CS 591B1: ALGORITHMIC ASPECTS OF COMPUTER NETWORKS **Spring 2002**

Lecture 7 — March 11, 2002

Lecturer: John Byers *BOSTON UNIVERSITY* *Scribe: D. Barman, A. Maram*

In today's lecture, we learned some more information dispersal techniques continued from previous lectures. In the previous lecture we learned Shamir's method for sharing information which is described in the paper, "How to share a secret" [3] and Rabin's Information Dispersal Algorithm [2].

7.1 Information Dispersal

In any information dispersal technique, the aim is to transmit a message consisting of d symbols represented by $b_0, b_1, \ldots, b_{d-1}$ (d symbols over any finite field say, Z_p where p is a prime). The encoding of the d message symbols are $e_0, e_1, \ldots e_{n-1}$ where $n > d$ and the encoding has the property that any d of the n suffice to reconstruct the original message i.e up to $n - d$ losses can be tolerated.
We now study several methods of information dispersal encoding techniques.

Method 1

Encoding:

Construct a degree $d - 1$ polynomial where the coefficients are the messages. The polynomial is given by,

$$P(x) = b_0 + b_1 x + b_2 x^2 + \ldots + b_{d-1} x^{d-1} \tag{7.1}$$

Evaluate this polynomial at n points $0, 1, 2, \ldots, n-1$ and transmit the coordinate pairs $(0, P(0)), (1, P(1)), \ldots, (n-1, P(n-1))$ which a receiver will receive if no data is corrupted. All the operations are taken over a finite field.

Decoding:

If a receiver receives any d of n pairs, he can recover the original message by constructing a polynomial of degree d using Lagrange Interpolation. Once the polynomial is constructed, the coefficients of this polynomial are the messages transmitted.

Lagrange Interpolation Theorem:

Given d points $(a_1, b_1), (a_2, b_2) \ldots (a_d, b_d)$ where a_i's are distinct, then there is a unique polynomial P of degree $d - 1$ where $P(a_i) = b_i \ \ \forall i \ 0 \le i \le (d-1)$. The Polynomial is :

$$P(x) = \sum_{j=0}^{d-1} b_j \prod_{j \neq k} \frac{x - a_k}{a_j - a_k} \tag{7.2}$$

Method 2

Encoding:

Construct a polynomial $P(x)$ of degree $d - 1$ such that

$$P(0) = b_0, P(1) = b_1 \ldots, P(d-1) = b_{d-1} \tag{7.3}$$

Evaluate this polynomial at n points $0, 1, 2, \ldots, n-1$ and transmit the coordinate pairs $(0, P(0)), (1, P(1)), \ldots, (n-1, P(n-1))$.

Decoding: In general, same as before.

But if there is no loss of information, first d transmissions are the messages. In this method, the original message is a part of transmitted message.

7.2 Erasure Codes vs Error-Correcting Codes

Error Correcting Codes: We wish to transmit a message (strings of symbols) over a noisy channel. Using Error Correcting Codes we encode the message in redundant way so that despite the noise, the receiving party can reconstruct the original message.

Erasure Codes: The key idea behind erasure codes is that k blocks of *message* data are encoded at the sender to produce n blocks of *encoded* data in such a way that any subset of k encoded blocks suffices to reconstruct the message data. Such a code is called (n, k) code and allows the receiver to recover from upto $n-k$ losses in a group of n encoded blocks.

Erasure Codes [1] can be implemented via Reed-Solomon codes. These Reed-Solomon codes can be implemented using

- Polynomial Interpolation
- Rabin's Method [2]: Cauchy-based Reed Solomon. Here "Cauchy" refers to the A matrix used in Rabin's paper.

Decoding of Erasure codes can be done either in the matrix world or polynomial world.

Polynomial Codes and the Berlekamp-Welch Algorithm

Let Z_p be the field modulo p. The message we wish to transmit is a string of elements from this field : $x_1, \ldots, x_k \in Z_p$ where $k < p$. Notice that the x_i define a unique polynomial f of degree $\leq k$ over Z_p such that $f(i) = x_i$ for all $0 \leq i \leq k$. The redundant encoding in this case is an evaluation of f at all points of the field,i.e. the transmitted message is: $f(0), f(1), \ldots, f(p-1)$.

Because of possible noise, the receiving party may receive values of a different function (evaluated at all the points of this field). We let $f' = f + E$ be the received function where $E(i)$ is the noise when i^{th} element was transmitted. As we will next see, if the number of errors is not too large then the receiving party can reconstruct the original message by evaluating f at the points 0,..., k.

Let the distance between two functions f and g be : $|f, g| = \{i | f(i) \neq g(i)\}$. We now show that if the number of errors $|f, f'| \leq \frac{p-(k+1)}{2}$ then f can be efficiently reconstructed from f'. We start by showing that f is the unique polynomial of degree at most k within distance at most $\frac{p-(k+1)}{2}$ from f'. Notice that any degree k polynomial is completely determined by values at $k+1$ points. Thus any two distinct polynomials f and g of degree k can agree on at most k points, i.e. $|f, g| \geq p-k$. Now, if both f and g are within distance $\frac{p-(k+1)}{2}$ from f' then by the triangle inequality $|f, g| \leq p-(k+1)$, a contradiction.

We now present the Berlekamp-Welch algorithm for recovering f from f'. The input for the algorithm is f' evaluated at the p points of the field. We know that $|f, f'| \leq \frac{p-(k+1)}{2}$ where f is a degree k polynomial. The goal is to output f in polynomial time.

Let $m = |f, f'| \leq \frac{p-(k+1)}{2}$. Consider a non-zero error-locater polynomial defined by $e(i) = 0$ if an error occured in i, i.e. if $f(i) \neq f'(i)$. Such a polynomial exists with degree$(e) \leq m$.

We can see that $f(i) \cdot e(i) = f'(i) \cdot e(i) \ \ \forall i \in Z_p$. If an error occurs at any point x_i, $e(x_i)$ will be 0, then both the sides of the equation will be 0. If there is no error, $f(x_i) = f'(x_i)$ for all i.

Let $h = f \cdot e$. Then $h(i) = f'(i) \cdot e(i) \forall i \in Z_p$. Furthermore,
degree(h) = degree(f) + degree$(e) \leq k + m \leq k + \frac{p-(k+1)}{2} = \frac{p+k-1}{2}$.

Since $f'(i)$ are known, it gives p equations in $\frac{p-(k+1)}{2} + \frac{p+k-1)}{2} = p - 1$ unknowns, which are the coefficients of e and h.

We now show that these equations have a unique non-zero solution (taking $e = h = 0$ is always a solution) up to a common multiplicative factor, i.e that the quotient polynomial $\frac{h}{e}$ is the same for all the non-zero solutions.
Let e_1, h_1 and e_2, h_2 be the two distinct non-zero solutions to the above set of equations. We first observe that

$$h_1(i) \cdot e_2(i) \quad = \quad h_2(i)e_1(i) \ \ \forall i \in Z_p$$

This is clear when $e_1(i) = 0$ (resp, $e_2(i) = 0$) since in this case $h_1(i) = 0$ (resp $h_2(i) = 0$) becasue $f'(i)e(i) = h(i)$ must hold. If both $e_1(i)$ and $e_2(i)$ are non-zero then $\frac{h1(i)}{e1(i)} = \frac{h2(i)}{e2(i)} = f'(i)$ and thus above equation holds.
Summing the bounds on the degrees of e and h we get that the degree of both $h1 \cdot e2$ and $h2 \cdot e1$ are strictly less than p. Thus, since these two polynomials agree on p points , they are the same. Hence, the quotient polynomials $\frac{h_1}{e_1}$ and $\frac{h_2}{e_2}$ are the same polynomials.
As we mentioned before, there is always at least one non-zero solution for which $f = \frac{h}{e}$ (take e to be the error locating polynomial defined before, and $h = f \cdot e$). Thus, for any non-zero solution e', h' of the above equations, $f = \frac{h'}{e'}$.

7.3 Efficient Erasure Correcting Codes(EECC)

Reed-Solomon codes gives the perfect information recovery but its running time for encoding and decoding is $O(n^2)$. In EECC, the running time is improved to linear time. It is proportional to $n\ln(\frac{1}{\epsilon})$ for all $\epsilon > 0$ at the price of slight increase in stretch factor $(1 + \epsilon)$.

7.3.1 Performance Metrics

The following metrics can be used to analyse the different erasure correcting codes.

1. Time complexity: Running time to encode/decode. In Rabin/Reed-Solomon techniques, matrix inversions take $O(n^2)$ where n is the number of rows.

2. Decoding inefficiency which is defined as

$$\frac{\textit{Number of pkts need to receive for decoding}}{\textit{Number of input symbols}}$$

   Decoding inefficiency can be weakened using erasure code. In other words, any $(1 + \epsilon)n$ symbols are sufficient to recover input symbols with high probability. Here, the decoding inefficiency is $(1 + \epsilon)$

Reed-Solomon coding is the perfect with decoding inefficiency equal to 1.

7.3.2 Erasure Codes via Bipartite Graphs

Main idea

1. Cascade sequence of "Error-reduction" codes. The k input symbols are associated with the leftmost n nodes of a bipartite graph. The graph has redundant or check nodes on the right hand side.The number of redundant check nodes is βk. We can assume that each node corresponds to a packet or bit as shown in Figure 7.1.
2. Extract the check nodes first.
3. Extract the message nodes.

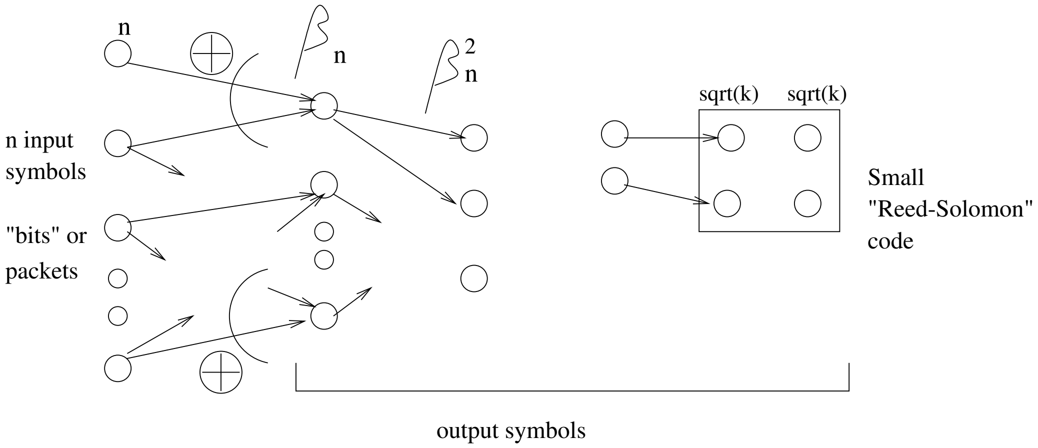

Figure 7.1. A graph defines a mapping from message nodes to check nodes. Mapping: A check node on the right is XOR of all neighboring message nodes on the left

Erasure Decoding

1. Decode right to left.
2. First decode final layer using a more conventional erasure correcting code such as Reed-Solomon code.
3. Then, assuming all check nodes are known for layer i and assuming some of the nodes in layer $i-1$ are directly recovered, decode the other nodes in layer $i-1$.

To illustrate through an example, let x_i^s be the input symbols and y_i^s be the output symbols . Also linear constraints give us:

$$\begin{aligned} y_1 &= x_3 \\ y_2 &= x_2 \oplus x_3 \\ y_3 &= x_1 \oplus x_3 \\ y_4 &= x_1 \oplus x_2 \oplus x_4 \end{aligned}$$

The process of recoving the left hand nodes (Figure 7.2) is as follows.

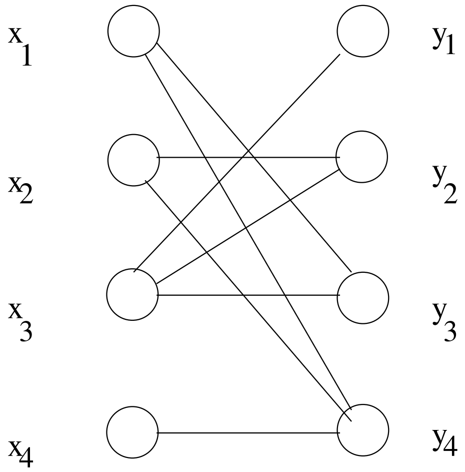

Figure 7.2. A graph showing recovery provess

Recovery rule : Recover a node on left hand side if it has a degree one neighbour on the right hand side.

Using the above rule, we can directly get x_3. x_2 and x_1 can be recovered by XOR-ing x_3 with y_2 and y_3 respectively. We can solve for x_4 by XOR-ing x_1 and x_2 with y_4

Invariant at any level, all of the right hand symbols are recovered, plus some of the left hand symbols. To produce codes that can correct erasures of check nodes as well as message nodes, we cascase the codes.

The decoding time is proportional to the number of edges across all the levels.

The construction of cascaded bipartite graphs

Let's denote the sequence of bipartite graphs as $B_1, B_2, \ldots$ in the cascaded bipartite graph.

1. Pick up a sequence of bipartite graphs $B_1, B_2, \ldots$.
2. Choose right hand nodes in B_{i+1} with left hand nodes in B_i. The edges can be chosen randomly or with some distribution.
3. Terminate when the number of nodes$\approx \sqrt{k}$
4. Truncate with $O(n^2)$ time decodable code such as Reed-Solomon codes.

Encoding of Bipartite graphs

1. First, place the values of initial k left nodes in B_1 (here n is assumed to be total number of nodes, k is input symbols).
2. Encode from left to right by setting the values in the right nodes to parity (XOR) of their left neighbors.

7.3.3 Analysis of 3-6 Bipartite graphs

3-6 Bipartite graph is a bipartite graph in which every node on the left hand side has a degree of 3 and all the nodes on the right hand side has a degree of 6.

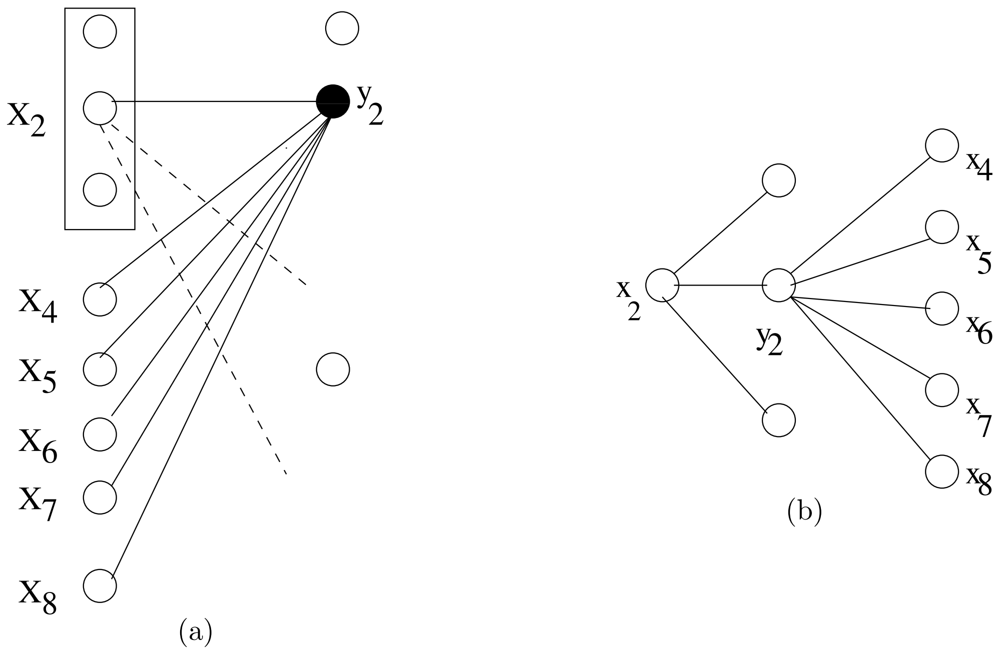

Figure 7.3. (a) x_i on the left has degree 3 and y_i on the right hand side has degree 6 (x_4, x_5, x_6, x_7 and x_8 are directly recovered) (b) shows dependence of x_2 on its neighbors

Let us try to analyze the probability that we can solve for any x_i in a 3-6 bipartite graph. Here, we are assuming at any stage, the right hand nodes y_i are already recovered for each left hand node. We also assume that a section of x_i^s will be directly recovered and a section will be not. Let p be the probability that any x_i is already recovered.
By following any of of three edges from a left hand node, probability that we can recover that left hand symbol is p^5.
Total chance of recovering $Z = \alpha(1 - (1 - p^5)^3)$
where α = fraction of total number of nodes in contention to be resolved (i.e., αn is the number of nodes which are not directly recovered).
The decoding efficiency of d-2d graph is shown in Figure 7.4.

7.3.4 Moving to irregular graphs

A random graph can have arbitrary degree distribution as in Figure 7.5 in which we can have,

1. More chances for nodes of higher degree.
2. Some nodes have better chances of resolving left neighbours.
3. High degree nodes act as an insurance in providing dense connectivity to other nodes.

Let us refer to the edges that are adjacent to a node of degree i on the left (right) as *edges of degree i* on the left(right). Each of the degree sequences is specified by a pair of vectos$(\lambda_1, \ldots, \lambda_m)$ and $(\rho_1, \ldots, \rho_m)$, where λ_i and ρ_i are given as,

$$\begin{aligned} \lambda_i &= \text{Fraction of edges of degree } i \text{ on left} \\ \rho_i &= \text{Fraction of edges of degree } i \text{ on right} \end{aligned}$$

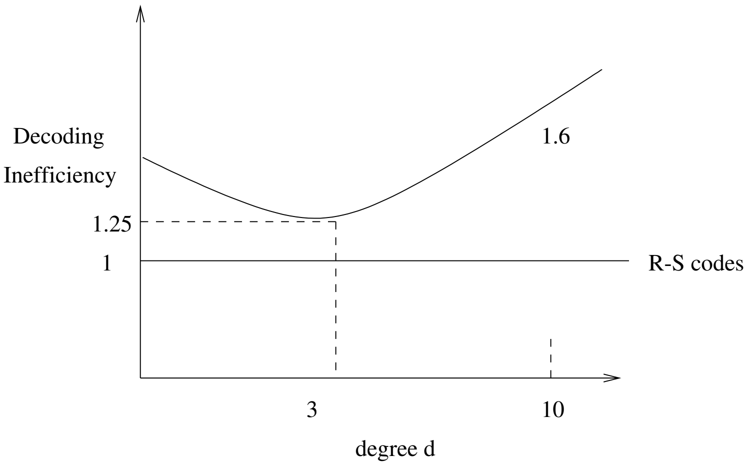

Figure 7.4. Performance bound on d-2d graphs

The sequence λ and ρ give rise to generating polynomials $\lambda(x) = \sum_i \lambda_i x^{i-1}$ and $\rho(x) = \sum_i \rho_i x^{i-1}$. The coefficient of x^{i-1} in the polynomial is the value of λ_i.

Example: Assume that half the message nodes have degree three and half have degree four and that there is a total of n nodes. Since every degree three node has three edges emanating from it, whereas every degree four nodes has four edges emanating from it we see that there are in total $\frac{1}{2}3n$ edges which emanate from degree three nodes and that there are a total of $\frac{1}{2}4n$ edges which emanate from degree four nodes. Therefore we have

$$\begin{aligned} \lambda_3 &= \frac{1/2 \cdot 3}{1/2 \cdot 3 + 1/2 \cdot 4} = \frac{3}{7} \\ \lambda_4 &= \frac{1/2 \cdot 4}{1/2 \cdot 3 + 1/2 \cdot 4} = \frac{4}{7} \end{aligned}$$

So, in this case $\lambda(x) = \frac{3}{7}x^2 + \frac{4}{7}x^3$. The fraction of edges which emanate from a degree i node is the coefficient of x^{i-1} rather than x^i as one might expect at first.
Let p be the probability that a node is not yet recovered (lost in the erasure channel).

$$\begin{aligned} \Pr[\text{all right neighbours were recovered}] &= \rho_1 + \rho_2 \cdot (1-p) + \rho_3 \cdot (1-p)^2 + \dots \\ &= \sum \rho_i \cdot (1-p)^{i-1} \\ &= \rho(1-p) \end{aligned}$$

Let x_i be a left node which is adjancent to a right node y_j of degree d. By right neighbors we mean the set of x_k^s which are also adjacent to y_j.

The probability that a node x_i is adjacent to a right node y_j of degree 1 is ρ_1 and such a node y_i is assumed to be recovered. Similarly, the probability that a node x_i is adjacent to a right node y_j of degree k is $\rho_k(1-p)^{k-1}$. The probability that y_j has $k-1$ other neighbors recovered is $(1-p)^{k-1}$.

The best distributions found from the experiments are: left hand side : heavy-tail, right hand-side : Poisson

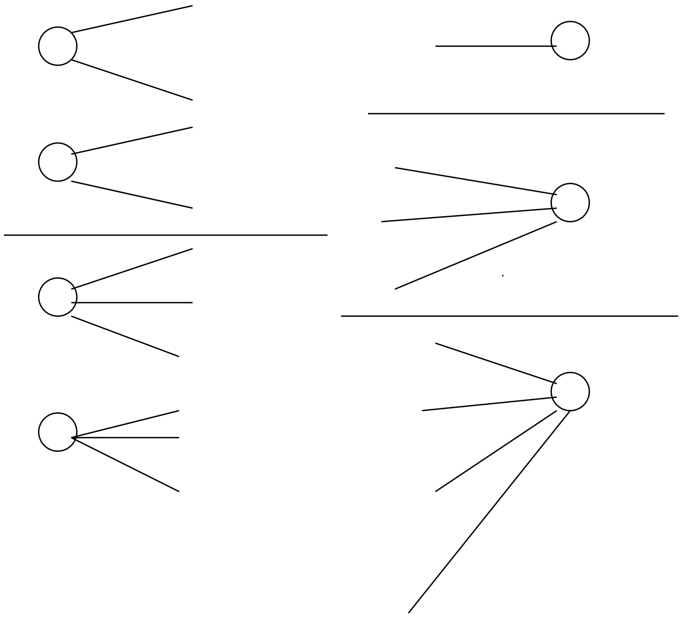

Figure 7.5. Irregular graph with some degree distribution of the node degrees

Bibliography

[1] M. Amin Shokrollahi Michael G. Luby, Michael Mitzenmacher and Daniel A. Spielman. "Efficient Erasure Correcting Codes". In *IEEE Transactions on Information Theory, Vol 47 No 2*, February 2001.

[2] M. Rabin. "Efficient Dispersal of Information for Security, Load Balancing and Fault Tolerance". In *Journal of the ACM 38, pp. 335-348*, 1989.

[3] Adi Shamir. "How to Share a Secret". In *Communications of the ACM 22(11), pp. 612-613*, 1979.