CS 559: Algorithmic Aspects of Computer Networks **Fall '07**

Lecture 2 — September 10, 2007

Lecturer: John Byers *BOSTON UNIVERSITY* *Scribe: Raymond Sweha*

Today's lecture covers Information Dispersal Algorithm (IDA) presented by Rabin in his seminal work [1]. The main idea is to create n Encoding packets out of m Original packets (where n>>?m), the recovery of *any* m Encoding packets would enable us to retrieve the m Original packets.

2.1 Key Property

Given a file F of size L, create F_i dispersions where:

$$|F_i| = \frac{L}{m} \quad \forall 1 \leq i \leq n \tag{2.1}$$

Any m of the F_i's is sufficient to reconstruct F.

2.1.1 Remarks

- Notice that the received data of size $m * \frac{L}{m} = L$ is sufficient to reconstruct the original file (of size L as well).
- one of the problems with IDA is that we can't get a prefix of F using $t < m$ pieces. (PET93)[2] developed a hierarchal version where getting $m_1 < m$ pieces would get you a rough version of F, while getting m_2 pieces (where $m_1 < m_2 < m$) would give you a better version.
- $n >> m$ is good for some applications (e.g. AKAMAI), while $n \simeq m$ is good for other applications (e.g. disk application) as we need to create date of size $n * \frac{L}{m} = L * \frac{n}{m}$ which is preferred to be near L.

2.2 Construction

For a vector b of m entries, we can construct a vector c of n entries using an m*n matrix A, Where:

$$A_{mn}.b_m = c_n \tag{2.2}$$

the knowledge of any m independent equations (i.e. any m values of c, given that the corresponding m rows of A are independent) is enough to get b_m.
Without Loss of Generality (WLOG), we can divide the file F into columns of size m; b_i where i = 1,2,...,L/m and apply the same transformation on them (i.e A), resulting in L/m columns of length n; c_j where j =1,2,...,L/m. As illustrated in Figure 2.1.
we call each row of C a *dispersion* and collecting any m dispersions would enable the user of retrieving F. More formally, let the retrieved pieces be C* and the corresponding rows of A be A*.

$$B = [A^*]^{-1}[C^*] \tag{2.3}$$

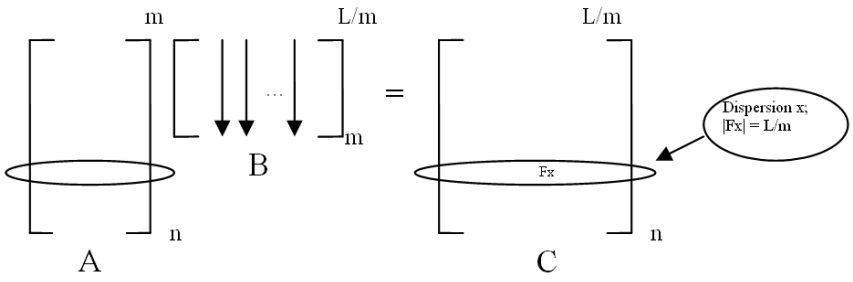

Figure 2.1. IDA construction. Each row of C is called dispersion F_x, collecting any m dispersion would enable us retrieve B.

2.3 Selecting A

In selecting A, we have two main consideration:

- Any combination of m rows are linearly independent.
- Any combination of m rows are easily invertible.

We have two ways of constructing A, namely;

- Using a well known matrix construction algorithm like, hadamard, Van der Monde, Read-Muller or Reed-Solomon. Any m rows of these matrices are known to be independent and easily invertible (quadratic order).
- Generating A randomly. $A : a_{ij} \in_R \mathbb{Z}_p$
  The pros of this approach are: it is elegant, simple, sharable and securable, while the cons are that we need matrix inversion and Gaussian elimination to get the file which are costly operations. Also, there is no guarantee that any m rows are invertible, but this problem is not so crucial as the decoder can wait for more dispersions which would allow it to decode.

The real difference is in the decoding time, Gaussian Elimination needs $O(m^2 * \frac{L}{m}) = O(Lm)$ operations to decode and retrieve the original file, which is ok for small to moderate file size L, But it imposes serious problems when L is large. Notice that "small" and "large" here relates to the RAM size.
On the other hand, specially constructed matrices can be decoded in O(L* polylog(m))

2.3.1 The proof that any m rows of a random matrix are likely independent

A sketch of the proof was presented in the lecture, and the paper states the bound without explanation, Here is the formal proof:
$Pr(|A| \neq 0 : a_{ij} \in_R \mathbb{Z}_p)$
$= Pr(A_1 \neq 0) * Pr(A_2 = \lambda A_1) * Pr(A_3 \neq \alpha A_1 + \beta A_2) * ... * Pr(A_m \neq \sum_{i=1}^{m-1} \alpha_i A_i)$
$= (1 - \frac{1}{p^m})(1 - \frac{p}{p^m})(1 - \frac{p^2}{p^m})...(1 - \frac{p^{m-1}}{p^m})$
$= \prod_{i=1}^{m}(1 - \frac{1}{p^i})$
$\geq 1 - \sum_{i=1}^{m} \frac{1}{p^i}$
$\geq 1 - \sum_{i=1}^{\infty} \frac{1}{p^i}$
$\geq 1 - (\frac{1}{1-\frac{1}{p}} - 1)$
$\geq 1 - (\frac{p}{p-1} - 1)$

$\geq 1 - (\frac{p-(p-1)}{p-1})$
$\geq 1 - \frac{1}{p-1}$ #

Bibliography

[1] M. Rabin. "*Efficient dispersal of information for security, load balancing, and fault tolerance*," J. ACM, 1998.

[2] Bernd Lamparter and Andres Albanese and Malik Kalfane and Michael Luby, "*PET: priority encoding transmission (video): a new, robust and efficient video broadcast technology*," MULTIMEDIA '95: Proceedings of the third ACM international conference on Multimedia.