Erasure-Coding Based Routing for Opportunistic Networks

Yong Wang, Sushant Jain[†], Margaret Martonosi, Kevin Fall[‡]

Princeton University, [†]University of Washington, [‡]Intel Research Berkeley

ABSTRACT

Routing in Delay Tolerant Networks (DTN) with unpredictable node mobility is a challenging problem because disconnections are prevalent and lack of knowledge about network dynamics hinders good decision making. Current approaches are primarily based on redundant transmissions. They have either high overhead due to excessive transmissions or long delays due to the possibility of making wrong choices when forwarding a few redundant copies. In this paper, we propose a novel forwarding algorithm based on the idea of erasure codes. Erasure coding allows use of a large number of relays while maintaining a constant overhead, which results in fewer cases of long delays.

We use simulation to compare the routing performance of using erasure codes in DTN with four other categories of forwarding algorithms proposed in the literature. Our simulations are based on a real-world mobility trace collected in a large outdoor wild-life environment. The results show that the erasure-coding based algorithm provides the best worst-case delay performance with a fixed amount of overhead. We also present a simple analytical model to capture the delay characteristics of erasure-coding based forwarding, which provides insights on the potential of our approach.

Categories and Subject Descriptors

C.2.2 [**Network Protocols**]: Routing protocols

General Terms

Algorithms, Performance, Theory

Keywords

Routing, Delay Tolerant Network, Erasure Coding

1. INTRODUCTION

Opportunistic networks are an important class of DTNs in which *contacts* (time-window when data can be exchanged) appear *opportunistically* without any prior information. Examples of such networks are sparse mobile ad hoc networks, such as ZebraNet [8], where no contemporaneous end-to-end path may exist due to radio range limitations. Routing becomes challenging in such networks because contact dynamics are not known in advance and no single path can be relied upon. Most current approaches are based on some kind of data replication over multiple paths [14, 8]. In this paper, we propose an alternate method of improving delay performance. The basic idea is to erasure code a message and distribute the generated code-blocks over a large number of relays. Compared to sending a full copy of the message over a relay, only a fraction of code-blocks are sent over each relay. This fraction allows us to control the routing overhead in terms of bytes transmitted. For scenarios like ZebraNet, where nodes are energy constrained, limiting such overhead is an important design goal.

The basic idea of using erasure coding is simple and has been explored in many applications [11]. However, it is not clear if and when it will perform better than simpler alternatives based on pure replications in DTNs. In this paper, we study the performance of an erasure coding approach and other existing alternatives on a diverse mobility scenarios with different node densities and moving patterns. We use both synthetic and real-world DTN mobility traces as input to our simulations. We discover that the erasure coding approach can provide good delay guarantees by using a fixed overhead. Fundamentally, the benefits of erasure coding arise in eliminating cases when long delays arise due to bad choice of forwarding relays. Erasure coding allows the transmission to be spread over multiple relays while using a fixed amount of overhead. This results in a protocol much more robust to failures of a few relays or some bad choices. We find that the erasure-coding based algorithm is the least sensitive to different parameters in terms of message latency and message delivery rate. Also, we derive an expression for the delay distribution under a simple network model to argue when and why the erasure coding approach outperforms other simpler alternatives. In one extreme case, we show that the average delay of a simple replication strategy will be infinite, whereas, by using erasure coding the average delay can be reduced to a small constant.

Erasure coding can also help combat packet loss due to bad channel quality or packet drops due to congestion. A full investigation of the benefits of this aspect is outside the scope of this paper. Here, our focus is on a less-conventional use of erasure coding: to achieve better delay performance using a fixed amount of replication.

2. BACKGROUND

In an opportunistic network, reliable data delivery is often achieved using replication to send identical copies of a message over multiple paths to mitigate the effects of disconnections. Typical algorithms differ based on their decisions as to *who* forwards the data, *at what time* is the data forwarded, and to *whom* is the data sent. In

SIGCOMM'05 Workshops, August 22–26, 2005, Philadelphia, PA, USA.

the following discussions, we define a *contact* as an opportunity to communicate between two nodes and a *relay* denotes a forwarding node.

- **Flooding (`flood`):** each node forwards any non duplicated messages (including messages received on behalf of other nodes) to any other node that it encounters. `flood` delivers messages with the minimum delay if there are no resource constraints, such as link bandwidth or node storage.

- **Direct contact (`direct`):** the source holds the data until it comes in contact with the destination. `direct` uses minimal resources since each message is transmitted at most once. However, it may incur long delays.

- **Simple replication (`srep(`r`)`):** this is a simple replication strategy in which identical copies of the message are sent over the *first* r *contacts*. Here, r is the replication factor. Only the source of the message sends multiple copies. The relay nodes are allowed to send only to the destination; they cannot forward it to another relay. This leads to small overhead as the message flooding is controlled to take place only near the source. This class of forwarding algorithms is also known as the *two-hop* relay algorithm [3, 2]. There is a natural trade-off between overhead (r) and data delivery latency. A higher r leads to more storage/transmissions but has lower delays.

- **History-based (`history(`r`)`):** here *history* is used as an indicator of the probability of delivery. Each node keeps track of the probability that a given node will deliver its messages. r highest ranked relays (based on delivery probability) are selected as forwarding nodes. ZebraNet uses the frequency at which a node encounters destination as an indicator of the delivery probability. We use the same implementation as [8] in our simulations.

A summary of these forwarding algorithms is listed in Table 1.

Algorithm	Who	When	To whom
`flood`	all nodes	new contact	all new
`direct`	source only	destination	destination only
`srep(`r`)`	source only	new contact	r first contacts
`history(`r`)`	all nodes	new contact	r highest ranked

Table 1: Summary of various forwarding algorithms.

3. THE ERASURE-CODING BASED FORWARDING ALGORITHM

As discussed in the previous section, most current approaches for routing in opportunistic networks are based on sending multiple identical copies over different paths. There is a fundamental trade-off between overhead and delay. On one extreme, flooding achieves the best possible delay but results in very high overhead. The other extreme is protocols like `direct` which have low overhead because they send only few copies or none at all. Lack of knowledge about the topology dynamics prevents distinguishing good paths from bad ones. Therefore, these protocols may result in long delays if bad paths are selected. In this section, we describe a forwarding algorithm based on the idea of erasure coding. Our algorithm achieves better worst-case delay performance than existing approaches with a fixed overhead.

3.1 Erasure coding background

Erasure codes operate by converting a message into a larger set of code blocks such that any sufficiently large subset of the generated code blocks can be used to reconstruct the original message. More precisely, an erasure encoding takes as input a message of size M and a replication factor r. The algorithm produces $M * r/b$ equal sized code blocks of size b, such that any $(1+\epsilon)\cdot M/b$ erasure coded blocks can be used to reconstruct the message. Here, ϵ is a small constant and varies depending on the exact algorithm used, such as Reed-Solomon codes or Tornado codes. The selection of algorithms involve trade-offs between coding/decoding efficiency and the minimum number of code blocks to reconstruct a message. For example, Tornado codes have efficient encoding and decoding steps based on simple operations such as XOR, at the cost of slightly higher ϵ. A thorough discussion of the various trade-offs is presented in [11]. The choice of exact erasure coding algorithm is not important in our forwarding algorithm. The key aspect is that when using erasure coding with a replication factor of r, only $1/r$ of the code blocks are required to decode the message. Therefore, we ignore constant ϵ for simplicity. Constant b is the block-size and is implementation dependent.

3.2 Erasure coding based forwarding (ec)

Our erasure-coding based forwarding algorithm can be understood as an enhancement to the simple replication algorithm (`srep`) described in Section 2.

In `srep` with a replication factor r, the source sends r identical copies over r contacts and relays are only allowed to send directly to the destination. In the erasure-coding based algorithm, we first encode the message at the source and generate a large number of code blocks. The generated code blocks are then equally split among the first kr relays, for some constant k. In comparison with `srep`, this approach uses a factor of k more relays and each relay carries a factor of $1/k$ less data. However, the number of bytes generated are rM, the same as the number of bytes generated by `srep` (r).

Now by definition of erasure coding (with rate r, message size M), the message can be decoded at the destination if $1/r$ of the generated code blocks are received. Since code blocks are divided equally among kr relays, the message can be decoded as soon as any k relays deliver their data if we assume that no code blocks are lost during transmissions to and from a relay.

When $k = 1$, the erasure coding approach has the same effect as the simple replication approach, which is, to use the first r relays and to each carry a copy of the original message.

3.3 Benefits of erasure coding in forwarding

In simple replication, r relays are used to improve the delay performance. The erasure-coding based approach, instead, utilizes kr relays for the same amount of overhead. Therefore, one can expect that the chances of at least some relays having low delays are higher, compared to using only r relays. At the same time, erasure coding requires at least k relays to succeed (instead of 1 in `srep`) before the data can be reconstructed. Therefore, if the number of such low-delay relays are larger than k, the erasure-coding based approach will successfully deliver the message with a lower delay than simple replication. Thus, the fundamental question is whether to use r relays and wait for one to succeed or use $r * k$ relays and wait for k relays to succeed. We answer this question using a simple analytical model in Section 5. The main observation is that if k is large, the delay distribution converges to a constant. Therefore, with the erasure-coding based approach, one can be *almost* assured of a constant delay.

4. EVALUATION

In this section, we use simulation to compare forwarding algorithms discussed in Section 2 and the erasure-coding based approach presented in Section 3.

4.1 Methodology

We use `dtnsim`, the discrete event simulator for DTN environments from [6]. We implemented the following routing algorithms in `dtnsim`: flooding (`flood`), direct contact routing (`direct`), history-based routing (`history`), simple replication routing (`srep`) and erasure-coding based routing (`ec`). For `srep` and `ec`, we represent different replication factors and number of relays used to split, using `srep`-rep$r$ and ec-rep$r$-pn. Here, r is the replication factor and n are the number of relays among which code blocks are divided.

We simulate using a real-world mobility trace collected as part of a wildlife tracking experiment in Kenya. The mobile network was deployed by the ZebraNet group in January, 2004 [15]. Tracking collars are placed on the necks of selected zebras. Each collar uses GPS to record its position data every 8 minutes, and periodically sends back position log data to a mobile base station (e.g., a vehicle). Due to extreme weather and waterproofing issues, as well as antenna problems, only one tracking collar returned a complete set of uninterrupted movement data for the whole 32-hour duration. Due to such limitations [1], we create a semi-synthetic mobility model as follows: we synthesize node speed and turn angle *distributions* from the observed data and create other node movements following the same distribution. We scale the grid size to 6km$\times$6km with a radio range of 1km. Initially, the nodes are randomly distributed in the grid. The base station moves along a rectangular path near the grid boundary. All messages are of size 1M. Each node generates 12 messages every day. The total duration of simulation is 16 days. Another mobility model based on heavy-tailed inter-contact times is discussed in Section 4.4.

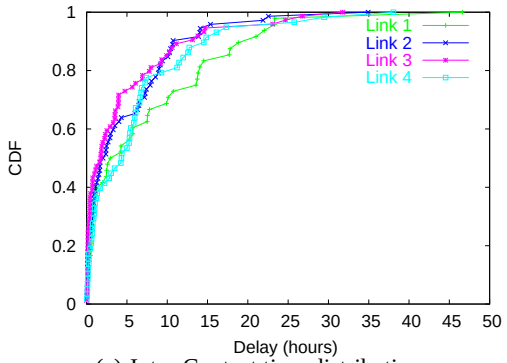
We compare the routing performance of different forwarding algorithms using the following three metrics:

- **Data success rate:** the ratio of the number of messages that are delivered to the destination within a time T (**deadline**). If T is unspecified, it is considered to be the whole duration of the simulation, i.e 16 days.

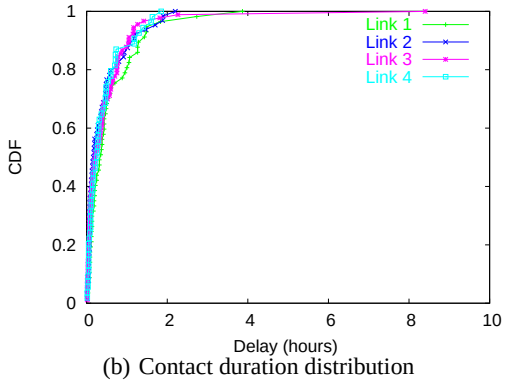
- **Data latency:** the duration between message generation and message reception (at its destination). In a DTN, latency may not be the most critical issue. However, it is always desirable to have fast data delivery whenever possible. The latency distribution metric measures how efficiently a protocol uses the available contact opportunities.

- **Routing overhead:** the ratio of the number of bytes transmitted to the number of bytes generated during the simulation time. This metric measures the extra data transmitted for each message generated, while a metric based solely on the number of message transmissions will overlook the fact that `ec` has smaller message sizes. The radio transmission energy is proportional to the total number of byes transmitted. Therefore, this metric reflects the energy efficiency of the forwarding algorithm.

[1]At the moment, we are working on collecting more node traces during our second field trip in June, 2005. We will work on adjusting the model once we have those node traces available.



(a) Inter-Contact time distribution



(b) Contact duration distribution

Figure 1: Cumulative distribution plots for inter-contact times and contact durations for the ZebraNet trace. The figure plots these two metrics for four randomly selected links. Other links show similar characteristics. The contact duration distribution uses a different x-axis range to separate different curves. Observe that inter-contact time patterns show significant variation and can be very long in some cases.

4.2 Zebra trace analysis

To begin our analysis, we first characterize the contact opportunities in the ZebraNet trace, with a focus on inter-contact time and contact durations. These two metrics are important in understanding the behavior of different forwarding algorithms on the ZebraNet trace. Simply put, inter-contact time is the time interval for which a link is down (no communications are possible during this time) and contact duration is the interval for which a link is up.

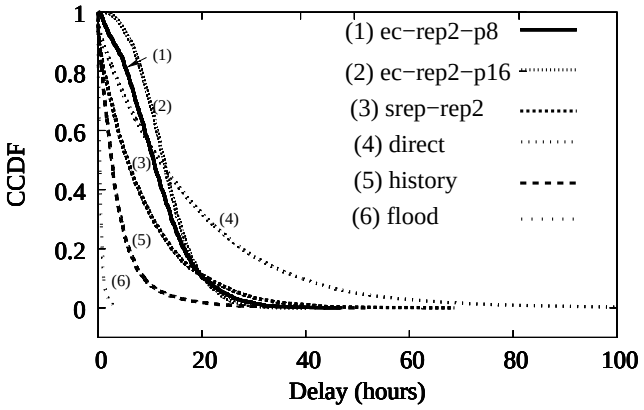
Figure 1 plots the distribution of these two metrics for four randomly selected pairs of nodes (links) in the ZebraNet trace. Since almost all the links in the trace show similar characteristics, we just use these four random links as examples.

As shown in Figure 1(a), the inter-contact time distribution has few cases when a link is broken for a very long time. This observation is important because such inter-contact time patterns can lead to extremely long delays when using a naive forwarding algorithm. As expected in such a sparse network, link up-times are relatively short (as compared to the link down times) and therefore, it is important to efficiently utilize the available communication opportunity.

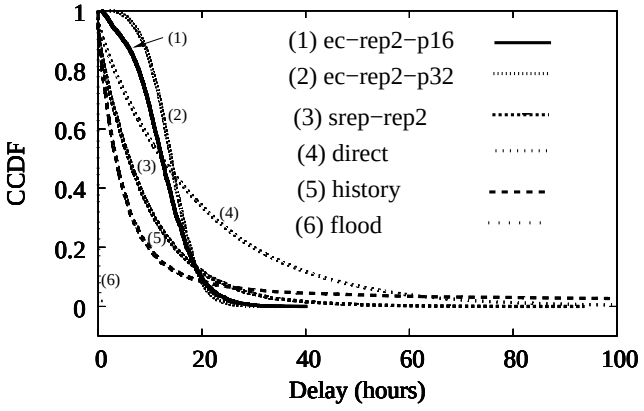
4.3 Impact of node density

4.3.1 Data latency distribution

Figure 2(a) and 2(b) show the data latency distribution for the



(a) 34 nodes



(b) 66 nodes

Figure 2: Latency distribution for different forwarding algorithms. Traffic injection rate is 12 messages per day. The distribution is shown in Complementary CDF (CCDF) curve. A numeric presentation of this figure is in Table 2 which lists the exact 50^{th}, 90^{th} and 99^{th} percentiles delay. The erasure-coding based approach has significantly smaller tail than other approaches (except flood). `flood` has the lowest latencies but has high overhead as discussed later.

ZebraNet trace with 34 nodes and 66 nodes respectively. Discounting source and destination, the total number of relays are 32 and 64 respectively. The distribution is shown in Complementary CDF (CCDF) curve.

Table 2 shows various data latency percentiles for both 34-node and 66-node experiments to facilitate the comparison of worse-case delay performance among all the algorithms considered.

Generally, `ec` has a higher 50^{th} percentile compared to other algorithms as shown in both Figure 2(a) and Figure 2(b) but a lower 99^{th} percentile. This is because it takes longer to find enough relays to distribute data replicas. However, once `ec` distributes enough code blocks by forwarding along multiple relays (the number of relays is larger than that used by `srep`), it takes a much shorter time to transfer the messages to the destination since any n/r relays are required to be successful. Since n is much larger than r, `ec` can fully utilize the diversity of multiple relays and is very robust to bad performance of individual relays. That is, in the presence of unpredicted failures or mobility of some of the relays, `ec` still has a good chance of sending the messages to the destination by routing code blocks through other functional relays.

Algorithm	34 nodes			66 nodes		
	50%	90%	99%	50%	90%	99%
ec-rep2-p8	0.44	0.84	1.32	—	—	—
ec-rep2-p16	0.53	0.85	1.21	0.51	0.83	1.17
ec-rep2-p32	—	—	—	0.59	0.82	1.04
srep-rep2	0.24	0.88	1.70	0.25	0.89	1.91
direct	0.49	1.63	3.27	0.51	1.79	3.54
history	0.18	0.87	9.50	0.14	0.72	10.83
flood	0.013	0.044	0.12	0.00012	0.0091	0.032

Table 2: Latency (in days) for different algorithms for two different node densities. This is the same data as shown in Figure 2. We see that `ec` has significantly lower 99^{th} percentile latency. This indicates that `ec` is effective in getting rid of very high latency cases.

Therefore, erasure-coding based routing is a promising candidate for opportunistic networks where (1) relay failures are prevalent and delays are unpredictable, and (2) minimizing the worst-case delay is important.

This observation is further supported by the data shown in Figure 2(b) where node density is higher. Given more contacts and relays, the CCDF curves of all forwarding algorithms become steeper. This is because there are more contacts overall. `ec`, as we have explained, still has the lowest 99^{th} percentile and the sharpest data latency curve. Therefore, given enough relay opportunities, `ec` has the best performance in delivering most of the messages the fastest among all the algorithms considered.

Simple replication, direct contact, and history-based algorithms, on the other hand, have very long tails (messages with much longer delays). This is because they use a small number of relays. Therefore, they cannot guarantee when these relays will see the destination. Very likely, some packets may encounter very long delays by selecting some relays that fail to deliver the message promptly. In the long run however, with sufficient buffer space, all messages will eventually be delivered. The lower the replication factor r, the longer the tail will be. This is illustrated by comparing the CCDF of `srep-rep2` and `direct`. Since `srep-rep2` replicates its data to two other relays, the chance of losing contact opportunities is lower than that for `direct`. Hence, `srep-rep2` has a shorter tail than `direct`.

The history approach, though having the lowest 50^{th} percentile delay, also has the longest tail among all the algorithms considered. The performance of `history` is dependent on the accuracy of its selection of highest ranked relays based on past statistics. If the decision is relatively accurate, it tends to find relays that will forward the data to the destinations very quickly. On the contrary, if the relays selected based on this heuristic do not reflect future forwarding probabilities, very long delays may be incurred. However, using certain timeout and retransmission schemes, these long-delay messages might be masked out which makes the history approach more attractive over the others in networks with predictable node movement. This is an interesting research direction to explore.

Finally, observe that the `flood` protocol in Figure 2(a) and Figure 2(b), has latency distribution curves which are almost vertical. This shows that `flood` has very low delays for all messages.

4.3.2 Routing overhead

Table 3 lists the routing overhead corresponding to each forwarding algorithm. Routing overhead is measured using the ratio of bytes transmitted to the bytes generated. Since both `ec` and `srep` transmit a fixed amount of data with respect to the data generated,

Algorithm	Overhead	
	(34 nodes)	(66 nodes)
ec-rep2-p8	3.96	—
ec-rep2-p16	3.96	3.98
ec-rep2-p32	—	3.98
srep-rep2	3.98	3.99
direct	1.0	1.0
history	30.28	59.61
flood	68.0	132.0

Table 3: Routing overhead of different forwarding algorithms for two node densities. Forwarding algorithms (such as `ec` and `srep`) which employ replication only at the source has significantly lower overhead. `flood` has almost an order of magnitude higher overhead and does not scale well as the number of nodes increase. The high overhead of `history` results from our implementation in `dtnsim2` where a copy of message is transmitted even when some copy of the original message has been delivered. Some timeout scheme can solve this problem by reducing unnecessary message transmissions.

their overhead is constant. For an algorithm with a replication factor of 2, the overhead should be 4, with 2 from the source to the relay and from the relay to the destination and the other 2 for the other relay. On the other hand, in both `history` and `flood` where relays also forward to other relays (and there are no restrictions on replication factor), multiple identical copies of the original message are transmitted even after the first delivery of the original message. As Table 3 shows, normally `history` has a higher overhead than `srep` and `ec`. This situation becomes worse when more contacts are available and very likely, more duplicate messages will be transmitted. For `flood`, almost all the nodes could receive a copy potentially and the overhead is proportional to $2n$, where n is the number of nodes. The factor of two comes because every relay sends to the destination (even if the destination has already received the message) in our implementation. Some simple timeout scheme, such as one that imposes a maximum number of hops a message can traverse, can alleviate this problem. However, data delivery rate will decrease if the number of hops a message can traverse is too small. The exploration of such a trade-off is part of our future work.

In summary, in terms of routing overhead, `ec` and `srep` scale well with node density and network size, while `flood` does not.

4.3.3 Data success rate

Algorithm	0.25 day	1 day	2 days	4 days	8 days
ec-rep2-p8	22.6%	95.9%	100%	100%	100%
ec-rep2-p16	9.2%	94.6%	100%	100%	100%
srep-rep2	51.8%	92.5%	99.6%	99.9%	99.9%
direct	32.0%	74.6%	94.2%	99.5%	99.9%
history	58.4%	87.9%	92.7%	94.6%	95.3%
flood	100%	100%	100%	100%	100%

Table 4: Data success rate of different algorithms for different deadlines. Even with extremely large deadline of 8 days simple replication can not transfer all its data. Also note that, `ec` has low data success rate when deadlines are extremely small and hence, caution must be used before deciding to use erasure coding.

Table 4 shows the data success rate for different algorithms with

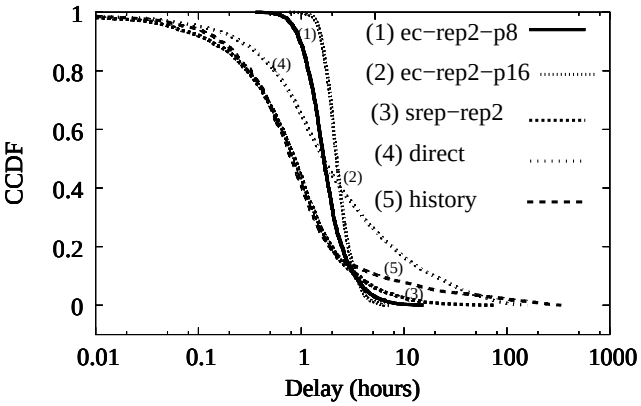


Figure 3: Latency distribution of different forwarding algorithms for the Pareto trace. We use a log-scaled x-axis for clarity. Similar to the ZebraNet trace we observe that tails are significantly smaller when `ec` is used, i.e., the worst case delays for other approaches are significantly higher. Since x-axis is log scale, the ratio of the worst case delay values is higher than in the ZebraNet trace.

deadlines smaller than the total simulation time. All deadlines are specified in units of **days**. The data success rate for `ec` is low if the deadlines are less than 6 hours long. However, for relatively long deadlines (between 1 and 2 days), `ec` has the highest data success rate. This result can be observed directly by looking at the data latency distribution curve. Because `ec` has a lower 99^{th} percentile of latency distribution, it will deliver more messages before that time and hence a higher data success rate. Therefore, if achieving low latencies for all messages or high success rate within certain reasonable deadlines are the application requirement, `ec` should be used.

On the other hand, `history` has the highest data success rate when the deadline is less than 6 hours. This is because `history` can find good relays without the need to distribute copies of data to many relays. The performance improvement of `history` upon `direct` and `srep` comes directly from the efficiency of its selection of good relays. However, since `history` has long tails in its data latency distribution curve, its data success rate is relatively low compared to other approaches.

4.4 Impact of mobility model

In this section, we evaluate the performance of `ec` and other approaches on a different mobility model. Our results here demonstrate that the idea of using erasure-coding based routing can be applied to different scenarios other than the ZebraNet trace. We find that the benefits of erasure coding are greater when the inter-contact times are heavy-tailed. We use such a heavy-tailed distribution for simulations in this section. The mobility model is based on the approximate power law distribution for inter-contact times observed for another set of real-world traces described in [2].

Figure 3 plots the CCDF of the data latency distribution for the Pareto trace. The other simulation parameters are exactly the same as in Section 4.1. Observe that all curves are much sharper than the ZebraNet traces. Again, `ec` has the sharpest CCDF curve and the lowest 99^{th} percentile delay, while all the other algorithms have higher worst case delays.

5. DELAY DISTRIBUTION ANALYSIS

This section discusses the theoretical behavior of the delay dis-

tribution when the erasure-coding based approach is used. We also theoretically compare the simple replication approach with the erasure-coding based approach.

5.1 Network model and assumptions

We consider the following scenario. Consider a source and destination pair which can communicate via one of the n distinct relays. Delay seen if relay i is used is a random variable with distribution X_i. Using a random variable allows us to model the fact that the delay encountered in using a relay is not known precisely. We assume the delay variables (X_i) to be identical and independent. Although a strong assumption, this should be seen as a first step towards a more sophisticated analysis. Even with these assumptions, we show interesting behavior which gives fundamental insights on the potential of using an erasure-coding based approach. We now introduce the concept of *order statistics*, the mathematical theory that is used to derive results.

5.2 Order statistics background

Consider n identical and independent (IID) random variables $X_1, X_2 \ldots X_n$. Let $Y_1^n, Y_2^n, \ldots Y_n^n$ be the random variables obtained by sorting the set $X_1, X_2 \ldots X_n$ in increasing order. The random variable Y_k is called the k^{th} *order statistic*.[2] One can obtain the distribution of Y_k in the following form [12]. Let $F(x)$ be the cumulative distribution function (CDF) of the variables X_i and $f(x)$ be the probability distribution function (PDF) (X_i are identical). Then the PDF (denoted by $g_k^n(y)$) of Y_k^n is given by:

$$g_k^n(y) = f(y)\frac{n!}{(k-1)!(n-k)!}F(y)^{k-1}(1-F(y))^{n-k} \quad (1)$$

5.3 Results for the delay distribution

In the erasure-coding based approach, the source erasure-codes a message and divides the code blocks equally among n relays. For simplicity, we assume that n is a multiple of r, and $n = rk$. A destination can decode a message as soon as k relays successfully deliver data. Let EC_r^n denote the delay distribution when `ec` is used with r replication and n relays. Using the terminology of order-statistics EC_r^n is defined as the k^{th} order statistic over the variables $\{X_1, X_2, \ldots, X_n\}$.

$$EC_r^n = Y_k^n \quad (2)$$

In simple replication, r relays are used and the message can be decoded when one of these relays successfully delivers data. Although the choice of relays could vary with the amount of information available to the forwarding algorithm, for simplicity we assume that the relays $1 \ldots r$ are used by the simple replication approach. Therefore, for simple replication, the observed delay is the minimum of the random variables $\{X_1, X_2, \ldots, X_r\}$.

$$srep_r = Y_1^r \quad (3)$$

Equation 1 (for distribution of the k^{th} order statistic) can now be used to obtain the delay distribution. However, it is hard to directly compare the statistics of these distributions from the above formula. In the rest of this section, we discuss two aspects based on the above analysis. First, we discuss the behavior of EC_r^n as a function of n (n measures how aggressively erasure coding is used). In particular, we derive the asymptotic distribution for EC_r^n (when $n \to \infty$). Second, we compare `srep` and EC_r^n analytically, for both finite n and as $n \to \infty$.

[2] $Y_1^n \equiv min_j(X_j)$ and $Y_n^n \equiv max_j(X_j)$.

5.4 Understanding the nature of EC_r^n

The exact distribution of EC_r^n depends on the distribution of X_i. For our analysis, we consider the case when X_i are given by the Pareto distribution. Recent work on mobility analysis has hinted that such delay distributions are heavy-tailed and thus it is an interesting case to discuss in-depth [2]. We also briefly consider the more traditional case when X_i are exponential (for example, when the underlying mobility model is random waypoint).

For a Pareto distribution, the probability distribution function is given by $f(x) = (b^\alpha)\alpha x^{-\alpha-1}$. The constant b is its scale parameter, and α is the power law coefficient. For $\alpha < 1$, $E[X] = \infty$ (i.e., using only one relay will have an infinite average delay). [3]

To understand the nature of EC_r^n, we plot the PDF of EC_r^n for different values of n. Figure 5.4 shows the PDF and CDF respectively. We consider the case when $\alpha = .6$. The choice of α is arbitrary and the only important aspect is that it presents a case when the distribution is extremely heavy-tailed.

Key Observations. As n increases, the distribution becomes thinner and more bell-shaped. In particular, the variance of the distribution reduces as n increases. Low variance means that the probability of encountering very large delays due to bad choice of relays is reduced. Also note that the mean delay reduces as n increases. We find that with $n = 32$, the mean delay is within 10% of the delay obtained when $n = \infty$ relays are used. This shows that convergence is reasonably quick, and most of the advantages of erasure coding can be obtained by using a moderate number of relays.

The following result formalizes the observations made above. The proof is based upon limiting analysis of quantiles of large number of independent variables and is beyond the scope of this paper [12].

RESULT 1. *Let the underlying delay variables X_i have a continuous PDF $f(x)$. Let $\zeta_{\frac{1}{r}}$ denote the $\frac{1}{r}^{th}$ quantile of the random variable X_i.* [4] *Then for sufficiently large n, the distribution of EC_r^n converges to the normal distribution in the following manner:*

$$EC_r^n \text{ is } \mathcal{N}(\zeta_{\frac{1}{r}}, \frac{(r-1)}{nf^2(\zeta_{\frac{1}{r}})}) \quad (4)$$

COROLLARY 1.

$$\lim_{n\to\infty} E[EC_r^n] = \zeta_{\frac{1}{r}} \quad (5)$$

$$\lim_{n\to\infty} Variance[EC_r^n] = 0 \quad (6)$$

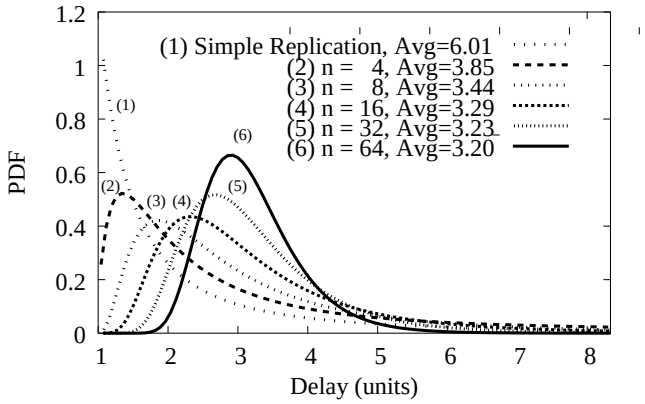
It follows immediately that, for large n, the distribution of EC_r^n converges to a constant. The mean delay is $\frac{1}{r}^{th}$ quantile of the original distribution. This result is also interesting because it indicates that only the $\zeta_{\frac{1}{r}}$ of the original distribution is relevant to characterize the performance when erasure coding is used.

COROLLARY 2. *When X_i have Pareto distributions with scale parameter b and power-law coefficient α, $E[EC_r^n]$ is given by,*

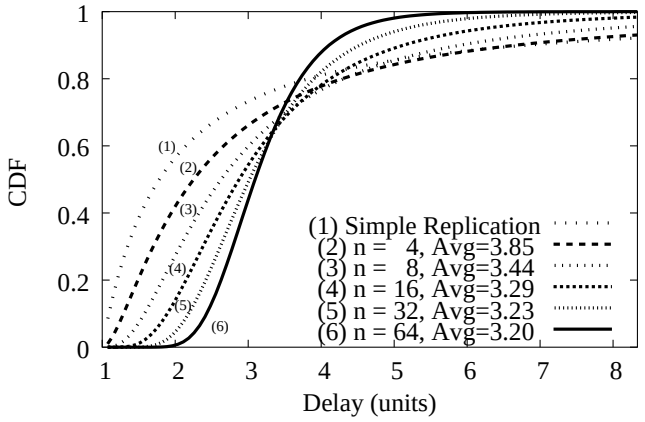
$$\lim_{k\to\infty} E[EC_r^n] = b\,(\frac{r}{r-1})^{\frac{1}{\alpha}} \quad (7)$$

[3] $E[\cdot]$ is used to denote expectation of a random variable.

[4] By definition, $Prob(X_i < \zeta_{\frac{1}{r}}) = \frac{1}{r}$. $\zeta_{\frac{1}{r}}$ is obtained by the inverse CDF function.



(a) Probability Distribution Function (PDF)



(b) Cumulative Distribution Function (CDF)

Figure 4: PDF and CDF of the delay distribution for different values of n when using the erasure-coding based approach. Underlying delay distribution is Pareto with $\alpha = .6$, and $b = 1$. $r = 2$. For each line, the value after the label *Avg* denotes the average value of the distribution ($E[EC_r^n]$). We observe that as n increases the distribution becomes thinner and more bell-shaped.

The above equation shows that for any value of α, the delay behavior of an erasure-coding based approach converges to a constant (for any finite amount of replication factor r, such that $r > 1$). This is particularly interesting because for $\alpha < 1$, the delay of using any single relay by itself is ∞. In fact, the `srep` algorithm, which uses the first r relays, will also have an infinite delay on average if the product $r\alpha < 1$. For this case, the erasure-coding based approach has infinitely better performance. This shows that for heavy-tailed delay distributions, the erasure-coding based approach is able to get rid of the tail (i.e long delays) whereas simple replication can not.

For the Pareto case, we can also derive the value of $E[EC_r^n]$ for a finite value of n. This is given by the following expression,

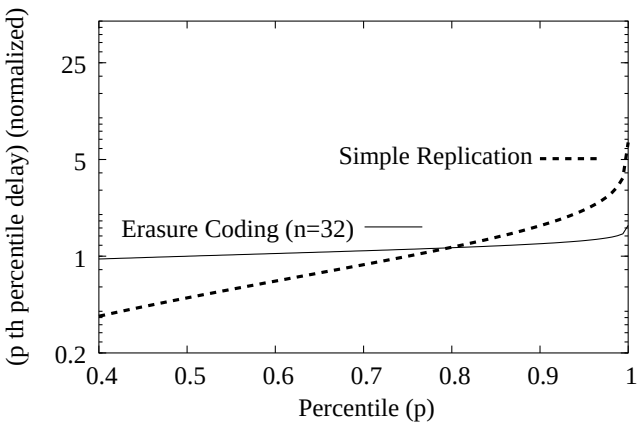
$$\frac{b}{E[EC_r^n]} = (1-\frac{1}{n\alpha})(1-\frac{1}{(n-1)\alpha})\ldots(1-\frac{1}{(n-(n/r)+1)\alpha})$$

This expression is useful in determining the value of n at which the expected delay converges to its eventual value $\zeta_{\frac{1}{r}}$.

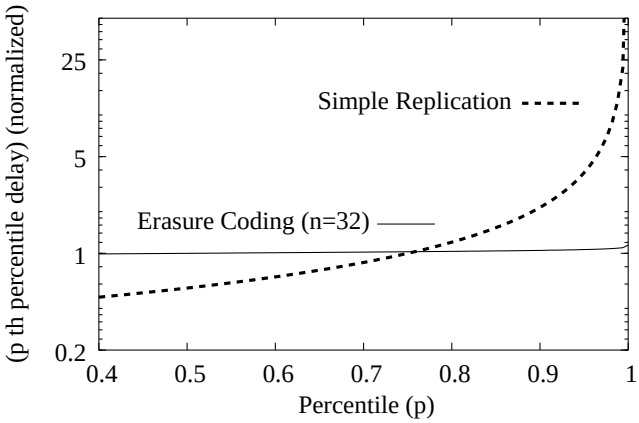
5.5 Comparing `ec` and `srep`

We argued above that the delay distribution when using erasure coding has low variance and therefore, it has lower worst case delay than simple replication. To support this, we compare the p^{th} percentile delay for `ec` and `srep`. The p^{th} percentile delay for a random delay variable Z is defined as the value w such that $Prob(Z \leq w) = p$. Figure 5 compares percentiles delay for `srep` and `ec` for two underlying delay distributions (X_i), a) Exponential, b) Pareto with $\alpha = .6$. $r = 2$ for both `srep` and `ec`, and $n = 32$ for `ec`. The delay values for each approach are normalized to the $\zeta_{\frac{1}{r}}$ of the underlying delay distribution X_i. This makes the plots independent of the scale parameters of the underlying distribution.

For lower percentiles, the erasure-coding based approach has higher delays. This implies that for the few best cases, simple repli-



(a) X_i are Exponential



(b) X_i are Pareto

Figure 5: Comparing percentiles delay for `ec` and `srep`. $r = 2$, $n = 32$. a) X_i are exponential b) X_i are Pareto ($\alpha = .6$). Delays are normalized to the $\zeta_{\frac{1}{r}}$ of the underlying distribution (X_i), which makes the plots independent of the scale parameters of the underlying distribution. The plot is obtained by first determining delay distributions in accordance to Equation 2 for the `ec` approach and Equation 3 for the simple replication approach. The percentiles are then analytically computed by inverting the cumulative distribution function. `ec` approach performs significantly better than `srep` when comparing higher percentiles of the resultant delay distribution. In fact, `ec` approach has almost non increasing delay values for different percentiles.

cation has lower delays than ec. However, the delays observed with ec towards the higher percentiles are much lower than srep. For ec, even for a moderate value of $n = 32$, delay values across percentiles are more or less the same, and close to the $\zeta_{\frac{1}{r}}$. For simple replication, delays for higher percentiles are very high. In this example, for the Pareto case, the 99^{th} percentile delay for srep is 25 times more than the erasure-coding based approach.

6. RELATED WORK

A series of efforts [8, 14, 9] in the context of sensor and mobile ad-hoc networks explore various forwarding algorithms for opportunistic networks. These techniques employ a form of data duplication (typically based on controlled flooding) to achieve eventual delivery.

The ZebraNet system uses a history-based protocol that leverages the contact history of nodes to the base station [8]. Simple replication approach which uses only one relay was shown to achieve optimal throughout in a mobile ad-hoc network [3] and has been further analyzed in [7, 2].

Erasure codes have recently been discussed and applied to many network applications, including achieving efficient distribution of bulk data in overlay networks [1] and P2P networks [10], coping with unreliable transmissions in wireless sensor networks [13], and achieving reliability in large-scale distributed storage systems [4]. Applying erasure coding to combat uncertainty in delay performance is the focus of our work. Since the idea of erasure-coding based forwarding is orthogonal to all other forwarding approaches, it can potentially be combined with them.

7. CONCLUSIONS AND FUTURE WORK

We presented a novel forwarding algorithm based on the use of erasure coding for opportunistic networks. The use of erasure coding spreads the responsibility of forwarding over many nodes while maintaining a fixed overhead. We showed using both analysis and simulation against a real-world mobility trace that our erasure-coding based approach significantly improves the worst case delay. At the same time, it has no "very small delay" cases. This is a natural consequence of how this approach works. We believe that the basic idea holds promise and an approach that combines erasure coding with other techniques, such as simple replication, may give us good performance on both fronts.

We have introduced the erasure coding idea in a simple setting in which all nodes are equally good relays. If different nodes have different characteristics, a more sophisticated approach to spread erasure code blocks is required. This is part of our ongoing effort and is discussed in [5]. We also plan to investigate the utility of the erasure coding approach on more mobility traces with diverse characteristics. Our analysis is preliminary and assumes that different relays used for forwarding data have independent delay distributions. This assumption breaks down if erasure coding uses the *best n* relays. Extending our analysis to incorporate such cases is an interesting future direction. Finally, based on the observation that different approaches have distinct advantages under certain mobility characteristics, it is desirable to have an adaptive strategy that selects one routing scheme over another on-the-fly, trying to achieve the best aspects of various forwarding algorithms.

ACKNOWLEDGMENTS

We gratefully acknowledge Andre Adler from IIT Chicago for pointing us to Sterfling [12] for the results regarding the distribution of order-statistics.

8. REFERENCES

[1] J. Byers, J. Considine, M. Mitzenmacher, and S. Rost. Informed Content Delivery Across Adaptive Overlay Networks. In *Procs. of ACM SIGCOMM*, 2002.

[2] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott. Pocket Switched Networks: Real-World Mobility and its Consequences for Opportunistic Forwarding. Technical Report UCAM-CL-TR-617, University of Cambridge, Computer Laboratory, February 2005.

[3] M. Grossglauser and D. Tse. Mobility Increases the Capacity of Ad-hoc Wireless Networks. *Transactions on Networking*, vol. 10, no. 4, Aug. 2002.

[4] Hakim Weatherspoon and John Kubiatowicz. Erasure Coding vs. Replication: A Quantitative Comparison. In *Procs. of IPTPS*, 2002.

[5] S. Jain, M. Demmer, R. Patra, and K. Fall. Using Redundancy to Cope with Failures in a Delay Tolerant Network. In *Procs. of ACM SIGCOMM*, 2005.

[6] S. Jain, K. Fall, and R. Patra. Routing in a Delay Tolerant Network. In *Procs. of ACM SIGCOMM*, 2004.

[7] S. Jain, R. Shah, W. Brunette, G. Borriello, and S. Roy. Exploiting Mobility for Energy Efficient Data Collection in Sensor Networks. In *Procs. of WiOpt*, 2004.

[8] P. Juang, H. Oki, Y. Wang, M. Martonosi, L.-S. Peh, and D. Rubenstein. Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet. In *Procs. of ACM ASPLOS*, 2002.

[9] A. Lindgren, A. Doria, and O. Schelén. Probabilistic Routing in Intermittently Connected Networks. In *Procs. of SAPIR*, 2004.

[10] P. Maymounkov and D. Mazieres. Rateless Codes and Big Downloads. In *Procs. of IPTPS*, 2003.

[11] M. Mitzenmacher. Digital Fountains: A Survey and Look Forward. *Information Theory Workshop*, 2004.

[12] R. J. Sterfling. *Approximation Theorems of Mathematical Statistics*. Wiley Series in Probability and Statistics, 1980.

[13] Sukun Kim and Rodrigo Fonseca and David Culler. Reliable Transfer on Wireless Sensor Networks. In *Procs. of IEEE SECON*, 2004.

[14] A. Vahdat and D. Becker. Epidemic Routing for Partially Connected Ad Hoc Networks. Technical Report CS-200006, Duke University, 2000.

[15] P. Zhang, C. M. Sadler, S. Lyon, and M. Martonosi. Hardware Design Experiences in ZebraNet. In *Procs. of ACM Sensys*, 2004.