Efficient Erasure Code for Wireless Sensor Networks

Sukun Kim
467 Soda Hall
Computer Science Division
University of California at Berkeley
Berkeley, CA 94720
binetude@cs.berkeley.edu

ABSTRACT

Reliable communication in Wireless Sensor Networks is hard to achieve efficiently using methods in conventional systems like Internet. End-to-end retransmission is inefficient, or in some cases impossible. Erasure code is code with which we can reconstruct m original messages by receiving any m out of n code words ($n > m$). Erasure code provides efficient option for achieving reliability. Erasure code based on Reed-Solomon code is implemented. And further modifications were made to optimize for resource-constrained Wireless Sensor Networks mote. Systematic code is used to increase speed of decoding. Multiple independent code words are packed into a single packet to reduce the size of code word, and to reduce memory space and computation time eventually. Operation lookup table is used for quicker decoding. Memory usage and computing time depend on the size of code word. For most of practical usage in Wireless Sensor Networks, 4 or 8 bit long code word will be sufficient. Then moderate amount of memory is used, and computation is quick with operation table. Redundant information and flexibility in loss distribution give benefit to convergence routing, divergence routing (multicast), and point-to-point routing.

Keywords

erasure code, wireless sensor networks

1. INTRODUCTION

Technology advance as stated in Moores law enabled more powerful chip with lower power consumption at low-cost. This enabled small battery-powered device with wireless communication, having computational power strong enough to sense and actuate environment. Wireless Sensor Networks (WSN) is a network of those small devices. And Berkeley mote running TinyOS realized it in a real world.

Wireless Sensor Networks faces new challenges which were not critical in conventional computing systems. Even though mobile computing devices like laptop and PDA are also constrained by power source, Wireless Sensor Networks (WSN) have additional challenge for low-power sleep mode, and quick wakeup to support long life with low duty cycle.

From the perspective of wireless communication, 802.11 Wireless LAN and cellular network already handles queer radio behavior. However they are more focused on asymmetric communication between access point (tower) and mobile node. WSN also involves asymmetric communication between powerful super-node and normal node. But large portion of communication involves two normal nodes talking to each other through multiple other nodes, requiring real ad-hoc network solution. Moreover, ratio of communication overhead compared to computational power and energy source, is more significant than existing systems. WSN also need to support diverse routing layers: convergence, divergence in addition to point-to-point. Convergence is forwarding data to one sink, and divergence is spreading data from one sink like multicast. These two types can be realized using point-to-point, but with substantial penalty to efficiency. Therefore, WSN support all different routing layers.

Reliable communication, for which we focus, can not be achieved efficiently merely by methods used in other systems. For example, end-to-end retransmission may not perform well. In case of convergence, only convergence tree is maintained, so there is no reverse path to send acknowledgement from root to each leaf. In case of divergence, we will suffer ACK or NACK flooding.

Moreover, link-level retransmission does not go together well with end-to-end retransmission. If link-level retransmission is used, end-to-end delay varies enormously, and estimating round trip time becomes difficult. Upper bound can be used, but this introduces another problem. Since upper bound is pessimistic and communication is slow, sender should keep session information for a long time. This solution is not feasible for motes with small memory. However, we can not exclude link-level retransmission, because without link-level retransmission, we lose too much efficiency. Let us assume loss rate of each link is 10%. The probability to send data and to get acknowledgement successfully from a node 10 hops away without link-level retransmission, is 12%. It means that 88% of time, all link capacity used to forward data or acknowledgement will be wasted.

In this harsh condition, WSN can benefit from special encoding scheme: erasure code. Using erasure code, we can reconstruct m original messages with any m out of n code words. Erasure code relieves constraints on packet loss dis-

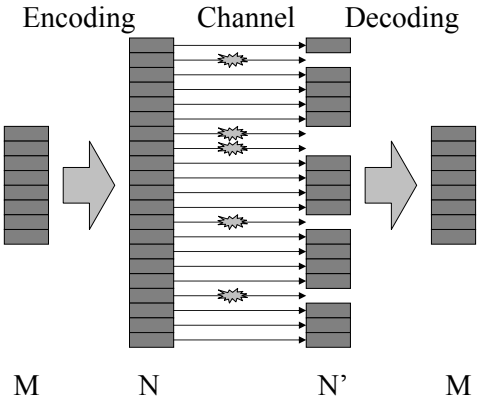

Figure 1: Mechanism of Erasure Code

tribution. In case where we simply duplicate packets, one of ith packet should be delivered. Otherwise, retransmission is necessary. With erasure code, any m packets are enough. Since erasure code does not require reverse path, it easily increases reliability of convergence routing, where there is no backward path. And for divergence case like code image dissemination for network reprogramming, even when each node is missing different set of packets, any additional packets will enable every node to reconstruct original message.
However, there also exists challenge in using erasure code. Since motes in WSN are severely constrained in computational power, and memory space, we need an efficient implementation. It turned out that in practical assumption for typical configuration of mote and operational environment, we can find sweet spot.
Next section will look at related work. Section 3 introduces concept and proof of erasure code. And one particular algorithm for erasure code, Reel-Solomon is explained. Then Section 4 explains additional optimization to adapt erasure code to resource-constrained WSN. Evaluation and conclusion follow in Section 5, and future work will be discussed in Section 6.

2. RELATED WORK

There exist diverse algorithms for erasure code which can be implemented in either software or hardware [6, 2]. [6] exploits diverse optimizations, from which this work gained many hints. Rateless code [3, 5] 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.

In Wireless Sensor Networks, there exist different types of routings. There is convergence routing [7], 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. We will discuss how all three types of routing layers can benefit from erasure code in achieving reliability later in Section 5.

3. ERASURE CODE

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

Figure 2: Vandermonde Matrix

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 1 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.

3.1 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.
This code is very useful, since encoding and decoding are computationally inexpensive. This is especially attractive in resource-constrained Wireless Sensor Networks.

3.2 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 2. 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.

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

Figure 3: High level diagram showing how Reed-Solomon code works

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

Figure 4: Systematic code

3.3 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 messages 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 3. 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.

4. MODIFICATION 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.

4.1 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!

4.2 Systematic Code

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 4 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.

4.3 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 operation on such a large field requires huge space and time.

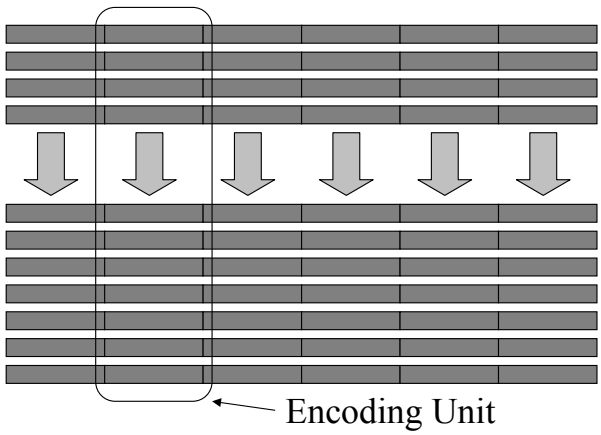

Figure 5: Divide packet into multiple independent code words

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 ith code words from each independent k data into a single packet. We either get all ith 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 5 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.

4.4 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 α. For any generator α, when we keep multiplying α, we can produce all $q-1$ non-zero elements. And then α 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.

5. EVALUATION AND CONCLUSION

In implementing erasure code, it is found that memory space is more stringent than computational power in Wireless Sensor Networks mote. One reason is that computation/memory ratio is higher than normal PC. Second reason is that wireless communication is slow compared to CPU especially when operations are done by table lookup.

We can argue that usage of operation table is waste. However, we are using small messages and code words by dividing packet into multiple independent code words. So operation table is moderate overhead to space. Let r be the size of code word, and p be the size of packet. For each operation table (exponent, log), there are 2^r elements in the field, each of size r bits. Space needed for one operation table is $r2^r$ bits. For both exponent and log tables, $r2^{r+1}$ bits are required. When r is 4, this is 16 bytes. When r is 8, memory consumption is 512 bytes. But in the case with $r = 8$, if memory space is very limited, we can divide data further to 2 data sets with four messages in each set. Then we can just use erasure code with $r = 4$. This process puts more restriction on loss distribution, and can eventually decrease effectiveness of erasure code. Tolerating any 6 packet losses provides more robustness than tolerating 3 packet losses from each of two transfers.

For the purpose of tolerating loss, even 4 bit long code word can survive 73% of loss rate. This would be more than

enough in practice, because communication over channel with higher than 73% of loss rate will need additional solution from different perspective like link-level retransmission. Packet buffer requires rp bits. However, this buffer is unavoidable. If we can reconstruct original data with less than r code words, it means that we can produce r pieces of information out of less than r pieces of information, which is impossible.

Erasure code can be used to improve reliability on wireless sensor networks, where end-to-end retransmission is very expensive. In case of convergence routing where there is no backward path, high reliability can easily be achieved. Erasure code also can be used in divergence routing like code dissemination. Let us assume that node 1 has 8 program images to distribute, and 8 bit long code word is used. Assume node 2 is missing image 3 and 6. Node 3 is missing image 1 and 8. For node 4 image 2 and 7 are missing, for node 5 image 4 and 5. With just retransmission, node 2, 3, 4, and 5 will send request for missing images. Then node 1 will send those missing images. This process involves 12 packets: 4 for requests, and 8 for missing images. Using erasure code, we only need to send 2 additional packets, and everyone will get every image. Even if some of nodes are still missing one image, which can be potentially different from one to another, one additional packet is enough to cure all. This is an improvement in the order of magnitude.

6. FUTURE WORK

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. If we can have larger M, as discussed in Section 5 tolerating any 6 packet losses provides more robustness than tolerating 3 packet losses from each of two transfers.

In an experiment for characterizing loss behavior of WSN, it is found that stale routing table after link failure is a significant problem. It is critical not only because it is too frequent, but also because it is hard to cure with conventional methods used in Internet. To reduce delay between link failure and routing table update, interval between control packets for probing link quality should decrease, which means more traffic overhead for finer-grain update. In case of internet, link failure is not very frequent, therefore this is not a significant problem. However, in WSN space or wireless network environment in general, link failure is quite frequent.

In case of stale routing table following link failure, end-to-end retransmission is not useful, since packets will follow the same failed link, just leading to loss. Even erasure code will not work well in this case. Correlated failure will prohibit delivering even M code words. It seems that we need fix routing table after some number of link-level transmission fails. One method would be forwarding to alternative next hop.

There is another method to overcome link failure with stale routing table through redundancy. Rather than taking a single path, we can forward through a path with some width. Unless all links across the path fail, forwarding will succeed. This approach has a drawback also. Because we are duplicating path, the number of messages sent is multiplied by the width of path.

Encoding time, and decoding time evaluation need be performed on real mote. Since systematic code is used, decoding time depends on which set of code words it received. The ratio of code words containing original messages, and its effect on decoding time will be interesting. And loss rate, and average decoding time graph will be also good.

Code can be found at
http://cvs.sourceforge.net/viewcvs.py/tinyos/tinyos-1.x/contrib/GGB/apps/ErasureCode
Future updates will be added.

7. ACKNOWLEDGMENTS

Discussion with Rodrigo Fonseca was invaluable in analyzing and optimizing algorithm.

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

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

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