Title
Using Redundancy to Cope with Failures in a Delay Tolerant Network

Journal
Computer Communication Review, 35(4)

ISSN
0146-4833

Authors
Jain, Sushant
Demmer, Michael
Patra, Rabin
et al.

Publication Date
2005-10-01

Peer reviewed

Using Redundancy to Cope with Failures in a Delay Tolerant Network

Sushant Jain
University of Washington
sushjain@cs.washington.edu

Michael Demmer, Rabin Patra
University of California, Berkeley
{demmer,rkpatra}@cs.berkeley.edu

Kevin Fall
Intel Research, Berkeley
kfall@intel.com

ABSTRACT

We consider the problem of routing in a delay tolerant network (DTN) in the presence of *path failures*. Previous work on DTN routing has focused on using precisely known network dynamics, which does not account for message losses due to link failures, buffer overruns, path selection errors, unscheduled delays, or other problems. We show how to split, replicate, and erasure code message fragments over multiple delivery paths to optimize the probability of successful message delivery. We provide a formulation of this problem and solve it for two cases: a 0/1 (Bernoulli) path delivery model where messages are either fully lost or delivered, and a Gaussian path delivery model where only a fraction of a message may be delivered. Ideas from the modern portfolio theory literature are borrowed to solve the underlying optimization problem. Our approach is directly relevant to solving similar problems that arise in replica placement in distributed file systems and virtual node placement in DHTs. In three different simulated DTN scenarios covering a wide range of applications, we show the effectiveness of our approach in handling failures.

Categories and Subject Descriptors:
C.2.2: Routing Protocols
General Terms: Algorithms, Performance, Theory
Keywords: Routing, Delay Tolerant Network

1. INTRODUCTION

Several issues related to the handling of data communications in the presence of interruptions and heterogeneity have been raised by recent work in the area of delay tolerant networking (DTN) [7]. DTNs are characterized by a lack of consistent end-to-end paths due to interruptions that may be either planned or unplanned. Selecting routing paths is considered to be the most challenging problem in such networks. Previous work in DTN routing addresses this problem in cases when the link up/down periods, capacities, and delays are known in advance [8].

If such information is not known with certainty, or if links may drop or corrupt messages that are in transit, the effectiveness of these approaches will likely suffer. Many real-world scenarios exhibit exactly these problems. For example, consider a scenario in which mobile entities (called data MULEs [18, 25]) are used to carry data through the network. As the physical path of a MULE through the environment may not be fully known, messages on a MULE may expire before reaching their destinations. Other causes of data loss in DTNs include link transmission errors, message drops due to insufficient buffer space, or failure of a node along the path. In this work, we take a step towards understanding the best techniques available to cope with path failures in DTNs.

In many transport protocols, reliability is achieved using acknowledgments and retransmissions. Due to the intermittent nature of DTNs, timely feedback may not be possible and therefore retransmission schemes may be of limited efficacy. A more attractive approach is to use *replication* (redundancy) and send identical copies of a message *simultaneously* over multiple paths to mitigate the effects of a single path failure [9, 21]. This is in contrast to retransmission schemes which typically wait for a message to be lost before sending another copy. At the same time, *erasure coding* techniques have long been used to cope with partial data loss efficiently [3]. A natural question to investigate is whether erasure coding techniques can be beneficial in DTN context as well.

Applying erasure coding to DTNs turns out to be an interesting and non-trivial exercise. The path model in a DTN is substantially different than in traditional erasure coding applications, as a DTN path is available for a short period of time and has a finite volume to carry data. Furthermore, messages have a finite lifetime (deadline) before they expire which adds another dimension to the problem. Our first contribution is a full characterization of these issues by formulating an optimization problem to cope with path failures and uncertainties. The problem is to determine an *optimal allocation* of erasure code blocks over multiple paths to maximize the probability of successful message delivery. This allocation problem is challenging and has some surprising results.

We show through both derivations and simulations that there is no simple "one size fits all" answer to the question of whether erasure coding is beneficial. We outline three different "regimes" based on the underlying path failure probabilities and redundancy used. Using ideas from modern portfolio theory [1], we propose an efficient algorithm to solve the above problem and demonstrate its efficacy as compared to simple replication and other heuristics in three different DTN scenarios. In carrying out this investigation, we were surprised by the complex nature of many of the issues involved. We now present some motivating examples to illustrate these complexities.

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

1.1 Replication and Erasure Coding

Consider the topology shown in Figure 1. Assume that transmissions over each edge succeed or fail independently with a certain probability, and that all failures are Bernoulli (i.e., if an edge succeeds all the data is received, otherwise nothing is received). Given a message to be transmitted from the source to the destination, our objective is to determine which paths to use to maximize the probability of successful reception.

We assume that an erasure coding algorithm can be used to encode a message into a large number of *code blocks*, such that if a fraction $1/r$ or more of the code blocks is received, the message can be decoded. The parameter r determines the degree of redundancy and is called the *replication factor*. In this example r is two. Thus, if at least 50% of the code blocks are received, the message can be decoded. The *allocation* problem is to determine the optimal fraction x_i of the erasure code blocks that should be sent on the i^{th} path. Figure 1 illustrates five different scenarios of edge success probabilities and the corresponding optimal allocations.

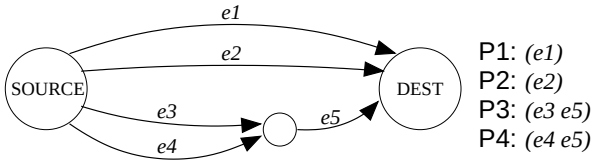
In the first scenario, all paths have the same success probability of .80. The optimal approach is to allocate the code blocks equally among all four paths. With this allocation, the success probability is $\binom{4}{2}(.8)^2(.2)^2 + \binom{4}{3}(.8)^3(.2) + \binom{4}{4}(.8)^4 = .9728$. If we were to *not* employ erasure coding and instead simply send two copies of the message over two paths (any two), the success probability would be $1-(.2)^2 = .96$. Informally, by spreading code blocks over more paths, we reduce the chance that failures of a few *specific* paths causes messages to be lost.

In the second scenario, even though the path probabilities are different, the optimal allocation is still to assign an equal number of code blocks to each path. It may be surprising that it is not beneficial to send more code blocks on paths with higher probability. However, the overall success probability is a function of the number of *subsets* of paths that contain at least $1/r$ of the code blocks. If the algorithm were to send more than a quarter of the code blocks on the higher probability paths $P1$ and $P2$, then together $P3$ and $P4$ would be allocated less than 50% of the code blocks, and if only paths $P3$ and $P4$ are successful, the message could not be successfully decoded. This shows the complex combinatorial nature of this problem. Interestingly, if the path failure behavior is partial (not Bernoulli), the optimal allocation can be different (Section 4).

The third scenario illustrates how the optimal allocation depends on the *absolute* value of the path probabilities, not solely on the relative probabilities between paths. Here, each path succeeds with equal probability (as in scenario 1), yet the optimal allocation is to place 50% of the blocks on any two of the four paths. The success probability of this allocation is $1-(.6)^2 = 0.84$. If the allocation were spread equally as in the previous example, then the probability that at least two of the four paths succeed is $\binom{4}{2}(.6)^2(.2)^2 + \binom{4}{3}(.6)^3.2 + \binom{4}{4}(.6)^4 = .8208$.

This example shows that when individual paths are less reliable, it is better to carry the code blocks on fewer paths. This allocation is equivalent to a simple replication scheme in which identical copies of the original message are sent on the two paths. This result also depends on the number of paths available, as with more choices, equal allocation among all paths is better. We discuss this property in Section 3.

Finally, by comparing the fourth and fifth allocation scenarios, we illustrate the effect of path dependency. Paths $P3$ and $P4$ share a common edge $e5$ and are thus dependent when $e5$ has a success probability less than 1.0. In both scenarios all *paths* have an equal success probability of 0.81. When the paths are independent (scenario 4), the optimal allocation is to spread code blocks equally. However, when paths are dependent (scenario 5), the benefits of using both $P3$ and $P4$ are reduced since they both fail if the common edge $e5$ fails. The optimal allocation is to choose any two of the paths, except for $P3$ and $P4$ together.



Scenario	Path Success Probability						Allocation			
	P1	P2	P3		P4					
	e_1	e_2	e_3	e_5	e_4	e_5	x_1	x_2	x_3	x_4
1	.80	.80	.80	1.0	.80	1.0	.25	.25	.25	.25
2	.89	.86	.83	1.0	.80	1.0	.25	.25	.25	.25
3	.60	.60	.60	1.0	.60	1.0	.50	0	.50	0
4	.81	.81	.81	1.0	.81	1.0	.25	.25	.25	.25
5	.81	.81	.90	.90	.90	.90	.50	.50	0	0

Figure 1: A simple topology in which the source and the destination are connected by four paths; two of which, P3 and P4, are dependent because they share an edge (e5). Table shows different scenarios of success probabilities for different edges. The right half shows the optimal allocation x_i of erasure code blocks for the corresponding scenario ($r = 2$, in all cases). The optimal allocation is obtained using the algorithm in Section 3.2. These examples illustrate that optimal allocation is complex and depends vastly on input probabilities.

These examples illustrate that even for simple topologies determining the optimal allocation can be complex. The optimal allocation is determined not only by the path probabilities, but also on the number of paths, the replication factor, path dependencies and the failure model.

1.2 Related Problems

We now briefly present two open problems from other areas which are similar to our allocation problem. Other related areas are discussed in Section 10.

The first problem is of optimal replica placement in a large scale distributed data management system. The goal is use redundancy to minimize the probability of data loss. Encouragingly, some basic studies show the advantages of using erasure coding with replication [23, 16]. Our work is a generalization of this basic problem and, as we will show, outlines the range of cases in which erasure coding may or may not be beneficial.

A related problem exists in mapping of virtual nodes to physical nodes in a distributed hashtable (DHT). Assume that there are N virtual nodes. By definition, a DHT is consistent only if at least K ($K < N$) virtual nodes are available. The mapping problem is to determine on which physical servers should the virtual nodes be mapped to maximize the probability that the DHT remains consistent.

2. ALLOCATION PROBLEM FORMULATION

We now formulate the allocation problem in a DTN context.

A DTN network can be considered as a *time-variant* set of *contacts* (a contact is defined as an opportunity to send data). The maximum amount of data that can be transmitted on a

contact is termed the *contact volume*, and is defined as the product of the contact duration and the link bandwidth during that duration. A *path* is defined as a sequence of contacts. The *path volume* is the minimum contact volume of all contacts of the path.

Messages are transferred along a path in a store-and-forward manner. If the next-contact (or next-hop) is unavailable, messages are buffered until the contact becomes available or the message expires. For a given message, there will typically be several paths that can be used to deliver it to the destination. We use the term *feasible paths* to denote all paths to the destination which have a delay less than the lifetime of a message. This approach allows us to treat delay as a constraint, rather than an optimization metric. Feasible paths are discussed again later in this section when discussing optimization metrics.

As mentioned in the introduction, path failures may result from a variety of problems that may be difficult to characterize precisely. To capture this uncertainty, we represent the outcome of a message transmission over path i using a random variable S_i. S_i is defined as the fraction of the data sent on path i that is received at the destination. Each S_i takes values ranging from 0 (no data received) to 1 (all data received). Although the examples in Figure 1 assumed each S_i is Bernoulli, in general S_i can have any probability distribution. With these definitions, we now formally state our optimization problem.

Formal Problem Definition:

Consider a node $\boldsymbol{s}$ sending a message of size m to node $\boldsymbol{d}$, and let there be n feasible paths from $\boldsymbol{s}$ to $\boldsymbol{d}$. For each path i, let V_i be the volume of the path, and let S_i be a random variable that represents the fraction of data successfully transmitted on path i.

Assume that an erasure coding algorithm can be used (with a replication factor r) to generate $b = (mr)/l$ code blocks of size l such that any m/l code blocks can be used to decode the message.

The Optimal Allocation problem is to determine what fraction (x_i) of the b code blocks should be sent on the i^{th} path, subject to the path volume constraint, to maximize the overall probability of message delivery.

Formally, let $Y = \sum_{i=1}^{n} x_i S_i$. *Find* $(x_1, x_2, \ldots, x_n)$ *that maximize* $Prob\left(Y \geq r^{-1}\right)$, *where* $\sum_{j=1}^{n} x_j = 1$ *and* $\forall i \in 1 \ldots n$, $0 \leq x_i \leq \frac{V_i}{mr}$.

Explanation of the formulation

The number of code blocks received over the i^{th} path is the random variable bx_iS_i, where $b = \frac{mr}{l}$ is the total number of code blocks generated. The total number of code blocks received over all paths are, therefore, $Y' = \sum_{i=1}^{n} bx_iS_i$. The message can be decoded if at least b/r code blocks are received. Therefore, maximizing the probability that $Y' \geq b/r$ is same as maximizing the probability that $Y \geq r^{-1}$, where $Y = \frac{Y'}{b} = \sum_{i=1}^{n} x_iS_i$.

The i^{th} path has maximum volume V_i (i.e., at most V_i bytes can be sent over it), so it can carry at most V_i/l code blocks. If two paths share a common contact, we assume that the bandwidth is pre-allocated to each path. When expressed as a fraction of the total number of code blocks, this corresponds to an upper bound (u_i) for the i^{th} path. $u_i = \frac{V_i/l}{b} = \frac{V_i}{mr}$. This constraint will be called the *volume constraint*, and the constraint $\sum_{i=1}^{n} x_i = 1$ will be called the *normalization constraint*.

Optimization Metrics

Several metrics can be used to characterize performance in our problem, including message delay, probability of successful delivery, bandwidth overhead, etc. The replication factor r in the formulation captures the bandwidth overhead. By using a higher replication factor, we can achieve lower delay and higher success probability at the expense of increased bandwidth overhead. For example, flooding achieves the smallest possible delay at the cost of high bandwidth overhead. In some cases, delay and success probability can be in competition, as in the choice between a highly reliable, long-latency path and a lossy but low-latency path.

In our formulation, we have chosen to fix the replication factor and the maximum delay as constraints, and then try to maximize the probability of delivery. In the presence of unreliable paths, maximizing the chance of a message arriving "on time" is preferable to minimizing the transit time if the message arrives at all. Attempting to minimize delay and maximize delivery probability *simultaneously* presents a challenging multidimensional optimization problem. In particular, even computing the average delay for a given allocation is hard because it depends on the *precise* combination of which paths were successful.

Finally, note that by using feasible paths we are able to capture delay constraints. To find feasible paths, we assume that the routing algorithm selects only those paths that have delay less than the message lifetime. The routing algorithm presented in prior work [8] satisfies this requirement. If there are paths that *may* meet the delay constraint only with a certain probability p ($p < 1$), then those paths are included by scaling the corresponding S_i by a factor of p.

Why is this problem hard?

Despite a relatively straightforward formulation, the optimization problem as described above is quite complex. The random variable Y is a convolution of multiple distributions and does not have a closed form expression for arbitrary S_i. In the special case where all the S_i can be approximated by Gaussians, then Y also has a Gaussian distribution, and we are able to obtain closed form results. On the other hand, if the S_i are Bernoulli, then optimizing $P(Y \geq r^{-1})$ becomes hard because of the combinatorics involved. We are currently investigating the computational complexity of the problem. As a preliminary result, we have proved that computing $P(Y \geq r^{-1})$ for given x_i is NP-hard [17].

3. BERNOULLI (0-1) PATH FAILURES

We now address the allocation problem when path failures are Bernoulli. This represents a complete success or failure case for paths: when a path fails, all the messages sent on it are lost; when a path succeeds, all the messages sent on it are successfully received. This model is appropriate for many failures that occur in real-world scenarios, such as message expiration due to long delays, contact failures, node failures, etc.

3.1 S_i are identical and independent (IID)

We first examine a simplified case when the random variables S_i are IID and there are no volume constraints. The observations in this section lay the foundation for the more general examination in subsequent analysis. Let there be n paths, each with a success probability p. Consider the following family of

allocation strategies, where the k^{th} strategy is defined by:

$$x_i \quad = \quad \begin{cases} \frac{1}{k} & \text{if } 1 \leq i \leq k \\ 0 & otherwise \end{cases}$$

Strategy k *splits* the code blocks equally among the first k paths (for simplicity we assume that k is a multiple of r) and ignores the rest of the paths. The term *split* denotes that code blocks from the original message will be spread equally over multiple paths. k measures the degree of splitting. By comparing performance of strategies for different k we can gauge whether splitting is beneficial. Let $\mathcal{P}(k)$ denote the probability of success for the k^{th} allocation strategy. It is the probability that at least k/r out of the first k paths succeed, which is:

$$\mathcal{P}(k) = \sum_{i=k/r}^{k} p^i (1-p)^{k-i} \binom{k}{i}$$

Figure 2 plots $\mathcal{P}(k)$ as a function of the number of paths (k) used for different values of p. A line with increasing values indicates that the success probability increases as we split over more paths. For each probability p, we would like to identify the k for which $\mathcal{P}(k)$ has the maximum value. Observe that different values of p have vastly different behavior. This range of values can be classified into three main regimes:

Case 1: Large p ($.67 < p \leq 1$): These lines are always increasing and asymptotically converge to 1.0. In this case, $\forall k,\ \mathcal{P}(k+1) > \mathcal{P}(k)$, hence, it is beneficial to split as much as possible.

Case 2: Medium p ($.5 < p \leq .67$): These lines have an initial dip but they eventually increase and converge to 1.0. Here, $\exists\ k_o,\ such\ that\ \forall k > k_o,\ \mathcal{P}(k+1) > \mathcal{P}(k)$, hence it is beneficial to split only if k is sufficiently large.

Case 3: Small p ($0 \leq p \leq .5$): The lines are decreasing and asymptotically approach 0. In this case, $\forall k,\ \mathcal{P}(k+1) < \mathcal{P}(k)$, hence it is never beneficial to split on more than r paths.

Intuitively, a larger k is equivalent to more trials of a random experiment with success probability p. If $p > .5$, then the odds are in favor of success, and more trials reduces the variance and increases the overall probability. This behavior for large k can also be deduced from the weak law of large numbers. Similarly, if $p < .5$, then the odds of success are reduced with more trials.

Further evaluation with different values of r also shows a similar characterization into these three regimes. However, the three regimes are defined by the product pr, rather than the absolute value of p. For all values of r, case 1 corresponds to values of p such that $pr > \frac{4}{3}$, case 2 occurs when $1 < pr \leq \frac{4}{3}$, and case 3 occurs when $pr \leq 1$.[1] Details and proofs are covered in the technical report [17].

3.2 S_i are different

We now turn our attention to the case when paths may have different and dependent probabilities. Our solution approach uses a Mixed Integer Program (MIP) to capture the objective function $Prob(Y > r^{-1})$.

We start with a few definitions. Given n paths, there are 2^n possible outcomes corresponding to the different combinations of successful paths. Let the possible outcomes be numbered $0 \ldots (2^n - 1)$. The binary representation of an n-bit integer j can be used to encode the success of the j^{th} outcome in the following manner. Define c_{ji} to be 1 if the binary representation of j has a 1 at the i^{th} position and 0 otherwise. Let c_{ji} encode whether the i^{th} path was successful in the j^{th} outcome:

$$c_{ji} \quad = \quad \frac{j}{2^{i-1}} \ mod\ 2 \ \text{ for } i \in 1 \ldots n$$

Let w_j denote the probability that the j^{th} outcome occurs, i.e., $Prob(\cap_{i=1}^{n} S_i = c_{ji})$. Note that w_j is an input to the formulation as it is a function of the underlying path probabilities. For the case when the paths S_i are independent, w_j is given by $\prod_{i=1}^{n} (p_i c_{ji} + (1-p_i)(1-c_{ji}))$. Recall that u_i represents the path volume constraint and is also an input to the formulation. If paths are *dependent*, we need the joint probability distribution of S_i's to determine w_j.

To capture the probability corresponding to the j^{th} outcome, we define a binary variable y_j. The variable y_j is 1 only if the x_i's are chosen such that the sum of x_i corresponding to the successful paths in the j^{th} outcome is greater than or equal to $1/r$. We now formulate the MIP.

$$\text{Maximize} \sum_{j=0}^{2^n-1} y_j w_j \qquad \text{subject to,} \tag{1}$$

$$y_j = \{0,1\}, \quad \sum_{i=1}^{n} c_{ji} x_i \geq y_j / r \qquad \text{for } j \in 0 \ldots (2^n - 1) \tag{2}$$

$$0 \leq x_i \leq u_i \qquad \text{for } i \in 1 \ldots n \tag{3}$$

$$\sum_{i=1}^{n} x_i = 1 \tag{4}$$

Equation 3 captures volume constraints u_i on path p_i (Section 2) on x_i. Equation 4 captures the fact that all the x_i should sum up to 1.0. Finally, note that the objective function of the formulation $Prob(Y > r^{-1})$ can be rewritten as:

$$\sum_{j=0}^{2^n-1} Prob\left(Y > r^{-1} \,\middle|\, j^{th}\ outcome\right) Prob\left(j^{th}\ outcome\right)$$

This is because all the 2^n outcomes are mutually exclusive and exhaustive. Therefore, the objective function has a term $y_j w_j$ for the j^{th} outcome.

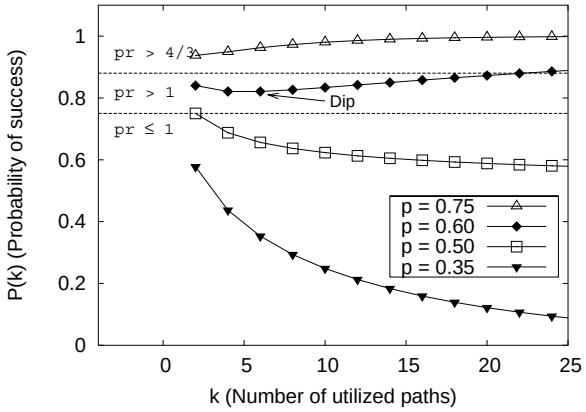


Figure 2: $\mathcal{P}(k)$ as a function of the number of utilized paths k. $r = 2$, and different lines correspond to different values of p. The nature of $\mathcal{P}(k)$ varies greatly depending upon whether $pr > 1$ or not. As k approaches infinity, if $pr > 1$, the line converges to 1. If $pr = 1$, the line converges to .5. If $pr < 1$ the line converges to 0. A dip in the line occurs when pr is slightly greater than one and k is relatively small.

[1]The value $\frac{4}{3}$ can be understood relatively easily when $r = 2$. Here, $\mathcal{P}(k+1) > \mathcal{P}(k)$ iff $(k+1)(1-p) > kp$. Substituting $k = 1$, we get that $\mathcal{P}(k+1) > \mathcal{P}(k)$ iff $pr > \frac{4}{3}$.

The formulation has an exponential number of constraints (2^n) and uses integer variables. This is not surprising given that even computing $Prob(Y > r^{-1})$ is NP-hard [17]. If paths are dependent, the description of the input (w_j) itself may be exponential in n. Despite these difficulties, we were able to use this formulation to solve problems with n as large as 16 using the CPLEX optimization suite [5] in typically less than 15 minutes.[2] The MIP formulation also allows us to gain insight into simple examples such as those discussed in the introduction. For many cases with larger n, we used the MIP approach on the best 16 paths to get a lower bound on the solution. We can also ignore the integrality constraints on y_j and convert the MIP to a linear program which can be solved much more efficiently. This will give us an upper bound on the solution.

4. PARTIAL PATH FAILURES

We now consider the case when Y can be approximated by a Gaussian distribution. If n is moderately large, we can use the central limit theorem to argue that Y can be approximated by a Gaussian distribution. This approximation, however, requires that the mean and the variance of each individual term x_iS_i be very small relative to the mean and the variance of Y. For our application, this means that as long as the path probabilities have comparable mean and variance, this approximation is reasonable. Also, if S_i can be captured by a Gaussian distribution, then Y is also Gaussian because a linear combination of Gaussian random variables is also Gaussian.

We approach this case in two steps. First, we show that maximizing $Prob(Y > r^{-1})$ is equivalent to maximizing the *Sharpe-Ratio* . We then apply results from the economic theory literature to maximize this ratio.

4.1 New objective function

If Y is Gaussian with mean μ_Y and variance σ_Y^2, then:

$$Prob(Y > r^{-1}) = \frac{1}{2}\left(1 + erf\left(\frac{\mu_Y - 1/r}{\sigma_Y\sqrt{2}}\right)\right), \text{where,}$$

$$erf(z) \equiv \frac{2}{\sqrt{\pi}}\int_0^z e^{-t^2}dt$$

$erf(z)$ is the *Gaussian error function* and is strictly increasing. Therefore, maximizing $Prob(Y > r^{-1})$ is equivalent to maximizing the argument of the erf function, $\frac{\mu_Y - 1/r}{\sigma_Y}$. This is a significant simplification (compared to the Bernoulli case) because the objective function has a closed form expression in terms of the mean and the variance of Y. Recall that $Y = \sum_{i=1}^{n} x_iS_i$ and hence, we can express μ_Y and σ_Y as:

$$\mu_Y = \sum_{i=1}^{n} x_ip_i, \qquad \sigma_Y^2 = \sum_{i=1}^{n}\sum_{j=1}^{n} x_ix_j\sigma_{ij}$$

Here, p_i is the success probability for the i^{th} path ($E[S_i]$) and σ_{ij} is the covariance between S_i (i^{th} path) and S_j (j^{th} path).

The ratio $\frac{\mu_Y - 1/r}{\sigma_Y}$ is called the *Sharpe-Ratio* [1], and plays an important role in the theory of allocating assets in investment portfolios. It measures the expected added return per unit of added risk. We now investigate how to maximize it.

[2]The CPLEX solver was configured to accept any solution which was within 1% of the optimal solution. CPLEX was run on a 2-processor Xeon (3.2GHz) machine with 2 GB of RAM.

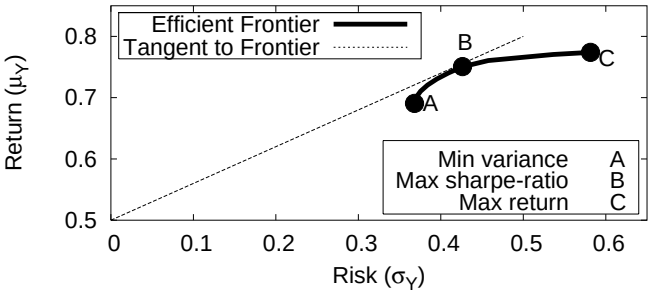


Figure 3: Efficient frontier generated from an experiment with six paths with probabilities .85, .7, .65, .65, .6 and .6 and $r = 2$. For any point (x, y) on the frontier, the slope of a line from the point $(0, 1/r)$ to (x, y) measures the *Sharpe-Ratio* at (x, y). This ratio is maximized by the line (drawn from the point $(0, 1/r)$) that is tangent to the frontier.

4.2 Maximizing the Sharpe-Ratio ($\frac{\mu_Y - 1/r}{\sigma_Y}$)

Maximizing the Sharpe-Ratio is a natural exercise in asset management, and we adopt the related economic terminology by first formulating a standard portfolio optimization problem:

Consider an investor who has the choice to invest in n assets. The return on the i^{th} assets is described by a random variable S_i. Given a utility function that represents the interests of the investor, in what proportion should she invest her wealth on various assets to maximize her utility function. The vector of allocation weights (amount invested in each asset) is called a "portfolio," and is represented by $(x_1, x_2 \ldots x_k)$. In these terms, different assets correspond to different paths in a DTN, and x_i becomes the amount of traffic assigned to each path. We seek to maximize a utility function, which for DTN will mean trying to maximize the probability that a message can be successfully decoded from its code blocks when transported over lossy paths. For Gaussian failures this is same as maximizing a portfolio's return while minimizing its risk and is captured by the Sharpe-Ratio.

To optimize the Sharpe-Ratio, we recall the notion of an *efficient frontier*, proposed by Markowitz [1]. The efficient frontier represents all the portfolios which have the "highest return for a given risk" or, equivalently, the "lowest risk for a given return."[3] Risk is measured by σ_Y. Figure 3 shows an example efficient frontier. It is easy to see that the portfolio that maximizes the Sharpe-Ratio also lies on the efficient frontier. This is because the Sharpe-Ratio has the mean in the numerator and the variance in the denominator.[4] Our problem is now reduced to finding a point on the efficient frontier that maximizes the Sharpe-Ratio. Geometrically, this is the tangent from the point $(0, 1/r)$ to the efficient frontier, as shown in Figure 3.

To find a portfolio that maximizes the Sharpe-Ratio, we use a simple numerical search on the efficient frontier. It can be shown that under mild conditions the Sharpe-Ratio function is concave. This allows us to do an efficient one dimensional search over the efficient frontier.[5] The complexity of this approach is

[3]A point (σ_1, μ_1) on the efficient frontier is obtained by solving the following quadratic optimization problem:
Minimize ${\sigma_1}^2$, where $\sigma_1^2 = \sum_{i=1}^{n}\sum_{j=1}^{n} x_ix_j\sigma_{ij}$,
Subject to: $\sum_{i=1}^{n} x_ip_i = \mu_1$, $\sum_{j=1}^{n} x_j = 1$, and $0 \le x_i \le u_i$

[4]For now, assume that the numerator is positive for the optimal allocation. The case when the numerator is always negative occurs when paths have very low probability and is less interesting. This also guides us that the replication factor should be large enough to make $(\mu_Y - 1/r)$ positive.

[5]A one dimensional search is based upon finding a point for

$O(In^6)$, where I is the number of iterations in the numeric search and n^6 is the complexity of solving a convex quadratic optimization problem. This numerical approach will be referred to as the **Markowitz Numeric** algorithm.

4.3 Approximation for optimizing the Sharpe-Ratio

We now present an efficient approximation to optimize the Sharpe-Ratio. The key observation is that if we remove the constraints that bound the value of x_i ($0 \leq x_i \leq u_i$), a closed form solution can be obtained. The following equation gives a portfolio $\hat{x} \equiv (x_1, x_2 \ldots x_n)$ that maximizes the Sharpe-Ratio.

$$\hat{x} = \frac{\hat{V}^{-1}(r\hat{p} - \hat{1})}{\hat{1}\hat{V}^{-1}(r\hat{p} - \hat{1})} \tag{5}$$

Here, $\hat{V}$ is the covariance matrix, i.e $v_{ij} = \sigma_{ij}$, $\hat{p}$ is the n-dimensional vector of the path probabilities and $\hat{1}$ is the unit vector of dimension n.

The derivation of the optimal point is discussed elsewhere [1]. The above solution is obtained by assuming that the numerator of the Sharpe-Ratio is positive. This assumption does not hold true if all paths have low probability of success. This is similar to our discussion in the Bernoulli case when $p < 1/r$, and splitting has no benefits. For such cases, we take the greedy approach of maximizing μ_Y by choosing paths with the highest p_i, subject to the volume constraints. Note that, when the p_i are different the product $max(\mu_Y)\ r$ represents the same concept as the product pr (when p are identical, as in Section 3).

Equation 5 requires solving a set of n linear equations and has complexity $O(n^3)$. The solution is greatly simplified if we assume that the paths are independent. When paths are independent, $\hat{V}$ is a diagonal matrix and if we let $\hat{W} = \hat{V}^{-1}$, then $w_{ii} = 1/v_{ii}$. Using equation 5,

$$x_i \propto \frac{(p_i - 1/r)}{\sigma_{ii}} \tag{6}$$

The above solution is quite elegant. It states that for the i^{th} path, x_i should be proportional to the path's excess probability $(p_i - 1/r)$ and inversely proportional to the path's variance. The proportionality constant is determined by observing that $\sum_{i=1}^{n} x_i = 1$.

The above formula may lead to assignments of the x_i that do not satisfy the volume constraints. Algorithm 1 is a simple *normalization* algorithm devised to enforce the volume constraints. If any x_i is negative, it is set to zero and if $x_i > u_i$, it is set to u_i. This approach, i.e., using Equation 5 followed by the normalization algorithm, will be referred to as the **Markowitz** algorithm and is summarized as Algorithm 2.

4.4 Sharpe-Ratio for Other Distributions

Although the argument that optimizing the Sharpe-Ratio maximizes the $Prob(Y > r^{-1})$ relies on the Gaussian assumption, we now discuss why maximizing the Sharpe-Ratio is a good approach for other distributions as well. The numerator in the Sharpe-Ratio denotes the average amount of data that is received in excess of what is required to decode the message $(1/r)$. If the standard deviation of Y (denominator) is small, the probability that Y deviates from its mean is also small, by Chebyshev's inequality. Therefore, if the numerator is positive, i.e., $\mu_Y > 1/r$, it is reasonable to minimize the standard deviation (thereby maximizing the Sharpe-Ratio). The ratio captures both the excess return and the risk of not obtaining the expected return. It gives us one way of systematically approaching the problem. The beauty of this approach is that it accounts for complex aspects such as path correlations and is computationally efficient.

which the slope of the function is close to zero.

Normalization algorithm
Input: n, $\{u_i\}$, $\{x_i\}$ $i \in 1..n$
Output: $\{x_i\}$
1: $\forall i \in 1..n,\ x_i = x_i / \sum_1^n x_j$
2: $\forall i \in 1..n$, **if** $x_i < 0$ **then** $x_i = 0$
3: $\forall i \in 1..n$, **if** $x_i > u_i$ **then** $x_i = u_i$
4: **if** $\sum_1^n x_i > 1$ **then**
5: $\quad \forall i \in 1..n,\ x_i = x_i / \sum_1^n x_j$
6: **else**
7: $\quad l = 1 - \sum_1^n x_i$
8: $\quad$ **while** $l > 0$ **do**
9: $\quad\quad t = arg\max_{i \in 1..n}\{p_i \mid x_i < u_i\}$
10: $\quad\quad \Delta = min(u_t - x_t, l)$
11: $\quad\quad l = l - \Delta\ ;\ \ x_t = x_t + \Delta$
12: $\quad$ **end while**
13: **end if**

Algorithm 1: Normalization algorithm to enforce volume constraints ($0 \leq x_i \leq u_i$) and normalize x_i ($\sum_{i=1}^{n} x_i = 1$). Lines 7-12 use a simple heuristic to adjust for slack variables after the constraints are enforced. Its complexity is $O(nlog(n))$.

Markowitz algorithm
1: **if** $\max(\mu_Y) > 1/r$ **then**
2: $\quad \hat{x} = \frac{\hat{V}^{-1}(r\hat{p} - \hat{1})}{\hat{1}\hat{V}^{-1}(r\hat{p} - \hat{1})}$
3: **else**
4: $\quad$ Maximize μ_Y
5: **end if**
6: Use normalization algorithm

Algorithm 2: Markowitz algorithm. For independent S_i, step 2 is trivial as $x_i = \frac{(p_i - 1/r)}{\sigma_{ii}}$. In step 4, μ_Y is maximized by using greedy allocation. The computational complexity of this approach is $O(n^3 + nlog(n))$ if S_i are dependent and $O(nlog(n))$ if they are independent.

5. EVALUATION

In an effort to better understand the problem we are attacking, and the benefits of the mathematical approach we have discussed, we evaluate different techniques in three simulation scenarios. The first scenario examines DTN routing over data MULEs, where paths are independent and data loss is Bernoulli due to message expiration when MULEs get delayed. We then move on to DTN routing over a set of city buses, where paths are multi-hop and dependent. Finally, we discuss a large sensor network scenario, where there are many paths and losses are partial in nature. Our main goal is to evaluate the performance differences between using simple replication and the erasure coding based Markowitz approach.

5.1 Performance Metric

The basic performance metric used is the failure probability (FP) for a message, defined as $1 - Prob(Y > r^{-1})$. For simulations in which multiple messages are transmitted, we consider the distribution of FP, also referred to as the failure rate (FR).

5.2 Allocation Techniques Considered

Simple Replication (SRep)
In this technique, identical copies of a message are sent over the r highest probability paths. No erasure coding is used. This approach effectively does a greedy allocation by considering only the best paths. Its complexity is $O(n)$.

All the following techniques use erasure coding and differ in how they allocate code blocks to different paths.

Simple Replication with Coding (SRep-Code)
Here, an equal proportion of erasure code blocks are sent over the r highest probability paths. Normalization algorithm is used to enforce volume constraints. This has the same performance as SRep when the path failure model is Bernoulli and the contact volume is sufficient for an entire message.

Proportional (Prop)
This is a simple heuristic in which allocation is proportional to the path probability $\left(x_i = \frac{p_i}{\sum p_i}\right)$, along with the normalization algorithm to enforce volume constraints. Its complexity is $O(n)$.

Markowitz Numeric (MkwNu)
This is the numeric method for maximizing the Sharpe-Ratio, as discussed in Section 4.2. It requires solving a series of *quadratic optimization problems with $O(n)$ constraints*.

Markowitz (Mkw)
This is the efficient approximation for maximizing the Sharpe-Ratio, as discussed in Section 4.2. The computational complexity of this approach is $O(nlog(n))$ if S_i are independent and $O(n^3 + nlog(n))$ otherwise. Although the Markowitz approach is derived under a Gaussian path failure model, it is applicable as an approximation for other failure models (see Section 4.4).

Mixed Integer Programming (MIP)
MIP is the optimal technique when the failure model is Bernoulli (see Section 3.2) and acts as yardstick for Bernoulli failures; unfortunately, it has exponential complexity. Our current implementation uses CPLEX and and we were able to solve for problem sizes as large as 16.

6. DATA MULE SCENARIO

In our first evaluation scenario, we consider the case in which mobile entities (called MULEs) are used to "carry" data from the source to the destination [9, 18, 25]. For example, in a sparse sensor network, Data-MULEs roam in the environment, retrieving data from the sensor(s) when they are physically close to them (via limited-range radios), and then transporting the data to an access-point, potentially located far away from the source.

For this experiment, we assume that MULEs do not forward messages to each other, and therefore, different MULEs act as independent forwarding nodes from the source to the destination. A message may expire if a MULE takes a long time to reach the destination. If the MULE mobility is random, it is hard to predict whether a MULE will deliver the data on time. This unpredictability makes the problem of selecting the best forwarding MULE(s) hard. By replicating data on multiple independent MULEs, the probability is increased that at least one MULE will reach the destination in time. In this scenario, message expiration due to delayed MULEs is the only cause of data loss. In other simulation scenarios we investigate the implications of multiple hops and other sources of data loss.

In this scenario, the path failure model is Bernoulli: either the MULE arrives on time at the destination and delivers all the data or the message expires and no data is transmitted. The probability that the i^{th} path is successful is $p_i = Prob(D_i \leq T)$, where D_i is the delay distribution of using the i^{th} MULE and T is the message expiration time. The delay distribution D_i is highly dependent on the environmental parameters, such as the topology, MULE velocity, radio range, mobility model, etc. In our simulations, each node records a contact history for each MULE along with the last time that MULE contacted the destination. This history is used to estimate the delay distribution.

p	Algorithm	# of MULEs (→) (n) 4	8	16	32	64
.41	SRep	36%	35%	37%	36%	36%
	Prop	48%	58%	70%	82%	88%
	Mkw	36%	35%	37%	36%	36%
	MIP	36%	35%	37%	–	–
.61	SRep	15%	15%	15%	15%	15%
	Mkw, Prop	19%	17%	11%	3%	1%
	MIP	15%	15%	11%	–	–
.86	SRep	2%	2%	2%	2%	2%
	Mkw, Prop	1%	1%	0%	0%	0%
	MIP	1%	1%	0%	–	–

Table 1: Failure rates with different MULE densities and success probabilities. $r = 2$ in all cases. When $p \geq .5$ both Proportional and Markowitz divide the code blocks equally among all MULEs and hence, are shown together.

Simulation Setup and Parameters
For the purposes of this section we use the following mobility model. We assume a planar environment with a dimension of 1km x 1km, with a single source and a single destination at opposite corners of the environment. Messages are generated once per hour, the message size is 10KB, the contact bandwidth is 100Kbps, and the storage capacity for MULE is 1MB. For these default settings, the message can be transmitted completely during a single contact with MULE. The effect of splitting due to small contact volume (relative to the message size) is considered in Section 6.2.

Each MULE follows a *random waypoint* mobility model, and unless otherwise specified, the MULE velocity is 10m/s (36 km/h). The radio range is 25m, and contact occurs whenever a MULE moves within range of a node. With these settings, the average value of delay is approximately 120 minutes. With this setup, we can create different delivery probabilities p (given by $Prob(D \leq T)$) by varying the relationship between the data deadline T (default is two hours) and/or changing the MULE velocities.

In the next few sections we evaluate and compare different replication techniques for various aspects of this scenario.

6.1 MULE Density

We now demonstrate the effect of the MULE density on the failure probability (FP) by varying the number of MULEs in the environment. All MULEs have the default velocity (10m/s), so they have approximately the same probability of success for a given deadline T. Table 1 shows the FP for three scenarios with deadlines of 80, 120, and 200 minutes (corresponding to $p = 0.41, 0.61, 0.86$, respectively). We keep r fixed at two, so these choices illustrate three "regimes" based on the product $p \cdot r$. Results for Markowitz-Numeric and SRep-Code are omitted because they have the same performance as Markowitz and SRep, respectively.

The first regime occurs when $p = .41$, hence $p \cdot r$ is less than

Algorithm	# of Fragments per Message (→) 1	2	4	8	16
SRep	15%	28%	48%	72%	92%
SRep-Code	15%	16%	15%	11%	3%
Mkw	3%	3%	3%	3%	3%

Table 2: Failure rates as more fragments are required for each message. $p = .61$, $r = 2$, and 32 MULEs. While Markowitz is robust to forced fragmentation, SRep degrades quickly. SRep-Code shows that SRep is greatly improved by using erasure coding when fragmentation occurs.

Algorithm	Number of slow MULEs (→) 0	4	8	10	12	14	16
MIP	0.4%	0.8%	2%	3%	4%	6%	52%
Mkw	0.4%	0.8%	2%	3%	4%	6%	52%
SRep	6%	6%	6%	6%	6%	6%	52%
Prop	0.4%	3%	10%	18%	35%	63%	95%

Table 3: Failure rates with 16 MULEs of two types: fast MULEs ($p = .76$) and slow MULEs ($p = .28$), and fixed $r = 2$. Prop (proportional) is unable to adapt to variations in MULE types, whereas, Markowitz maintains good performance until all MULEs are slow.

1. In this regime, splitting is harmful. The optimal allocation is thus to send a full copy of the message on the best two MULEs. This is the approach of SRep, and both the MIP and Markowitz identify this optimal allocation. However, Prop (proportional allocation) splits the data equally among all the MULEs and, as expected, has poor performance which gets worse as the number of MULEs increase.

The second regime corresponds to the case when $p \cdot r$ is only slightly greater than 1 (the "gray zone" when $p = .61$). In this case, when the number of paths is small (≤ 16), the splitting-based techniques (Markowitz, Prop, MIP) have the same or slightly worse performance than SRep. However, as n increases, the benefits of increased splitting become evident, and Markowitz has close to zero failure probability when n is 64. Finally, MIP has the same allocation as SRep when there are only a few paths, but employs splitting as the number of MULEs increase. However, even with only eight MULEs, the absolute difference between Markowitz and MIP is small.

The third regime is the case when $p \cdot r$ is significantly greater than 1 ($p = .86$). In this case, all techniques achieve good performance, which is intuitive because all MULEs are generally "good bets". Furthermore, with a large number of MULEs, the techniques that use erasure coding can achieve a near-zero failure probability. In contrast, SRep will require a very large replication factor to achieve such low failure probabilities. This could be important in situations when very high assurance of delivery is required.

6.2 Forced Splitting

In this section, we scale the message size to demonstrate the effects of being forced to split a message to fit onto a contact. SRep handles this constraint by splitting the message into the largest fragments that can fit over a single contact and then choosing the best paths for each fragment sequentially. Note that Markowitz, MIP, and Prop already account for volume constraints.

Table 2 shows the results of these experiments. We first observe that the performance of SRep drops sharply as the message size increases. This is because, in SRep, at least one copy of every *distinct* fragment must be received to successfully reconstruct the message. The probability of receiving all the fragments decreases exponentially as the number of fragments increase. The larger the message size, the the larger the number of fragments, which explains the decrease in FR as overall message size increases.

The performance of Markowitz is not affected by forced splitting. This is because for $p = .61$, Markowitz spreads the code blocks equally over the paths, and thus obeys the contact volume constraint.

Finally, when splitting is required, SRep is greatly enhanced by erasure coding (SRep-Code). Interestingly, the performance here improves with a larger message size, simply because the contact volume constraint forces more splitting and SRep-Code approaches the equal-split allocation that is done by Markowitz. This shows that if fragmentation is required, erasure coding is almost essential for good performance.

6.3 Fast and Slow MULEs

We now consider the case where paths have different success probabilities. To construct such a scenario, we introduce two kinds of MULEs: "fast" MULEs with a velocity of 10m/s and "slow" MULEs at 5m/s. Recall that the only way data is lost in this experiment is by message expiration. Thus, slower MULEs will tend to lose data more often than faster ones. We fix the total number of MULEs at 16, the replication factor at 2, and the deadline at two hours. These settings correspond to $p = .76$ for the fast MULEs and $p = .28$ for the slow ones. We vary the slow MULEs to fast MULEs to understand how different techniques adapt to a heterogeneous set of MULEs.

As shown in Table 3, SRep has a relatively constant failure probability of 6% as long as there are at least two fast MULEs, but jumps to 52% when all the MULEs are slow. In contrast, Prop is incapable of adapting to the mixture of path probabilities, and demonstrates notably worse performance when more than half the MULEs are slow. Finally, we see that the optimal MIP solution and Markowitz take advantage of as many fast MULEs as possible, and therefore obtain a much better FP when there are many fast MULEs. As the number of slow MULEs increases, the FP gradually degrades to approach that of simple replication, until there are no more fast MULEs, when the FP similarly jumps to 52%. We also considered other mixtures of path success probabilities, and find that Markowitz performs well in all cases.

6.4 Tolerance to Probability Estimation Errors

In this experiment, we discuss the impact of imperfect knowledge of the path delivery probabilities on performance of the various allocation techniques. This is an important consideration for real-world deployment of DTN systems, because path probabilities will generally need to be estimated and will not be completely accurate. We estimate path success probabilities by using a history of observations for each MULE; naturally, the accuracy of these predictions depends on the number of observations. In this experiment, we vary the number of samples used to estimate path probabilities and compare the reliable delivery performance. We divide 16 MULEs equally into four types, with velocities of 15, 12.5, 10, and 7.5m/s, respectively (corresponding to $p = .81$, .73, .61, and .47, respectively). We then examine five configurations (10, 20, 50, 100, and 1000) of the number of samples used to make the probability estimate. For each configuration, we ran 100 simulations with different

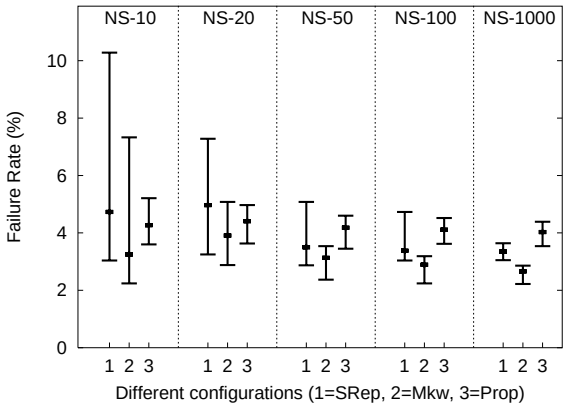


Figure 4: Failure rate distribution as the number of samples (denoted by NS-#) is varied, with 16 MULEs divided equally with varied speeds (p = .81, .73, .61, and .47), and $r = 2$. For each technique, we show the median, 5^{th}, and 95^{th} percentiles.

random seeds and therefore potentially different estimates of the underlying probabilities.

Figure 4 shows failure rates for SRep, Markowitz, and Prop for different number of samples. For each technique, the line indicates the median failure probability along with the 5^{th} percentile and the 95^{th} percentile (of the results from the 100 runs) to show the variability. The MIP had similar behavior as Markowitz and is omitted.

When only a few samples are used (NS-10 case), it is likely that some slow MULEs will be mistakenly identified as fast ones and vice-versa. Therefore, techniques such as SRep and Markowitz that aggressively use path probabilities to guide decisions will be adversely affected by errors in probability estimation; this is clear from high variability in the failure rates.

Having more samples leads to a more accurate estimate of the probabilities and improved performance for both SRep and Markowitz. Markowitz converges faster than SRep. This is because SRep will choose only two MULEs that it thinks have the highest probability. If only a few samples are used to estimate probabilities, it may choose the wrong MULEs. Markowitz, on the other hand, will split code blocks among a larger number of MULEs; it needs only a rough idea of which MULEs are fast and which are slow. Somewhat surprisingly, Prop is only marginally affected by the number of samples, and reveals good performance even with large estimation error. This is because it spreads data over all paths. Even with a large estimation error, it will send data on both the high and low probability paths, and therefore achieve reasonable performance. In fact, for this case, even a naïve technique that allocates equal code blocks among all paths (ignoring path probabilities) has performance similar to Prop.

7. BUS NETWORK SCENARIO

Now we consider the San Francisco city bus network scenario, as presented in previous work [8]. In this scenario, 20 city buses are equipped with radio transceivers to move data. Although the buses act as MULEs, this scenario has significantly different properties from the previous MULE scenario. Here, path failures are not independent because a given destination is serviced by a small number of bus routes. Therefore, it is possible that multiple paths to that destination use the same bus contact, introducing dependency between the paths.

Prior work assumed that buses always deliver messages with probability one [8]. We extend that scenario by adding a failure model in which all data transmissions during a contact may fail with a fixed probability. Such failures might occur because of channel contention, large unpredictable delays in bus movements (resulting in timeouts), or buffer overflows. Although in many ways similar to the MULE scenario, this case differs in four important ways:

- Buses follow published routes and this can be used to determine an efficient path. Unlike the MULE scenario, such a delivery path may be multi-hop, complicating the scenario.
- Paths or buses may suffer complete outages, resulting in all messages held in the failed bus to be discarded, irrespective of their timeouts. In the previous scenario, messages were discarded only on expiration.
- Paths are dependent and this requires a solution approach which can incorporate this dependency among paths.
- The solution space for path selection is significantly richer. That is, messages with different sources, sinks, departure times, and expiration times may be assigned completely different paths and replication strategies. In the previous scenario, delivery of all messages was constrained to a homogeneous set of MULEs.

Simulation Setup

The bus mobility is assumed to follow the published city schedule. We assume a radio bandwidth of 400kbps and a radio range of 100 meters. Each simulation examines a 12 hour duration. For each hour, 20 messages are sent between a random pair of buses at a random start time within the hour. The message size is 10KB, each bus has 1MB of storage, and the message expiration time was set to six hours. Contact success probability was chosen between 0.8 and 1.0, and made available to the allocation technique. The average *path* delivery probability, however, is .643 because paths are multi-hop. The path failure model is still Bernoulli. To apply Markowitz we need to compute the covariance matrix (V). We compute V_{ij} ($Cov(S_i, S_j)$) as follows.

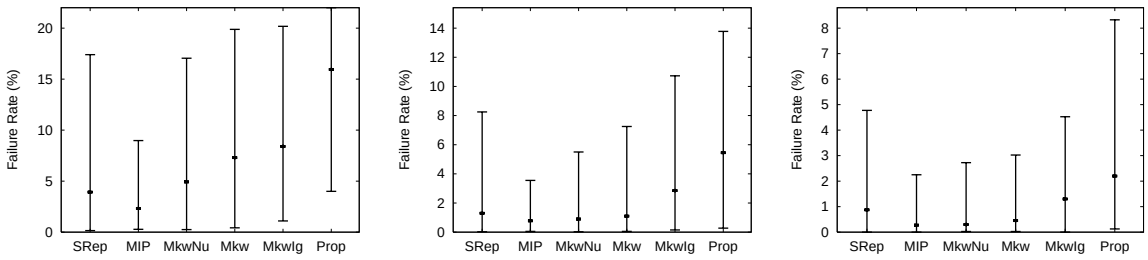
Consider two paths i and j and let C_q denote the set of contacts common between the two paths. Let C_i and C_j be the set of contacts that exclusively belong to path i and j respectively. Now, $V_{ij} = Cov(S_i, S_j) = E[S_i S_j] - E[S_i]\ E[S_j]$
$= Prob(S_i = 1)\ (Prob(S_j = 1|S_i = 1) - Prob(S_j = 1))$
$= \prod_{l \in C_i \cup C_q} p_l\ (\prod_{l' \in C_j} p_{l'} - \prod_{l' \in C_j \cup C_q} p_{l'})$
$= \prod_{l \in C_i \cup C_q \cup C_j} p_l\ (1 - \prod_{l' \in C_q} p_{l'})$

The dynamic Dijkstra algorithm (ED) presented in [8] is used to find multiple path(s) between two buses within a given deadline. Taking all messages into account, the number of paths for a message to be delivered ranges from 16 to 60, with a median of 34. For most messages, there are a few high probability paths and many more (longer) ones of low probability. Furthermore, many of these paths contain common buses, so they have dependent delivery probabilities. The probability dependence can be summarized in terms of the average correlation coefficient ($\bar{\rho}$), which is 0.2. Though it is not immediately obvious how this statistic corresponds to probability of delivery, we note that a value of $\bar{\rho} = 0$ would imply independent paths, and $\bar{\rho} = 1$ implies completely dependent paths.[6]

7.1 Multi-hop and dependent paths

Figure 5 shows failure rates for three different replication factors of 2, 3 and 4. Each line shows the 5^{th}, 50^{th} and the 95^{th} percentiles of the the distribution of failure rates across 240

[6] $\bar{\rho} = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{\sigma_{ij}}{\sigma_i \sigma_j}$. $\sigma_{ij} = Cov(S_i, S_j), {\sigma_i}^2 = Var(S_i)$



(a) r = 2 (b) r = 3 (c) r = 4

Figure 5: Failure rate distributions for different techniques and three different replication factors. For each technique, the plot shows the median, 5^{th} and 95^{th} percentiles across 240 simulated messages. MkwIg denotes the Markowitz approach that ignores correlations. For clarity, the y-axis has a different range and scale for the different plots.

simulated messages. In all cases, the MIP technique produces the optimal solution with the tightest range.[7] Examining the less expensive techniques, when $r = 2$, SRep performs slightly better than Markowitz. The average path success probability in this scenario is around .6, and so the product $p \cdot r$ is in the "gray zone". The effectiveness of splitting over multiple paths is further reduced because of dependencies among paths.

As we increase r, Markowitz shows a faster improvement than SRep. The difference between Markowitz and SRep when comparing the 95^{th} percentile FR is substantial. For example, when $r = 4$, Markowitz has almost half the failure rate as SRep. The most surprising revelation, however, is that even for higher replication factors, when comparing the median failure rates, Markowitz is only marginally better than SRep. Due to the dependencies between paths and the limited number of viable path alternatives, there is not much room for Markowitz to improve in many cases. At the same time, for some messages, paths are independent and Markowitz utilizes the path diversity to achieve better performance.

While Markowitz-Numeric and Markowitz have different performance, the differences are not generally significant. To understand the importance of accounting for correlations, we consider a modified version of Markowitz (labeled MkwIg) that ignores any correlations between paths, and note that its performance is notably degraded. This highlights the importance of systematically incorporating correlations. As a final note, we see that Proportional performs worse in all cases.

In summary, the benefits of using erasure coding are limited because of dependencies among paths. In many cases, simple replication will likely be sufficient, and the complexity of the erasure coding based techniques may not be justified. However, if very high assurance is required, using erasure coding techniques allows us to increase the "nines" of delivery success (i.e., go from a success probability of .9 to one of .99).

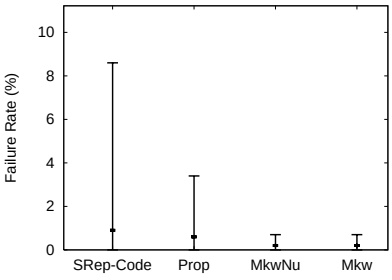
8. SENSOR NETWORK SCENARIO

In this section, we briefly discuss some early results obtained by applying our ideas to address the reliable communication problem in a multi-hop dense sensor network scenario. This scenario is different from the previous two cases as it represents a case of partial failures. We consider a loss model in which packet loss occurs because of poor channel quality. Poor channel quality may be the result of obstacles, interference, hidden terminal problems, multi-path effects, etc. Every link is characterized by a success probability, which is the probability that a transmitted packet is received successfully at the receiver. If multiple code-blocks are sent over a contact opportunity, then the resultant behavior can be approximated by a binomial distribution [24]. This is different from the previous two scenarios where S_i were Bernoulli. We also assume that the sensors sleep periodically with a low duty cycle to save energy [19]. This causes frequent disconnection in the network and it represents an instance of a DTN where no immediate feedback may be possible.

[7]For computation tractability MIP uses only the best 16 paths.

Simulation Setup and Parameters

Our simulation topology is borrowed from a real 97 node deployment of a sensor network in an office lab environment [15]. The sensor nodes are placed in a 40x16 foot grid with a grid size of 8ft. We use prior results to characterize packet loss based on distance [24]. The basic routing algorithm used was a minimum-cost algorithm. Edge costs were defined based on the distance between the endpoints of edges. The algorithm was extended to find the k best edge-disjoint paths. We found that the average number of paths from a node to the base-station was 7, and the path loss rates ranged from 0.4 to 0.6. This is consistent with an ongoing study to measure path loss rates for that network [15]. r was chosen to be two.



Technique	Avg FR
SRep	95%
SRep-Code	2.88%
Prop	2.738%
MkwNu	1.71%
Mkw	1.70%

Figure 6: Failure rates in the sensor network scenario. The path success probabilities range from 0.4 to 0.6. $r = 2$. For each technique, we show the median, 5^{th} and 95^{th} percentiles failure rate across 50 messages. The average failure rates are listed in the table and show that SRep has a significantly high failure rate.

Results

The mean failure rates for different techniques and the percentiles (5^{th}, 50^{th} and the 95^{th}) are shown in Figure 6. As expected, SRep has significantly higher failure rates (95%). This is natural because we only receive fragments of a message over the two paths, and since erasure coding is not used, at least one copy of each fragment is needed to be able to reconstruct the

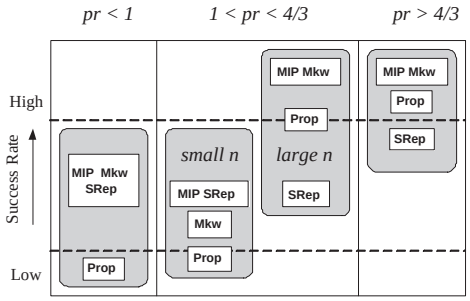


Figure 7: Qualitative performance of different techniques in three regimes. The first region corresponds to a low product pr, in which the use of more paths is detrimental, and simple replication performs well. In the second regime, the "gray zone," the benefits of erasure coding are only evident when many paths are used. In the third regime, all techniques have high success rate since all paths are good.

message. This is similar to the situation in the MULE scenario where splitting occurs (Section 6.2). We again use SRep-Code to understand the effect of a greedy allocation technique which works by sending code blocks on the best r paths. SRep-Code has a significantly better performance with a failure rate of only 2.88%. The Markowitz approach again gives additional benefits, more evident in terms of higher percentiles. For example, the 80^{th} percentile (i.e., for 20% of the messages) failure probability for Markowitz was less than two percent and more than six percent for SRep-Code.

9. SUMMARY OF RESULTS

The three scenarios are interesting because they exhibit a range of network scales and path failure models. The MULE scenario considers single-hop, uncorrelated delivery paths without volume constraints. The SFO city bus scenario uses multi-hop, correlated paths with volume constraints. Finally, the sensor network scenario has a large number of disjoint routing paths available, allowing us to explore partial path failures.

One consistent theme throughout our results is the existence of three operating regimes (based on the concept of the product $p \cdot r$) that profoundly affect the best choice. Figure 7 shows the classification of the efficacy of the different techniques in these different regimes.

The first question we addressed is to determine the benefits of erasure coding in the various regimes. When paths have lower probability (regime 1), erasure coding does not provide additional benefits over simple replication, regardless of the number of available paths. In the second regime, the situation is slightly more complex, as the benefits of erasure coding are only evident when there are many paths on which to split code blocks. Finally, in the third regime, while all techniques achieve high success probabilities, the erasure coding based techniques can approach a near-perfect success rate.

An encouraging result is that our Markowitz technique is able to handle all three regimes and performs close to optimal in most cases. Using the Sharpe-Ratio as an alternate objective function turned out to be efficient and effective in capturing complex aspects such as path dependencies. While simple heuristics like the proportional approach may do well in certain situations, in others they have notably worse performance. Finally, for the case of partial path failures and forced splitting due to limited contact volume, the use of erasure coding is a clear win.

10. RELATED WORK

Use of replication to improve reliability in a DTN context has been suggested in recent work [8, 9]. To our knowledge, this work is the first attempt to systematically examine how replication and erasure coding can be used to combat uncertainties, lack of information, and failures in DTN scenarios. We now discuss various related fields from which we draw our techniques.

Portfolio Theory

Modern portfolio theory has a vast literature and an extensive survey is not possible here. The theory of efficient frontiers and utility maximization was developed by Henry Markowitz in the late 1950s. The concept of the Sharpe-Ratio was proposed by William Sharpe, in the context of stock-market equilibrium and optimal investment when a bank provides a risk free rate r. Markowitz and Sharpe (along with Merton Miller) won the Nobel Prize for Economics in 1990 for this work.

To our knowledge, numerical search is the only method for maximizing the Sharpe-Ratio if there are volume constraints. However, unconstrained optimization is covered in detail in several sources [1, 13, 6]. Furthermore, the existence of a unique maximum on the efficient frontier and the various approaches for finding it using numerical methods are also covered in prior work [13, 20].

Optimizing the utility function $Prob(Y > c)$ for general distributions is not well studied due to the complex nature of the distribution of Y. Most of the literature focuses on concave utility functions expressed in terms of the mean and the variance of the return Y [1, 6]. Some approaches discuss the minimization of the *shortfall probability* $Prob(Y < c)$ [6]. However, this work also assumes that Y has a Gaussian distribution.

Waterfilling in Gaussian channels

A problem related to ours arises in communication theory, where a fixed power budget is allocated to maximize the total capacity of a set of parallel Gaussian noise channels. The optimal amount of power allocated to each channel is roughly proportional to the difference between the computed *water-level* and the noise level of the channel [4].

Recent work in multiple antenna channels looks at similar problems where the objective is to find the capacity of multi-hop wireless ad-hoc networks [2]. Here, the formulation focuses on optimizing the expected value of a linear combination of random variables; therefore convex optimization techniques can be used. These approaches cannot be used in our case because the objective is a step function and therefore neither concave nor convex.

FEC, Erasure Coding, Internet Routing

Several different erasure coding techniques such as Reed-Solomon codes and Tornado codes [14] have been developed, and the various tradeoffs are described elsewhere [12]. The different coding techniques differ primarily in their computational complexity and bandwidth overhead. For this paper, the choice of exact erasure coding algorithm is not important.

There are several well known cases in which erasure codes are used to cope with packet transmission failures [3, 12, 10]. Much of this work is in the context of multicast video or data transmission, and therefore generally sends a continuous sequence of code blocks over a single path. Furthermore, in these contexts, the communication path is assumed to be always available with relatively constant capacity. For these reasons, the problem of choosing the best allocation among a set of paths with different reliability does not appear in these settings.

Combinatorial Optimization

The scenarios in which path failures are modeled as Bernoulli is at heart a combinatorial optimization problem. However, to our knowledge the problem has not been investigated in detail. The problem of computing $Prob(Y > c)$ is discussed in literature dealing with reliability of $k-$out-of$-n$ systems [11], where systems fail if more than k components fail. The work focuses on computing $Prob(Y > c)$ for a given configuration and not on optimizing $Prob(Y > c)$. We believe that the problem raised here presents an interesting avenue for future work.

11. CONCLUSIONS AND FUTURE DIRECTIONS

In this paper, we investigate the problem of improving the probability of successful message delivery in DTNs with path failures by applying a combination of erasure coding and replication. Although erasure coding is a well-known technique to address failures in traditional networks, the notions of path failures and volume constraints in DTNs significantly alter the best solution approaches. With attention paid to these differences, we formulate the optimal allocation problem of erasure code blocks over multiple paths. Solving this problem, even for the simple case of Bernoulli failures, turns out to be a challenging task. To do so, we used a *mixed integer program*. Unfortunately, it has too high a computational complexity to be practically useful, but it does serve as a yardstick to evaluate other methods.

Our primary solution technique relies on ideas from *modern portfolio theory*. This allows us to optimally solve the problem for a Gaussian failure model and gives us good approximations for other cases. In spite of a complex derivation, the final algorithm is simple and elegant. It is quite general and accounts for path dependencies and volume constraints. Through simulations of various DTN scenarios, we demonstrate that this approach offers significant benefits over simple heuristics such as greedy and proportional allocation and works well in most scenarios. Interestingly, a similar problem arises in two other contexts; replica placement in wide-area storage systems and virtual node mapping in distributed hash tables (DHTs). The ideas presented here should provide useful insights for these problems as well.

This paper is only a first step in understanding the reliability issues in DTN. Our formulation assumes that the underlying probabilities remain constant over time. Violations of this assumption present additional challenges we have not addressed here, and some form of adaptation might be required to handle dynamic conditions.

In investigating the combinatorial version (Bernoulli case) of our problem, we were unable to compute optimal solutions at any large scale. Efficient approximation algorithms (e.g. using Chernoff bounds) present an encouraging avenue of exploration for this and related problems. Another interesting direction would be to alter the optimization criteria. One possibility would be to to optimize the replication factor given constraints on delivery probabilities. The analysis included here provides hints along these lines, but this direction is still wide open.

Finally, we have some experience with the impact of erasure coding on the *delay distribution* of message delivery in opportunistic DTN networks, which we discuss separately in [22]. This is another potentially rich direction for future work.

Acknowledgments

We are grateful to Gaetano Borriello, David Wetherall and the SIGCOMM reviewers for providing helpful feedback on the paper. Haifeng Yu, Henry Lin, Christos Papadimitriou and Brighten Godfrey provided input regarding the hardness of the allocation problem. Thanks are also due to Harsha Madhyastha, Michael Rosenblum, Sergiu Nedevschi, Rahul Shah, Aditya Mohan, Bowei Du and Melissa Ho for providing feedback on a draft version of this paper.

12. REFERENCES

[1] G. J. Alexander and J. C. Francis. *Portfolio Analysis*. Prentice Hall, 1986.

[2] H. Boche and E. A. Jorswieck. Outage Probability of Multiple Antenna Systems: Optimal Transmission and Impact of Correlation. In *IEEE International Zurich Seminar (IZS)*, 2004.

[3] J. W. Byers, M. Luby, and M. Mitzenmacher. A Digital Fountain Approach to Asynchronous Reliable Multicast. *IEEE J-SAC, Special Issue on Network Support for Multicast Communication*, 20(8), 2002.

[4] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley Series in Telecommunications, 1991.

[5] CPLEX: Linear Programming Solver. `http://www.ilog.com/`.

[6] M. Engles. Portfolio Optimization: Beyond Markowitz. Master's thesis, Leiden University, 2004.

[7] K. Fall. A Delay-Tolerant Network Architecture for Challenged Internets. In *ACM SIGCOMM*, 2003.

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

[9] P. Juang, H. Oki, Y. Wang, M. Margaret, P. Li-Shiuan, and R. Daniel. Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet. In *ASPLOS-X*, 2002.

[10] S. Kim, R. Fonseca, and D. Culler. Reliable Transfer on Wireless Sensor Networks. In *SECON*, 2004.

[11] W. Kuo and M. J. Zuo. *Optimal Reliability Modeling: Principles and Applications*. Wiley, 2002.

[12] M. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman. Efficient Erasure Correcting Codes. In *IEEE Transactions on Information Theory*, 2001.

[13] D. Maillard. Some Remarkable Spots on the Efficient Frontier. *Conservatorie National des Arts et Metiers*, 2004.

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

[15] A. Mohan, W. Hong, D. Gay, P. Buonadonna, T. Doeppner, and A. Mainwaring. End-to-End Performance Characterization of Sensornet Multihop Routing. In *IEEE ICPS*, 2005.

[16] R. Rodrigues and B. Liskov. High Availability in DHTs: Erasure Coding vs. Replication. *IPTPS*, 2005.

[17] S. Jain et al. Additional Proofs and Discussion Related to the Optimal use of Redundancy to Cope with Failures in a Delay Tolerant Network. Technical Report 2005-06-04, University of Washington, 2005.

[18] R. Shah, S. Roy, S. Jain, and W. Brunette. Data MULEs: Modeling a Three-tier Architecture for Sparse Sensor Networks. In *IEEE SNPA*, 2003.

[19] R. Shah, S. Wietholter, A. Wolisz, and J. Rabaey. Modeling and Analysis of Opportunistic Routing in Low Traffic Scenarios. In *IEEE WiOpt*, 2005.

[20] R. H. Tutuncu. Optimization in Finance. *Advance Lecture on Mathematical Science and Information Science*, 2003.

[21] A. Vahdat and D. Becker. Epidemic Routing for Partially-connected Ad hoc Networks. Technical Report CS-2000-06, Duke University, 2000.

[22] Y. Wang, S. Jain, M. Martonosi, and K. Fall. Erasure coding based routing in opportunistic networks. In *ACM SIGCOMM Workshop on Delay Tolerant Networking*, 2005.

[23] H. Weatherspoon and J. Kubiatowicz. Erasure Coding vs. Replication: A Quantitative Comparison. In *IPTPS*, 2002.

[24] A. Woo, T. Tong, and D. Culler. Taming the Underlying Challenges of Reliable Multhop Routing in Sensor Networks. In *SenSys*, 2003.

[25] W. Zhao, M. Ammar, and E. Zegura. A Message Ferrying Approach for Data Delivery in Sparse Mobile Ad Hoc Networks. In *ACM MobiHoc*, 2004.