# Reliable Transfer on Wireless Sensor Networks

Sukun Kim
binetude@cs.berkeley.edu

Rodrigo Fonseca
rfonseca@cs.berkeley.edu

## ABSTRACT

In Wireless Sensor Networks, many applications like structure monitoring require collecting all data without loss from motes. End-to-end retransmission which is used in Internet for reliable transport layer, does not work well in Wireless Sensor Networks, since wireless communication, and constrained resources give new challenges. We looked at factors affecting reliability, and searched possible options. Information redundancy like retransmission, erasure code, and thick path are candidates. However, if loss is not randomly distributed, those methods does not work well. For example, when link fails, but routing table is not updated, all packets through that path will be dropped. Route fix, which tries alternative next hop after some failure, reduces correlated consecutive drops, so that information redundancy can perform well.

Experiment on real testbed in Soda Hall shows that route fix with erasure code provides good reliability. And encoding and decoding of erasure code is efficient.

More investigation of data (overhead, delay) will give deeper insight in comparing options.

## Keywords

wireless sensor networks, reliable transfer, retransmission, erasure code, route fix

## 1. INTRODUCTION

There exist many applications which require all data without loss. For example, structure monitoring needs the entire data from all measuring points to build a model and analyze it. Moreover, data collection can be done over multi-hop network. Challenges to achieve reliability on Wireless Sensor Networks can be divided to three main categories.

Challenges in the first category are related to the problems coming from wireless communication. Asymmetry of link makes link quality estimation hard. Correlated losses (obstacles, interference) can lead to consecutive losses decreasing effectiveness of erasure code. Weak correlation between quality and distance, hidden terminal problems, and dynamic change of connectivity complicates the situation further.

The second sort of problems come from the constrained resources of Wireless Sensor Networks motes. A mote is battery powered, so has limited power source. And it has small computational power and memory space. Even for communication, its bandwidth is narrow. Therefore we can't play a complicated algorithm to achieve reliability. We can't send many control packets to tune network, nor can we run sophisticated algorithm in motes.

Diverse routing layers add more challenges. Since motes are constrained in resource, there are different routing layers customized for specific purpose. Even if we can run point-to-point routing for dissemination of information or collection of data, this approach is very inefficient. For collecting data (convergence routing), each node only need to keep which nodes are candidates for its parent. This reduces burden to keep additional information to support routing to any node. Dissemination of information like code image distribution is similar to multicast (divergence routing). In this case, we can benefit from broadcasting nature of wireless communication. By injecting one packet into channel, everyone around can hear the packet. Compared to sending packet to each single receiver, this can save huge effort. So there are three main routing layers categories: point-to-point routing, convergence routing, and divergence routing. One transport layer or one method may not work for all three cases well. But it is not a good idea to keep three separate versions of reliable transfer either. At least it will be desirable to share some components if possible, wherever it might be located in network stack.

In this paper, we will look at diverse options for improving reliability over multiple-hops, focusing mainly on point-to-point routing. First of all, it would be worthy seeing fundamental factors determining reliability. Then we will see possible options which improve each factor. Let us simplistically look at the following equation

$$\text{numberofpacketsreceived} = P_{success} \times \text{numberofpacketssent}$$

The goal is to increase (number of packets received) sufficiently so that we can get all data. Even though it is also important which packets are received as we will see later, the basic limitation is delivering a sufficient amount of packet. This in turn amounts to increasing either (number of packets sent) or increasing the probability to get through ($P_{success}$).

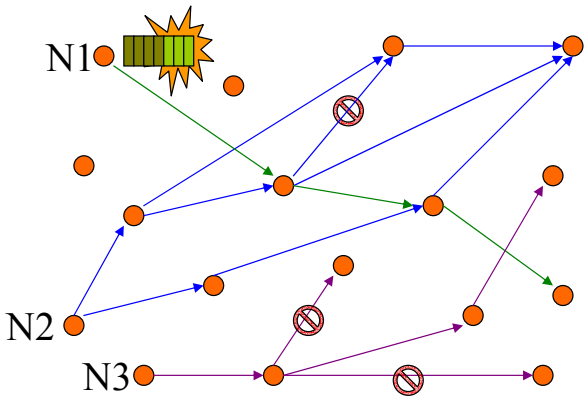


**Figure 1: Possible options to achieve reliability**

Increasing the number of packets sent can be interpreted as adding redundancy to information. One option is retransmission. End-to-end retransmission is used in TCP of Internet. Link-level retransmission is used in wireless communication where loss rate of link is high. Adding redundancy to data is also an option. Sending additional parity packet for some number of previous packets is a good example. Erasure code can be thought as a generalization of parity code. Rather than sending one additional packet, erasure code can send multiple additional packets. In case of parity packet, any $M$ out of $M+1$ packets will reconstruct original $M$ data. Likewise, erasure code enables reconstruction of $M$ original data only if $M$ out of $M+R$ packets are received. In the Figure 1, $N1$ is sending data with erasure code. We can also add redundancy to path. As $N2$ in Figure 1, thick path can be used. Every node within the path width (actually long rectangular area) will participate in transferring data.

Increasing the probability to deliver packet has a critical benefit. If $P_{success}$ is not randomly distributed, some techniques of information redundancy can not provide reliability efficiently. For example, erasure code can survive up to $R$ losses. Correlated consecutive $R+1$ or more losses make erasure code unable to decode. Congestion control is an approach used in Internet to prevent queue overflow, and to decrease delay. Adding alternative routes alleviates correlated losses coming from dynamic change in connectivity graph. Wireless link can be unusable any time, but it takes time to update routing table to incorporation this information. Then from link failure to routing table update, every packet through that link will be dropped. We can attenuate the problem by increasing the update frequency. However this means more control traffic. Or after some delivery failure, we can try alternative next hop. This is the case of $N3$ in Figure 1.

In this paper, we will look at link-level retransmission, end-to-end retransmission, erasure code, and alternative route. Other options remain as future work. We examined those options on real-world testbed. We will provide the result in Section 8. And then we will see which options and which combinations of options are good choices.

## 2. RELATED WORK

In Wireless Sensor Networks, there exist different types of routings. There is convergence routing [9], which can be used for collecting data. Deluge is for divergence routing [4], and program image dissemination is a good example of application. There also exists point-to-point routing with geographical information.

There exist diverse algorithms for erasure code which can be implemented in either software or hardware [8, 2]. [8] exploits diverse optimizations, from which this work gained many hints. Rateless code [3, 7] is a class of erasure code in which arbitrary number of code words can be produced. However, those works are not optimized for systems with low capability: not much attention was paid to cases with extreme space limitation. Work in this paper puts heavy weight on optimization for node with very limited resources.

## 3. LINK-LEVEL RETRANSMISSION

Loss rate on wireless link is much higher than that of wired link. And this effect accumulates quickly as the number of hops increases. For example, when loss rate is 10% per hop, after 15 hops loss rate becomes 80%! If a message is lost at nth hop, all previous n-1 transfers become waste. To deliver the packet to nth hop again, we need n-1 additional transfer, if all n-1 transfers succeed. If we try link-level retransmission, just one retransmission can bring packet to the same point. For efficient use of wireless channel, link-level retransmission is a very good choice.

There exists drawback in link-level retransmission, when used together with some particular technique. Delivery time depends on number of retransmission in the middle, and Round trip time (RTT) varies significantly. This situation makes end-to-end retransmission inefficient. Since we do not clearly know the RTT, an upper bound need be used. Then in case of a packet loss, the sender should hold its buffer for a longer time. Holding memory space for a long time is not desirable in resource-constrained Wireless Sensor Networks. And for link-level acknowledgement, there is 20% decrease in channel utilization. Another minor downside is that middle node needs to hold buffer until it receives acknowledgement from the next hop.

## 4. END-TO-END RETRANSMISSION

End-to-end retransmission is method used in Internet. This can guarantee eventual reliability regardless whatever happens in the middle. However, as pointed out in the previous subsection, this solution may not work well when combined link-level retransmission is also used. And there exists overhead for opening session and acknowledgement. For small data, control overhead becomes relatively large. Furthermore, in many situations end-to-end acknowledgments may not be practical, for example when the reverse path may not exist.

For our experiments we use an implementation of end-to-end retransmission, Large-scale Reliable Transfer (LRX) component. The following is overview of protocol in LRX, which is quoted from [6].

Large-scale Reliable Transfer (LRX) component assumes that data resides in RAM. Upper layer should handle non-volatile storage. LRX transfers one data cluster, which is composed of several blocks. One block fits into one packet, so the number of blocks is equal to window size. Each data cluster has a data description. After looking at data description,

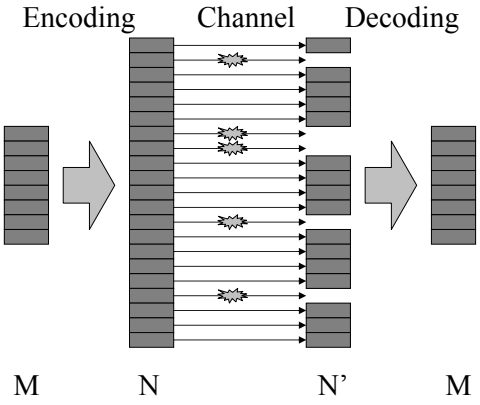


**Figure 2: Mechanism of Erasure Code**

receiver may deny data (receiver already has that data, or that data is not useful anymore).
Explicit open handshake is used. Data description and size of cluster is sent as a transfer request. If receiver has enough RAM, and application layer agrees on data description, then receiver sends acknowledgement for transfer request.
Once connection is established, actual data is transferred. Protocol at high level can be summarized as selective acknowledgement and retransmission. Data transfer is composed of multiple rounds. In each round, sender sends packets missing in the previous round. At the end of each round, receiver sends acknowledgement saying which packets are missing. Then sender, after looking at this acknowledgement, sends packet missing again. The first round can be thought of as a special case where every packet was missing in the previous (imaginary) round.
Tear-down is implicit. Successful tear-down for both sides cannot be guaranteed anyway, however close phase will introduce overhead, and delay. We favored quick movement to next connection, and eliminated close phase.

# 5. ERASURE CODE

Erasure code is code with which we can reconstruct $m$ original messages by receiving any $m$ out of $n$ code words ($n > m$). If $n$ is sufficiently large compared to the loss rate, we can achieve high reliability without retransmission. Figure 2 shows high level mechanism of erasure code. A particular erasure code algorithm called Reed-Solomon code is used. To explain Reed-Solomon code, linear code will be presented first, followed by Vandermonde matrix.

## 5.1 Linear Code

For encoding process, there exists encoding function $C(X)$ where $X$ is a vector of $m$ messages. Then $C(X)$ will produce a vector of $n$ code words ($n > m$). If code has a property that $C(X) + C(Y) = C(X + Y)$, then it is called a linear code. Linear code can be represented with a matrix $A$. code word vector for message vector $X$ is simply $AX$. Encoding is matrix-vector multiplication. Decoding is finding $X$ such that $AX = Z$ for a received code word vector $Z$. This is finding solution to linear equation $AX = Z$. We can see that $A$ should have $m$ linearly independent rows so that linear equation has unique solution, and in turn unique message vector.

$$\begin{pmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{m-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{m-1} \\ 1 & x_3 & x_3^2 & \cdots & x_3^{m-1} \\ \vdots & \vdots & \vdots & & \vdots \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_{n-1} & x_{n-1}^2 & \cdots & x_{n-1}^{m-1} \\ 1 & x_n & x_n^2 & \cdots & x_n^{m-1} \end{pmatrix}$$

**Figure 3: Vandermonde Matrix**

$$\begin{pmatrix} 1 & x_1 & \cdots & x_1^{m-1} \\ 1 & x_2 & \cdots & x_2^{m-1} \\ 1 & x_3 & \cdots & x_3^{m-1} \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ 1 & x_{n-1} & \cdots & x_{n-1}^{m-1} \\ 1 & x_n & \cdots & x_n^{m-1} \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_{m-1} \end{pmatrix} = \begin{pmatrix} p(x_1) \\ p(x_2) \\ p(x_3) \\ \vdots \\ \vdots \\ p(x_{n-1}) \\ p(x_n) \end{pmatrix}$$

**Figure 4: High level diagram showing how Reed-Solomon code works**

This code is very useful, since encoding and decoding are computationally inexpensive. This is especially attractive in resource-constrained Wireless Sensor Networks.

## 5.2 Vandermonde Matrix

There is one more thing we need to look at before Reed-Solomon code. Vandermonde matrix is a matrix with element $A(i, j) = x_i^{j-1}$ where each xi is nonzero and distinct from each other, as shown in Figure 3. For $n$ *by* $m$ Vandermonde matrix ($n > m$), any set of $m$ rows forms nonsingular matrix. That is to say, whatever set with m rows we may choose, rows in the set are linearly independent. Lets define this property as *Property V* for future use.

**Definition** (*Property V*): For a $n$ *by* $m$ ($n > m$) matrix $A$, if any set $S$ of $m$ rows of $A$ form nonsingular matrix such that all rows in $S$ are linearly independent, then $A$ is said to have *Property V*.

We can see that in linear equation $AX = Z$ where $A$ is Vandermonde matrix, any $m$ rows and corresponding $m$ elements of $Z$ form $m$ *by* $m$ square matrix and a vector of size $m$, where matrix is nonsingular. Then we can uniquely determine $X$. This sounds somewhat similar to erasure code. Next subsection will describe Reed-Solomon code, which uses the same idea.

## 5.3 Reed-Solomon Code

Basic idea of Reed-Solomon code is producing $n$ equations with $m$ unknown variables. Then with any $m$ out of $n$ equations, we can find those $m$ unknowns.
For a given data, let us break down it into $m$ messages $w_0, w_1, w_2, \ldots, w_{m-1}$. And construct $P(X)$ using these mes-

sages as coefficients such that

$$P(X) = \sum_{i=0}^{m-1} w_i x^i$$

And evaluate this polynomial $P(X)$ at $n$ different points $x_1, x_2, , x_n$. Then $P(x_1), P(x_2), , P(x_n)$ can be represented as multiplication of matrix and vector as shown in Figure 4. Here we can see that matrix $A$ is Vandermonde matrix, $W$ is a vector of messages, and code words are contained in a vector $AW$. If we have any $m$ rows of $A$ and their corresponding $P(X)$ values, we can obtain vector $W$ which contains coefficients of polynomial, which is again the original messages. Reed-Solomon code can be also used to correct error. However, in current implementation of TinyOS, each packet has CRC to detect bit error. We can assume that there will be no bit error in packet containing code word (otherwise, it must be dropped at lower layer). Therefore, error correction is not used in the implementation.

# 6. MODIFICATION OF ERASURE CODE FOR WIRELESS SENSOR NETWORKS

We need more process to bring erasure code to real world implementation, especially in resource-constrained Wireless Sensor Networks (WSN). Several methods are used to improve efficiency in mote.

## 6.1 Extension Field

For efficient use of bits in packet, we use extension field with base 2. First of all, we need to look at field, and prime field. We follow definition of mathworld [1]. Field is any set of elements which satisfies the field axioms for both addition and multiplication and is commutative division algebra. Field axioms include commutativity, associativity, ditributivity, identity, and inverse. An archaic name for a field is rational domain. The French term for a field is corps and the German word is Korper, both meaning "body."

A field with a finite number of members is known as a finite field or Galois field. For a given Galois field of size $q$, if $q-1$ powers of an element $x$ $(x^1, x^2, \ldots, x^{q-1})$ produce all non-zero elements, that element $x$ is called a generator of the given Galois field.

Prime field is a Galois field, whose elements are integers in $[0, p-1]$, where $p$ is prime. Addition and multiplication are normal integer addition and multiplication with modulo operation at the end. Prime field always have generator. The size of prime field is $p$, and we need $\lceil \log_2(p) \rceil$ bits to represent all elements. Since $p$ is not power of 2, there exists waste in bit usage. For example, to represent prime field with prime 11, we need 4 bits with which 16 numbers can be represented.

Extension field is a Galois field whose elements are integers in $[0, p^r - 1]$. Extension field can be though as polynomials on $primefield(p)$. Operations follow rules of polynomial operation with modulo operation at the end. Primitive polynomial is generator of extension field. Interestingly, this set with polynomial operations stated above, still satisfies properties of field. Moreover, by setting $p = 2$, we can fully utilize bits in message. *Property V* of Vandermonde matrix still holds for prime field, and even for extension field!

## 6.2 Systematic Code

$$\begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 \\ 1 & x_{m+1} & \cdots & x_{m+1}^{m-1} \\ 1 & x_{m+2} & \cdots & x_{m+2}^{m-1} \\ \vdots & \vdots & & \vdots \\ 1 & x_n & \cdots & x_n^{m-1} \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_{m-1} \end{pmatrix} = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_{m-1} \\ p(x_{m+1}) \\ p(x_{m+2}) \\ \vdots \\ p(x_n) \end{pmatrix}$$

**Figure 5: Systematic code**

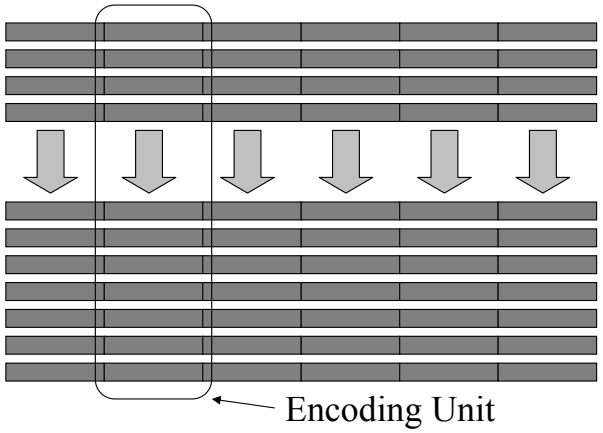


**Figure 6: Divide packet into multiple independent code words**

If some part of erasure codes are original messages, when every packet in this part arrives we can get original messages without decoding. Such code is called systematic code. Another good property of Vandermonde matrix is that even if any $m$ rows of $n$ *by* $m$ $(n > m)$ Vandermonde matrix are substituted with rows of $m$ *by* $m$ identity matrix, the new matrix still have *Property V*, even for extension field. Figure 5 shows one possible case.

When we use systematic code in this way, at the encoding side, we dont need any computation for the portion of code words containing original messages. This reduces encoding overhead. At the decoding side, when we have all code words containing original messages, we get messages without any further computation. In this case, we can achieve huge save in decoding computation (actually no computation is needed). Systematic code can give benefit even when we loose some packets. At the decoding side, the more original message part we have, the closer decoding matrix would be to identity matrix, and the quicker decoding process becomes. Therefore, if we have most of code words containing original messages, and some messages constructed by Vandermonde matrix, decoding will be very fast.

## 6.3 Multiple Independent Code Words in a Packet

If one packet carries one code word, each code word will be very large. This makes implementation intractable since op-

eration on such a large field requires huge space and time. One solution would be using small messages and small code words. Then, however, payload in a packet gets too small. By putting multiple independent code words into a packet, we can fully utilize payload space of a packet without problems of large code word.
Imagine dividing one big data into $t$ pieces of small data. Then each data is again divided into $m$ messages, and encoded into $n$ code words. We have total of $tn$ code words to send. Pack $i$th code words from each independent $k$ data into a single packet. We either get all $i$th code words for $k$ data, or we get nothing. Any $m$ packets will provide $m$ code words for all $k$ data, and we can reconstruct original $k$ data. Since all $k$ data have code words with same sequence set, decoding process is the same: same decoding matrix can be used. This further enhances decoding efficiency. Figure 6 shows example. Here data is divided into 6 small data chunks. Each data chunk is divided again to 4 messages. Messages from each chunk are encoded to 7 code words independently. Then code words from all data chunks with same sequence number are packed into the same packet.
The drawback of dividing packet into multiple code words is the limitation for the number of messages and code words. The number of messages can not exceed number of bits used to present message. The number of code words should be smaller than the size of extension field. For example, if each code word is 8 bit long, maximum number of messages is limited to 8, and maximum number of code words is limited to 255.

## 6.4 Operation Table

As describe before, extension field is used to efficiently use packet payload. Operations on extension field are not simply addition and multiplication combined with modulo operation. They are polynomial operations with modulo. Therefore, rather than performing complex computation on the fly, table is used to lookup result of operation. Addition is XOR of two numbers, and we dont need table. For multiplication and division operation, exponent and log values are computed and stored as tables.
Here we need to look at generator. Let the size of extension field be $q = p^r$, where $p$ is prime. Extension field has generators. Let one of them be $\alpha$. For any generator $\alpha$, when we keep multiplying $\alpha$, we can produce all $q-1$ non-zero elements. And then $\alpha$ is produced again starting cycle. That means

$$\alpha^q \ mod \ q = \alpha, \ \alpha^{q-1} \ mod \ q = 1$$

Let

$$x = \alpha^{k_x} \ mod \ q, \ y = \alpha^{k_y} \ mod \ q$$

Exponent and log are defined as follows

$$exp(k_x) = x, \ log(x) = k_x \ where \ x, k_x \in GF(p^r)$$

Then multiplication of $xy$ is

$$xy \ mod \ q = \alpha^{k_x}\alpha^{k_y} \ mod \ q = \alpha^{k_x+k_y} \ mod \ q$$

$$= \alpha^{k_x+k_y \ mod \ (q-1)} \ mod \ q = exp(k_x + k_y \ mod \ (q-1))$$

$$= exp(log(x) + log(y) \ mod \ (q-1))$$

Inverse of $x$ is

$$\frac{1}{x} = \alpha^{-k_x} = \alpha^{q-1-k_x} = exp(q-1-k_x)$$

$$= exp(q-1-log(x))$$

Therefore, multiplication involves two log table lookups, one addition, one modulo, and one exponent table lookup. Inverse involves one log table lookup, one subtraction, and one exponent table lookup.
These tables consume memory space. However, frequent use of multiplication operation justifies usage of table. And currently computation on the fly takes $O(size\ of\ extension\ field)$ time which grow exponentially with the size of code word. As we will see in the next section, the memory usage is moderate when the size of message and code word is small.

# 7. ALTERNATIVE ROUTE

Adding an alternative route in the case of the failure of a given link is yet another way to increase reliability. When a link between two nodes fails, the messages sent by the first to the second will successively be dropped, until the link estimation component is triggered and selects a new route. This process, if prevalent, can eliminate the benefits obtainable from erasure coding, since many losses on the same encoded messages will very likely be above the redundancy added in the coding process. In this case, it should be clear that link-level retransmissions are of no great help, unless used to an prohibitively long extent, because no messages will get through. A sensible strategy, then, is to detect the failure as soon as possible, and send the packet to an alternative route, if possible.

This points to the need of special support from the routing layer for enabling alternative paths towards the destination. This flexibility ultimately depends on the routing geometry of the routing algorithm, to paraphrase the work done for Peer-to-Peer DHT algorithms in [5]. For example, in the case of aggregation, in which nodes route to a parent in the tree to the root, there may be many nodes within communication range that decrease the distance to the root. In geographic routing, there may also be more than one neighbor that allows progress towards the destination. In our evaluation we use an implementation of Beacon Vector Routing (BVR). We describe the algorithm in some detail in Section 8, but for now it suffices to say that it allows flexibility in the selection of routes.

# 8. EVALUATION

We thought of three metrics: success rate, overhead, and delay. Success rate is percentage of packets which arrived at final destination. Overhead is the number of packets injected to network to deliver one packet from source to destination. Overhead includes both loss rate, and average number of hops from source to destination. Since some options may take more reliable path even though it could be longer, overhead is more meaningful than packet loss rate. But this makes hard to compare results from two different pairs of nodes. Delay is time during which a packet travels from source to destination. In case of end-to-end retransmission, it is from opening connection to tearing down connection, since sender should preserve buffer space during that amount of time. For erasure code, it is time until receiver gets sufficient number of packet to decode data, or times out. This is reasonable since receiver should keep buffer for during that period.

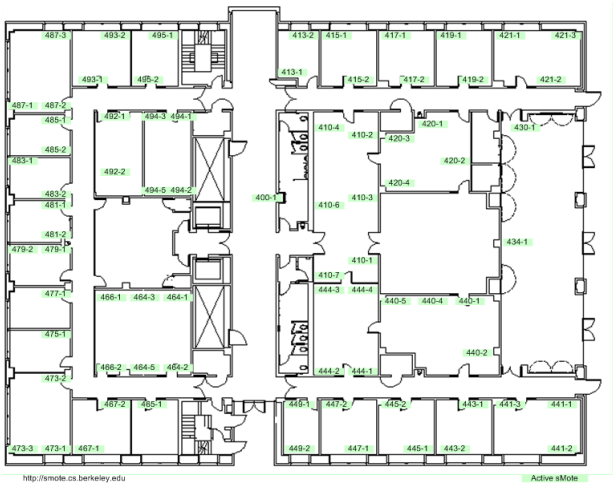

**Figure 7: Floor plan of testbed in Soda Hall**

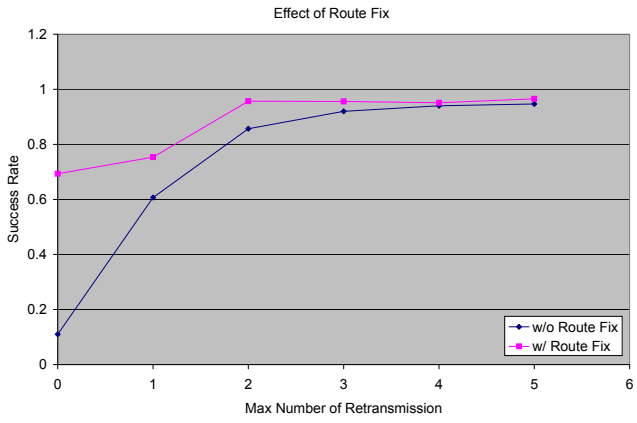


**Figure 8: Effect of route fix**

Comparison of options is performed on testbed in Soda Hall. Figure 7 shows floor plan of testbed.

## 8.1 Beacon Vector Routing

In our experimental evaluation, we use an implementation of Beacon Vector Routing (under submission), a point-to-point routing algorithm for wireless sensor networks. For the purpose of our evaluation, it is not necessary to describe the routing algorithm in much detail, except for its aspects that provide flexibility in selecting routes.

Beacon vector routing assigns virtual coordinates to nodes, derived solely from the network connectivity information. A subset of the nodes is selected as "beacons", and these beacons flood the network at least once, so that all nodes learn their distance to the set of beacons. The beacons act as reference points for routing. A node's coordinate is then the set of tuples $< B_i, d_i >$, for the beacons $i \in B$. Each node in BVR is required to know its distance to each of the beacons, and the coordinates of its one-hop neighbors. This set can also be extended to the $k$-hop neighbors, but the current implementation uses $k = 1$.

The basic routing exported by BVR is a route-to-coordinate interface. Routing in BVR is a form of greedy routing, similar to the routing used in geographic routing algorithms. When given a packet to route to a coordinate, a node selects the neighbor whose coordinates are the closest to the destination's coordinates, by some distance metric. The simplest such metric is given by Equation 1 below, and is equal to the sum of the absolute component-wise differences of the two coordinates (a form of an $L1$ metric).

$$\delta(\mathcal{P}(p), \mathcal{P}(q)) = \sum_{i=1}^{r} \omega_i |B_{ip} - B_{iq}|, \qquad (1)$$

This greedy-routing procedure may fail when no neighbor makes progress in the coordinate space towards the destination. To get out of these 'local minima', BVR employs a fallback routing mode that ultimately guarantees that the destination will be reached. In fallback mode, the node forwards the packet towards the beacon that is closest to the destination. This beacon is readily determined by the smallest component of the destination's coordinates. The minimum distance reached by the packet is recorded in the packet; this allows each node to resume normal greedy routing when one of the neighbors makes progress. Eventually, a packet may reach the beacon which is closest to the destination. In this situation, normal greedy routing cannot be used without the guarantee of loops. The beacon then initiates a scoped flood that will reach the destination. The choice of the fallback beacon as the closest to the destination minimizes the flood scope.

We can now explain how in BVR one can get flexibility for choosing next hops. At each step of greedy routing , there may be many nodes which make progress in coordinate space to the destination. Also, when doing fallback-mode routing, any node that is reachable and is closer to the desirable root is good to be used. In the BVR implementation, we fix the maximum number of alternative routes to 6, and the routing layer returns these alternatives ordered by progress and link quality.

## 8.2 Success Rate

Test data is collected from two particular pairs of nodes: one pair for test without route fix, and another for test with route fix. Both pairs are around 8 to 12 hops away, and link loss rate is around 20%. Network outage is excluded.
Figure 8 shows effect of link-level retransmission and route fix on success rate. Sending packets in alternate routes can be seen as a type of retransmission, and so some care must be exercised when counting the number of retransmissions for a given situation. Routing layer provides up to 6 candidates. Without route fix, 3 retransmission touches plateau. With route fix, 2 retransmissions approach ceiling.
Figure 9 shows effect of link-level retransmission and erasure code. Each line is related to how many redundant code words are added to 8 original messages. Both show significant improvement. When end-to-end reliability is over 80%, more than 2 redundant packets does not seem to be neces-

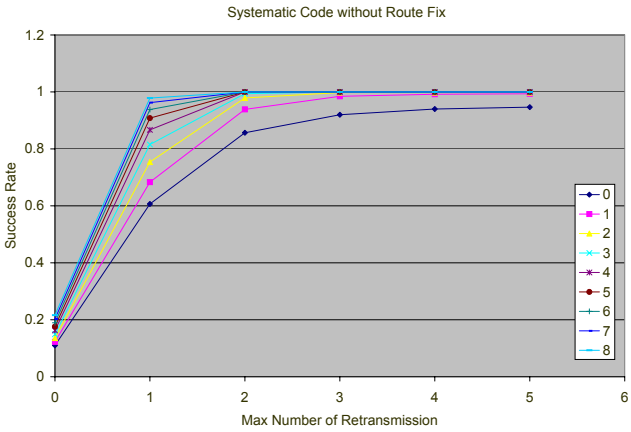


**Figure 9: Effect of erasure code**

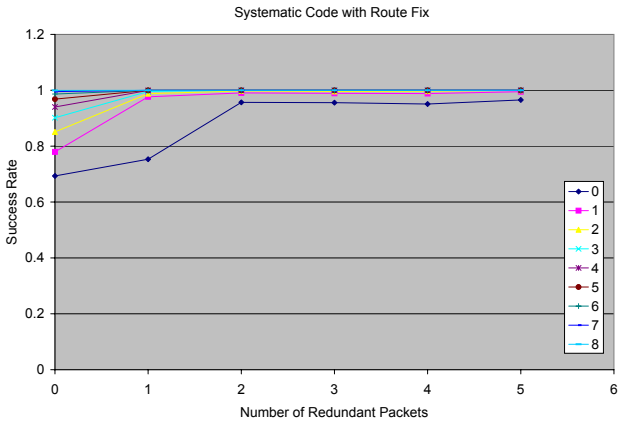


**Figure 10: Erasure code with route fix**

sary. Its gain diminishes quickly above 2 more redundant packets.
And Figure 10 shows the effect of combining all link-level retransmission, erasure code, and route fix. With route fix, even small amount of information redundancy achieves high reliability.

Overhead information will provide more precise comparison. Especially in case with route fix, it is hard to see success rate/overhead tradeoff, and compare relative efficiency: like which achieve better success rate given same amount of overhead, or which option require less overhead to achieve given success rate.

## 8.3 Erasure Code

Figure 11 shows how much reliability can be gained from erasure code. Each line in the graph represents how many additional packets are sent per 8 packets. When there is

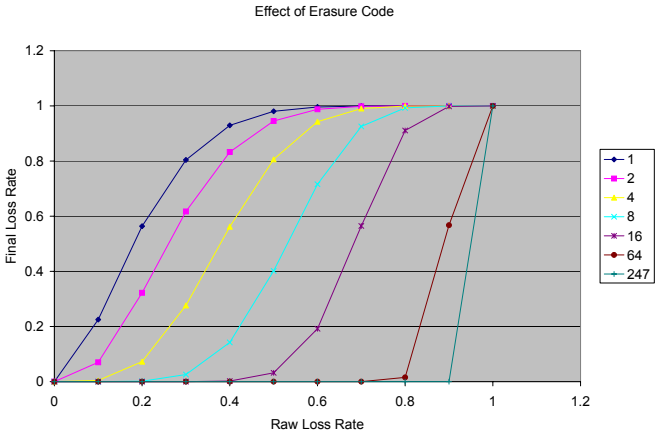


**Figure 11: Effect of Erasure Code on Loss Rate**

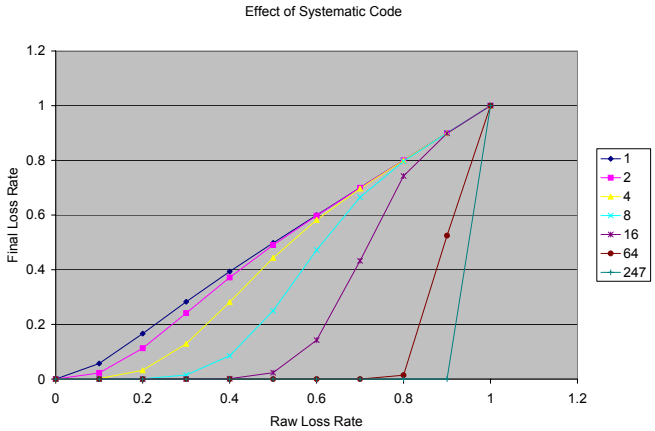


**Figure 12: Effect of Systematic on Code Loss Rate**

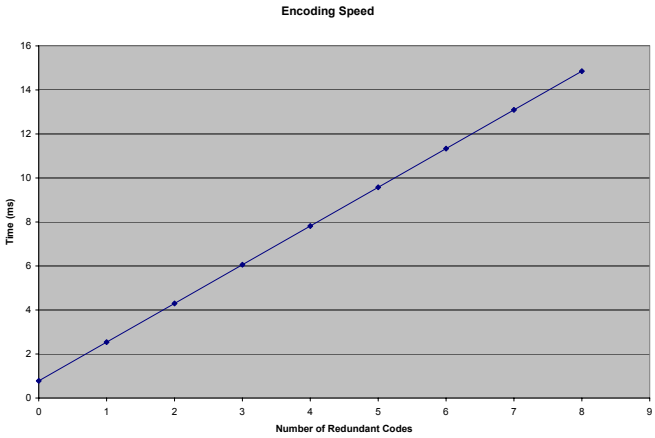


**Figure 13: Encoding Time**

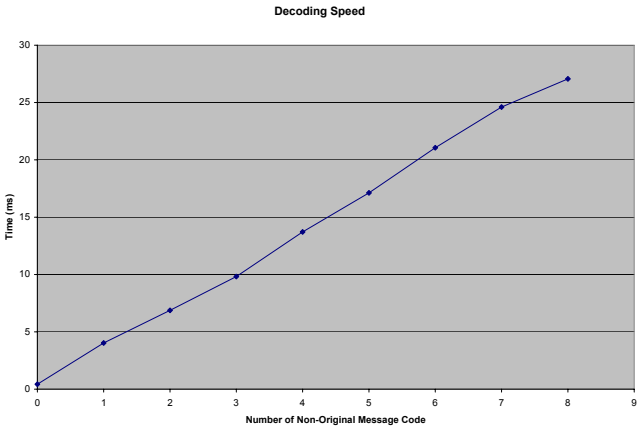


**Figure 14: Decoding Time**

small amount of redundancy, loss rate provided by erasure code is higher than raw loss rate. If we can not decode, we loose everything. So receiving 7 packets is effectively same as receiving 0 packets. Systematic code is good not only for saving computation, but also for increasing reliability. By using systematic code, even if we receive 7 packets, when 3 packets are codes containing original messages, we get 3 packets. Figure 12 shows improvement with systematic code. Final loss rate is always smaller than raw loss rate. Figure 11 and 12 are mathematically calculated. In this paper, systematic code is implemented, and all experiments used systematic code.

Figure 13 shows encoding time. In systematic code, first 8 packets dos not require any computation, they are just memory copies. Additional packet requires 1.7ms, which is smaller than transmission time of packet (20ms) by order of magnitude. Figure 14 shows decoding time. Decoding time depends on mix of code words as stated previously. Even in

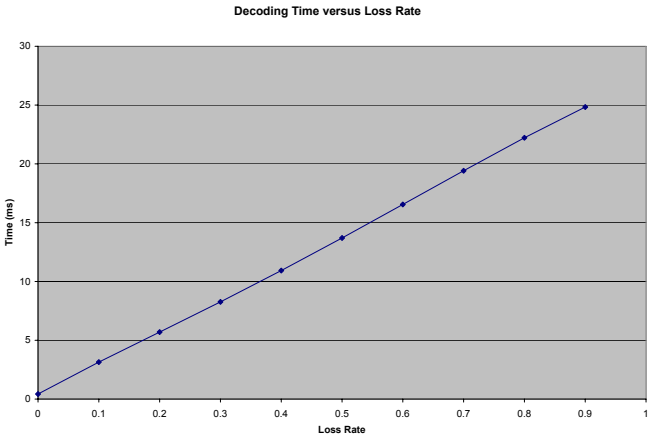


**Figure 15: Effect of Loss Rate on Decoding Time**

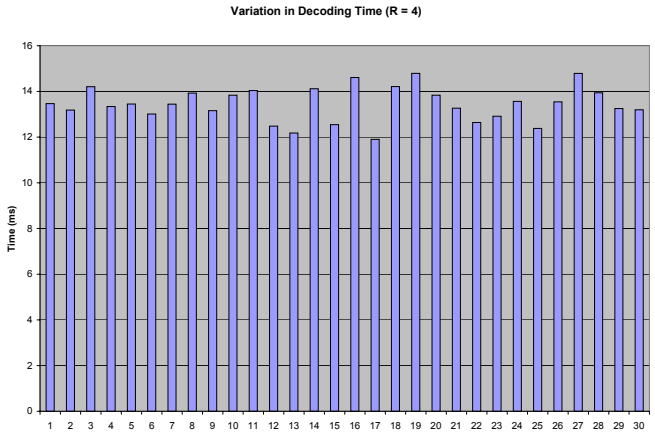


**Figure 16: Variation in Decoding Time**

worst case, it takes less than 30ms. Those times are measured on real mote. Considering that it takes at least 160ms to receive 8 packets, we can decode in real time. Based on this experiment value, we calculated expected amount time to decode given loss rate in Figure 15. Mix of code words (how many codes are original messages) determines decoding time. 'how many' affects decoding time, but 'which' does not. This is shown in Figure 16. 30 random cases are produced, and decoding time is measured. Difference is mostly within 15

# 9. CONCLUSION

Link-level retransmission is effective in most combination of options. Route fix is important to make the loss distribution less bursty. Especially elimination of long series of correlated consecutive drops makes erasure code very attractive solution.
Route fix basically provides flexibility in next-hop choice. Route fix and erasure code works greatly, but still not enough to survive bursts.

# 10. FUTURE WORK

Overhead information will provide more precise comparison. Especially in case with route fix, it is hard to see success rate/overhead tradeoff, and compare relative efficiency: like which achieve better success rate given same amount of overhead, or which option require less overhead to achieve given success rate.
End-to-end test is missing. We need to run LRX on Soda Hall testbed.
Thick path is another possible option. It achieves reliability only through information redundancy, and can survive link failure. Moreover it has low delay to deliver packet. Downside is that it injects a large amount of packets: it is multiplication of path length and path thickness. Since traffic is correlated locally, channel contention will not significantly affect whole network. However, in terms of energy consumption this would be a bad choice. It will be interesting to see tradeoff of success rate, overhead, delay, and energy consumption.
Some form of congestion control is need. Admission control would be a good candidate solution.
Initially it looked like that this implementation works as long as $M + N < 2^r - 1$. In experiments, when $M > r$ it worked in most cases. But there were cases it failed to work. Mathematical reasoning of this phenomenon is the future work. And if we can avoid these cases without expensive operation, it would be helpful.

# 11. REFERENCES


[1] http://mathworld.wolfram.com/.

[2] J. Blomer, M. Kalfane, R. Karp, M. Karpinski, M. Luby, and D. Zuckerman. An xor-based erasure-resilient coding scheme.

[3] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege. A digital fountain approach to reliable distribution of bulk data.

[4] A. Chlipala, J. Hui, and G. Tolle. Deluge: Data dissemination for network reprogramming at scale.

[5] K. P. Gummadi, R. Gummadi, S. D. Gribble, S. Ratnasamy, S. Shenker, and I. Stoica. The impact of dht routing geometry on resilience and proximity. *Proceedings of the ACM SIGCOMM 2003*, Aug 2003.

[6] S. Kim, D. Culler, J. Demmel, G. Fenves, S. Glaser, T. Oberheim, and S. Pakzad. Structure health monitoring using wireless sensor networks.

[7] P. Maymounkov. Online codes. *NYU, Technical Report TR2002-833*, November 2002.

[8] L. Rizzo. Effective erasure codes for reliable computer communication protocols. *ACM Computer Communication Review*, 27(2):24–36, April 1997.

[9] A. Woo, T. Tong, and D. Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. *ACM Sensys*, November 2003.