Encoding for Persistent Sensor Networks

Abhinav Kamra[1], Jon Feldman[3], Vishal Misra[1,2] and Dan Rubenstein[2,1]

[1]Department of Computer Science

[2]Department of Electrical Engineering
Columbia University in the City of New York

[3]Google Inc.

Allerton Talk - September 29, 2005

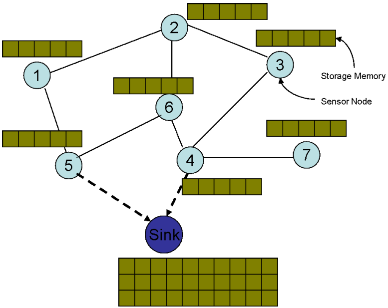
Outline

Outline

Background

Sensor Networks

- A network of low memory, low processing power nodes
- Each node generates some data
- An Access Point node (or Sink node), connected to one or more sensor nodes
- Sensor nodes route generated data to sink node for post-processing



Background (contd.)

Failures in Sensor Networks

- Processor crash
- Accidental Breakage
- Interference attacks
- Collision attacks
- Atmospheric Phenomena
- Natural Disasters

Consequences of Failures

- Network gets partitioned
- Sink may not be able to receive all data

Problem ?

How to receive maximum amount of data at the sink before the network goes down?

Background (contd.)

Failures in Sensor Networks

- Processor crash
- Accidental Breakage
- Interference attacks
- Collision attacks
- Atmospheric Phenomena
- Natural Disasters

Consequences of Failures

- Network gets partitioned
- Sink may not be able to receive all data

Problem ?

How to receive maximum amount of data at the sink before the network goes down?

Background (contd.)

Failures in Sensor Networks

- Processor crash
- Accidental Breakage
- Interference attacks
- Collision attacks
- Atmospheric Phenomena
- Natural Disasters

Consequences of Failures

- Network gets partitioned
- Sink may not be able to receive all data

Problem ?

How to receive maximum amount of data at the sink before the network goes down?

Data Transfer Protocols

Characteristics of a Data Transfer Protocol

- Nodes exchange data with their neighbours
- Nodes are unreliable and may fail at any time
- Nodes may not know in which direction the sink lies
- Nodes are unaware of the size of the network

Is Coding necessary?

- Sensor nodes should cooperate in getting the data across to the sink
- Protocol should be simple: nodes do not have much computational power
- Sink might receive multiple copies of the same data
- Encoding of data required to minimize such wastage

Data Transfer Protocols

Characteristics of a Data Transfer Protocol

- Nodes exchange data with their neighbours
- Nodes are unreliable and may fail at any time
- Nodes may not know in which direction the sink lies
- Nodes are unaware of the size of the network

Is Coding necessary?

- Sensor nodes should cooperate in getting the data across to the sink
- Protocol should be simple: nodes do not have much computational power
- Sink might receive multiple copies of the same data
- Encoding of data required to minimize such wastage

Data Transfer Protocols

Characteristics of a Data Transfer Protocol

- Nodes exchange data with their neighbours
- Nodes are unreliable and may fail at any time
- Nodes may not know in which direction the sink lies
- Nodes are unaware of the size of the network

Is Coding necessary?

- Sensor nodes should cooperate in getting the data across to the sink
- Protocol should be simple: nodes do not have much computational power
- Sink might receive multiple copies of the same data
- Encoding of data required to minimize such wastage

Data Transfer Protocols

Characteristics of a Data Transfer Protocol

- Nodes exchange data with their neighbours
- Nodes are unreliable and may fail at any time
- Nodes may not know in which direction the sink lies
- Nodes are unaware of the size of the network

Is Coding necessary?

- Sensor nodes should cooperate in getting the data across to the sink
- Protocol should be simple: nodes do not have much computational power
- Sink might receive multiple copies of the same data
- Encoding of data required to minimize such wastage

Data Transfer Protocols

Characteristics of a Data Transfer Protocol

- Nodes exchange data with their neighbours
- Nodes are unreliable and may fail at any time
- Nodes may not know in which direction the sink lies
- Nodes are unaware of the size of the network

Is Coding necessary?

- Sensor nodes should cooperate in getting the data across to the sink
- Protocol should be simple: nodes do not have much computational power
- Sink might receive multiple copies of the same data
- Encoding of data required to minimize such wastage

Related Work

Related Coding schemes

- Tornado Codes and LT Codes
  - Useful for Bulk Data Transfer
  - Use XOR based encoding for linear encoding/decoding complexity
- Decentralized Erasure Codes
  - Useful for Distributed Data Storage
  - Use erasure codes for storage of data in network nodes
- Random Linear Codes
  - Also useful for Distributed Data Storage
  - Use linear combinations of data as codes
  - High encoding/decoding complexity

Related Work

Related Coding schemes

- Tornado Codes and LT Codes
  - Useful for Bulk Data Transfer
  - Use XOR based encoding for linear encoding/decoding complexity
- Decentralized Erasure Codes
  - Useful for Distributed Data Storage
  - Use erasure codes for storage of data in network nodes
- Random Linear Codes
  - Also useful for Distributed Data Storage
  - Use linear combinations of data as codes
  - High encoding/decoding complexity

Related Work

Related Coding schemes

- Tornado Codes and LT Codes
  - Useful for Bulk Data Transfer
  - Use XOR based encoding for linear encoding/decoding complexity
- Decentralized Erasure Codes
  - Useful for Distributed Data Storage
  - Use erasure codes for storage of data in network nodes
- Random Linear Codes
  - Also useful for Distributed Data Storage
  - Use linear combinations of data as codes
  - High encoding/decoding complexity

Outline

Data Encoding for Failure-Prone Sensor Networks

Properties of a data encoding protocol in failure-prone sensor networks

- Low encoding/decoding complexity
- Distributed: A sensor node may have information only about its limited storage
- A sensor node may encode data using only its limited storage
- A sensor nodes may exchange encoded data with only its neighbours

Protocols based on XOR codes

Encoding

- Encoded Symbols:
  - Data from all nodes assumed to be of the same size
  - Each encoded symbol is an XOR of 1 or more data units, e.g. $x_1 \oplus x_5 \oplus x_4$
  - Degree of an encoded symbol = Number of components forming the XOR code

Protocols based on XOR codes

Decoding

- Decoding of received symbols:
  - Sink node receives symbols of various degrees
  - Decodes degree 1 symbols first (since these are just the original data units)
  - Any higher degree symbol, for which all but one components have been decoded can now be decoded
  - Follows the previous step iteratively
- Decoding example:
  - Received symbols: x_2, $x_3 \oplus x_5$, $x_1 \oplus x_4$ and x_5
  - Decode degree 1 symbols first: x_2 and x_5 recovered
  - Decode higher degree symbols: $x_3 = (x_3 \oplus x_5) - x_5$
  - Encoded symbol $x_1 \oplus x_4$ is unusable at this point

Protocols based on XOR codes

Decoding

- Decoding of received symbols:
  - Sink node receives symbols of various degrees
  - Decodes degree 1 symbols first (since these are just the original data units)
  - Any higher degree symbol, for which all but one components have been decoded can now be decoded
  - Follows the previous step iteratively
- Decoding example:
  - Received symbols: x_2, $x_3 \oplus x_5$, $x_1 \oplus x_4$ and x_5
  - Decode degree 1 symbols first: x_2 and x_5 recovered
  - Decode higher degree symbols: $x_3 = (x_3 \oplus x_5) - x_5$
  - Encoded symbol $x_1 \oplus x_4$ is unusable at this point

Protocols based on XOR codes

Decoding

- Decoding of received symbols:
  - Sink node receives symbols of various degrees
  - Decodes degree 1 symbols first (since these are just the original data units)
  - Any higher degree symbol, for which all but one components have been decoded can now be decoded
  - Follows the previous step iteratively
- Decoding example:
  - Received symbols: x_2, $x_3 \oplus x_5$, $x_1 \oplus x_4$ and x_5
  - Decode degree 1 symbols first: x_2 and x_5 recovered
  - Decode higher degree symbols: $x_3 = (x_3 \oplus x_5) - x_5$
  - Encoded symbol $x_1 \oplus x_4$ is unusable at this point

Protocols based on XOR codes

Decoding

- Decoding of received symbols:
  - Sink node receives symbols of various degrees
  - Decodes degree 1 symbols first (since these are just the original data units)
  - Any higher degree symbol, for which all but one components have been decoded can now be decoded
  - Follows the previous step iteratively
- Decoding example:
  - Received symbols: x_2, $x_3 \oplus x_5$, $x_1 \oplus x_4$ and x_5
  - Decode degree 1 symbols first: x_2 and x_5 recovered
  - Decode higher degree symbols: $x_3 = (x_3 \oplus x_5) - x_5$
  - Encoded symbol $x_1 \oplus x_4$ is unusable at this point

Degree Distributions of Protocols

- Degree Distribution: The distribution of the fraction of symbols of various degrees generated by a protocol
- π_i = Fraction of degree i symbols

LT Codes

- Use the Soliton distribution
  - $\pi_1 = \frac{1}{N}$
  - $\pi_i = \frac{1}{i(i-1)}$ for $i \in 2, N$
  - In Expectation: One data unit recovered per symbol
  - But variance very high
- Also propose the Robust Soliton distribution
  - Less variance
  - More than N symbols required to decode N data units
- Unencoded distribution is simply $\pi_1 = 1$

Degree Distributions of Protocols

- Soliton and Robust Soliton degree distributions good when more than N encoded symbols can be recovered
- When very few symbols can be recovered: What is a good degree distribution?

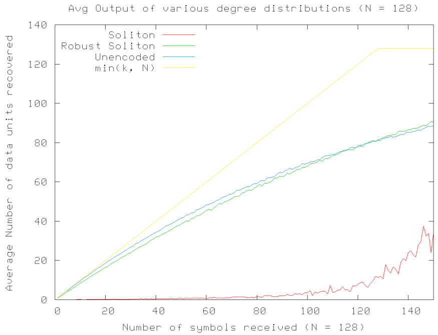
Degree Distributions of Protocols

- Soliton and Robust Soliton degree distributions good when more than N encoded symbols can be recovered
- When very few symbols can be recovered: What is a good degree distribution?

Exploring Optimal Degree Distributions

Some Results

- When very few symbols recovered: Use no coding
- When symbols recovered $\leq$ approx. $\frac{3N}{4}$, use no coding



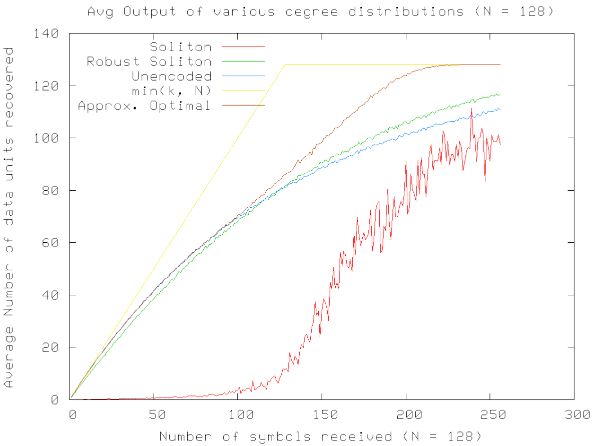
- Unencoded can recover the maximum data when very few symbols recovered
- Robust Soliton works as well

Exploring Optimal Degree Distributions

Some Results

- When symbols recovered $>$ approx. $\frac{3N}{4}$, also use degree 2 symbols
- In particular, define boundaries $K_1, K_2, \ldots, K_{N-1}$ such that
- When symbols recovered is $k : K_{i-1} \leq k < K_i$, use degree i symbols
- K_1 can easily be evaluated from the Coupon Collector's argument
- Upper bounds for $K_2, \ldots, K_{N-1}$ can be obtained

Exploring Optimal Degree Distributions



- Approx. Optimal degree distribution works as unencoded when very few symbols are expected to be recieved
- Switches to higher degrees when more symbols are expected to be received

An Optimal Distributed Data Transfer Protocol

1. A sensor network where every node holds a limited number of encoded symbols in its storage
2. At the start, each node generates a data unit and stores it in all locations of its storage
3. In every round, each node exchanges one of its symbols with a random neighbour
4. In every round, the sink node gets a symbol from one of its neighbouring sensor nodes
5. After round K_{i-1}, all nodes exchange degree i symbols by XORing their own data unit with the exchanged symbol if it is of a smaller degree

In this way, if the network fails at any time, the sink has the best possible degree distribution of the symbols

An Optimal Distributed Data Transfer Protocol

1. A sensor network where every node holds a limited number of encoded symbols in its storage
2. At the start, each node generates a data unit and stores it in all locations of its storage
3. In every round, each node exchanges one of its symbols with a random neighbour
4. In every round, the sink node gets a symbol from one of its neighbouring sensor nodes
5. After round K_{i-1}, all nodes exchange degree i symbols by XORing their own data unit with the exchanged symbol if it is of a smaller degree

In this way, if the network fails at any time, the sink has the best possible degree distribution of the symbols

An Optimal Distributed Data Transfer Protocol

1. A sensor network where every node holds a limited number of encoded symbols in its storage
2. At the start, each node generates a data unit and stores it in all locations of its storage
3. In every round, each node exchanges one of its symbols with a random neighbour
4. In every round, the sink node gets a symbol from one of its neighbouring sensor nodes
5. After round K_{i-1}, all nodes exchange degree i symbols by XORing their own data unit with the exchanged symbol if it is of a smaller degree

In this way, if the network fails at any time, the sink has the best possible degree distribution of the symbols

An Optimal Distributed Data Transfer Protocol

1. A sensor network where every node holds a limited number of encoded symbols in its storage
2. At the start, each node generates a data unit and stores it in all locations of its storage
3. In every round, each node exchanges one of its symbols with a random neighbour
4. In every round, the sink node gets a symbol from one of its neighbouring sensor nodes
5. After round K_{i-1}, all nodes exchange degree i symbols by XORing their own data unit with the exchanged symbol if it is of a smaller degree

In this way, if the network fails at any time, the sink has the best possible degree distribution of the symbols

An Optimal Distributed Data Transfer Protocol

1. A sensor network where every node holds a limited number of encoded symbols in its storage
2. At the start, each node generates a data unit and stores it in all locations of its storage
3. In every round, each node exchanges one of its symbols with a random neighbour
4. In every round, the sink node gets a symbol from one of its neighbouring sensor nodes
5. After round K_{i-1}, all nodes exchange degree i symbols by XORing their own data unit with the exchanged symbol if it is of a smaller degree

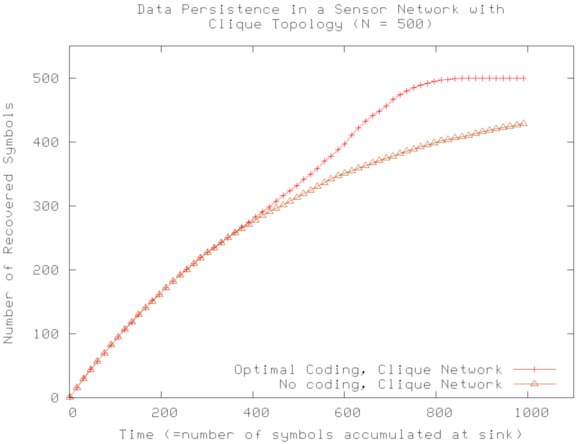
In this way, if the network fails at any time, the sink has the best possible degree distribution of the symbols

An Optimal Distributed Data Transfer Protocol

1. A sensor network where every node holds a limited number of encoded symbols in its storage
2. At the start, each node generates a data unit and stores it in all locations of its storage
3. In every round, each node exchanges one of its symbols with a random neighbour
4. In every round, the sink node gets a symbol from one of its neighbouring sensor nodes
5. After round K_{i-1}, all nodes exchange degree i symbols by XORing their own data unit with the exchanged symbol if it is of a smaller degree

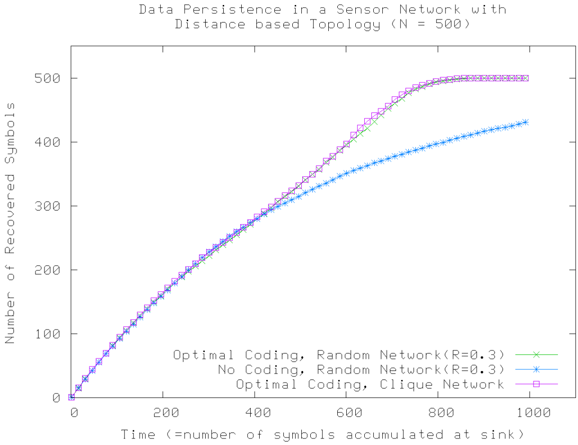
In this way, if the network fails at any time, the sink has the best possible degree distribution of the symbols

Evaluating the Optimal Distributed Data Transfer Protocol: Clique Topology



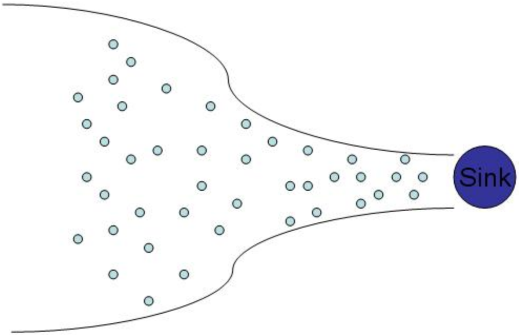
- A Clique network of 500 sensor nodes
- Sink attached to any 1 node, so receives one new encoded symbol per round

Evaluating the Optimal Distributed Data Transfer Protocol: Random Topology



- 500 nodes randomly placed in 1x1 square
- 2 nodes are neighbours if distance ≤ 0.3
- Sink attached to 1 random node, so receives one new encoded symbol per round

Countering the Funnel Effect



- Proposed protocol handles well a sudden burst of information throughout the sensor network
- Congestion at the sink (Funnel Effect)
- Nodes in the periphery exchange data amongst themselves