RESEARCH ARTICLE

HUBCODE: hub-based forwarding using network coding in delay tolerant networks†
Shabbir Ahmed* and Salil S. Kanhere

School of Computer Science and Engineering, The University of New South Wales, Sydney, Australia

ABSTRACT

Most people-centric delay tolerant networks have been shown to exhibit power-law behavior. Analysis of the temporal connectivity graph of such networks reveals the existence of hubs, a fraction of the nodes, which are collectively connected to the rest of the nodes. In this paper, we propose a novel forwarding strategy called HubCode, which seeks to use the hubs as message relays. The hubs employ random linear network coding to encode multiple messages addressed to the same destination, thus reducing the forwarding overheads. Further, the use of the hubs as relays ensures that most messages are delivered to the destinations. Two versions of HubCode are presented, with each scheme exhibiting contrasting behavior in terms of the computational costs and routing overheads. We formulate a mathematical model for message delivery delay and present a closed-form expression for the same. We validate our model and demonstrate the efficacy of our solutions in comparison with other forwarding schemes by simulating a large-scale vehicular delay tolerant network using empirically collected movement traces of a city-wide public transport network. Under pragmatic assumptions, which account for short contact durations between nodes, our schemes outperform comparable strategies by more than 20%. Copyright © 2011 John Wiley & Sons, Ltd.

KEYWORDS
DTN; Routing; Network Coding; Power-law; Hubs; Forwarding

*Correspondence
Shabbir Ahmed, School of Computer Science and Engineering, The University of New South Wales, Sydney, Australia.
E-mail: shabbira@cse.unsw.edu.au

1. INTRODUCTION

Delay tolerant networks (DTN) are a type of challenged networks, wherein the contacts between the communicating devices are intermittent. Consequently, a contemporaneous end-to-end path between the source and destination rarely exists. Of particular interest are the networks that are formed by people in urban environments. These include (i) pocket switched networks [1], wherein personal communication devices carried by humans self-organize to form an intermittently connected network, and (ii) vehicle-based DTN [2], in which WiFi routers mounted on vehicles can communicate with each other.

Message forwarding is one of the most challenging aspects of DTN because of the inherent intermittent connectivity [3]. However, knowledge of fundamental properties of the underlying network can be helpful in making better forwarding decisions. In particular, the aforementioned people-centric networks have been shown to follow power-law behavior [1,4,5]. In such networks, a small percentage of the nodes, often referred to as hubs [6], are known to have significantly higher connectivity (i.e., high node degree) as compared with the rest of the nodes. Consequently, most nodes can be reached from every other node by a small number of hops, via the hubs.

A few forwarding schemes have been proposed, which exploit these power-law properties [7,8]. The general idea is to rank nodes based on popularity metrics such as node centrality. A node then forwards a message to another node, if the latter has a higher rank than the former. These schemes have been shown to perform effectively, under the assumption that the nodes can exchange unlimited data in an encounter. However, in reality, contact durations between nodes in people-centric DTN are often only few seconds long [1]. Consequently, the most popular nodes, which concentrate all of the forwarding traffic, can often only exchange limited number

†Parts of this work have been published in “HUBCODE: message forwarding using hub-based network coding in delay tolerant networks”. Proceedings of MSWiM ’09, October 2009; 288–296.
of messages and in effect act as bottlenecks in the forwarding process.

In this paper, we seek to address this particular problem by employing the theory of network coding [9], which has been shown to attain maximum information flow in a network. We propose a novel forwarding strategy, HubCode, which exploits the power-law properties of the network by directing all forwarding traffic to the hubs. In other words, the hubs form a data conduit. Messages are then forwarded within the data conduit (i.e., only among hubs) using random linear network coding, wherein multiple messages addressed to the same destination are combined to form a single encoded message. Because randomly selected coefficients are used in the coding process, each encoded message is useful to the destination, thus reducing the propagation of redundant messages.

In the basic version of HubCode, the hubs exchange the coefficient matrices of the encoded messages prior to data exchange to select the messages for forwarding. The resulting overhead, which is $O(n^2)$ for $n$ messages, can be fairly significant. As a result, during short contact durations, the hubs may not get a chance to forward coded messages, because most of the contact opportunity is used for exchanging coefficients. To reduce this overhead, we propose an alternate approach, wherein the hubs do not exchange the entire coefficient matrices but rather only exchange a list of native messages. The resulting overhead is just $O(n)$. However, the hubs now need to decode the messages (i.e., solve linear equations), which is computationally expensive. On the contrary, in the basic version, only the destination decodes the messages, thus simplifying the processing at the hubs. These two versions address the important trade-off between routing overhead and computational complexity.

We evaluate the performance of our proposed schemes and compare them with other forwarding protocols using traces collected from a large-scale (>1000 nodes) real-world bus-based DTN. Under realistic assumptions, which account for the limited data exchange possible during short encounters, our schemes achieve 20% higher delivery ratio than comparable strategies. In addition, our schemes achieve about 50% savings in delivery cost. Empirical evaluations also suggest that our schemes are more resilient against random node failure compared with other protocols. Besides, comprehensive analyses have been carried out to examine the effect of varying traffic loads, message lengths, and hub sizes on delivery performance of HubCode.

We have also formulated a mathematical model to estimate the message delivery delay. Closed-form expressions, which serve as an upper bound in estimating message delivery delay, are presented for our proposed schemes. Simulation results corroborate our analytical formulation especially when there are sufficient numbers of hubs that act as message relays.

The rest of the paper is organized as follows: Section 2 discusses related work. Section 3 presents the details of the HubCode schemes. The mathematical model to estimate message delivery delay is described in Section 4. In Section 5, we present the results from our simulations, and finally, Section 6 concludes the paper. A preliminary version of this work has been presented in [10].

## 2. RELATED WORK

Ahlswede et al. [9] first introduced the theory of network coding and showed that it can achieve maximum information flow in a network, in the context of multicasting. In recent years, researchers have demonstrated that network coding can improve not only the performance of wired networks [11] under link disruption but also the throughput of wireless networks for unicast [12–14] as well as broadcast transmissions [15]. Li and Li [14] presented theoretical results on the application of network coding for unicast transmissions. The works presented in [12] and [13] focus on practical issues. They demonstrate that network coding can benefit from leveraging the broadcast advantage in wireless networks. In [12], the authors also presented empirical results from testbed deployments and showed that their proposed method can increase the throughput several folds. However, their methods are suited for densely connected networks such as mesh networks, where the nodes can overhear their neighbors’ transmissions. Consequently, these schemes are not effective for intermittently connected networks such as DTN.

A few papers [16,17] have studied the use of network coding in DTN. Zhang et al. [17] and Widmer and Boudec [16] have studied the benefits of using random linear coding (RLC) for unicast transmissions in DTN. RLC uses simple flooding to distribute the messages in the network. However, rather than transmitting the native messages, a node combines these messages to form an encoded message and forwards this encoded message to its neighbors. The coefficients used in the encoding process are also transmitted along with the message. The messages are only decoded at the destination, when it receives a sufficient number of encoded messages (i.e., $n$ linearly independent encoded messages are required to decode $n$ messages). Our proposed scheme also employs network coding for forwarding messages. However, there are two key differences. First, instead of flooding the encoded messages in the network, we leverage the power-law properties of the network and only choose a small fraction of the nodes that have high connectivity (i.e., hubs), as the relay nodes. Second, only the hubs are responsible for coding messages.

In recent years, several researchers [1,5,18] have analyzed the properties of people-centric DTN using empirically collected traces. They have found that in all these networks, a small percentage of popular nodes are connected to most of the other nodes. In other words, the degree distribution follows a power law. Freeman [19] defined several centrality metrics to measure the importance of a node in a network. Researchers in [7,8] have proposed forwarding strategies that exploit the existence
of the scale-free structure in the underlying network. In BubbleRap [7], nodes are formed into communities and also ranked according to their centrality. Both global and community rankings are used to find suitable forwarders by using a gradient forwarding approach. Similar ideas are proposed in [8], where each node is assigned a quality metric based on its popularity. Gradient forwarding is then employed. In our work, we also make use of the popular nodes (called hubs) as relay nodes. However, unlike these schemes, which employ gradient forwarding, in HubCode, messages are disseminated amongst the hubs using network coding.

Mathematical modeling of DTN forwarding strategies is a mature field of research. In particular, forwarding schemes based on epidemic forwarding principles have been extensively analyzed in [20–23]. It is commonplace to use differential equations for modeling system dynamics in DTN [23] because of their simplicity as compared with Markov chains. For example, the system dynamics of forwarding a single packet have been modeled using ordinary differential equations [23]. In [24], the authors have extended this to account for a batch of packets for both replication-based and network-coding-based forwarding. There also exist some efforts that analyze the performance of two-hop relay protocols using redundant copies [25,26]. As mentioned earlier, these works are based on epidemic principle where a node distributes its messages to any other nodes it meets, treating all nodes equally. Consequently, it is difficult to adopt these techniques for modeling our proposed schemes, which differentiate between nodes based on their encounter patterns (hubs versus normal nodes). Instead, in this paper, we use the network model proposed in [22] for studying routing in mobile ad hoc networks. In this model, the characteristics of ad hoc network are captured through a single parameter: the inter-contact rate between nodes. We have amended the model to incorporate the effects of network coding and the forwarding policies of our hub-based forwarding scheme in a simplified manner.

3. HUBCODE

As highlighted in Section 1, empirical analysis of the mobility patterns of several people-centric DTN [1,4,5] have revealed that the degree distribution of the network graph follows a power law. This implies the existence of a small percentage of hubs, which are individually connected to a large number of nodes as compared with other nodes. Further, collectively, the hubs are connected with most of the other nodes in the network (i.e., they achieve nearly 100% coverage). Motivated by these properties, we propose a novel forwarding strategy called HubCode, which uses the hubs as message relays. The hubs are identified by analyzing historical movement patterns of the nodes (e.g., in this paper, we have identified the hubs based on their node degrees). Because most people-centric networks exhibit significant repeatability (e.g., most people have the same daily routine; buses follow the same schedule), this classification of nodes is reasonably time invariant. Also, if the network characteristics change, the new set of hubs can be readily identified by repeating the analysis. An alternative to this static hub-labeling approach is to enable each node to count the number of unique nodes it meets over time. As time elapses, each node can quite effectively use its counter as its rank (i.e., degree) in the network. This hub-labeling approach is de-centralized and do not depend on the history of nodes’ degree distribution. A more sophisticated approach that can be employed in our hub-labeling problem can be found in [2].

All traffic in the network is forwarded to the hubs. Because each hub concentrates significant traffic, we propose the use of network coding at the hubs to encode multiple messages (addressed to the same destination) into a single encoded message. A hub forwards an encoded message to a neighboring hub if this message is linearly independent with the encoded messages carried by the neighbor. The use of network coding results in significant savings in bandwidth, because a single encoded message is forwarded in place of multiple native messages. Further, because the hubs collectively have contact opportunities with all other nodes, most of the messages can be delivered to the destination.

We first present the basic version of our scheme, HubCodeV1, which makes use of the traditional approach to network coding [17]. We argue that this scheme requires the hubs to exchange significant auxiliary information. Next, we present an alternate approach, HubCodeV2, which requires the intermediate hubs to decode the coded messages (in addition to the normal encoding operations). As a result, the hubs only need to exchange message IDs, which reduces the auxiliary data overhead. However, because the hubs decode messages, the computational complexity increases.

3.1. HubCodeV1

In our schemes, message forwarding is a simple three-step process: (i) Source nodes forward messages to a hub; (ii) a hub encodes multiple messages headed to the same destination and disseminates the encoded messages among other hubs; and (iii) a hub delivers the encoded message to the destination. To simplify the explanation, we classify nodes into three groups: (i) source, (ii) destination, and (iii) hubs. We provide a detailed description of the tasks undertaken by each category of node. Note that a source or destination can also be a hub, but for simplicity, we assume the groups are mutually exclusive.

3.1.1. Source

When a source encounters a hub, it creates a copy of the message and forwards the copy to the hub. Recall that the hub nodes are appropriately labeled by analyzing past behavior of the network. Each node broadcasts its label along with any auxiliary data in the periodic beacons.
As a result, the neighbors of a node can easily identify whether the peer (i.e., the beacon sender) is a hub or not. If the source carries a single native message, it is forwarded as-is. However, if more than one message are destined to the same address, then the source combines them into a single encoded message using linear network coding (Equation (1)) and forwards the encoded message to the hub. The coding technique is described below.

### 3.1.2. Hubs

When two hubs encounter each other, they first exchange certain auxiliary information that is used to decide if the hubs should forward messages to each other (these details are explained later). If a hub needs to forward messages to another hub, it encodes all messages with a common destination using RLC and forwards the single encoded message. This results in significant savings in the bandwidth. Assume that a hub currently has \( k \) messages, \( X_1, X_2, \ldots, X_k \), with a common destination. Then the hub creates a linear combination \([17,27]\) of these \( k \) messages to form a single encoded message \( F_1 \), using Equation (1),

\[
F_1 = \sum_{i=1}^{k} a_i X_i, a_i \in \mathbb{F}_q
\]

where \( a_1, a_2, \ldots, a_k \), represent the coefficients, which are randomly selected from a finite field \([28]\), \( \mathbb{F}_q \), where \( q = 2^{16} \). All the additions and multiplications are performed over the finite field \( \mathbb{F}_q \), so that the encoded message has the same size as the native message. The coefficients \( a_i \) and the message IDs \( \text{id}(X_i) \) of all the native messages are appended to the encoded message prior to transmission. This is because the receiving hub may perform further encoding. Because the coefficient vectors are chosen from a large random space, there is a high probability that they are linearly independent. As a result, two encoded messages that are created from the same native messages are still useful to the destination (decoding is explained later).

Note that hubs do not decode the messages. The encoding and forwarding process described previously continue at all intermediate hubs. If a hub holds multiple encoded messages, then these can be further combined into a single message. For example, assume that a hub has received two encoded messages, \( F_1 \) and \( F_2 \), that have been created as follows:

\[
F_1 = a_{11} X_1 + a_{12} X_2 + a_{13} X_3
\]

\[
F_2 = a_{21} X_1 + a_{22} X_2 + a_{23} X_4
\]

Then the hub can combine these two messages to create a single encoded message, \( F_3 \), such that \( F_3 = a_1 F_1 + a_2 F_2 \) where \( a_1 \) and \( a_2 \) are two randomly selected coefficients.

The discussion has focused on the coding process. We now explain the decision making process involved before a hub encodes messages. Each hub maintains a coefficient matrix for all the encoded messages that it currently holds. There is one such matrix for each destination. The columns of the matrix correspond to the message IDs, and there is one row for each encoded message. When two hubs encounter each other, they first exchange the coefficient matrices. These are generally included in the beacons, which are periodically exchanged by nodes. We explain the decision process for a single destination. These steps are repeated for each destination. When a hub receives its neighbor’s matrix, it has to decide if transmitting a linear combination of all its messages will be useful to the neighbor. The hub can determine this by checking if this encoded message is linearly independent to the encoded messages carried by the neighbor. Consider the following example.

Let \( F_1 \) be the encoded message created by this hub, which is composed of two native messages \( X_1, X_2 \) and respective coefficients set \( A_1, < a_1, a_2 > \) (i.e., \( F_1 = a_1 X_1 + a_2 X_2 \)). Also, assume that the hub receives coefficient matrix \( A_2 \) from its neighbor. \( A_1 \) and \( A_2 \) are shown below:

\[
A_1 = \begin{bmatrix} id(X_1) \\ a_1 \\ a_2 \end{bmatrix}, A_2 = \begin{bmatrix} id(X_2) \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{bmatrix}
\]

Because the coefficient matrix is accompanied by the message IDs (i.e., \( id(X_i) \) of the corresponding columns of the matrix, the receiving hub can determine which column is associated with which message. The receiving hub then inserts the coefficient set \( A_1 \) in the corresponding columns of \( A_2 \). In this particular case, \( A_2 \) does not contain any coefficient for the native message \( X_3 \) (i.e., there is no column in \( A_2 \) for the message ID of \( X_3 \)). So, a new column for message \( X_3 \) is created. The coefficients for the message \( X_3 \) in \( A_2 \) will be zero. Similarly, the coefficient of the message \( X_3 \) in \( A_1 \) will also be zero. The modified \( A_2 \) is shown below:

\[
A_2 = \begin{bmatrix} id(X_1) & id(X_2) & id(X_3) \\ a_3 & a_4 & 0 \\ a_5 & a_6 & 0 \\ a_1 & 0 & a_2 \end{bmatrix}
\]

If the coefficient sets (i.e., rows of the matrix \( A_2 \)) are linearly independent, then it is assumed that the newly encoded message \( F_1 \) by the hub is useful to its neighbor. Although this requires the hubs to exchange significant information, they can make more informed decisions about forwarding encoded messages and hence avoid the transmissions of redundant messages. Eventually, when a hub meets the destination, it forwards an encoded message composed of all messages addressed to that destination.

### 3.1.3. Destination

When the hub encounters a destination, it forwards an encoded message to it. Similar to the hubs, the destination also maintains a coefficient matrix. The columns represent the native message IDs, and each row corresponds to an encoded message. Recall that each encoded message is a linear combination of the native messages. Consequently,
to decode $n$ messages, the destination should receive $m$ linearly independent combinations of these messages, such that $m \geq n$. Note that because the coefficients are randomly chosen from a large finite space, there is a high probability that all encoded messages are linearly independent. Hence, $n$ encoded messages are sufficient for decoding (i.e., $m = n$). The $n$ linear equations can be solved using matrix inversion.

For example, if the destination receives the following linearly independent encoded messages, $F_1, F_2,$ and $F_3$ $(F_1 = a_{11}X_1 + a_{12}X_2 + a_{13}X_3, F_2 = a_{21}X_1 + a_{22}X_2 + a_{23}X_3, F_3 = a_{31}X_1 + a_{32}X_2 + a_{33}X_3)$, then the set of linear equations can be written in matrix form $\mathbf{f} = \mathbf{A}\mathbf{x}$.

$$
\mathbf{A} = \begin{bmatrix}
a_{11} & a_{12} & a_{13} \\
a_{21} & a_{22} & a_{23} \\
a_{31} & a_{32} & a_{33}
\end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix}, \quad \mathbf{f} = \begin{bmatrix} F_1 \\ F_2 \\ F_3 \end{bmatrix}
$$

The native messages can be retrieved by matrix inversion:

$$
\mathbf{x} = \mathbf{A}^{-1}\mathbf{f} \quad (4)
$$

Figure 1 presents an illustrative example of HubCodeV1. There are three hubs: $A$, $B$, and $C$. $Q$, $R$, and $S$, and $D$ are regular nodes. Let us assume that $R$, $S$, and $Q$ are source nodes that wish to transmit messages $X_2$, $X_1$, and $X_3$, respectively, to a common destination $D$. The arrows in the figure indicate that the two nodes can communicate with each other. For example, at $t_1$, both $R$ and $S$ are in the communication range of $A$. The direction of the arrow indicates the flow of data messages. $C_v$ is the coefficient vector that is appended to the encoded message. It takes the form $\{idx_j : a_i\}$, where $idx_j$ represents the message ID and $a_i$ denotes the coefficient. The figure is self-explanatory and shows the sequence of operations that are involved in delivering the messages to the destination, $D$.

### 3.2. HubCodeV2

The main drawback of HubCodeV1 is that the hubs need to exchange their coefficient matrices to make the forwarding decision. The overhead of this exchange is $O(n^2)$ for $n$ messages. This overhead is particularly of concern when the contact durations with other hubs are short lived. This is because in such instances, the exchange of auxiliary information may dominate the entire contact opportunity. Empirical measurements have shown that in real-world DTN [1], contact durations can often be quite short. To solve this problem, we present an alternate approach that seeks to reduce this overhead without penalizing message delivery.

In V1, a hub uses the coefficient matrices received from a neighbor to determine if forwarding an encoded message is beneficial to this neighbor. However, if a hub can decode the coded messages to recover the native messages, then it can simply send a list of native message IDs to its neighbors instead of the coefficient matrix. As a result, the neighboring hub can make the same decision. Sending a list of message IDs reduces the auxiliary data overhead to $O(n)$ for $n$ messages, as compared to $O(n^2)$ with V1. However, this gain comes at the expense of extra computation. Because the hubs now decode messages, the computational complexity increases to $O(n^2)$ (solving $n$ linear
equations has a complexity of $O(n^2)$. On the other hand, in V1, the hubs only encode messages, which incurs a complexity of $O(n)$. Most personal communication devices (such as smart phones, personal digital assistants) and in-vehicle routers have sufficient processing capabilities and battery power to perform the decoding operations. Hence, this scheme can be readily deployed in most people-centric DTN. However, V2 is not suitable for resource-constrained devices such as sensor nodes. The two versions address an important trade-off between computational complexity and routing overhead.

As in V1, we classify nodes in three different groups: (i) source, (ii) hubs, and (iii) destination. We explain the operations performed by each type of node.

### 3.2.1. Source

As in V1, the source creates a copy of the native message (without encoding) and forwards it to a hub. However, unlike V1, in this scheme, the hubs may possess native messages (because they decode messages). As a result, a source forwards the native message to a hub only if the latter does not have this message. The source can determine this by examining the auxiliary information (i.e., message IDs) transmitted by the hub in the beacons.

### 3.2.2. Hubs

When a hub receives an encoded message for a destination, it examines the other encoded messages in its queue heading to the same destination. If sufficient encoded messages are present, then the hub decodes these messages (decoding was explained in V1) and stores the native messages. In the event that sufficient messages have not been received, the encoded messages are stored as-is.

When two hubs encounter each other, they exchange the message IDs of the native messages that they carry. If a hub contains an encoded message, which has not been decoded yet, then the coefficients of this message are not included in the auxiliary information. In other words, only the information of the native messages is exchanged. When a hub receives the native message list of its neighboring hub, it compares this list to the native messages waiting in its queue and also to the messages that are used to compose the encoded messages (if any). If the hub finds at least one message (either native or a part of an encoded message) that is not in its neighbor’s list, then the hub encodes the missing messages along with any other messages (native or encoded) for that destination and forwards the combinations to that neighbor. As in hubcode V1, the hubs do not forward any message (either native or encoded) to a non-hub node. Recall that the rationale behind confining message dissemination among hubs is to reduce redundant messages without penalizing delivery ratio, because hubs collectively cover most of the network.

When the hub meets the destination and if it only has encoded messages for that destination (i.e., no native messages), then it sends an encoded combination of these messages. If the hub has one or more native messages in its queue, then it simply forwards them to the destination without coding.

### 3.2.3. Destination

The decoding operation at the destination is exactly similar as in HubCodeV1. Hence, we do not provide details here. The only difference is that unlike V1, the destination may receive native messages in addition to coded messages. Figure 2 highlights the basic operation of HubCodeV2. We have used the same scenario as in the example for HubCodeV1.

### 4. MATHEMATICAL ANALYSIS

In this section, we present a mathematical model to estimate the message delivery delay in our hub-based forwarding schemes. In particular, we derive closed-form expressions for message delivery delay in hub-based forwarding schemes when (i) hubs utilize network coding to disseminate a message among themselves (e.g., HubCode V1, V2) and (ii) hubs do not perform network coding and merely send copies of the message to other hubs (we refer to this as Hub-only scheme). We include the latter scheme in our analysis to quantify the improvements achieved by using network coding.

We use the widely used network model introduced in [22] as the starting point for our analysis. In this model, the characteristics of ad hoc networks are captured through two parameters of the network: (i) the number of nodes in the network and (ii) the intensity of identical and independent Poisson processes that model the meeting instances between any pair of nodes. Although, originally, the network model was proposed for ad hoc networks, it fits well in the context of DTN. This is because of the fact that the meeting instances of the nodes (or inter-meeting durations) determine the message forwarding rate in DTN.

To determine the message delivery delay, we need to estimate the average message forwarding rate. In addition to the inter-meeting durations, the message forwarding rate also depends on various forwarding properties, such as nature of the forwarding protocol and importance of the messages. In the following sections, we start our discussion by introducing the generic routing model. Following this, we begin our mathematical analysis by introducing the factors that affect the message forwarding rate. Once we model the message forwarding rate, we can formulate the expected message delivery delay in a straightforward manner.

#### 4.1. Stochastic forwarding model

We assume a network that consists of one source, one destination, and $N - 1$ relay nodes (i.e., hubs). We also assume that non-hub nodes will not affect forwarding mechanism because message forwarding is performed only by the hubs. Two nodes may only communicate when they are within communication range. The duration when two
nodes are connected is referred to as the meeting time, and the time that elapses between two consecutive meeting times of a given pair of nodes is called the inter-meeting time. For simplicity, the transmission time is assumed to be instantaneous. This is a valid assumption if the size of the message is very small. Obviously, if we had a realistic message traffic model available, then we could have taken a more pragmatic approach in modeling the transmission time. However, modeling message traffic is non-trivial because of its dependence on the network characteristics and network protocol behavior. We have also ignored the effect of processing delays.

Figure 3 shows the state diagram of the generic routing model. The system is in state $i \in 1, 2, \ldots, N$ when there are $i$ copies of the message in the network. It is in state A when the message has been delivered to the destination. Let $R^F_i$ ($i \in 1, 2, \ldots, N - 1$) be the average message forwarding rate to another hub and $R^A_i$ ($i \in 1, 2, \ldots, N$) be the average message forwarding rate to the destination when there are $i$ copies of the message in the network. As can be
seen from Figure 3, the mean time for a message to go from state 1 to state A (i.e., from source to destination) directly (i.e., without the forwarding via hub) is 1/R^1. If the message takes one hop to reach the destination, then the mean delay becomes

$$\frac{1}{R_1} + \frac{1}{R^A}$$

Similarly, if the message takes two hops to reach the destination, then the mean delay becomes

$$\left[ \frac{1}{R_1} + \frac{1}{R_2} \right] + \frac{1}{R^A}$$

Generalizing, the mean time for a message to go from state 1 to state A in k-th (k ∈ 1, 2, 3, …, N - 1) hop is given by

$$\sum_{i=1}^{k} \left[ \frac{1}{R_i} \right] + \frac{1}{R^A_{k+1}}$$

In other words, the above expression provides the mean message delivery delay when the message takes k relays to reach the destination. Using this simple reasoning, we develop our message delay model for both HubCode and Hub-only forwarding schemes in the next sections.

As mentioned earlier, the message forwarding rate depends not only on the inter-meeting durations between nodes but also on other factors, such as forwarding principles and number of copies in the network. Next, we discuss the major factors that affect the message forwarding rates (R^F and R^A) and quantify the impact of these factors on the forwarding rate.

### 4.2. Factors that influence the message forwarding rate

#### 4.2.1. Contact rate between nodes

Because nodes can only forward messages during contact periods, the contact rate between nodes directly influences the message forwarding rate. Inter-meeting durations determine the contact rate. In a study on the association patterns of wireless access points [18], the authors found that the inter-meeting durations between nodes of a large network can be modeled using generalized Pareto distributions. Our study with VANET trace also shows similar findings. For example, in Figure 8, we have plotted the probability mass function of inter-contact durations of the Seattle trace during the 3PM–8PM period and a close-fit Pareto distribution to show the goodness of fit.

If the inter-meeting durations between a pair of nodes are random variables (X) with a power-law distribution (e.g., Pareto distribution), then the expected inter-meeting duration E(X) is given by the following relation. Readers are referred to Appendix A in [29] for further elaboration.

$$E(X) = (\alpha x_{min})/(\alpha - 1),$$

where $$x_{min}$$ is the minimum value of X and $$\alpha$$ is a positive parameter often termed as the tail index of the power-law distribution. The value of $$\alpha$$ and $$x_{min}$$ can be obtained empirically by using the method of maximum likelihood. In fact, the value of $$\alpha$$ (i.e., the tail index) can be approximated graphically by plotting the data sample on a log–log plot and taking the slope of close-fit straight line of the data points.

The average contact rate R is the reciprocal of the average inter-contact duration E(X). Therefore, R is

$$R = \frac{\alpha - 1}{\alpha x_{min}}$$

#### 4.2.2. Number of copies of messages

At the beginning, when the number of copies of a message is low, the message forwarding rate increases exponentially. This exponential growth can be explained intuitively by considering the fact that at the beginning, the source copies the message to another node; then these two nodes can copy the message to another two nodes, thus making the total copies equal to four. Now, these four nodes can copy the message to another four nodes, thus making the total copies to eight, and so on. However, this rate declines as the copies saturate the network. We elaborate this effect of the number of copies (of a message) in forwarding rate in the next section.

#### 4.2.3. Forwarding mechanism

The mechanism of any particular routing protocol may change the message forwarding rate. For example, in our routing protocol (e.g., non-coding Hub-only version), a node forwards a message to only a fraction of the total nodes (i.e., highly connected nodes) and thereby reduces the message forwarding rate by a factor f. Intuitively, f is the probability that the next encountered neighbor is a hub.

#### 4.2.4. Relevance of the messages

The message forwarding rate also depends on the relevance of the message to other nodes (i.e., a redundant message is not relevant to a node). Let p be the probability that a message is important to the forwarded node. For network coding based schemes, p ≈ 1, because almost all messages are important (i.e., linearly independent) because of the application of random linear encoding scheme. In particular, it has been shown [27] that for network coding based schemes, $$p = 1 - (1/d)$$ where d is the coefficients space. Intuitively, the linear independence among the messages depends on the size of the random coefficients. The larger the coefficients, the greater the probability that the messages are linearly independent. Because the finite field size d is often chosen as 2^16, p approaches to 1. On the other hand, when network coding is not employed, analogous to the coupon collector’s problem [30], p becomes $$1 - ((f - 1)/M)$$, where M is the total number of unique
messages destined to the same address, and $j$-th ($j \leq M$) message has been received successfully.

$$p = \begin{cases} 1 & \text{for HubCode} \\ 1 - \frac{i-1}{M} & \text{for Hub-only (without coding)} \end{cases} \quad (7)$$

In the following two sections, we accommodate the previously mentioned factors in the average message forwarding rates ($R^F_i$ and $R^A_i$) and estimate the message delivery delay for both Hub-only and HubCode schemes.

### 4.3. Message delivery delay in Hub-only scheme

In this subsection, we develop the message delay model of Hub-only forwarding scheme. A brief definition of the notations used here can be found in Table I. Figure 4 shows the state diagram of the Hub-only (without coding) forwarding scheme. Recall that in Hub-only forwarding scheme, the source forwards the message to hubs, and hubs disseminate it among themselves in epidemic manner. Eventually, one of the copies reaches the destination.

The message forwarding rate $R^F_i$ at each state $i$ is a function of the number of copies $i$ of the message in the network. When there are $i$ ($1 < i \leq N$) copies of the same message in the network, then each of those nodes either sends a new copy to the $N-i$ hubs, which do not have a copy yet, at a rate of $R^F_i = i(N-i)f$ or meets the destination at a rate of $R^A_i = iRp$. The rationale behind the introduction of the factors $p$ and $f$ are discussed in Section 4.2.

Therefore, the mean time for the $j$-th ($j \in 1, 2, 3, \ldots, M$) message to go from state 1 to state A (i.e., from source to destination) in $k$-th ($k \in 1, 2, 3, \ldots, N-1$) step (Figure 4) is

$$\sum_{i=1}^{k} \frac{1}{iR(N-i)f} + \frac{1}{(k+1)Rp}$$

Because in real-world people-centric DTN, most contact durations are very short [5], it can be assumed that only one message can be transferred at each contact opportunity. Therefore, the mean time for all $M$ messages to go from state 1 to state A in $k$-th step is given by

$$M \sum_{i=1}^{k} \frac{\alpha x_{\text{min}}}{i(\alpha-1)(N-i)f} + \sum_{j=1}^{M} \frac{\alpha x_{\text{min}}}{(k+1)(\alpha-1)\left(1 - \frac{i-1}{M}\right)}$$

(8)

(9)

(10)

Therefore, the expected message delivery delay $E[T_{HO}]$ for all $M$ messages to reach state A from state 1 is

$$E[T_{HO}] = \frac{1}{N-1} \sum_{k=1}^{N-1} \left[ \sum_{i=1}^{k} \frac{\alpha x_{\text{min}}}{i(\alpha-1)(N-i)f} \right] \sum_{j=1}^{M} \frac{\alpha x_{\text{min}}}{(k+1)(\alpha-1)\left(1 - \frac{i-1}{M}\right)}$$

(8)

Table I. Notations.

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E[T_{HO}]$</td>
<td>Expected message delivery delay of Hub-only (without coding) scheme</td>
</tr>
<tr>
<td>$E[T_{HC}]$</td>
<td>Expected message delivery delay of HubCode scheme</td>
</tr>
<tr>
<td>$R$</td>
<td>Average rate of contact opportunities</td>
</tr>
<tr>
<td>$R^F_i$</td>
<td>Average message forwarding rate to another hub when $i$ copies of message are in network</td>
</tr>
<tr>
<td>$R^A_i$</td>
<td>Average message forwarding rate to destination from a hub or source when $i$ copies of the message are in network</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>Power-law tail index</td>
</tr>
<tr>
<td>$x_{\text{min}}$</td>
<td>Minimum inter-contact duration</td>
</tr>
<tr>
<td>$p$</td>
<td>Probability of messages relevance to the peer</td>
</tr>
<tr>
<td>$f$</td>
<td>Probability that a hub meets another hub</td>
</tr>
<tr>
<td>$N$</td>
<td>Number of hubs</td>
</tr>
<tr>
<td>$M$</td>
<td>Number of messages destined to common address</td>
</tr>
</tbody>
</table>

Figure 4. State diagram of Hub-only forwarding scheme.
A closed-formed expression (Equation (9)) of the expected message delivery delay is obtained from Equation (8) by employing calculus approximation methods. The details have not been included because of space limitations. Interested readers are referred to Appendix C in [29] for further elaboration.

\[
E[T_{\text{HC}}] = \frac{M \alpha x_{\text{min}}}{Nf (N - 1)(\alpha - 1)} \left[ (N - 2)\ln(N - 1) + fN \ln(M \ln \left( \frac{N}{2} \right)) \right]
\]  

(9)

### 4.4. Message delivery delay in HubCode

In this sub-section, we sketch the message delivery model of our HubCode forwarding schemes. Recall that the HubCodeV1 and HubCodeV2 only differ in how they broadcast auxiliary data in beacons. For example, HubCodeV1 exchanges coefficient matrix, whereas HubCodeV2 exchanges native message IDs in beacons. In HubCodeV2, to meet the purpose of exchanging the auxiliary data (i.e., message IDs), the hubs try to decode the messages. However, the basic principle of forwarding messages to hubs, which in turn disseminate messages amongst other hubs using network coding, is the same for both schemes. Because our model does not incorporate the beaconing mechanism, the same forwarding model is applicable to both HubCodeV1 and HubCodeV2. We use the generic term HubCode to refer to both HubCodeV1 and HubCodeV2.

Figure 5 shows the state diagram of the HubCode forwarding scheme. Note that in HubCode, the source forwards the message to hubs, and hubs encode and disseminate the encoded message among themselves. The destination receives multiple encoded messages from different hubs. Unlike the Hub-only (non-coding) scheme, practically all the encoded messages are important to the destination because of the application of linear encoding method. The destination decodes the original messages by solving the set of linear equations (i.e., encoded messages together with coefficient vector).

When there are \(i\) (\(1 < i < N\)) copies of the same message in the network, then unlike Hub-only (non-coding) forwarding scheme, each of those nodes either sends a newly encoded message to the \(N\) hubs (because all messages are important to all hubs) at a rate of \(R^F_i = iNRf\) or meets the destination at a rate of \(R^D_i = iRp\) or \(iR\) (because \(p \approx 1\) in case of HubCode (Equation (7)).

Now, the mean time for a message to go from state 1 to state \(A\) in \(k\)-th \((k \in [1, 2, 3, \ldots, N - 1])\) step (Figure 5) is

\[
\sum_{i=1}^{k} \left[ \frac{iRNf}{(k+1)R} \right] + \frac{1}{(k+1)R}
\]

Therefore, the mean time for all \(M\) messages to go from state 1 to state \(A\) in \(k\)-th step is (assuming that only one message can be transferred at each contact opportunity)

\[
M \sum_{i=1}^{k} \left[ \frac{\alpha x_{\text{min}}}{i(\alpha - 1)Nf} \right] + \frac{M \alpha x_{\text{min}}}{(k+1)(\alpha - 1)}
\]

after replacing the value of \(R\) (from Equation (6)).

Therefore, the expected message delivery delay \(E[T_{\text{HC}}]\) for all \(M\) messages to reach state \(A\) from state 1 is

\[
E[T_{\text{HC}}] = \frac{1}{N - 1} \sum_{k=1}^{N-1} \left[ \frac{M \alpha x_{\text{min}}}{(k+1)(\alpha - 1)} \right] + \frac{1}{N - 1} \sum_{k=1}^{N-1} \frac{M \alpha x_{\text{min}}}{(k+1)(\alpha - 1)}
\]

(10)

A closed-formed expression (Equation (11)) of the expected message delivery delay is obtained from Equation (10) by using calculus approximation methods. The details have been omitted because of space limitations. Interested readers are referred to Appendix D in [29] for details.

\[
E[T_{\text{HC}}] = \frac{M \alpha x_{\text{min}}}{Nf (N - 1)(\alpha - 1)} \left[ (N - 1)\ln(N - 1) + fN \ln \left( \frac{N}{2} \right) - N + 2 \right]
\]

(11)

### 4.5. Model validation

To validate our theoretical model, we compare it with the simulation results. The details of the mobility trace and other simulation parameters have been discussed thoroughly in Section 5.
In the first set of our simulations, 10 randomly selected sources (from a pool of 1065 nodes) sent 10 messages each to a single common destination (randomly selected). That is, a total of 100 messages are heading towards a single common destination. The message payload is 1000 bytes. The source nodes generate messages between the period 3000 to 4000 s after the start of the simulation. The average delivery delay is measured for only the messages that reach the destination. The messages, which are not received during the 5 h lifetime of the simulation period, are ignored. We have found that approximately 63% of the messages reach the destination in average. The number of hubs is varied from 5 to 150 in the following steps: (5, 25, 50, 75, 100, 125, 150). The simulation is repeated 20 times for each instance. We measure the message delivery delay and plot the average value in Figure 6. We also plot the 95% confidence intervals. We compare the simulations results to the message delay derived in Equations (11) and (9). The parameters \((\alpha, x_{\text{min}}, f)\) in Equations (11) and (9) are derived empirically from the mobility traces as \((1.01, 30, 0.8)\), respectively. To observe the impact of increasing the traffic, we increase the number of messages sent by each random source from 10 to 20 and repeat the above simulations. The corresponding results are plotted in Figure 7. The theoretical curve of Hub-only scheme (without coding) is also plotted to compare its delivery delay with that of HubCode. Recall that in the Hub-only scheme, unlike HubCode, the hubs do not encode multiple messages into a single message.

One can observe from Figures 6 and 7 that there is significant disparity between the simulation and analytical results when the number of hub nodes are small (<50). However, as the number of hub nodes increases (>50), the two results converge. Further, both results exhibit the same general trend, in that the delivery delay decreases as the number of hubs increase.

In the following, we discuss the possible causes of the disparity between the theoretical model and the simulation results when the number of hubs are small (<50).

- **Theoretical limitation.** First of all, the approximate closed-form mathematical models impose some theoretical limits on possible values of hubs. For example, a cursory look at the closed-form Equations (11) and (9) reveals that the equations are undefined when \(N<3\) (i.e., hubs <3). This explains the peculiar shape of the theoretical model at low values of \(N\).

- **Biased expectation of inter-contact rates.** The accuracy of the expected inter-contact rate directly affects the message delivery delay in Equations (11) and (9). In an empirical study, we have found that the inter-contact durations follows a power-law distribution and model it with a simple Pareto distribution. However, because of the nature of power-law distribution (i.e., most inter-contact durations are short lived, but there exist few very long inter-contact durations), the expected inter-contact duration exhibits bias toward fewer very long inter-contact durations. For example, Figure 8 plots a theoretical Pareto curve that fits...
the empirical inter-contact distribution of the mobility trace (from 3PM–8PM slot) that we have used in our simulation. The expected value of the theoretical Pareto distribution is found to be $\approx 11$ min. However, from Figure 8, it can be seen that the probability of occurrence of this expected value is only $\approx 0.008$. Hence, the expected value of inter-contact durations does not accurately represent the rate of contact opportunities. In other words, it contributes toward a more pessimistic message delay.

- **Multiple transfer per contact opportunity.** Recall that in our model, we assumed that only a single message can be transferred at a contact opportunity. Although most of the contact durations are very small (e.g., $<10$ s), few contact durations are indeed large enough to transfer multiple messages (this is the typical behavior of power-law based distributions). As a direct consequence of the fact, the theoretical model tends to predict larger message delivery delay than the results obtained from simulation settings, especially when the number of messages destined to common address is increased (Figure 7).

- **Problem of averaging with fewer terms.** Our model exhibits instability when the number of hubs (acting as message relays) is very small (between 1–6). If we delve into the message delay model in Equations (5), (10), and (8), we find that averaging with fewer terms (i.e., when number of hubs is very small) is responsible for the instability. We explain this by considering the generalized delay model in Equation (5), which is restated below for convenience:

\[
\text{Mean delay for arriving at state } A \text{ in } k \text{ steps} = \left( \frac{1}{R_1^A} + \frac{1}{R_2^A} + \ldots + \frac{1}{R_{k-1}^A} \right) + \left( \frac{1}{R_k^1} \right)
\]

1st part 2nd part

In the above equation, the first part represents the cumulative forwarding delay among hubs, and the second part represents the delay when a relay delivers the message to the destination. The values of the trailing terms in the first part rapidly decrease as the value of $k$ (i.e., hubs) increases because of the rapid growth rate of $R_k^A$. As a consequence, the average value calculated (in Equations (10) and (8)) with fewer terms becomes significantly larger than that with more terms because first few terms are much larger than the rest of the terms. For example, the average value of $\left( 1/R_1^A + 1/R_2^A \right)$ is greater than that of $\frac{1}{R_1^4} + \frac{1}{R_2^4} + \frac{1}{R_3^4}$. This explains the initial large discrepancy between the theoretical and simulation results at fewer number of hubs.

As mentioned earlier, the theoretical curve of Hub-only scheme (without coding) is included to compare its delivery delay with that of HubCode. As expected, the theoretical message delivery delay (Figures 6 and 7) in Hub-only scheme is greater than that in HubCode. The difference becomes even more prominent when the number of messages to a common destination are increased (Figure 7). Because more messages are headed towards common destination (Figure 7), the coding opportunity of HubCode increases, hence showing improved performance than Hub-only scheme. In both Hub-only and HubCode schemes, the theoretical models provide us with upper bounds on the delivery delay.

5. SIMULATION-BASED EVALUATIONS

In this section, we present simulation-based evaluations that compare the performance of the proposed HubCode schemes with other DTN forwarding schemes. We use mobility traces of a large-scale vehicular DTN network. In all our simulations, we took a pragmatic approach, wherein the data exchanged by two adjacent nodes are proportional to the contact duration. Further, we also accounted for the auxiliary information exchanged by the nodes. In the first set of experiments, we compared the performance of our forwarding schemes with others schemes in terms of message delivery ratio and delivery cost. We also evaluated the robustness of our schemes against random node failure. In the second set of simulations, we investigated the impact of various parameters on the forwarding performance. In particular, we studied the impact of varying the per cent of nodes classified as hubs, the message size, and the traffic load.

5.1. Mobility trace details

In recent years, several researchers have conducted empirical measurements to study the behavior of people-centric DTNs. In these experiments, communicating devices (bluetooth, zigbee, etc) are either handed to a volunteer group [1] or are mounted on moving vehicles [2]. The devices record all opportunistic contacts with other devices in the participant set and also with external devices. The external contacts are often excluded in the analysis, because complete information about their encounters is not available. Because of practical limitations of conducting empirical experiments, the node population in all the traces is quite small (see Table II). Further, it has been observed that few hubs are connected to all other nodes (i.e., have full degree distribution). This is an artifact of the small population in the traces and is not representative of real-world networks. As a result, we have found that schemes that exploit the power-law properties, such as BubbleRap, achieve close to optimal performance (results are excluded for brevity). Hence, employing network coding at the hubs offers little advantage.

Therefore, in our evaluations, we use mobility traces from a significantly larger network that capture the
movement of public transport buses from the King County Metro bus system in Seattle [32]. This transport network consisted of 1163 buses plying over 236 distinct routes covering an area of 5100 km². The traces were collected over a 3 week period in October–November 2001. The traces are based on location update messages sent by each bus. Each bus logs its current location using an automated vehicle location system, its bus and route ID along with a timestamp. The typical update frequency is 30 s. The traces can be readily used to simulate a bus-based DTN, similar to DieselNet [5]. As in [5], we assumed that each bus is equipped with a 802.11b radio. Buses exchange messages when they are within the communication range of each other, which was assumed to be 250 m.

The traces were post-processed to generate fine-grained location information. The details have been omitted for reasons of brevity. The trace also shows power-law behavior. However, unlike the other traces, no single hub connects to most of the other nodes (the maximum degree of a hub is 0.15). This is representative of a large real-world people-centric DTN.

### 5.2. Simulation settings

We used a custom discrete event simulator. We assumed that each node broadcasts a beacon every 5 s, for neighbor discovery. The beacons also contained additional information as required by the routing scheme (e.g., HubCode). Because we wished to study the performance of the routing schemes in isolation, the 802.11 MAC was not implemented. In each simulation, we injected 1000 messages to 100 destinations (both source and destination are randomly selected from the pool of 1065 nodes). We assumed that the message inter-arrival duration at the source is exponentially distributed, with the average inter-arrival duration set to 30 s. The message payload was 1000 bytes. The source nodes generated messages between the period 3000 to 4000 s after the start of the simulation. This is because several nodes are inactive in the initial period. Each simulation lasted for 5 h (18 000 s). We chose the 3PM–8PM period from two successive weekdays, 31 October 2001 (Wednesday) and 1 November 2001 (Thursday). We simulated each trace 20 times for statistical significance. The results presented were averaged over the 20 runs. The 95th percentile confidence intervals were all within 10% of the average.

To evaluate the performance, we used the following metrics: (i) delivery ratio, which is the ratio of the messages delivered to the messages created, (ii) delivery cost, which is measured as the total number of messages transmitted, normalized by the total number of unique messages created, and (iii) delivery delay, which is the average end-to-end delay of the delivered messages. Note that the delivery cost does not include the auxiliary messages exchanged by the nodes in the beacons.

We compared the performance of the HubCode schemes (we use the term HubCode to refer both HubCodeV1 and HubCodeV2) with several other DTN forwarding schemes. Epidemic (i.e., flooding) [33] was included because it achieves the highest delivery ratio when the network is not congested. We chose spray and wait [34] as a representative restrictive multiple-copy forwarding scheme. In this scheme, the source initially makes \( n \) copies of a message and forwards half of those to a neighbor it meets. This node in turn repeats this strategy; that is, it forwards half of the messages to the next encountered node and retains the other half. This process repeats until a node is left with one copy, which is only forwarded to the destination. In our simulation, we chose initial number of copies \( n \) to 8, which is a fair compromise between the delivery ratio and the cost [34]. RLC [17] and BubbleRap [7] were included because they are closely related to our work (see Section 2). We used the same methods as described in [7] to rank the nodes.

To implement HubCode, it is necessary to identify the hub nodes in the network. For this, we analyzed the traces from an entire weekday and ranked the nodes according to the number of unique nodes they have encountered. We found that the ranking is similar for all the weekdays, because the buses follow repeatable patterns. Most people-centric networks are known to exhibit such repeatable behavior. The top 10% of nodes (i.e., 116 buses) were classified as the hubs. We have utilized the centralized approach to rank hubs for simplicity. However, distributed approaches similar to distributed community detection methods [7,35] might be applicable to decentralize the ranking problem. Because of lack of space, we omit the discussion of the distributed ranking approaches.

Contact durations are finite and short lived in the real-world scenario. For example, analyzing the bus traces, we found that several contact durations are smaller than 30 s. Similar behavior is observed in other real-world networks [1]. Hence, we assumed that the amount of data exchanged is proportional to the length of the contact duration. For simplicity, we assumed the following linear relationship. The amount of data, \( D \), exchanged during a contact duration of \( T_c \) seconds is given by \( D = (T_c - T_s) \times 4 \text{ Mbps} \).

### Table II. Properties of mobility traces.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Period</td>
<td>05-01-06 to 05-01-11</td>
<td>05-01-25 to 05-01-31</td>
<td>05-03-07 to 05-03-10</td>
<td>01-10-30 to 01-11-26</td>
</tr>
<tr>
<td>Device</td>
<td>iMote</td>
<td>iMote</td>
<td>iMote</td>
<td>None</td>
</tr>
<tr>
<td>Network</td>
<td>Bluetooth</td>
<td>Bluetooth</td>
<td>Bluetooth</td>
<td>WiFi (simulated)</td>
</tr>
<tr>
<td>Nodes</td>
<td>9</td>
<td>12</td>
<td>41</td>
<td>1163</td>
</tr>
<tr>
<td>Coverage</td>
<td>Top 2 covered 9</td>
<td>Top 2 covered 12</td>
<td>Top 4 covered 41</td>
<td>Top 100 covered 865</td>
</tr>
</tbody>
</table>

DOI: 10.1002/wcm
Empirical experiments have shown that in 802.11b, the typical goodput (accounting for overheads) at the highest data rate of 11 Mbps is around 4 Mbps [36]. $T_a$ refers to the association time, which includes typical time to associate with an access point. For simplicity, we assumed a fixed value of 10 s for $T_a$. We also accounted for the time required to exchange the beacon messages (note that the size of the beacons vary depending on the forwarding scheme employed).

5.3. Delivery ratio, cost, and delay

Figures 9, 10 and 11 plot the delivery ratio, delivery cost and delivery delay, respectively, for all the forwarding schemes. The delivery ratio for HubCodeV2 is approximately 72%, which outperforms all other schemes by about 15 – 20%. On the contrary, the delivery ratio for HubCodeV1 is about 60%. Recall that, in HubCodeV2, the hubs do not exchange the complete coefficient matrices as in HubCodeV1. Rather, they only exchange native message IDs. Hence, in HubCodeV1, particularly when the contact opportunities are short, significant time is utilized in exchanging the coefficient matrices, thus leading to several wasted opportunities for transferring data. Figure 10 shows that V1 still outperforms V2 slightly (by 20%) in terms of the delivery cost, as a consequence of the extra information exchanged in the beacons. Also, observe that the HubCode schemes are far superior in comparison with all other schemes (e.g., Epidemic incurs 300% excess costs as compared with V1). This is because our schemes restrict the message forwarding within the hubs and encode multiple messages together.

The delivery ratio of epidemic is about 60%, which is less than that of HubCodeV1. This is because a node may not be able to transfer all the messages it carries to other nodes during an encounter. When contact durations are bounded, epidemic essentially resembles a restricted flooding scheme such as spray and Wait, as evident from their similar delivery ratio. The delivery cost of epidemic is still significantly high as compared with other schemes. Spray and Wait reduces the cost by about 15% as a result of the cap on the number of copies exchanged between nodes. Despite employing network coding, the performance of RLC is poor in comparison with HubCode. This is because, in RLC, encoded messages are flooded in the entire network. Hence, the overhead of unnecessary message replications and dominating auxiliary data exchange exhaust scarce bandwidth.

Although a large message delay is common in DTNs, it is always desirable to receive a message as early as possible. To get an idea of message delivery delay, we plot (Figure 11) the average end-to-end delay for all delivered messages for the forwarding protocols in consideration. The $x$-axis represents the delivery ratios of the forwarding protocols as time elapses, whereas the $y$-axis represents the average delays of the delivered messages for the corresponding delivery ratios. Consider the instance when the delivery ratio is 60% (as indicated by the vertical line in Figure 11). In this case, the average delay for HubCodeV2 is approximately 3000 s lower than its nearest contender (i.e., RLC). This is because of the fact that the unnecessary copies of the messages created by the other protocols...
exacerbate over time and exhaust scarce bandwidth, which in turn has a negative impact on the message delivery ratio and delay. Recall that in HubCodeV1, nodes exchange their complete coefficient matrices, which grow as time elapses. As a consequence, the behavior of HubCodeV1 degenerates into that of a closely related protocol, RLC. This explains the similar delivery delays (~11000) experienced by HubCodeV1 and RLC.

5.4. Effect of node failure

In people-centric networks, node failures are a reality. A node failure may occur as a result of software/hardware failures or energy depletion. Further, a node may also cease to participate in message forwarding activity. In this section, we investigate the impact of node failures on the forwarding schemes under consideration. A randomly chosen fraction of nodes is made inactive over the entire duration of the simulation. We observe the impact of varying the percentage of inactive nodes on the delivery ratio. This allows us to investigate the robustness of our scheme to node failures. As before, the simulation is repeated 20 times, and the average delivery ratio is plotted. Note that we consider two different cases. In the first instance (Figure 13), we assumed that the inactive nodes are chosen from the entire set of nodes. In the second case, we only picked hubs as the inactive nodes (Figure 12).

As can be seen from the graph (Figure 12), even a 20% failed hub nodes has only a minor effect on the delivery ratio (a 7% decline). Recall that the hubs are usually well connected (i.e., have high out degrees) and that HubCode disseminates messages among hubs that collectively form a data conduit. The high inter-connectivity among the hubs provides alternate paths to disseminate messages among themselves. As a result, our schemes are robust to the failure of a few hubs. Figure 12 also shows that other schemes such as epidemic and RLC are also not significantly affected by node failures. This is because in both of these schemes, messages are copied to all encountered nodes. Therefore, failure of some random nodes does not impact the message delivery ratio.

If the number of failed nodes is drawn from the entire node population (i.e., not merely from the group of hubs), the probability that the failed nodes contains hubs decreases. Because HubCode only uses hubs (a fraction of total nodes) as delivery messengers, failure of non-hub nodes does not affect its delivery performance. Figure 13 shows the effect of random hub failure on the delivery ratio of HubCodeV2. The same trend is also observed in HubCodeV1 (not shown in Figure).

5.5. Effect of the number of hubs

Recall that HubCode uses the hubs as the message relays. In the previous experiments, we assumed that the top 10% of the nodes, ranked according to the degree distribution, are classified as the hubs. A natural question arises: what is the impact of increasing the number of hubs on the performance? In this set of experiments, we sought to answer this question. We considered the same parameters as in Section 5.3. Figure 14 illustrates the impact of increasing the percentage of nodes that constitute the hubs on the delivery ratio. The delivery ratio for both schemes initially increased as we increased the number of hubs. This was expected because the number of nodes that the hubs can collectively contact increases. As a result, the number of messages that can reach the destinations increase. However, after a certain knee point (around 20 – 30%), the delivery ratio began to decrease. This is because an increase in the number of hubs reduces the number of messages carried by each hub. As a result, the opportunities for coding of multiple messages decrease. Also, the increased overhead caused by the increasing number of hubs is also a key factor that reduces the delivery rate. Because the overhead of HubCodeV2 is less than that of HubCodeV1, the knee point in HubCodeV2 is slightly higher than that of

Figure 12. The effect of random node failure on message delivery.

Figure 13. The effect of random hub failure on message delivery (HubCode V2).
HubCodeV1 (30% as compared with 20% in HubCodeV1). Note that when 100% of the nodes were regarded as the hubs, HubCode degenerates into RLC. The result concerning delivery costs is not shown in this paper because the findings are obvious: the costs continue to increase with increasing number of hubs. These results suggest that only a small percentage of the nodes (20 – 30%), which are highly connected, should be designated as hubs.

5.6. Effect of message size

In this section, we investigate the impact of the size of the message blocks on the delivery ratio. When the message block size is small, the auxiliary headers make up for a significant percentage of the packet size, which reduces the forwarding efficiency. On the other hand, if the message block is too large, a node may fail to forward it if the contact duration between the nodes is small. In this experiment, we seek to find the optimum block size when transferring a long file.

We assumed that each file is 100 KB long and that there are 1000 files to send. The number of destinations is 100, which were selected randomly from our pool of 1065 nodes. We varied the message block size from 1 to 3000 (in 500 bytes step) and observed the delivery ratio. Other parameters remained the same as in the previous simulations. The experiment was repeated 20 times, and the average value of the delivery ratio was plotted (Figure 15).

In HubCode versions, the coding opportunities increase when block size is small because the probability of the number of blocks headed toward common destinations increases. However, this effect is diminished by the negative impact of the increased amount of overhead caused by larger coefficient vectors and headers. As a result, the HubCode versions showed very poor performance when the block size is < 50 bytes.

The delivery ratio begins to increase when the message block size is beyond 50 bytes. However, after reaching a cutoff point (~ 500 in this case), the delivery ratio declines in response to further increase in the message block size. This is because as the message size becomes larger, the contact opportunities required for successful message transfer becomes scarce (i.e., small contact opportunities may fail to transfer large blocks) and negatively affects the delivery ratio.

5.7. Effect of traffic load

In this experiment, we gradually increased traffic load to observe its effect on the delivery performance of the routing protocols. As before, 100 destinations were randomly selected from the pool of 1065 nodes. We varied the number of injected messages from 500 to 3000 (incremented in blocks of 500). Messages were injected in the network 3000 s after the simulation began at random intervals (using exponential random variable with $\lambda = 30$). The whole experiment was repeated 20 times, and the average values were plotted (Figure 16).

Because the number of destination remains the same (i.e., 100) in all cases, the increased traffic load implies
that more messages are directed to common destinations. The delivery ratio initially increases steadily as traffic load is increased up to a knee point (1500 messages in this scenario) and then declines slowly when the traffic load is increased further. This is because as the traffic load is increased, the probability that more messages have common destination increases, and so do the coding opportunities among the hubs. An increase in coding opportunity implies more efficient utilization of Bandwidth and hence increases the delivery ratio. However, if the traffic load is increased further beyond the knee point, the auxiliary data overhead (because of the increased size of coefficients) dominate and exhaust scarce bandwidth resources (recall that most contact durations are very small). As a result, the delivery ratio begins to decrease when the traffic load is increased beyond the threshold.

6. CONCLUSION

In this paper, we proposed a novel forwarding strategy called HubCode for people-centric DTN that exhibits power-law behavior. HubCode uses the highly connected nodes as message relays. Further, messages are forwarded amongst the hubs using linear network coding. We presented two alternate implementations of HubCode to address the important trade-off between routing overhead and computational complexity. Our simulation-based evaluations of a large-scale vehicular DTN demonstrated the efficacy of our schemes. In particular, under pragmatic assumptions, our schemes were shown to achieve 20% higher delivery ratio and less than half of the delivery costs of comparable strategies.

We have derived closed-form expressions for message delivery delay of our proposed scheme and validated our model by comparing with simulation results. We show that our mathematical model serves as a lower bound for the message delivery delay incurred under normal operational conditions.

As a future work, we would like to examine the effect of the bundle-protocol layer and the convergence layer on our forwarding protocols. Licklider protocol [37] is a good convergence layer to start with, because recent research [38] shows its effectiveness compared with other convergence-layer protocols (e.g., user datagram protocol-based convergence layer protocol, transmission control protocol-based convergence layer protocol).

REFERENCES


AUTHORS’ BIOGRAPHIES

Shabbir Ahmed received his B.Sc. degree in Applied Physics & Electronics from the University of Dhaka, Dhaka, Bangladesh in 1992; M.Sc. degree in Computer Science from the same university in 1993, and Ph.D degree in Computer Science from the University of New South Wales, Sydney, Australia in 2010. He is an Associate professor of Computer Science and Engineering at the University of Dhaka. His current research interest includes vehicular networks and data mining.

Salil S. Kanhere received the B.E. degree in electrical engineering from the University of Bombay, Bombay, India in 1998 and the M.S. and Ph.D. degrees in electrical engineering from Drexel University, Philadelphia, U.S.A. in 2001 and 2003, respectively. He is currently a senior lecturer with the School of Computer Science and Engineering at the University of New South Wales, Sydney, Australia. His current research interests include wireless sensor networks, vehicular communication, mobile computing, and network security. He is a member of the IEEE and the ACM.