Bottleneck Discovery and Overlay Management in Network Coded Peer-to-Peer Systems

Mahdi Jafarisiavoshani
EPFL
Switzerland
mahdi.jafari@epfl.ch

Christina Fragouli *
EPFL
Switzerland
christina.fragouli@epfl.ch

Suhas Diggavi
EPFL
Switzerland
suhas.diggavi@epfl.ch

Christos Gkantsidis
Microsoft Research
United Kingdom
chrisgk@microsoft.com

ABSTRACT

The performance of peer-to-peer (P2P) networks depends critically on the good connectivity of the overlay topology. In this paper we study P2P networks for content distribution (such as Avalanche) that use randomized network coding techniques. The basic idea of such systems is that peers randomly combine and exchange linear combinations of the source packets. A header appended to each packet specifies the linear combination that the packet carries. In this paper we show that the linear combinations a node receives from its neighbors reveal structural information about the network. We propose algorithms to utilize this observation for topology management to avoid bottlenecks and clustering in network-coded P2P systems. Our approach is decentralized, inherently adapts to the network topology, and reduces substantially the number of topology rewirings that are necessary to maintain a well connected overlay; moreover, it is integrated in the normal content distribution. This work demonstrates another advantage of using network coding and complements previous work that showed network coding achieves high network-resource utilization.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Algorithms, Management, Measurement

Keywords

Peer-to-Peer Networks, Network Coding, Topology Management, Network Algorithms

*This work was in part supported by the Swiss National Science Foundation under award No PP002–110483.

INM'07, August 27–31, 2007, Kyoto, Japan.

1. INTRODUCTION

Peer-to-peer (P2P) networks have proved a very successful distributed architecture for content distribution. The design philosophy of such systems is to delegate the distribution task to the participating nodes (peers) themselves, rather than concentrating it to a low number of servers with limited resources. Therefore, such a P2P non-hierarchical approach is inherently scalable, since it exploits the computing power and bandwidth of all the participants.

Having addressed the problem of ensuring sufficient network resources, P2P systems still face the challenge of how to efficiently utilize these resources while maintaining a decentralized operation. Central to this is the challenging management problem of connecting the peers in a way that ensures the fastest possible information dissemination. The goal is to create a network topology that efficiently uses the available bandwidth with minimal or no centralized coordination, in a manner scalable to the network size.

The main P2P network solutions (*e.g.*, Gnutella, Freenet, Napster, BitTorrent) effectively build complex topologies, such as rings, or meshes of multicast trees.These approaches have the disadvantage of imposing topological constraints on the bandwidth usage, *e.g.*, embedding trees in the existing network structure. Such a problem is hard to solve optimally even with perfect centralized control.

An alternate design philosophy is embodied by network coded P2P systems such as Avalanche [2, 3]: it employs randomized network coding, where peers randomly combine their received packets and propagate such linear combinations to their neighbors. A peer receiving a sufficient number of linear combinations solves a system of linear equations and retrieves the source packets. Thus peers no longer collect packets routed through pre-specified paths from the source, but instead attempt to collect and propagate "degrees of freedom" (linear combinations). This approach dispenses with the need of creating and preserving specific routing structures. In fact it can be shown that the optimal routing problem becomes polynomial-time. Moreover, there is potential for significant gains in network throughput making network coding a very promising paradigm for P2P systems as has been demonstrated in the Avalanche system. However, to optimize the use of these resources, we still need to build a network topology that allows the fastest possible dissemination of information in a scalable manner.

In both systems, the task of topology management is hindered by the fact that the peers need to form overlay-network connections using underlying physical links whose available bandwidth is hard to estimate. In this paper, we will argue that for P2P networks

employing randomized network coding, we can use the structure of the exchanged packets to passively infer physical-link bottlenecks. In particular, the packets a node observes from its neighbors contain implicit information about the underlying physical-link bandwidths. The specific application of this observation is to use such information to break clusters, using the minimum required number of topology rewirings. Topology rewirings could be costly in P2P networks, since each rewiring of the topology may need to be accompanied by an authentication and set-up of a security protocol (as done in Avalanche). This could incur significant delay and hence motivating methods that seek to reduce the rewirings without affecting dissemination rates. We propose algorithms that (i) identify and re-connect only nodes whose re-wiring leads to breaking of clusters, (ii) use a variable number of reconnections, adapting (without centralized knowledge) to the requirements of the network topology, and (iii) are peer-initiated and scalable.

In Section 2, we briefly discuss Avalanche, illustrate our approach with a simple example, and put our work in context.We then develop a theoretical framework in Section 3, propose algorithms in Section 4 and present simulation results in Section 5. Section 6 concludes the paper.

2. DESCRIPTION AND MOTIVATION

We start by first briefly reviewing randomized network coding, and describing how Avalanche builds the overlay topology employed for content distribution. We then motivate our work through an example, and discuss related work.

2.1 Randomized Network Coding

Consider a source with n packets to be distributed to a set of nodes. In randomized network coding [4], each packet is assumed to consist of L symbols over a finite field $\mathbb{F}_q$. This simply means that, for $q = 2^m$, a packet of length mL bits is considered to contain L symbols over $\mathbb{F}_{2^m}$, where sets of of m bits are treated as one symbol over $\mathbb{F}_{2^m}$. Nodes in the network perform operations over $\mathbb{F}_q$. The source sends "coded packets", each packet consisting of a uniform at random chosen linear combination of the n packets over $\mathbb{F}_q$. Each packet has an appended *coding vector* of size n symbols over $\mathbb{F}_q$, that specifies which linear combination each coded packet carries. Note that for large L (mL bit packets), the overhead of n symbols (nm bits) can be made very small. Intermediate nodes in the network recursively and uniformly at random combine their collected packets and create new coded packets that they propagate through the network. Note that the coding vectors a node can create have to belong in the span[1] of the coding vectors it has already collected. Once a node has collected a set of coding vectors that spans the n-dimensional space, it has enough information to solve a system of linear equations and retrieve the source packets.

2.2 Avalanche Topology Management

In a nutshell, Avalanche relies on periodically renewed random selections for the peer neighbors to rewire the employed overlay network [2,3]. In more detail, the source forms the first node of the overlay network that will be used for the file distribution. All nodes in this network are connected to a small number of neighbors (four to eight). Neighbors for each arriving node are chosen uniformly at random among already participating nodes, which accept the solicited connection unless they have already reached their maximum number of neighbors. Each node keeps local topological information, namely, the identity of the neighbors it is directly connected to. A special node called registrat keeps track of the list of active peers. Nodes periodically drop one neighbor and reconnect to a new one, asking the registrat to randomly select the new neighbor from the active peers list.

[1]The vectors $v_1, \ldots, v_n$ span an n-dimensional space if they form a basis of this space. Their span, is the set of vectors that are *all* linear combinations of them.

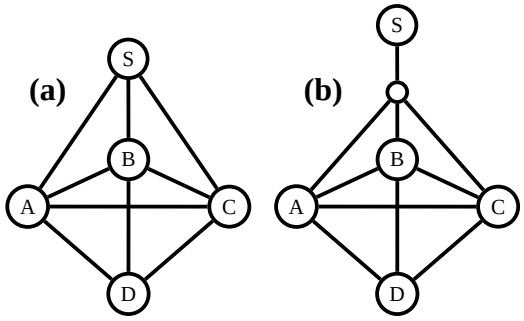


Figure 1: The source S distributes packets to the peers A, B, C and D over the overlay network (a), that uses the underlying physical network (b).

The randomized rewiring Avalanche employs results in a fixed average number of reconnections per node independently of how good or bad is the formed network topology. Thus to achieve a good, on the average, performance in terms of breaking clusters, it entails a much larger number of rewiring and requests to the registrat than required, and unnecessary topology changes.

Clearly the registrat, since it allocates to each peer its neighbors, could keep some structural information, *i.e.*, keep track of the current network topology, and use it to make more educated choices of neighbor allocations. However, the information the registrat can collect only reflects the *overlay* network topology, and is oblivious to bandwidth constraints from the underlying physical links. Acquiring bandwidth information for the underlying physical links at the registrat requires costly estimation techniques over large and heterogeneous networks, and steers towards a centralized network operation. We argue that such bottlenecks can be inferred passively, thus avoiding these drawbacks.

2.3 Our Approach

Our work starts from the observation that the coding vectors the peers receive from their neighbors can be used to passively infer bottleneck information. This allows individual nodes to initiate topology changes to correct problematic connections. In particular, peers by keeping track of the coding vectors they receive can detect problems in both the overlay topology and the underlying physical links. The following example illustrates these points.

Consider the toy network depicted in Fig. 1(a) where the edges correspond to logical (overlay network) links. The source S has n packets to distribute to four peers. Nodes A, B and C are directly connected to the source S, and also among themselves with logical links, while node D is connected to nodes A, B and C. In this overlay network, each node has constant degree three (three neighbors), and there exist three edge-disjoint paths between any pair of nodes (in particular, between the source and any other node).

Assume now (as shown in Fig. 1(b)) that the logical links SA, SB, SC share the bandwidth of the same underlying physical link, which forms a bottleneck between the source and the remaining nodes of the network. As a result, assume the bandwidth on each of these links is only $1/3$ of the bandwidth of the remaining links. The registrat, even if it keeps track of the complete logical network structure, is oblivious to the existence of the bottleneck and the asymmetry between the link bandwidths.

Node D however, can infer this information by observing the coding vectors it receives from its neighbors A, B and C. Indeed, when node A receives a coded packet from the source, it will forward a linear combination of the packets it has already collected to nodes B and C and D. Now each of the nodes B and C, once they receive the packet from node A, they also attempt to send a coded packet to node D. But these packets will not bring new information to node D, because they will belong in the linear span of coding vectors that node D has already received. Similarly, when nodes B and C receive a new packet from the source, node D will end up being offered three coded packets, one from each of its neighbors, and only one of the three will bring to node D new information.

More formally, the coding vectors nodes A, B and C will collect will effectively span the same subspace; thus the coded packets they will offer to node D to download will belong in significantly overlapping subspaces and will thus be redundant (we formalize these intuitive arguments in Section 3). Node D can infer from this passively collected information that there is a bottleneck between nodes A, B, C and the source, and can thus initiate a connection change.

2.4 Related Work

Overlay topology monitoring and management that do not employ network coding has been an intensively studied research topic, see for example [5]. Our proposed approach applies specifically to systems employing network coding.

We have initiated work on taking advantage of the network coding capabilities for active network monitoring in [6, 7] where the focus was on link loss rate inference. Passive inference of link loss rates has also been proposed in [8]. However, the idea of passive inference of topological properties is a novel contribution of this paper.

3. THEORETICAL FRAMEWORK

We start by formally introducing some notation. We assume that the source has n packets, each with independent information, to distribute to a set of nodes. We can think of each packet as corresponding to one dimension of an n-dimensional space, over a finite field $\mathbb{F}_q$. We thus associate with each packet one of the orthonormal basis vectors $\{e_1, \ldots, e_n\}$, where e_i is the n-dimensional vector with one at position i and zero elsewhere. Each packet has an associated n-dimensional coding vector attached to it.

We say that node j at time t observes a subspace Π_j, where Π_j is the vector space spanned by the coding vectors node j has received up to time t. Initially at time $t = 0$, Π_j is empty. If node j receives k linearly independent coding vectors, then

$$\dim(\Pi_j) = k.$$

A coded packet brings *innovative information* if the associated coding vector does not belong in Π_j (it increases the $\dim(\Pi_j)$ by one). When $\dim(\Pi_j) = n$, node j has collected a basis of the n-dimensional space and can decode the source information. To compare subspaces, we will denote

1. the dimension of each subspace as
$$d_i = \dim(\Pi_i), \quad \forall\ i,$$
2. the dimension of the intersection of two subspaces as
$$d_{ij} = \dim(\Pi_i \cap \Pi_j), \quad \forall\ i, j,$$
3. the distance between two subspaces as
$$D_{ij} = \dim(\Pi_i \cup \Pi_j) - \dim(\Pi_i \cap \Pi_j), \quad \forall\ i, j.$$

As also observed in [9], the distance D_{ij} defines a metric space and can be used to compare how "different" the subspaces are. In some occasions we will also need a measure that compares how the subspaces of one cluster of nodes $\mathcal{A}$ differ from the subspaces of another cluster of nodes $\mathcal{B}$. For this we will use the average pairwise distance

$$D_{\mathcal{AB}} = \frac{1}{|\mathcal{A}||\mathcal{B}|} \sum_{i \in \mathcal{A}, j \in \mathcal{B}} D_{ij}. \quad (1)$$

Note that the computation of distances d_i, d_{ij}, D_{ij} and $D_{\mathcal{AB}}$ can be reduced to calculating ranks of certain associated matrices. This can be done efficiently over finite fields using lattice reduction algorithms [11].

For simplicity we will assume that the network is synchronous. By this we mean that nodes transmit and receive according to a global clock tick[2]. Nodes are allowed to transmit linear combinations of their received packets only at clock ticks, at a rate equal to the adjacent link bandwidth. We normalize the transmitted rates so that the *maximum* rate a node can transmit is 1 packet per timeslot in *each* of its outgoing edges. A node transmitting information at a rate $\frac{1}{k}$ on an outgoing link, sends one coded packet every k clock ticks.

We will explore in this paper the topological information revealed by the coding vectors at each node. We distinguish two cases, depending on what information we are allowed to use.

- *Local Information:* at a given time t, each node j knows its own subspace, and the subspaces it has received from its parent nodes. This is the case we will examine in this paper.
- *Global Information:* at a given time t, we know the subspaces that all nodes in the network have observed. This is the maximum information we can hope to get from the coding vectors propagated through the network, and is studied in a companion paper [10]. Here we would like to briefly mention that, for a directed network where information only flows from parent to child nodes, and under some mild conditions, knowledge of the subspaces of all nodes in the network, allows to uniquely determine how the network nodes are connected. This is a surprising result, indicating that the topological information carried from the subspaces is in fact quite significant.

In the following, we will also use the notion of min-cut values. Let $\mathcal{A}$ and $\mathcal{B}$ denote two disjoint sets of vertices of the graph. A cut value is defined as the sum of the capacities of a set of edges we need to remove to disconnect sets $\mathcal{A}$ and $\mathcal{B}$. The min-cut is the minimum cut value. The celebrated min-cut max-flow theorem states that if the min-cut value between two nodes equals c, then the maximum information rate we can convey from one to another also equals c.

3.1 Local Information

Let $\Pi_i = \hat{\Pi}_1 \cup \ldots \cup \hat{\Pi}_c$ denote the subspace spanned by the coding vectors a node i has collected at a fixed time instance, where $\hat{\Pi}_1, \ldots, \hat{\Pi}_c$ denote the subspaces that it has received from its c neighbors $u_1, \ldots, u_c$. We are interested in understanding what information we can infer from these received subspaces $\hat{\Pi}_1, \ldots, \hat{\Pi}_c$. For example, the overlap of subspaces from the neighbors reveals something about a bottleneck. Therefore, we need to show that such overlaps occur due to topological properties and not due to particular random linear combinations chosen by the network code.

[2]This is not essential for the algorithms but simplify the theoretical analysis.

The following lemmas present some properties that the subspaces observed by the network nodes and the rates at which their size increases, need obey.

LEMMA 1. *Let Π_k be a k-dimensional subspace of the vector space $\mathbb{F}_q^n$, i.e., $d_k = k$. Construct the subspace Π_m by selecting $m \leq n$ vectors $\{w_1, \ldots, w_m\}$ uniformly at random from $\mathbb{F}_q^n$. Under the assumption that $q \gg 1$ it holds*[3] *that*

$$\Pr[d_m = m] \approx 1, \text{ and}$$
$$\Pr[d_{km} = d] \approx \begin{cases} 1 & \text{if } d = (m-(n-k))^+, \\ 0 & \text{otherwise.} \end{cases}$$

Proof. Let $\Pi_k^\perp$ be the subspace with the property $\Pi_k \cup \Pi_k^\perp = \mathbb{F}_q^n$, and $U = \{u_1, \ldots, u_k\}$ and $V = \{v_1, \ldots, v_{n-k}\}$ be sets of basis vectors for Π_k and $\Pi_k^\perp$ respectively. We can then expand the vectors $\{w_1, \ldots, w_m\}$ as

$$w_i = \sum_{j=1}^{k} \alpha_j^{(i)} u_j + \sum_{j=k+1}^{n} \alpha_j^{(i)} v_{j-k}, \qquad i = 1, \ldots, m.$$

Let A be the $n \times m$ matrix with columns the vectors $\alpha^{(j)}$, and denote by $\widetilde{U}_{k\times m}$, $\widetilde{V}_{(n-k)\times m}$ the sub matrices of A collecting the coefficients with the respect to U and V:

$$A = \begin{bmatrix} | & & | \\ \alpha^{(1)} & \cdots & \alpha^{(m)} \\ | & & | \end{bmatrix} = \left[\begin{array}{c} \widetilde{U}_{k\times m} \\ \hline \widetilde{V}_{(n-k)\times m} \end{array}\right].$$

For $q \gg 1$, the matrix A is full rank with probability approaching one, and thus $\Pr[d_m = m] \approx 1$ (see also [4]). To calculate $d_{km} = \dim(\Pi_k \cap \Pi_m)$, note that the vectors a that belong in $\Pi_k \cap \Pi_m$ satisfy the equation

$$\left[\begin{array}{c} \widetilde{U}_{k\times m} \\ \hline \widetilde{V}_{(n-k)\times m} \end{array}\right] \cdot \begin{bmatrix} a_1 \\ \vdots \\ a_m \end{bmatrix} = \left[\begin{array}{c} b_1 \\ \vdots \\ b_k \\ \hline 0_{(n-k)\times 1} \end{array}\right],$$

and thus belong in the kernel (null space) of the matrix $\widetilde{V}_{(n-k)\times m}$. For $q >> 1$ this matrix is full rank with high probability. As a result,

$$\begin{aligned} \dim(\text{Kernel}(\widetilde{V}_{(n-k)\times m})) &= m - \text{Rank}(\widetilde{V}_{(n-k)\times m}) \\ &\approx m - \min(m, n-k) \\ &= (m-(n-k))^+. \qquad \square \end{aligned}$$

LEMMA 2. *Let Π_i and Π_j be two subspaces of $\mathbb{F}_q^n$ with dimension d_i and d_j respectively, and intersection $\Pi_{ij} = \Pi_i \cap \Pi_j$ of size d_{ij}. Construct $\hat{\Pi}_i$ and $\hat{\Pi}_j$ by choosing $m_i \leq d_i$ and $m_j \leq d_j$ vectors uniformly at random from Π_i and Π_j respectively. Then the size of their intersection $\hat{d}_{ij} = \dim(\hat{\Pi}_i \cap \hat{\Pi}_j)$ satisfies*

$$\Pr[\hat{d}_{ij} = d] \approx \begin{cases} 1 & d = (m_i + m_j - (d_i + d_j - d_{ij}))^+, \\ 0 & otherwise, \end{cases}$$

Proof. The proof follows by applying the previous lemma twice, once on $\hat{\Pi}_i$ and once on $\hat{\Pi}_j$. We denote $\Pi = \hat{\Pi}_i \cap \Pi_{ij}$.

[3]The $\approx$ notation means that this is true with probability 1 as $q \to \infty$.

1. From Lemma 1, $\hat{\Pi}_i$ has dimension m_i w.h.p., and its intersection with Π_{ij} has dimension $d = (m_i - (d_i - d_{ij}))^+$. 2. From Lemma 1, $\hat{\Pi}_j$ has dimension m_j w.h.p., and its intersection with the subspace Π has dimension

$$\hat{d}_{ij} = (m_j - (d_j - d))^+ = (m_i + m_j - (d_i + d_j - d_{ij}))^+. \square$$

Lemma 3 can be proved following a very similar approach to the main theorem in network coding [1].

LEMMA 3. *In a synchronous network where the min-cut to a node i is c, after a transition phase of the network, node i receives c innovative packets per time slot from its neighbors.*

Assume for example that the subspaces $\hat{\Pi}_1, \ldots, \hat{\Pi}_c$ a node i receives from its set of neighbors $\{u_i\}$ have an intersection of dimension d. This implies that, (i) from Lemma 3, the min-cut between the nodes $\{u_i\}$ and the source is smaller than the min-cut between the node i and $\{u_i\}$, and (ii), from Lemma 2, the subspaces $\Pi_1, \ldots, \Pi_c$ of the neighbors have an intersection of size at least d. Next we will discuss algorithms that use such observations for decentralized topology management.

4. ALGORITHMS

Our peer-initiated algorithms for topology management consist of three tasks:

1. Each peer decides whether it is satisfied with its connection or not, using a *decision criterion*.
2. An unsatisfied peer sends a *rewiring request*, that can contain different levels of information, either directly to the registrat, or to its neighbors (these are the only nodes the peer can communicate with).
3. Finally, the registrat, having received rewiring requests, *allocates neighbors* to nodes to be reconnected.

The decision criterion can capitalize on the fact that overlapping received subspaces indicate an opportunity for improvement. For example, a node can decide it is not satisfied with a particular neighbor, if it receives $k > 0$, non-innovative coding vectors from it, where k is a parameter to be decided. The first algorithm we propose (**Algorithm 1**) uses this decision criterion; it then has each unsatisfied node directly contact the registrat and specify the neighbor it would like to change. The registrat randomly selects a new neighbor. This algorithm, as we demonstrate through simulation results, may lead to more rewirings than necessary: indeed, all nodes inside a cluster may attempt to change their neighbors, while it would have been sufficient for a fraction of them to do so.

Our second algorithm (**Algorithm 2**) uses a different decision criterion: for every two neighbors i and j, each peer computes the rate at which the received joint space $\hat{\Pi}_i \cup \hat{\Pi}_j$ and intersection space $\hat{\Pi}_i \cap \hat{\Pi}_j$ increases. If the ratio between these two rates becomes greater than a threshold $\mathcal{T}$, the node decides it would like to change one of the two neighbors. However, instead of directly contacting the registrat, it uses a decentralized voting method that attempts to further reduce the number of re-connections. A node unsatisfied with a particular neighbor sends a request to this neighbor indicating so. Every node collects all such requests it receives, and only after it collects a certain number Δ of them, it sends a request to the registrat requesting to be rewired. The registrat then randomly selects and allocates one new neighbor.

Our last proposed algorithm (**Algorithm 3**), while still peer-initiated and decentralized, relies more than the two previous ones in the computational capabilities of the registrat, and is specifically

targeted to breaking topological clusters. The basic observation is that, nodes in the same cluster will not only receive overlapping subspaces from their parents, but moreover, they will end up collecting subspaces with very small distance (this follows from Lemmas 1-3 and is also illustrated through simulation results in Section 5). Each unsatisfied peer i sends a rewiring request to the registrat, indicating to the registrat the subspace Π_i it has collected. A peer can decide it is not satisfied using for example the same criterion as in Algorithm 2.

The registrat waits for a short time period, to collect requests from a number of dissatisfied nodes. These are the nodes of the network that have detected they are inside clusters. It then calculates the distance between the identified subspaces to decide which peers belong in the same cluster. While exact such calculations can be computationally demanding, in practice, the registrat can use one of the many hashing algorithms to efficiently do so. Finally the registrat breaks the clusters by rewiring a small number of nodes in each cluster. The allocated new neighbors are either nodes that belong in different clusters, or, nodes that have not send a rewiring request at all.

We will compare our algorithms against the **Random Rewiring** currently employed by Avalanche. In this algorithm, each time a peer receives a packet, with probability p contacts the registrat and asks to change a neighbor. The registrat randomly selects which neighbor to change, and randomly allocates a new neighbor from the active peer nodes.

5. SIMULATION RESULTS

For our simulation results we will start from the topology illustrated in Fig. 2, that consists of 30 nodes connected into three distinct clusters. The source is node 1, and belongs in the first cluster. The bottleneck links are indicated with arrows (and thus indicate the underlying physical link structure). Our first set of simulation results depicted in Fig. 4 and 3 show that the subspaces within each cluster are very similar, while the subspaces across clusters are significantly different, where we use the distance measure in (1). Note that, the smaller the bottleneck, the larger the "similarity" of subspaces within the same cluster, and also, the larger the difference across clusters. These results indicate for example that knowledge of these subspaces will allow the registrat to accurately detect and break clusters (Algorithm 3).

Our second set of simulation results considers again a topology with three clusters: cluster 1 has 15 nodes and contains the source, cluster 2 has also 15 nodes, while the number of nodes in cluster 3 increases from 15 to 250. During the simulations we assume that the registrat keep the nodes' degree between 2 and 5, with an average degree of 3.5. All edges correspond to unit capacity links. An experiment terminates once all peers have collected all packets. The values presented are averaged over 10 experiments, where in each experiment the source sends 50 packets to the peers.

We compare the performance of the three proposed algorithms in Section 4 with random rewiring, currently employed by Avalanche. We implemented these algorithms as follows. For random rewiring, every time a node receives a packet it changes one of its neighbors with probability $p = \frac{8}{500}$. For Algorithm 1, we use a parameter of $k = 10$, and check whether the non-innovative packets received exceed this value every four received packets. For Algorithm 2, every node checks each received subspaces every four received packets using the threshold value $\mathcal{T} = 1$. Nodes that receive $\Delta = 2$ or more rewiring requests contact the registrat. Finally for Algorithm 3, we assume that nodes use the same criterion as in Algorithm 2 to decide whether they form part of a cluster, again with $\mathcal{T} = 1$. Dissatisfied node send their observed subspaces to the registrat. The registrat assigns nodes i and j in the same cluster if $D_{ij} \leq 7$, where D_{ij} is defined in Section 3.

Table 1 compares all algorithms with respect to the average collection time, defined as the difference between the time a peer receives the first packet and the time it can decode all packets, and averaged over all peers. All algorithms perform similarly, indicating that all algorithms result in breaking the clusters. It is important to note that these average collection times is in terms of number of exchanges needed and *does not* account for the delays incurred due to rewiring. We compare the number of such rewirings needed next.

Fig. 5 plots the average number of rewirings each algorithm employs. Random rewiring incurs a number of rewirings proportional to the number of P2P nodes, and independently from the underlying network topology. Our proposed algorithms on the other hand, adapt to the existence and size of clusters. Algorithm 3 leads to the smallest number of rewirings. Algorithm 2 leads to a larger number of rewirings, partly due to that the new neighbors are chosen randomly and not in a manner that necessarily breaks the clusters. The behavior of algorithm 1 is interesting. This algorithm rewires any node that has received more than k non-innovative packets. Consider cluster 3, whose size we increase for the simulations. If k is small with respect to the cluster size, then a large number of nodes will collect close to k non-innovative packets; thus a large number of nodes will ask for rewirings. Moreover, even after rewirings that break the cluster occur, some nodes will still collect linearly dependent information and ask for additional rewirings. As cluster 3 increases in size, the information disseminates more slowly within the cluster. Nodes in the border, close to the bottleneck links, will now be the ones to first ask for rewirings, long before other nodes in the network collect a large number of non-innovative packets. Thus once the clusters are broken, no new rewirings will be requested. This desirable behavior of Algorithm 1 manifests itself for large clusters; for small clusters, such as cluster 2, the second algorithm for example achieves a better performance using less reconnections.

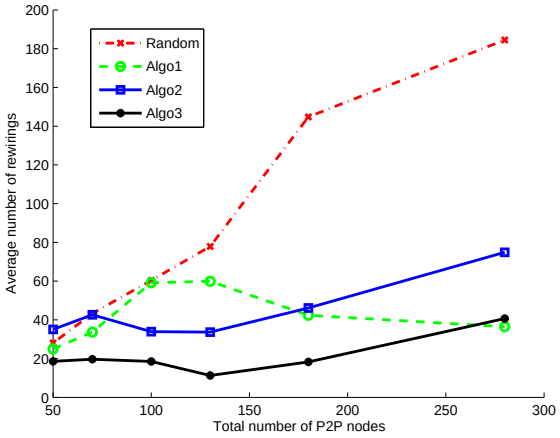


Figure 5: Average number of rewirings, for a topology with three clusters: cluster 1 has 15 nodes, cluster 2 has 15 nodes, while the number of nodes in cluster 3 increases from 20 to 250 as described in Table 1.

6. CONCLUSIONS

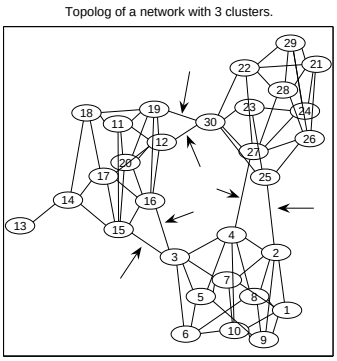


Figure 2: Topology with three clusters: cluster 1 contains nodes 1–10, cluster 2 nodes 11–20 and cluster 3 nodes 21–30.

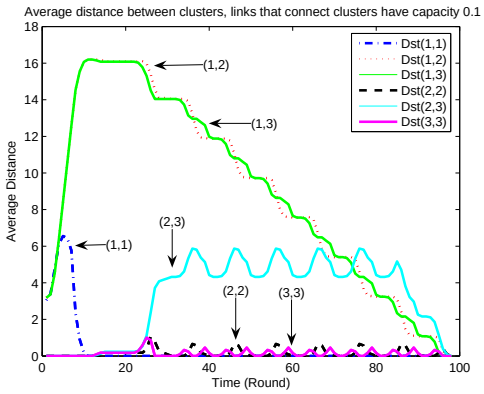
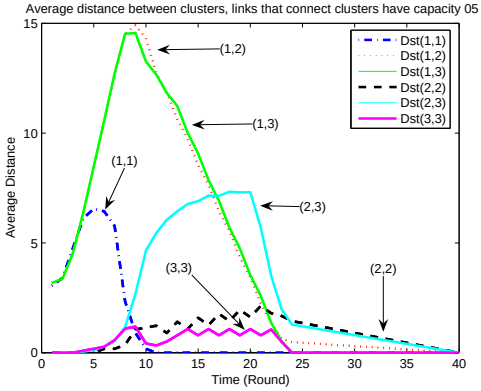


Figure 3: Simulation results for the topology in Fig. 2, with bottleneck link capacity values equal to 0.1.



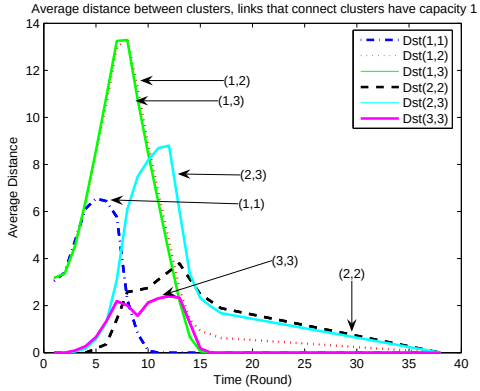


Figure 4: Additional results for the topology in Fig. 2, with bottleneck link capacity values equal to 0.5 (left) and 1 (right).

Table 1: Average Collection Time

Topology	Random	Algo 1	Algo 2	Algo 3
15–15–20	20.98	22.14	20.57	20.39
15–15–40	18.72	21.13	19.36	19.47
15–15–70	18.88	21.54	18.97	19.54
15–15–100	18.6	21.48	18.91	21.42
15–15–150	19.56	20.85	19.96	20.18
15–15–250	18.79	19.8	19.18	18.99

In this paper we observed that, in a P2P network utilizing network coding the linear combinations a peer receives from its neighbors unravel structural information about the network topology. We propose methods to capitalize on this fact for passive inference of network characteristics, and peer-initiated overlay topology management. This is a first paper introducing these ideas, and there are a number of associated questions and research directions that still need to be explored such as, combining with proposed techniques for distributed management of systems that do not employ network coding, and security concerns.

Acknowledgments: We thank A. Markopoulou, S. Mohajer and P. Rodriguez for many useful discussions.

7. REFERENCES

[1] R. Ahlswede, N. Cai, S-Y. R. Li, and R. W. Yeung, "Network information flow", *IEEE Trans. Inform. Theory*, vol. 46, pp. 1204–1216, July 2000.

[2] C. Gkantsidis and P. Rodriguez, "Network coding for large scale content distribution", *Infocom*, March 2005.

[3] C. Gkantsidis, J. Miller, P. Rodriguez, "Comprehensive view of a live network coding P2P system", *ACM SIGCOMM/USENIX IMC*, 2006.

[4] T. Ho, R. Koetter, M. Médard, M. Effros, J. Shi, and D. Karger, "A random linear network coding approach to multicast", *IEEE Transactions on Information Theory*, pp. 4413-4430, Oct. 2006.

[5] "RON: Resilient Overlay Networks" *http://nms.csail.mit.edu/ron*

[6] C. Fragouli and A. Markopoulou, "A network coding approach to overlay network monitoring", *Allerton*, Oct. 2005.

[7] C. Fragouli, A. Markopoulou, and S. Diggavi, "Active topology inference using network coding", *Allerton*, Oct. 2006.

[8] T. Ho, B. Leong, Y. Chang, Y. Wen and R. Koetter, "Network monitoring in multicast networks using network coding", *in International Symposium on Information Theory (ISIT)*, June 2005.

[9] R. Koetter and F. Kschischang, "Coding for errors and erasures in random network coding", *in International Symposium on Information Theory (ISIT)*, June 2007.

[10] M. Jafarisiavoshani, C. Fragouli and S. Diggavi, "Subspace properties of randomized network coding", *Information Theory Workshop*, July 2007.

[11] H. Coehn, *A course in computational algebraic number theory*, Springer, 1993.