Fault Tolerant Network Coding

Ravishankar Krishnaswamy
ravishankar.k@gmail.com
Guide: Pandurangan Chandrasekaran
rangan@shiva.cs.iitm.ernet.in
Indian Institute of Technology Madras

May 15, 2007

Abstract

Consider a communication network in which a source node wishes to multicast information to some sink nodes on the network. In the traditional setting, every node can only pass on the data it receives from incoming links to the links leaving it. We consider an extended setting which gives the intermediate nodes more "power". In our model, each node may send any *linear combination* of the received data on the outgoing links. Such protocols, known as *Linear Network Coding* schemes, have been proved to guarantee optimal multicast throughput (that is, the maximum number of different data packets the source can transmit to all the sinks in one execution of the protocol matches the theoretical bound). In this work, we address the problem of fault tolerance in network coding. More specifically, we are interested in obtaining reliable coding schemes which guarantee optimal (in a sense we shall explain later) throughput even when some edges stop functioning, or are corrupt by noise.

We modify an existing algorithm to provide a centralized scheme for finding such codes that tolerate any failure pattern from a polynomially bounded set of possible failure patterns in a network. A failure pattern is a set of edges that fail (transmits a zero message). Our algorithm is rate optimal in the asymptotic limit with respect to the message packet size, in the sense that it sustains a flow rate equal to the minimum of the *max-flows* from the source to the sinks with the failed edges being deleted. Such codes could be utilized in networks where certain edges are prone to failure, and yet optimal throughput is expected regardless of the edges failing. We also present simple extensions which optimally tolerate corruption of edges by white noise (they transmit a purely random message).

1 Introduction

The concept of *Network Coding* was first introducted in the ladmark paper [1] bt Ahlswede et al. The authors prove bounds on the capacity of network coding and show that it is possible to obtain optimal throughput using network coding. The implication in a multicast setting is that one can achieve a throughput equal to the minimum of the *max flows* to each of the sinks - the theoretical upper bound on network throughput. In [2], Jaggi et al obtain a polynomial centralized algorithm that identifies the throughput maximizing linear network code for a directed acyclic graph.

Our Focus: We primarily focus on the *robustness* (tolerance to the failure/corruption of certain edges) of network coding. We are interested in efficiently designing network codes that tolerate polynomially many failure patterns. A similar setting is also considered by [2] and [3], but they assume that the sinks have knowledge about the edges that failed. This assumption, however, may not be valid in real-life networks. To the best of our knowledge, this is the first centralized algorithm for network coding when edges fail, without any assumptions about apriori information of the failures.

2 Preliminaries

All networks considered are directed acyclic graphs $G = (V, E)$. Let $s \in V$ be the source node and $T = \{t_1, t_2, ..., t_{|T|}\} \subseteq V \setminus \{s\}$ be the set of sinks. The source s wishes to multicast h elements $x_1, x_2, \ldots, x_h$ where $x_i \in \mathbb{F}_{q^l}$. Each edge $e \in E$ can transmit a single *packet* of information. If an edge were to have a higher capacity k, we could replace it with k links of unit capacity. In linear network coding, the message m transmitted on any edge e is a linear combination of the messages in the incoming edges (and therefore a linear combination of the original message packets). In our framework, a packet p that is transmitted on an edge e is an $h+1$ tuple $(m, b_1, b_2, \ldots, b_h)$ where $m \in \mathbb{F}_{q^l}$ and $b_j \in \mathbb{F}_q$. The vector $b(e) = (b_1, b_2, \ldots, b_h)$ is called the *global encoding vector* of edge e. Also $b(e)$ is such that $m = \Sigma_{i=1}^{h} b_i x_i$.

3 Tolerance to Edge Failures - Results

This section deals with network codes that can tolerate failure of some edges. We first consider codes that tolerate single edge failures, and later extend it. We now provide an upper bound on the throughput an algorithm can achieve when some edges can fail when network coding is and is not employed.

Upper Bound:
Consider a network with one source and two sinks and n intermediate nodes. The source is connected to the n intermediate nodes, each of which has an outgoing arc to both the sinks. We classify the edges connecting the source and the intermediate nodes to be in *level 1*, and the edges connecting the intermediate nodes to the sinks to be in *level 2*. Clearly, the minimum of the *max-flows* from the source to each of the sinks in this setting is n. When we wish to tolerate an edge that can fail, we notice that this value drops to $n-1$. This is the bound that our routing protocols should try to achieve. If the network nodes do not employ network coding, we see that a rate of $n-1$ cannot be achieved. This is because, if the source were to send $n-1$ packets of information over to the intermediate nodes, it could repeat only one of the packets in the remaining *level 1* edge. If $n > 1$ and a *level 1* edge not carrying a packet that is repeated fails, there is no way by which that information could be retrieved. On the other hand, with network coding, the source could send the $n-1$ information packets $x_1, x_2, \ldots, x_{n-1}$ along the first $n-1$ *level 1* edges to the $n-1$ intermediate nodes and the packet $x_1 + x_2 + \ldots + x_{n-1}$ to the last intermediate node. Note that any $n-1$ out of the n vectors that are transmitted along *level 1* edges are *linearly independent*. Therefore, even when any one edge fails, the sink will definitely receive $n-1$ linearly independent equations in the variables $x_1, x_2, \ldots, x_{n-1}$. A similar argument works when upto k edges can fail in the network, and we can show that without network coding, the maximum possible throughput is $\lfloor n/(k+1) \rfloor$ - each message needs $k+1$ redundancies to be transmitted.

Lower Bound: Let G have a max flow of n from the source to each of the sinks.

Theorem 3.1 *Given any graph G whose minimum max flow from s to the sinks is n, there exists a centralized network coding scheme that guarantees a throughput of $n-1$ under any single edge failure, which is optimal in the asymptotic limit w.r.t message content size, i.e $l \to \infty$.*

```
1. Traverse the edges in the topological order $e_1$ to $e_{|E|}$. It is to be noted that when traversing $e_i$, all incoming edges to $head(e_i)$ have already been traversed. Let $e_j$ be the edge that is currently traversed. Perform steps 2 to 19
2.     For $k = 0$ to $j - 1$
3.         Initialize the list $U = \phi$.
4.         For $t \in sinks(e_j)$
5.             Obtain from the $b - a$ table for sink $t$, all values of $(b_i, a_j)$ such that $b_i = carried(prev(e_j, t), k)$ and add the $(b_i, a_j)$ tuple to the list $U$.
6.         End For
7.         Remove all redundant entries from $U$.
8.         Using the replacement vector algorithm ([2]), find a vector $b$ such that $b.a_j \neq 0$ for all $a_j$ such that $(b_i, a_j) \in U$.
9.         Set $carried(e_j, k) = b$.
10.        For $t \in sinks(e_j)$
11.            Mark $B_t^k$ for updating to $B_t^k \setminus \{carried(prev(e_j, t), k)\} \cup \{b\}$. Also run the pseudo-inverse procedure ([2]) to obtain the changes to be done to the $b - a$ table. Mark these changes as well.
12.        End For
13.    End For
14.    Set $carried(e_j, j) = 0$.
15.    For $t \in sinks(e_j)$
16.        Set $B_t^k = B_t^k \setminus \{carried(prev(e_j, t), k)\} \cup \{0\}$.
17.        Update the $j^{th}$ group of the $b - a$ table such that only the $b, a$ values corresponding to the non-zero $b$ vectors in $B_t^k$ remain (there will be $n - 1$ such vectors).
18.    End For
19.    Update all the changes in the sets $C_t$, $B_t^k$ and the $b - a$ table that were marked for updating in step 11.
20. End Traversal
```

Sketch: We modify the algorithm presented in [2] to accommodate for the edge failure. The basic idea is that we maintain the set of n global encoding vectors for each sink such that any $n-1$ of them are linearly independent. Therefore, even if 1 edge fails, the sink receives $h = n-1$ linearly independent equations from which it can identify the sender's information. Though the way we assign the global encoding vector *carried*(e, k) makes it appear that it depends on the edge e_k that has failed, we note that the global encoding vector is actually just a function of the incoming packets' global encoding vectors. This is so because, for two different sets of incoming global encoding vectors $\{b_1, b_2, \ldots, b_i\}$ and $\{b'_1, b'_2, \ldots, b'_i\}$, where i is the in-degree of the current node we are considering, the list U that is chosen would be the same if $b_j = b'_j$. Therefore, the outgoing global encoding vector would also be the same, regardless of which edge has failed. The fact that the above procedure maintains the invariant mentioned in 3 is because of step 8

Polynomial Edge Failure Patterns: We now extend the above result to an algorithm allowing failure in any one of polynomially many edge patterns.

Theorem 3.2 *Given any graph G, and a polynomial set Err of failure patterns, suppose the minimum max flow under any of the possible failure patterns to the sinks is h, then there exists a centralized network coding scheme that guarantees a throughput of h under any edge failure pattern, which is optimal in the asymptotic limit.*

The basic idea is to generate a set of n vectors such that any h of them are linearly independent. We observe that a purely random generation of n vectors satisfies this requirement with sufficient probability so long as the field size is large. We then run the centralized algorithm trying to maintain one set of h linearly independent vectors for every failure pattern.

Noisy Edges: To handle a scenario where a failed edge transmits a perfectly random message (instead of $\mathbf{0}$), we add some redundancy to the message packet transmitted. If we modify the transmitted packet on an edge e to an $h+2$ tuple $(m, b_1, b_1, b_2, \ldots, b_h)$ instead of the $h+1$ tuple $(m, b_1, b_2, \ldots, b_h)$, we could detect white noise on edges with probability $1/q$ by performing the check that the 2nd and 3rd elements of the received packet are the same. We then treat noise edges as 0, and perform the old algorithm assuming a 0 for such noise-detected edges.

References

[1] R. Ahlswede, N. Cai, S-Y.R. Li, and R.W. Yeung. Network information flow. IEEE Transcations on Information Theory, vol. 46, no. 4, 2000.

[2] S. Jaggi, P. Sanders, P.A. Chou, M. Effros, S. Egner, K. Jain, and L. Tolhuizen. Polynomial time algorithms for multicast network code construction. IEEE Trans. Info. Theory, vol. 51, no. 6, 2005.

[3] R. Koetter and M. Medard. Beyond routing: An algebraic approach to network coding. 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOMM), volume 1, pages 122130, 2002.