_k (denoted as $M_{k,j}$). If node j already has $M_{k,j}$, the fourth column shows whether the known vector of $M_{k,j}$ contains i. The fifth column explains the possible scenarios of message M_k.

IV. Evaluation

We use a Java-based simulator, called The ONE [2], to evaluate the performance of the Summary Vector and Known Vector schemes in terms of the delivery ratio and transmission overhead on the Epidemic and PRoPHET routing protocols. The delivery ratio is defined as: (the number of messages delivered to

TABLE I
POSSIBLE SCENARIOS FOR A MESSAGE M_k RESIDING ON NODE i WHEN NODE i ENCOUNTERS NODE j

Scenario	$KV_{M_{k,i}}$ contains j	Node j has $M_{k,j}$	$KV_{M_{k,j}}$ contains i	Explanation [1]
1	No	No	No	1.M_k is added to $SV_{i,j}$. 2.Node j requests M_k after receiving $SV_{i,j}$. 3.After M_k has been transmitted, j is added to $KV_{M_{k,i}}$, and i is added to $KV_{M_{k,j}}$.
2	No	Yes	No	1.M_k is added to both $SV_{i,j}$ and $SV_{j,i}$ (because i does not know j has M_k, and j does not know i has M_k). 2.No request for M_k from node i or node j. 3.Node i infers that node j has M_k and adds j to $KV_{M_{k,i}}$; node j infers that node i has M and adds i to $KV_{M_{k,j}}$.
3	No	Yes	Yes	1.M_k is added to $SV_{i,j}$ but not to $SV_{j,i}$ (because node j knows that node i has M_k from $KV_{M_{k,j}}$). 2.No request for M_k from node j. 3.Node i infers that node j has M_k and adds j to $KV_{M_{k,i}}$.
4	Yes	No	No	1.$KV_{M_{k,i}}$ shows that node j has M_k, but j no longer has M_k. This may happen because the buffer of node j is full and M_k is selected for deletion. 2.M_k is not added to $SV_{i,j}$, and node j does not request M_k from node i.
5	Yes	Yes	No	1.M_k is not in $SV_{i,j}$, but it is in $SV_{j,i}$ (node j does not know that node i has M_k because i is not in $KV_{M_{k,j}}$). 2.No request for M_k from node i. 3.Node j infers that node i has M_k and adds i to $KV_{M_{k,j}}$.
6	Yes	Yes	Yes	1.M_k is not added to $SV_{i,j}$ or $SV_{j,i}$. 2.No action regarding M_k.

the destinations) / (the number of messages created in the simulation); and the transmission overhead is defined as: (the total number of bytes transmitted, including messages and control bytes) / (the total number of bytes of all the messages created in the simulation).

We evaluated three network scenarios, namely, ZebraNet [1], SLAW [18], and Community [24], which are publicly available for research purposes. ZebraNet is an opportunistic wildlife network, while the other

[1]To check if KV *contains* the encountering party or not, a node searches its memory. The searching action is basically memory reading, which takes time and energy. But because both memory access time and energy is very slim, we choose to neglect it and focus on the transmission overhead in this paper.

TABLE II
THE PROPERTIES OF THE THREE NETWORK SCENARIOS

Trace Name	ZebraNet	SLAW	Community
Duration	16 days	3 days	11,500 seconds
Devices participating	34	30	67
Number of contacts	31,693	3899	37,266
Avg. # Contacts/pair/day	3.53086	2.98774	188.53945
Total contact time (seconds)	298,779	1,402,203.5	275,535
Avg. contact time/pair/day	33.2864	1,074.4854	908.7386

two are people networks. Table II lists the basic properties of the three scenarios.

ZebraNet [1] utilizes a random movement model to create multiple zebra movement traces based on the collected movement patterns of a particular zebra. In this study, we create 34 movement traces with a range of 5000 $\times$ 5000 m^2 in a 16-day period.

SLAW (Self-similar Least Action Walk) [18] is a mobility model that imitates human walks patterns. It produces synthetic human walk traces comprised of the following four features observed in human mobility: (i) truncated power-law flights and pause-times; (ii) heterogeneously bounded mobility areas; (iii) truncated power-law inter-contact times; and (iv) fractal waypoints. We use a tool provided by the authors to generate 30 human walk traces with a range of 5000 $\times$ 5000 m^2 in a 3-day period.

Community is a mobility model developed by the authors of [24]. We use the following settings, which are given in the paper: a 3000 $\times$ 1500 m^2 area is divided into 3 $\times$ 4 sub-areas (i.e., each sub-area is 100 $\times$ 375 m^2). The "gathering place" is in the third row and the fourth column, while other sub-areas are "communities". In each community, and at the gathering place, there is a fixed (non-mobile) node that can also act as a gateway for that community. Each node has a home community that it is more likely to visit than other communities; and for each community, there are five nodes that treat it as their home community. Hence, there are 67 nodes in total (55 mobile nodes and 12 fixed nodes). When a node is at its home community, there is a probability of 0.8 that it will go to the gathering place and 0.2 that it will go elsewhere; and when a node is elsewhere, there is a probability of 0.9 that it will return home and 0.1 that it will go elsewhere.

In each simulation run, the source and destination pairs are selected at random from among all participating peers. The source node transmits messages in the first 10% of the simulation time at a rate of 200 seconds/message in ZebraNet, and at a rate of 40 seconds/message in SLAW. In Community, each of

TABLE III
THE SIMULATION SETTINGS OF THE THREE NETWORK SCENARIOS

Trace Name	ZebraNet	SLAW	Community
Message generation period (seconds)	1∼138,240	1∼25,920	501∼3,500
Message generation rate (seconds/message)	200	40	2.5
# of messages generated	691	648	1200

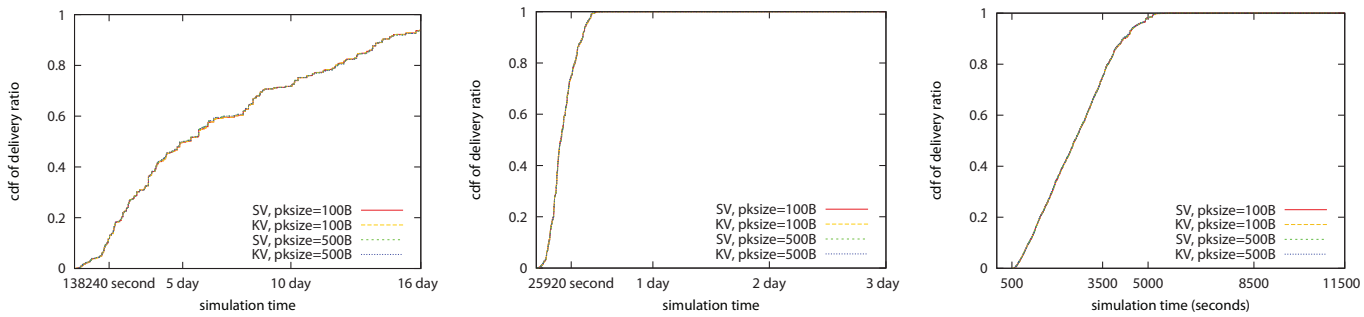
two randomly chosen community gateways generates a message for another gateway in the community or the gathering place every ten seconds. Then, five seconds after a message is generated by a gateway, each of two randomly chosen mobile nodes transmits the message to a randomly chosen destination (either a mobile node or a gateway). In the Community scenario, there is a warm up period of 500 seconds at the beginning of the simulations, after which messages are generated for 3000 seconds, and the simulation runs for another 8000 seconds. The above settings are summarized in Table III.

For simplicity, we assume that data transmission between nodes is via ZigBee [3] with a fixed rate of 240 Kbps, and the size of each message is either 500 bytes or 100 bytes. We vary the buffer size in each set of simulations, and compare the delivery and transmission performance of the two meta-message exchange schemes on the Epidemic and PRoPHET routing protocols. All the results are based on the average performance of 25 simulation runs for each network configuration.

In the following subsections, each figure contains three subfigures, which correspond to the three scenarios respectively. We evaluate cases of an infinite buffer in Section IV-A and cases of a finite buffer in Section IV-A. For the latter, we evaluate the performance by using three buffer sizes for each message size. The delivery performance and the transmission overhead are presented as cumulative distribution function (CDF) curves.

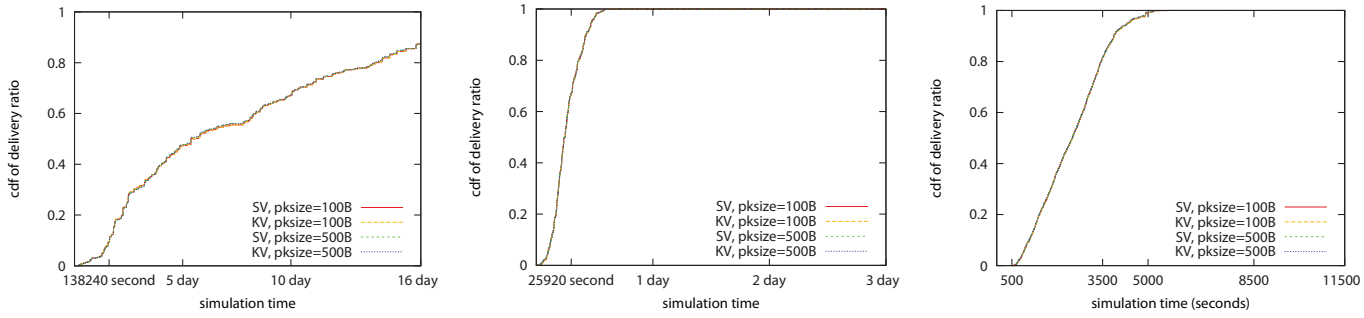
A. Evaluation I: The Infinite Buffer Case

In the first set of simulations, we evaluate the delivery and transmission performance of SV and KV schemes with infinite buffers. Figure 1 and Figure 2 show, respectively, the delivery performance of the two schemes on the Epidemic and PRoPHET routing protocols in the three scenarios. The results in the six subfigures indicate that the SV and KV schemes perform almost the same regardless of the message size. The delivery ratio in SLAW and Community reaches 100% quickly, which is much better than the delivery ratio in ZebraNet. This is because the total contact time in SLAW and Community is much longer than that in ZebraNet. We observe that ZebraNet's delivery performance is better on the Epidemic protocol



(a) ZebraNet (b) SLAW (c) Community

Fig. 1. The delivery performance of the Epidemic routing protocol with infinite buffers in the three scenarios.

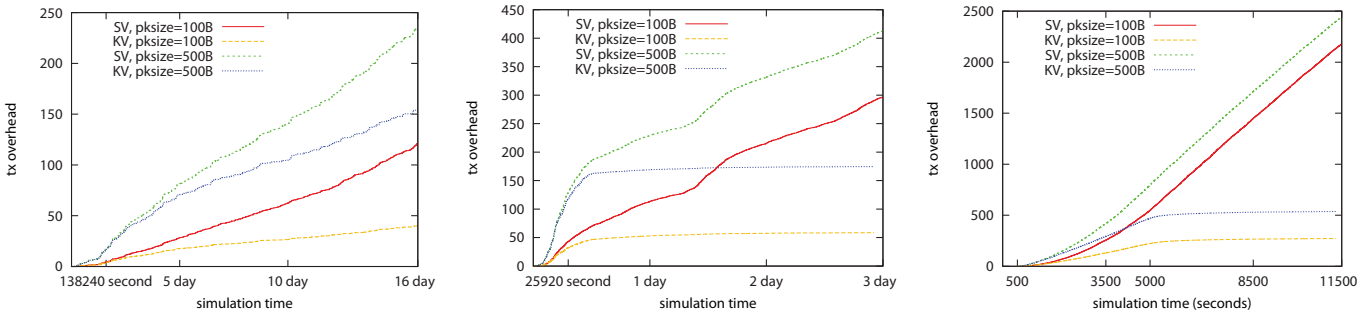


(a) ZebraNet (b) SLAW (c) Community

Fig. 2. The delivery performance of the PRoPHET routing protocol with infinite buffers in the three scenarios.

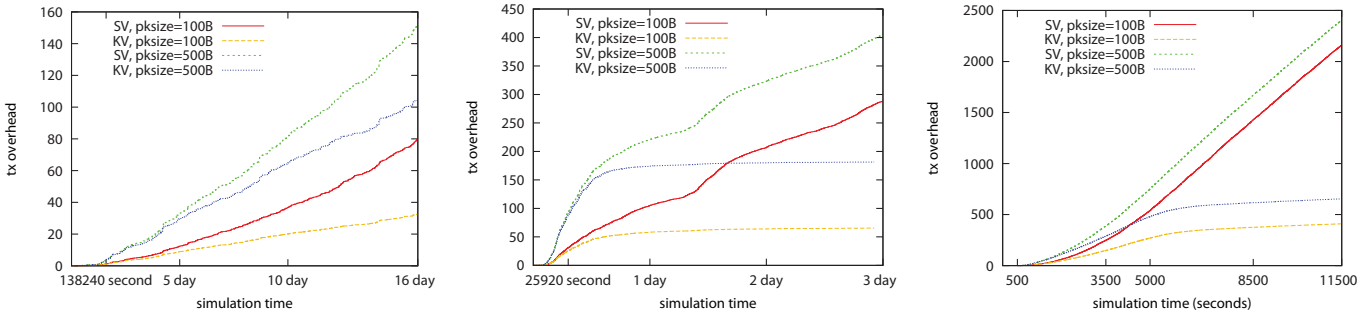
than on the PRoPHET protocol. In the SLAW scenario, the delivery ratios reach 100% on both routing protocols, but the delivery performance is a little better on Epidemic because it reaches 100% earlier. In contrast, the delivery performance of Community is a little better on PRoPHET because this scenario is more regular than the other two scenarios and the PRoPHET protocol can make better predictions in this scenario. There are no significant differences between delivery ratios of SV and KV in this case, because a node with infinite buffer can save a very long summary vector, or known vector, without taking up space for real messages. But we can see the differences in secion IV-B.

Figures 3 and 4 show the transmission overhead of KV and SV on the Epidemic and PRoPHET protocols respectively. In each case, the overhead increases with the simulation time. However, SV's transmission overhead increases much more rapidly than that of KV. It is worth noting that, in the SLAW and Community scenarios, KV's transmission overhead almost reaches a bound at about the same time that the delivery ratios reach 100%. This shows that KV can reduce the overhead by preventing the transmission of unnecessary summary vectors. By contrast, in the SV scheme, summary vectors are always exchanged for each message in the buffer.



(a) ZebraNet (b) SLAW (c) Community

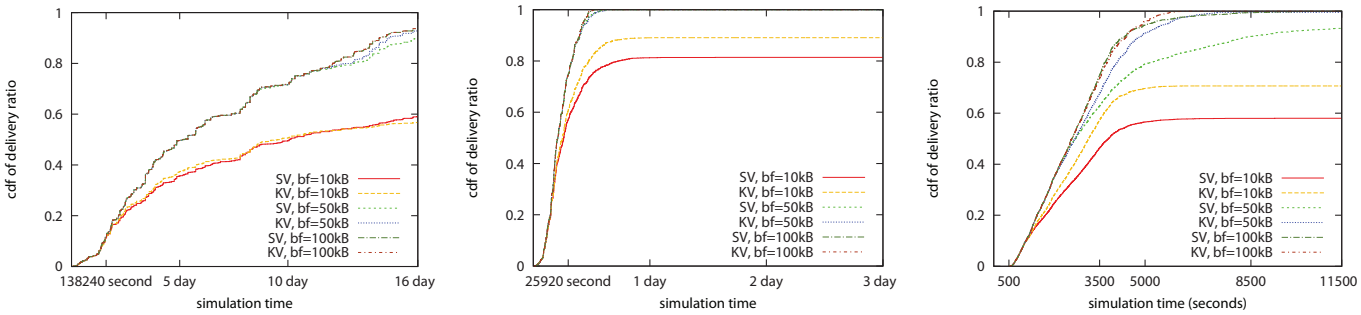
Fig. 3. The transmission performance of the Epidemic routing protocol with infinite buffers in the three scenarios.



(a) ZebraNet (b) SLAW (c) Community

Fig. 4. The transmission performance of the PRoPHET routing protocol with infinite buffers in the three scenarios.

We can use an analogy to explain this. When a new semester starts, students exchange their stories about summer vacations and trips. Questions like ”Have you heard story A, B, and C?” are the vectors for stories. Summary Vector is like a very patient friend, he tells anyone all the story codes he knows repeatedly, and reveals the stories to others who ask for them. And he will keep asking everyone he meets on the road if they have heard all the stories he knows, by reading all the story codes. Known Vector, on the other hand, has very good memory. She remembers who told her which stories, and to whom she told which stories. So when she meets someone on the road, she scans her memory and only tell the person the story codes that he dose not know, and vice versa. The price she has to pay, is memory. She has to keep ’who knows what’ in her memory. The gain, is energy. Unlike her friend Summary Vector, who has to repeat all the story codes to every person and is exhausted at the end of day, she only has to ask what is new to her encounter. And energy is more precious resource in WSNs than memory. We have more detailed discussion in V. In figure 3 and 4, KV saves from 35% to 88% of transmission overhead in different scenarios.



(a) ZebraNet (b) SLAW (c) Community

Fig. 5. The delivery performance of the Epidemic routing protocol with finite buffers in the three scenarios when the packet size is 100 bytes.

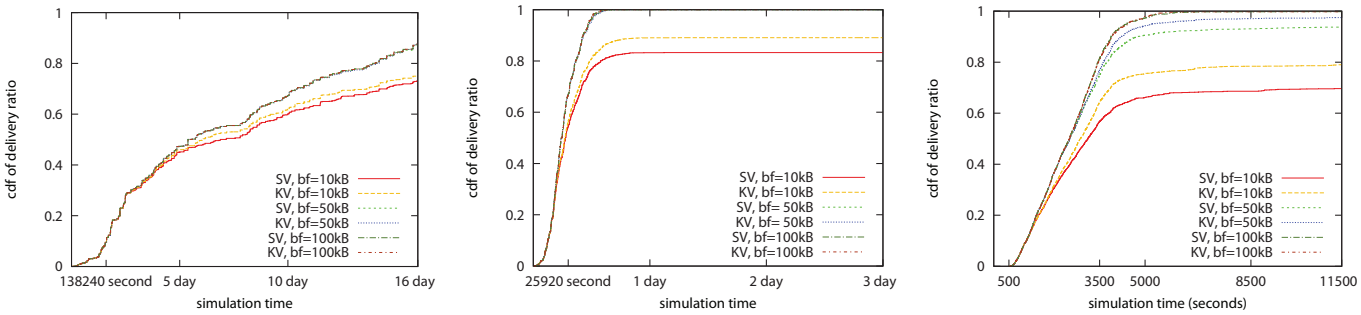
B. Evaluation II: The Finite Buffer Case

In this section, we evaluate the SV and KV schemes with finite buffers. Each simulation setting is evaluated with three buffer sizes, which are designed to accommodate 100, 500, and 1000 messages respectively. Therefore, for a 100-byte message, the respective buffer sizes are 10k bytes, 50k bytes, and 100k bytes. Although the buffer that can accommodate 1000 messages seems sufficient for all messages in ZebraNet (which generates 691 messages) and SLAW (which generates 648 messages), some space is needed to save known vectors. Furthermore, messages can be dropped when a node needs space to generate summary vectors for exchange or to receive summary vectors from other nodes.

In the following, we consider the delivery performance of KV and SV for three buffer sizes. Figure 5 and Figure 6 show the delivery performance on the Epidemic and PRoPHET protocols respectively when the message size is 100 bytes. For both figures, we can see that when the buffer is of size 10KB, KV performs better than SV. This is because SV memories the message IDs, and KV memories the node IDs those have the message, and message ID usually takes up more bytes than node ID since there are usually more messages than nodes in a system. When the memory is relatively small, SV will take up more space than KV, which causes more packet drop, and lower delivery ratio. As expected, the delivery performance improves as the buffer size increases.

The delivery performances of SV and KV are comparable when the buffer can accommodate 500 or 1000 messages; however, KV performs much better than SV in the SLAW and Community scenarios when the buffer can only accommodate 100 messages. The results indicate that the KV scheme is superior to the SV scheme, especially when the end devices are buffer-constrained (e.g., sensors and hand-helds devices).

In the SLAW and the Community scenarios, we observe that the delivery ratios of the buffer sizes that



(a) ZebraNet (b) SLAW (c) Community

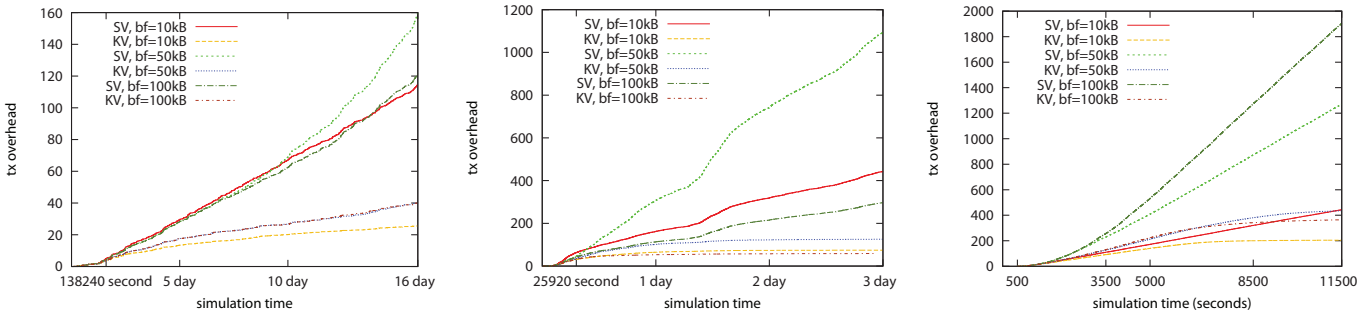
Fig. 6. The delivery performance of the PRoPHET routing protocol with finite buffers in the three scenarios when the packet size is 100 bytes.

can only accommodate 100 messages are bound to some points under both schemes. The reason is as follows. Although SLAW and Community have better contact opportunities than ZebraNet (as shown in Table II), some messages could be dropped because the buffer size is too small and there are only a few copies of messages. As a result, some messages may never be delivered, and the delivery ratios are bound to some points.

Next, we consider the transmission overhead of KV and SV with different buffer sizes. Figure 7 and Figure 8 show, respectively, the transmission performance of the schemes on the Epidemic and PRoPHET protocols when the message size is 100 bytes. Similar to the infinite buffer cases, although the transmission overhead increases with the simulation time, the KV scheme outperforms the SV scheme in all 6 subfigures. It is clear that the KV scheme is more effective than the SV scheme in reducing the traffic overhead. For example, for Epidemic routing in the SLAW scenario, given a 100-byte message, the KV scheme reduces the traffic overhead by about 83%, 89%, and 80% when the buffer size is 10k, 50k, and 100k bytes respectively. Again, the results show that the KV scheme is superior to the SV scheme when the buffer size of each network participant is limited.

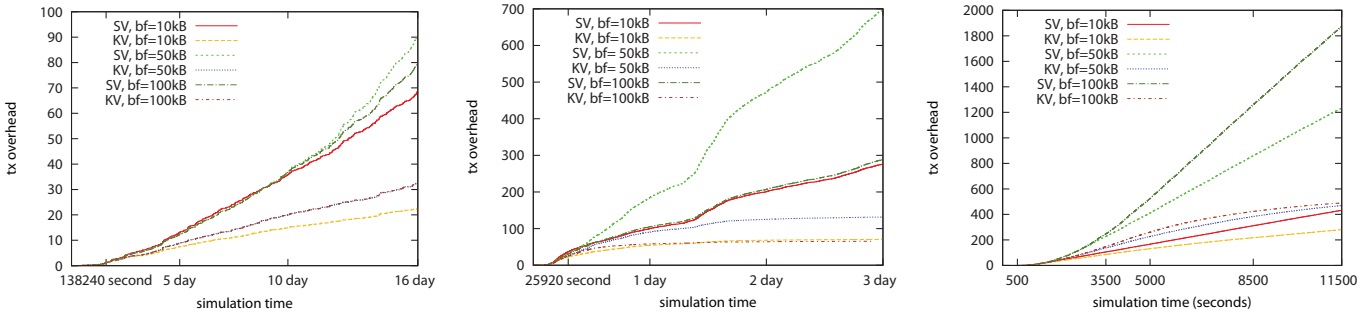
V. Discussion

Based on the survey of [4], the communication module has an energy consumption much higher than the computation module in WSNs, and transmitting one bit may consume as much as executing a few thousands instructions. Mathur et al. [26] also argued that the emergence of the new generation of NAND flash storage has added a new variable to the design equation, as storage may be used to reduce network traffic, and we support their point of view. We propose that all the KV can be saved in RAM if it is large



(a) ZebraNet (b) SLAW (c) Community

Fig. 7. The transmission performance of the Epidemic routing protocol with finite buffers in the three scenarios when the packet size is 100 bytes.



(a) ZebraNet (b) SLAW (c) Community

Fig. 8. The transmission performance of the Prophet routing protocol with finite buffers in the three scenarios when the packet size is 100 bytes.

enough. Otherwise, most of the KV can be saved in the NAND flash, while only the message ID and a pointer to the address of the message vector in the NAND flash reside in the RAM. According to the same paper[26], reading costs 0.0322μJ and has 1.761us latency per byte. Polastre et al.[31] measured that MCU+Radio Rx costs 21.8mA, MCU+Radio Tx 19.5mA, and MCU+Flash Read 4.1mA. And because flash reading speed is a lot faster than radio transmission, the energy cost is almost neglectable.

VI. Conclusion

In this paper, we study meta-message exchange in opportunistic network routing protocols, and observe that the transmission overhead may increase substantially as the number of messages buffered on each node increases. To resolve the problem, we propose a novel approach called Known Vector. Using a comprehensive set of simulations, as well as realistic network mobility traces, we compare the proposed scheme's performance with that of the Summary Vector meta-message exchange scheme on the Epidemic and ProPHET routing protocols. The results show that the two schemes are comparable when the network buffer is infinite. However, when the network buffer is constrained, the Known Vector scheme is far

superior to the Summary Vector in terms of delivery performance and transmission overhead. Moreover, the KV scheme is simple and it is generalizable to other opportunistic routing protocols.

VII. Acknowledgments

We are grateful to the editors and anonymous reviewers for their insightful comments. This paper is based on research supported by the National Science Council of Taiwan under Grant No. NSC 97-2628-E-001-007-MY3.

References

[1] CRAWDAD Project. http://crawdad.cs.dartmouth.edu/.

[2] The opportunistic network environment simulator. http://www.netlab.tkk.fi/tutkimus/dtn/theone/.

[3] ZigBee alliance. http://www.zigbee.org/.

[4] G. Anastasi, M. Conti, M. Di Francesco, and A. Passarella. Energy conservation in wireless sensor networks: A survey. *Ad Hoc Netw.*, 7(3):537–568, May 2009.

[5] M. H. Anisi, A. H. Abdullah, Y. Coulibaly, and S. A. Razak. Edr: efficient data routing in wireless sensor networks. *International Journal of Ad Hoc and Ubiquitous Computing*, 12(1):46–55, 2013.

[6] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13:422–426, 1970.

[7] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless Communication and Mobile Computing Journal*, 2(5):483–502, 2002.

[8] C.-M. Chao and M.-W. Lu. Energy-efficient transmissions for bursty traffic in underwater sensor networks. *International Journal of Ad Hoc and Ubiquitous Computing*, 13(1):1–9, 2013.

[9] L.-J. Chen, C.-L. Tseng, and C.-F. Chou. On using probabilistic forwarding to improve hec-based data forwarding for opportunistic networks. In *IFIP Conference on Embedded and Ubiquitous Computing*, pages 101–112, 2007.

[10] L.-J. Chen, C.-H. Yu, T. Sun, Y.-C. Chen, and Hao-huaChu. A hybrid routing approach for opportunistic networks. In *ACM SIGCOMM Workshop on Challenged Networks*, pages 213–220, 2006.

[11] L.-J. Chen, C.-H. Yu, C.-L. Tseng, H. hua Chu, and C.-F. Chou. A content-centric framework

for effective data dissemination in opportunistic networks. *IEEE Journal of Selected Areas in Communications*, 26(5):761–772, June 2008.

[12] Y.-C. Chen, J.-H. Huang, L.-J. Chen, and H. Polly. Yushannet: A delay-tolerant wireless sensor network for hiker tracking in yushan national park. In *Tenth International Conference on Mobile Data Management: Systems, Services and Middleware*, pages 379–380, 2009.

[13] V. Erramilli and M. Crovella. Forwarding in opportunistic networks with resource constraints. In *ACM MobiCom Workshop on Challenged Networks*, 2008.

[14] V. Erramilli, M. Crovella, A. Chaintreau, and C. Diot. Delegation forwarding. In *ACM MobiHoc*, 2008.

[15] M. Grossglauser and D. Tse. Mobility increases the capacity of ad-hoc wireless networks. *IEEE/ACM Transactions on Networking*, 10(4):477–486, August 2002.

[16] K. A. Harras, K. C. Almeroth, and E. M. Belding-Royer. Delay Tolerant Mobile Networks (DTMNs): Controlled Flooding in Sparse Mobile Networks. In *IFIP Networking*, pages 1180–1192, 2005.

[17] X. Hong, M. Gerla, R. Bagrodia, and G. Pei. A group mobility model for ad hoc wireless networks. In *ACM International Workshop on Modeling, Analysis and Simulation of Wirelessand Mobile Systems*, pages 53–60, 1999.

[18] K. Lee, S. Hong, S. J. Kim, I. Rhee, and S. Chong. Slaw: A mobility model for human walks. In *IEEE Infocom*, 2009.

[19] J. Leguay, T. Friedman, and V. Conan. Dtn routing in a mobility pattern space. In *ACM SIGCOMM Workshop on Delay Tolerant Networks*, 2005.

[20] J. Leguay, T. Friedman, and V. Conan. Evaluating mobility pattern space routing for dtns. In *IEEE Infocom*, 2006.

[21] Y. Liao, K. Tan, Z. Zhang, and L. Gao. Estimation based erasure-coding routing in delay tolerant networks. In *International Wireless Communications and Mobile Computing Conference*, 2006.

[22] Y. Lin, B. Li, and B. Liang. Efficient network coded data transmissions in disruption tolerant networks. In *IEEE Infocom*, 2008.

[23] A. Lindgren and A. Doria. Probabilistic routing protocol for intermittently connected networks. Technical report, draft-irtf-dtnrg-prophet-08.txt, IETF Internet draft, October 2010.

[24] A. Lindgren, A. Doria, and O. Schelen. Probabilistic routing in intermittently connected networks.

ACM SIGMOBILE Mobile Computing and Communications Review, 7(3):19–20, July 2003.

[25] C. Liu and J. Wu. An optimal probabilistic forwarding protocol in delay tolerant networks. In *ACM MobiHoc*, 2009.

[26] G. Mathur, P. Desnoyers, P. Chukiu, D. Ganesan, and P. Shenoy. Ultra-low power data storage for sensor networks. *ACM Trans. Sen. Netw.*, 5(4):33:1–33:34, Nov. 2009.

[27] S. C. Nelson, M. Bakht, and R. Kravets. Encounter-based routing in dtns. In *IEEE Infocom*, 2009.

[28] A. D. Nguyen, P. Senac, V. Ramiro, and M. Diaz. Pervasive intelligent routing in content centric delay tolerant networks. In *Dependable, Autonomic and Secure Computing (DASC), 2011 IEEE Ninth International Conference on*, pages 178–185, 2011.

[29] M.-S. Pan and Y.-C. Tseng. Zigbee-based long-thin wireless sensor networks: address assignment and routing schemes. *International Journal of Ad Hoc and Ubiquitous Computing*, 12(3):147–156, 2013.

[30] A. Petz, C.-L. Fok, C. Julien, B. Walker, and C. Ardi. Network coded routing in delay tolerant networks: an experience report. In *Proceedings of the 3rd Extreme Conference on Communication: The Amazon Expedition*, ExtremeCom '11, pages 4:1–4:6, New York, NY, USA, 2011. ACM.

[31] J. Polastre, R. Szewczyk, and D. Culler. Telos: enabling ultra-low power wireless research. In *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, pages 364–369, 2005.

[32] M. Shin, S. Hong, and I. Rhee. Dtn routing strategies using optimal search patterns. In *ACM MobiCom Workshop on Challenged Networks*, 2008.

[33] T. Small and Z. J. Haas. Resource and performance tradeoffs in delay-tolerant wireless networks. In *ACM SIGCOMM Workshop on Delay Tolerant Networks*, 2005.

[34] A. Vahdat and D. Becker. Epidemic routing for partially-connected ad hoc networks. Technical Report CS-2000-06, Duke University, 2000.

[35] Y. Wang, S. Jain, M. Martonosi, and K. Fall. Erasure coding based routing for opportunistic networks. In *ACM SIGCOMM Workshop on Delay Tolerant Networks*, pages 229–236, 2005.

[36] J. Widmer and J.-Y. L. Boudec. Network coding for efficient communication in extreme networks. In *ACM SIGCOMM Workshop on Delay Tolerant Networks*, 2005.