Data Persistence in Large-scale Sensor Networks with Decentralized Fountain Codes

Alex Washburn

Graduate Center, CUNY

Oct 19th 2022

Introduction

Introduction

Today's paper:

Data Persistence in Large-scale Sensor Networks with Decentralized Fountain Codes [1]

[1] (Lin, Liang, and Li 2007)

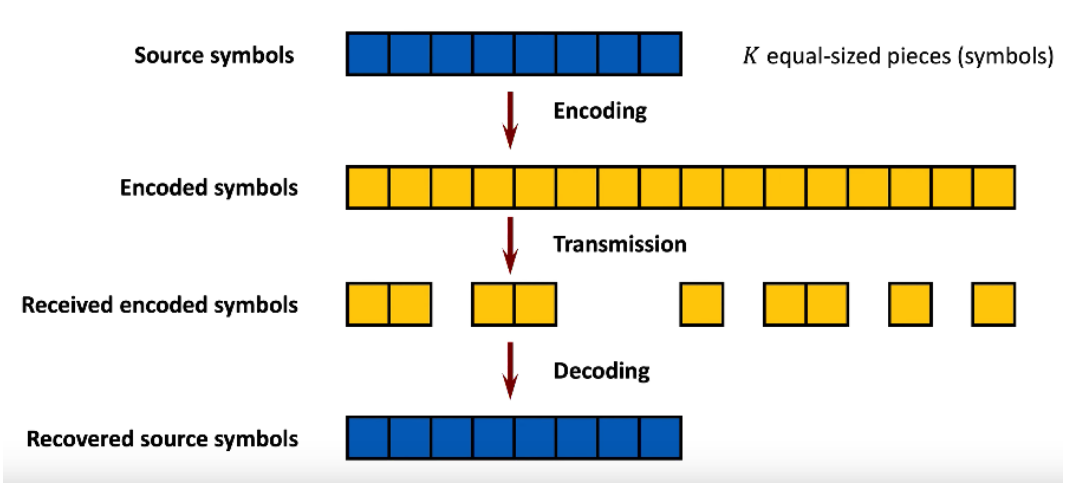
Overview

- Luby Transform (LT) code [2]
- Network of nodes, sensors, sinks
- Unreliable nodes
- Unavailable sinks

[2] (Luby 2002)

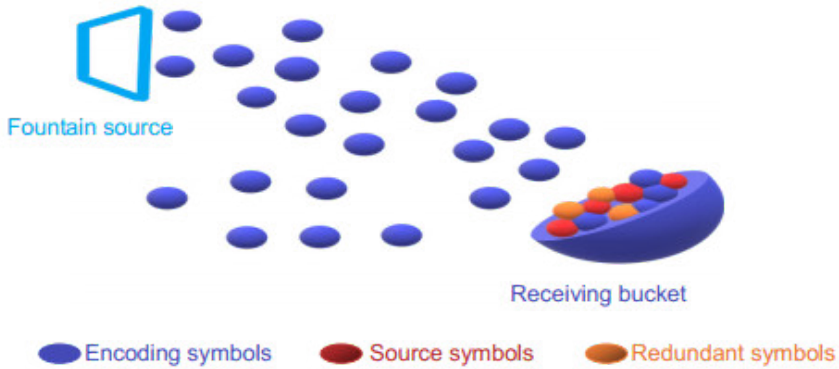
Fountain Codes (why)

What problem does a fountain code solve? Recovering erasures!



Fountain Codes (how)

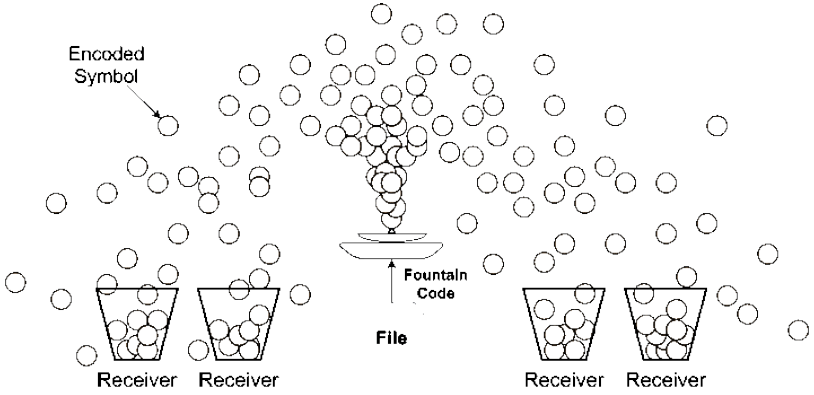
How does a fountain code recovering erasures? Redundancy!



LT Code Concept

Fountain Codes (who)

Who does a fountian code send messages to? Many neighbors!



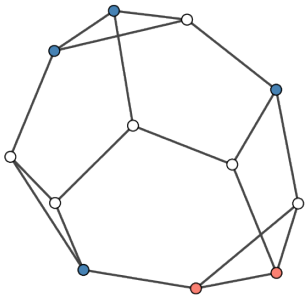
LT Code Concept (Networked)

Sensor Networks

Sensor Networks (concept)

What is the idea of a sensor "Sensor Network?"

- Many "geographically" connected nodes
- Blue "sensor" sources
- Red "receiver" sinks
- White "cache" nodes

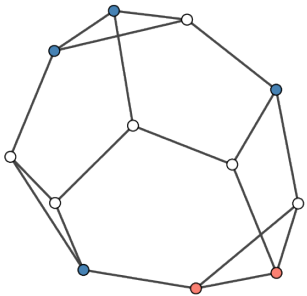


Example Network

Sensor Networks (scale)

What is the scale of a sensor "Sensor Network?"

- $N > 10^4$ nodes
- $K > 10^3$ sources
- 1 or more sinks
- $N - K$ "cache" nodes

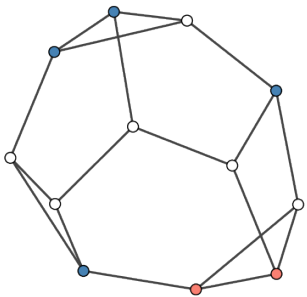


Example Network

Sensor Networks (difference)

What is **not** of a sensor "Sensor Network?"

- Fixed topology
- Centralized routing
- Solve using *max-flow*
- "Push-based" data flow

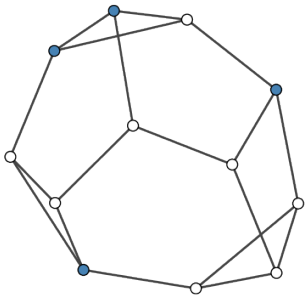


Example Network

Sensor Networks (sinks)

What are sinks in a sensor "Sensor Network?"

- Sinks are not persistent in the network
- Sinks transit through the network
- "Pull-based" data flow

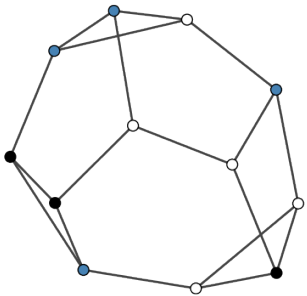


Sensor Network

Sensor Networks (topology)

What is network topology in a sensor "Sensor Network?"

- Nodes may permanently fail
- Topology changes, monotonically decreasing vertices/edges

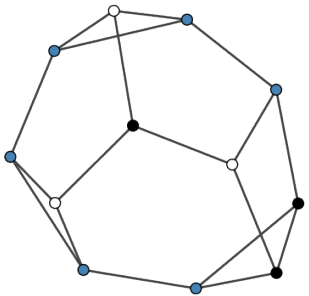


Sensor Network with Failures

Motivating Example (Science)

Example: Scientific sensor network

- Deploy many sensors collecting scientific data
- Difficult to access sensors after deployment
- Nodes damaged by flora, fauna, or weather
- Collect data via expensive collector vehicle
- Reliably recover as much data as possible

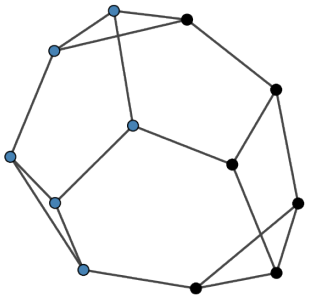


Scientific Network

Motivating Example (Warfare)

Example: Battlefield sensor network

- Battlefield sensors collecting inelligence
- Enemy has abundance of artillery
- Enamy shells portion of battlefield
- Carfully retrieve data from surviving nodes
- Reliably recover as much data as possible



Battlefield Network

Desiderata

Sensor network features

- Decenteralized routing
- Dynamic, decreasing topology
- Pull-based polls of vertex subsets
- Reliable data recovery from polling

Data Persistence

Data Persistence

- Lots of erasures in network
- Fountain codes handle erasures well
- Construct specialized LT code

Data Persistence (RAID lessons)

Sensor network as persistent storage

- View network as analogous to RAID device
- RAID used to recover data after disk failures
- RAID-6 already uses coding theory (Solomon codes)
- Hint that coding theory is the right approach

Data Persistence (Soliton Distribution)

$$\rho(i) = \begin{cases} 1/K & \text{if } i = 1, \\ 1/i(i-1) & \text{for } i = 2, 3, \ldots, K. \end{cases}$$

Ideal

$$\mu(i) = \frac{\rho(i) + \tau(i)}{\sum_i \rho(i) + \tau(i)}$$

Robust

$$R = c \ln(K/\delta)\sqrt{K}$$

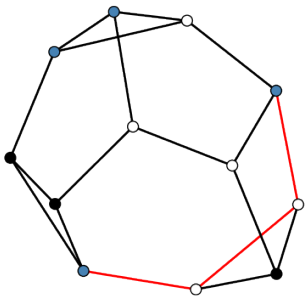
$$\tau(i) = \begin{cases} R/iK & \text{for } i = 1, \ldots, K/R - 1 \\ R \ln(R/\delta)/K & \text{for } i = K/R \\ 0 & \text{for } i = K/R + 1, \ldots, K \end{cases}$$

Extra

Data Persistence (Broadcasts)

Random walks

- Encoded blocks broadcasted into network
- Encoded blocks have appended source ID
- Routing follows a random walk
- Each receiving node caches encoded blocks
- Walk length depends on network size

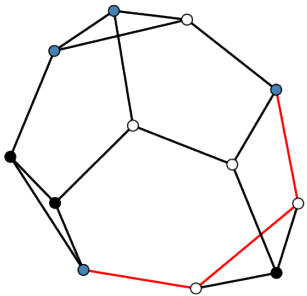


Random Walk in Network

Data Persistence (Markov properties)

Markov chains

- Time-reversible Markov chain
- Ergodic & suffciently long walk
- Steady-state distribution equals limiting distribution

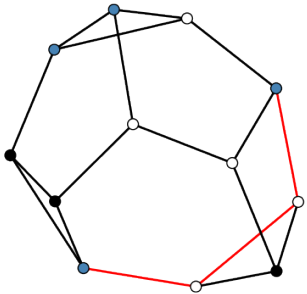


Random Walk in Network

Data Persistence (Walk length)

Minimize walk length

- Long walk, high network overhead
- Minimize walk length
- Preserve steady-state distribution convergence
- Use Metropolis algorithm [3]



Random Walk in Network

[3] (Upfal 2005)

Data Persistence (Random Transition)

Compute transition matrix

- Use Metropolis algorithm
- Use steady state distribution
- Each row needs local information
- Table construction is distributed

$$\pi = (\pi_1, \pi_2, \ldots)$$

Steady State Distribution

$$P_{ij} = \begin{cases} \min(1, \pi_j/\pi_i)/M & \text{if } i \neq j \text{ and } j \in N(i), \\ 0 & \text{if } i \neq j \text{ and } j \notin N(i), \\ 1 - \sum_{j \neq i} P_{ij} & \text{if } i = j. \end{cases}$$

Transition Matrix

Data Persistence (Routing)

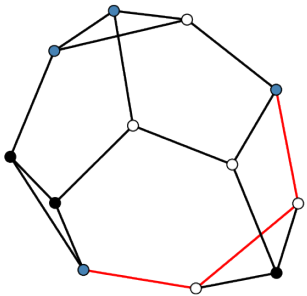
Probabilistic Forwarding Table

- Each transition matrix row corresponds to a node
- Each row element is probability of forwarding data
- Use the row i of transition matrix

Data Persistence (Result)

Resulting combination

- Sources broadcast encoded blocks
- Every node forwards blocks probabilistically
- Random walk length expectation minimized
- Sink transits network, polling for blocks
- Accumulate enough blocks to recover message

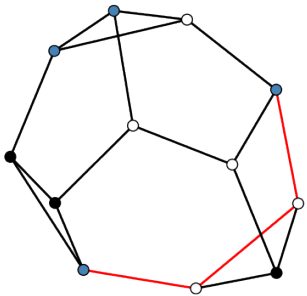


Random Walk in Network

Data Persistence (Other)

Other considerations

- Multiple-random walks for redundancy
- Repeat encoded block broadcasts for redundancy



Random Walk in Network

Algorithm

Algorithm

Each node *independently*:

1. Degree generation
2. Compute steady-state distribution
3. Compute probabilistic forwarding table
4. Compute the number of random walks
5. Block dissemination
6. Encoding & routing
7. GOTO [5]

Algorithm

Each node *independently*:

1. Chooses its code-degree d from the Robust Soliton distribution
2. Computes its steady-state distribution π_d from its chosen code-degree d
3. Computes probabilistic forwarding table T_d by the Metropolis algorithm using π_d
4. If sensor; computes b, the number of random walks from itself
5. If sensor; disseminates b copies of its source block with its node *ID* using T_d
6. Encoded block by (XOR) a subset of d received blocks and forward using T_d
7. GOTO [5]

Algorithm

- Decod using Belief Propagation algorithm
- Similar to LT codes

Results

Theory

$\mathcal{O}(\ln(K/\delta))$

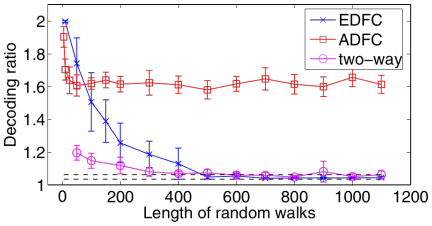
Average degree of an encoded block

$$\frac{\sum_{d=1}^{K} x_d d\mu(d)}{\sum_{d=1}^{K} d\mu(d)}$$

Overhead Ratio

Practice

As walk length increases converges to other codes



Simulated Results

Conclusion

Conclusion

- Treat sensor network as storage device
- Use coding theory and probability theory to define fountain code
- Have sink poll subset of network
- Robust erasure recovery in network with frequent faults

Conclusion

Thank you!

Questions?

Slides are publically available [4]

[4] https://github.com/recursion-ninja/CSc-85020/tree/master/Presentation

References I

Lin, Yunfeng, Ben Liang, and Baochun Li. 2007. "Data Persistence in Large-Scale Sensor Networks with Decentralized Fountain Codes." In *IEEE INFOCOM 2007-26th IEEE International Conference on Computer Communications*, 1658–66. IEEE.

Luby, Michael. 2002. "LT Codes." In *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.*, 271–71. IEEE Computer Society.

Upfal, Michael Mitzenmacher Eli. 2005. "Probability and Computing: Randomized Algorithms And."