On Randomized Network Coding Properties and their Application

Mahdi Jafari Siavoshani

advisor:
Prof. Christina Fragouli

External expert:
Prof. Frank Kschischang

Master Thesis
Communication and Computer Science Department
Ecole Polytechnique Fédérale de Lausanne

October 1, 2007

Acknowledgement

This project would not have been possible without the support of many people. I would be grateful to my advisor Prof. Christina Fragouli for her constant support and encouragement at the start of the project, during the work and at the end. Also thanks to my unofficial co-advisor Prof. Suhas Diggavi for his guidance and support.

Here I would like to thank Soheil Mohajer for the great discussion we had during this work. I should also thank many friends including:
Amin Karbasi: who always forces me to enjoy the life!,
Shirin Saeedi: who always shows me that there exists another way of thinking!,
Ghazale Hosseinabadi: who is a good neighbor and friend,
Pedram Pedarsani: for his free ride by his Ford!,
Hamid Khatibi and Maryam Javanmardi: for their so much help and support,
Ali Ajdari and Maryam Zaheri: for their friendship,
Hamed Alavi: who is really calm and enjoy the life.

Finally, I owe my family (my parents and my brother) a great debt of gratitude for their support.

Contents

Chapter 1

Introduction

Network coding is becoming a hot research area that promises to have interesting applications in practical networking systems. In today's communication networks, intermediate nodes just perform store and forward operations which means independent information routes independently through the network.

The advent of network coding [1] has changed the above methodology: instead of just storing and forwarding data, nodes may recombine the input packets to generate output packets. It has been shown that there are two main benefits of this approach. There is potential throughput improvement and also a high degree of robustness. Similar to erasure coding, successful reception of information does not depend on receiving original packet content but rather on receiving a sufficient number of independent packets.

In *Linear Network Coding*, the relation between the input and output packets is linearly determined. The reason for choosing a linear framework is that the algorithms for encoding and decoding are well studied and working with them is simpler. In this framework, the condition for decoding is receiving a sufficient number of linearly independent packets at a specific receiver.

In this context, the problem of network code design is to choose what linear operations each node of the network should perform. There are two different approach introduced in the literature. First, it is possible to use deterministic algorithms to design network codes. The polynomial-time algorithm for multicast introduced in [22] is one of such example. Alternatively, it is possible to simply let each node in the network select uniformly at random the coefficients over a finite field, in a completely independent and decentralized manner [6, 4]. This later approach is termed *randomized Network Coding*.

Randomized network coding is the approach adopted by almost all practical applications. For example, Avalanche, the first implementation of a Peer-to-Peer system that uses network coding, adopts randomized operation [13, 14]. As another example, in wireless and sensor networks [] most proposed protocols again adapted for randomized network operation.

The reason randomized network coding is so popular is because it allows to achieve a very simple and flexible operation without need of synchronization, that is very well tailored to packet networks. In particular to every packet, a coding vector is appended that determines how the packet is expressed with respect to the original data packets that exist at the source nodes. These coding vectors enable the receivers to solve a set of linear equations to decode and recover the original data packets.

Our work in this thesis started from the observation that coding vectors implicitly carry information about the network structure (as well as its state). We will explore these relation by investigating the properties of vectors spaces defined over a finite field. Especially we are interested in random sampling from the vectors spaces over a finite field. Then we will show that in a system that employs randomized network coding, we can take advantage of these properties towards different applications. Thus we will argue that, randomized network coding, apart from more well known desirable features it offers, can additionally provide us some information about the network itself. This allows us to design some centralized and decentralized algorithms for topology inference, network tomography and network management.

Our contributions include the following. First we will investigate the properties of vectors space in a system that uses randomized network coding in Chapter 3 and then use these properties to explore some applications in Chapters 4, 5, 6. These results have also appeared in [9, 10].

We will study the passive topology inference problem in Chapter 4 where we have found that under certain conditions it is possible to uniquely identify the topology of a network using the subspaces each node collects during the dissemination process.

Finding the location of Byzantine attackers is the problem that is explored in Chapter 5. We have proposed different methods, compared them, and shown that we can find the location of attacker up to a small uncertainty.

In Chapter 6, we show that the received subspaces at a specific node can give us some information regarding the bottlenecks in a network. So we use this idea to propose some algorithms that are peer initiated to change the topology of a network in a way that breaks these bottlenecks.

1.1 Related Work

Network coding started by the work of Ahlswede et al. [1] who showed that, using with network coding, and assuming the symbol size approaches infinity, a source can multicast information at a rate approaching the smallest mincut between the source and any receiver. Li et al. [2] showed that linear network coding with finite field size is sufficient for multicast. Koetter et al. [3] presented an algebraic framework for network coding.

Randomized network coding was originaly proposed by Ho et al. [6] where they showed that randomly choosing the network code leads to a valid solution for a multicast problem with high probability if the field size is large. It was later applied by Chou et al. [5] to demonstrate the practical aspects of random linear network coding. Gkantsidis et al. [13, 14] also implemented a practical file sharing system based on this idea.

Network error correcting codes, that are capable of correcting errors inserted in the network, have been developed during the last few years. For example see the work of Koetter et al. [15], Jaggi et al. [17], Ho et al. [18], Yeung et al.[19, 20] and Zhang [21]. These schemes are capable of delivering information despite the presence of Byzantine attacks in the network, as long as the number of such attacks is limited. These network error correcting schemes are designed to work without knowledge of the network topology.

Overlay topology monitoring and management that do not employ network coding has been an intensively studied research topic, see for example [23]. This subject for the networks using network coding is a new area of study. Fragouli et al. [7, 8] took the advantage of the network coding capabilities for active network monitoring where the focus was on link loss rate inference. Passive inference of link loss rates has also been proposed by Ho et al. [12]. However, the idea of passive inference of topological properties is a novel contribution of this work.

Chapter 2

Modeling

The motivation of this work is based, as we are going to show, on the fact that the coding vectors attached to every packet in a system running randomized network coding convey some information about the topology of the network as well as its status. We will investigate the properties of the subspaces each node gathers during the information flow and show how we can use them for different applications.

To the find the properties of subspaces we need to use different models, where each one can answer some of the desired questions. In this section we will investigate different aspects of the problem modeling.

2.1 Network Operation

Consider a network represented as a connected graph $G = (V, E)$, with $\vartheta = |V|$ nodes and $\xi = |E|$ edges. For an arbitrary edge $e = (i, j) \in E$, we denote $\text{head}(e) = j$ and $\text{tail}(e) = i$. For an arbitrary node $v \in V$, let us denote $\text{In}(v)$ the set of incomming edges to v and $\text{Out}(v)$ the set of outgoing edges from v. For an undirected graph we have $\text{In}(v) = \text{Out}(v)$.

We assume there is a source $S \in V$ that has a set of n independent packets $\{p_1, \ldots, p_n\}$, $p_i \in \mathbb{F}_q^\ell$, to distribute to a set of receivers using network coding techniques [1], where each packet is a sequence of symbols with fixed length ℓ over a finite field $\mathbb{F}_q$. We can think of each source packet as corresponding to one dimension of an n-dimensional space over $\mathbb{F}_q$. We can thus associate with each packet one of the orthonormal basis vectors $\{\alpha_1, \ldots, \alpha_n\}$, where $\alpha_i \in \mathbb{F}_q^n$ is a vector with one at position i and zero elsewhere. It should be noted that vector α_i is a part of packet p_i at a fixed position for all $i = 1, \ldots, n$.

In general, to enable the receivers to decode, a vector of length n, called coding vector, appended to each packet (see Figure 2.1) which determines the linear relation between the received packets and the original packets at the source. For each source packet p_i, the coding vector equals α_i. The

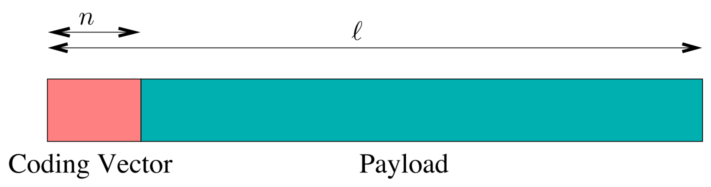

Figure 2.1: A coding vector appended to each packet.

coding vector in each packet is affected by the same operations that act on the whole packet during the propagation in the network. At a specific receiver node, after collecting n independent packets z_i with coding vectors $\hat{\alpha}_i$, it is possible to invert the transfer matrix

$$G = \begin{bmatrix} \hat{\alpha}_1 \\ \vdots \\ \hat{\alpha}_n \end{bmatrix},$$

and recover the original messages p_i.

Assume every node performs randomized network coding, where each node sends uniform at random linear combinations over $\mathbb{F}_q$ of its collected packets to its neighbors. We say that node $i \in V$ at time t observes a subspace $\Pi_i(t) \subseteq \mathbb{F}_q^n$, if $\Pi_i(t)$ is the space spanned by the received coding vectors at node i up to time t. When $\dim(\Pi_i) = n$, node i has collected a basis of the n-dimensional space, and can decode the source information.

It is possible to separate the dissemination protocols as described above into the following operation categories.

- *Synchronous:* All nodes are synchronized and transmit to their neighbors according to a global clock tick (time-slot). At time t node i sends linear combinations from all vectors it has collected up time $t-1$, chosen uniformly at random from $\Pi_i(t-1)$. Once nodes start transmitting information, they keep transmitting until all receivers are able to decode.

- *Asynchronous:* Nodes transmit linear combinations at randomly and independently chosen time instants.

For designing algorithms we can distinguish between different kinds of information that we have access to. We may assume that we have the following information.

- *Global information*: A central entity knows the subspaces that all ϑ nodes in the network have observed.

- *Local Information:* There is no such omniscient entity, and each node i only knows what it has received, its own subspace Π_i.

We may also have something between these two extream cases. Moreover, we may have a *static view*, where we take a snapshot of the network at a given time instant t, or a *non-static view*, where we take several snapshots of the network and use the subspaces' evolution to design an algorithm.

2.2 Randomized Network Coding Model

Let us denote $S_i \in V$, $i = 1, \dots, s$, as the set of source nodes that want to multicast unit rate process X_i[1] to a set $R_j \in V$, $j = 1, \dots, r$, of receiver nodes. Sources of larger rate can be modelled as multiple sources at the same node. This is a multi-source multicast problem.

In the algebraic framework of network coding introduced in [3], the source processes, the information processec transmitted on each link and the receiver processes, are sequences of symbols over finite field $\mathbb{F}_q$ of fixed length ℓ. The information process Y_j transmitted on a link j can be written as a linear combination of link j's inputs. For the delay-free case we can write,

$$Y_j = \sum_{\{i\,:\,S_i=\text{tail}(j)\}} a_{ji} X_i + \sum_{\{l\,:\,\text{head}(l)=\text{tail}(j)\}} f_{jl} Y_l.$$

The ith output process $Z_{v,i}$ at the receiver node v is a linear combination of the information process on its incomming links, where can be written as follows,

$$Z_{v,i} = \sum_{\{l\,:\,\text{head}(l)=v\}} b_{v,i,l} Y_l.$$

It is shown in [3] that for multicast on a network with link delays, memory is needed at the receiver nodes but memoryless operation is sufficient at all intermediate nodes. In this case the equation of the network can be written as

$$Y_j(t+1) = \sum_{\{i\,:\,S_i=\text{tail}(j)\}} a_{ji} X_i(t+1) + \sum_{\{l\,:\,\text{head}(l)=\text{tail}(j)\}} f_{jl} Y_l(t),$$

and the process received at node v as

$$Z_{v,i}(t+1) = \sum_{\tau=0}^{\mu} b'_{vi}(\tau) Z_{v,i}(t-\tau) + \sum_{\{l\,:\,\text{head}(l)=v\}} \sum_{\tau=0}^{\mu} b''_{v,i,l}(u) Y_l(t-\tau),$$

where μ represents the memory required. These equations can be represented algebraically in terms of a delay variable $\mathcal{D}$:

$$\begin{aligned} Y_j(\mathcal{D}) &= \sum_{\{i\,:\,S_i=\text{tail}(j)\}} a_{ji} X_i(\mathcal{D}) + \sum_{\{l\,:\,\text{head}(l)=\text{tail}(j)\}} \mathcal{D} f_{jl} Y_l(\mathcal{D}) \\ Z_{v,i}(\mathcal{D}) &= \sum_{\{l\,:\,\text{head}(l)=v\}} b_{v,i,l}(\mathcal{D}) Y_l(\mathcal{D}), \end{aligned}$$

[1]Each source process X_i has an entropy rate of one bit per unit time.

where

$$\begin{aligned} X_i(\mathcal{D}) &= \sum_{t=0}^{\infty} X_i(t)\mathcal{D}^t \\ Y_j(\mathcal{D}) &= \sum_{t=0}^{\infty} X_j(t)\mathcal{D}^t \\ Z_{v,i}(\mathcal{D}) &= \sum_{t=0}^{\infty} Z_{v,i}(t)\mathcal{D}^t, \end{aligned}$$

and

$$b_{v,i,l}(\mathcal{D}) = \frac{\sum_{\tau=0}^{\mu} \mathcal{D}^{\tau+1} b''_{v,i,l}(\tau)}{1 - \sum_{\tau=0}^{\mu} \mathcal{D}^{\tau+1} b'_{vi}(\tau)}.$$

We can collect the coefficient $\{a_{ji}, f_{jl}, b_{v,i,l}\}$ into $\xi \times s$ matrix A, $\xi \times \xi$ matrix F and $|\text{In}(v)| \times \xi$ matrix B_v whose structure is constrained by the network. In the framework defined above, a tuple $(A, F, B_{R_1}, \ldots, B_{R_r})$ can be called a linear network code. As it is shown in [3], the mapping from source processes $X = [X_1, \ldots, X_s]^T$ to output processes $Z_v = [Z_{v,1}, \ldots, Z_{v,\text{In}(v)}]^T$ can be presented as follows

$$Z_v = B_v(I - F)^{-1} AX,$$

for the delay-free case and

$$Z_v(\mathcal{D}) = B_v(\mathcal{D})(I - F(\mathcal{D}))^{-1} A(\mathcal{D}) X(\mathcal{D}),$$

for the network that has delay, where $A(\mathcal{D}) = \mathcal{D}A$ and $F(\mathcal{D}) = \mathcal{D}F$.

For a given multicast connection problem if some network code $(A, F, B_{R_1}, \ldots, B_{R_r})$ in a field $\mathbb{F}_q$ or $\mathbb{F}_q(\mathcal{D})$ satisfies the condition that $B_{R_i}(I - F)A$ has full rank s for each receiver R_i then we can see that the problem has a solution in the same field.

Definition 2.1. *A multicast connection problem for which there exist a solution in some field $\mathbb{F}_q$ or $\mathbb{F}_q(\mathcal{D})$ is called* feasible.

Definition 2.2. *If for a receiver R_i there exists some value of B_{R_i} such that $B_{R_i}(I - F)^{-1}A$ has full rank s, then (A, F) is a* valid *network code for receiver R_i. A network code (A, F) is valid for a multicast problem if it is valid for all receivers.*

It is shown that we can find a linear network code for a given multicast problem using the knowledge of topology; for example see [11, 22]. But in practical applications, usually it is very costly to implement such algorithms specially when the communication between nodes are limited or expensive. In [6], the authors consider a randomized approach in which network nodes independently and randomly choose code coefficients, (A, F), from some finite field $\mathbb{F}_q$. They give a lower bound on the probability of success and show that this probability goes to 1 as q goes to infinity. The following theorem represents this relation more clearly.

Theorem 2.1. *For a feasible multicast connection problem, and a network code in which some or all code coefficients (A, F) are chosen independently and uniformly over a finite field $\mathbb{F}_q$ (some coefficients can take fixed values as long as these values preserve feasibility), the probability that all the receivers can decode the source processes (the selected code is valid) is at least $(1 - r/q)^{\nu}$ for $q > r$, where r is the number of receivers and ν is the number of links that have random coefficients.*

2.3 Definitions and Notation

Let $\Pi_i(t)$ denote the subspace node i has collected up to time t. For simplicity of notation, we will drop t when not necessary and use Π_i. To compare subspaces Π_i and Π_j, we will denote:
the dimension of each subspace as

$$d_i \triangleq \dim(\Pi_i), \quad \forall i \in V,$$

the dimension of the intersection of two subspaces as

$$d_{ij} \triangleq \dim(\Pi_i \cap \Pi_j), \quad \forall i, j \in V,$$

and the dimension of the joint span of two subspaces as

$$D_{ij} \triangleq \dim(\Pi_i \cup \Pi_j), \quad \forall i, j \in V,$$

where by union we mean the common span of Π_i, Π_j, *i.e.*, $\Pi_i \cup \Pi_j = \text{span}\{\Pi_i, \Pi_j\}$. Note that $d_i + d_j = d_{ij} + D_{ij}$. For a set of nodes $\mathcal{U} = \{u_1, \ldots, u_m\}$, we will denote as $d_{\mathcal{U}} \triangleq \dim(\Pi_{u_1} \cup \ldots \cup \Pi_{u_m})$. We also use the following metric defined to measure the distance between two subspaces,

$$\begin{aligned} \delta_{ij} &\triangleq \dim(\Pi_i \cup \Pi_j) - \dim(\Pi_i \cap \Pi_j), \quad \forall i, j \in V, \\ &= D_{ij} - d_{ij}. \end{aligned}$$

This metric was also introduced in [15], where it was used to design error correction codes.

We will consider connected networks, where each node, apart from the source, has at least one node (parent) transmitting information to it. If node i has p_i parents $u_1, \ldots, u_{p_i}$, we will denote with $\Pi_i^{(u_j)}(t)$ the subspace node i has received from parent u_j up to time t, and with $\pi_i^{(u_j)}(t)$ the subspace node i receives from parent u_j at exactly time t. Then we define $\pi_i(t)$ as the whole subspace (from all parents) node i received at time t. Thus, $\Pi_i^{(u_j)}(t) = \Pi_i^{(u_j)}(t-1) \cup \pi_i^{(u_j)}(t)$, $\pi_i(t) = \cup_{j=1}^{p_i} \pi_i^{(u_j)}(t)$, and $\Pi_i(t) = \cup_{j=1}^{p_i} \Pi_i^{(u_j)}(t)$.

Initially, at time $t = 0$, the subspaces of all nodes (apart the source) are empty. We define the filling time as following,

Definition 2.3. *The* filling time threshold τ_f *of a dissemination algorithm, is the first time* τ_f *at which each edge of the network has been used at least once.*

For time $t < \tau_f$ there exist some edges in the network that have not used yet. For $t \geq \tau_f$, all nodes receive packets from all their incoming edges. It should be noted that the time τ_f depends upon the structure of the network as well as the dissemination protocol itself used by every node.

2.4 Oblivious Model for Randomized Network Coding

Here we explain the idea of communication using the subspaces in a network performing randomized network coding that was introduced in [15]. In the following, we use the same notation as used in Section 2.1. Let $\{p_1, \ldots, p_n\}$, $p_i \in \mathbb{F}_q^\ell$ denote the set of injected packets into the network. Assume that there is no error. An arbitrary receiver R_i collects m packets z_j where each z_j can be presented as $z_j = \sum_{i=1}^n h_{ji} p_i$. The coefficients h_{ji} are unknown and randomly chosed over $\mathbb{F}_q$. On the other hand, if t erroneous packets, $\{e_1, \ldots, e_t\}$, are injected by an adversary into the network we can write

$$z_j = \sum_{i=1}^{n} h_{ji} p_i + \sum_{k=1}^{t} g_{jk} e_k,$$

where again $g_{jk} \in \mathbb{F}$ are unknown coefficients. In matrix form, the transmission model can be represented as

$$z = Hp + Ge,$$

where H and G are random $m \times n$ and $m \times t$ matrices, p is the $n \times \ell$ matrix whose rows are the transmitted packets, and e is the $t \times \ell$ matrix whose rows are the error packets.

The matrices H and G are random but the network topology imposes some constraints on these matrices. As stated in [15], the above model naturally lead us to consider information transmission not via the choice of p_i but rather by the choice of the vector space spaned by $\{p_i\}$.

Let W be a fixed ℓ-dimensional vector space over $\mathbb{F}_q$. Let also $\mathcal{P}(W)$ denote the set of all subspaces of W. It is obvious that all the transmitted and received packets are vectors of W. We will give the definition of "operator channel" which is a usefull model in the context of randomized network coding [15].

Definition 2.4. *An operator channel associated with space W is a channel with input and output alphabet in $\mathcal{P}(W)$. The channel transfer relation between input Π_p and output Π_z is as follows,*

$$\Pi_z = \mathcal{H}_k(\Pi_p) \oplus E,$$

where $\mathcal{H}_k$ is an erasure operator, $E \in \mathcal{P}(W)$ is an arbitrary error space, and $\oplus$ denotes the direct sum. If $\dim(\Pi_p) > k$, the erasure operator $\mathcal{H}_k$ projects Π_p onto a randomly chosen k-dimensional subspace of Π_p; otherwise, $\mathcal{H}_k$ leaves Π_p unchanged.

In fact, the operator channel takes a vector space as its input and puts out another vector space where it may delete some vectors from the input space and add some new vector to it.

In reference [15], the authors used this model to design codes that allow communication in networks performing randomized network coding. In Chapter 5, we will assume this model of communication when searching the location of Byzantine adversaries.

2.5 Algebraic Model for Synchronous Networks

In this section an algebraic approach will be introduced that is useful to model synchronous networks. This approach is similar to [3], with a slightly changed dissemination protocol. Suppose the network is synchronous and each link has unit delay. We assume that each node waits for a finite time before it starts to send out packets. After starting the transmission, each node sends packets on all its outgoing edges at every time slot. If the waiting time is zero for all nodes we will be in the case of usual synchronous networks. The waiting times will be used in the following chapters to enforce the subspaces of different nodes be distinct.

Suppose we are interested in finding the transfer relation between the source and an arbitrary node v. Let $X(t)$ be an $|\mathrm{Out}(S)| \times n$ matrix which represents the coding vectors that the source node injects to the network at time t. The $\xi \times n$ matrix $Y(t)$ collects the coding vectors of all messages that pass through the edges of the network at time t. Suppose $\mathcal{F}$ is the adjacency matrix of the labeled line graph of the graph G which is defined as follows

$$\mathcal{F}_{ij} = \begin{cases} 1 & \mathrm{head}(e_i) = \mathrm{tail}(e_j), \\ 0 & \text{otherwise.} \end{cases}$$

From the definition, we know that $\mathcal{F}$ is a $\xi \times \xi$ matrix. Let $F_0^{(t)}, F_1^{(t)}, \ldots, F_t^{(t)}$ be a sequence of random matrices over $\mathbb{F}_q$ which conform to $\mathcal{F}$, for $i \neq j$, $(F_k^{(t)})_{ij} = 0$ wherever $\mathcal{F}_{ij} = 0$ and having random numbers from $\mathbb{F}_q$ in other places.

Using the waiting time of each node we can compute the time at which an edge starts to convey packets. Let $\delta_1, \ldots, \delta_\xi$ denote these times. Using the step function $u(t)$ defined as follows,

$$u(t) = \begin{cases} 1 & t \geq 0, \\ 0 & \text{otherwise}, \end{cases}$$

let us write the $\xi \times \xi$ diagonal matrix $U(t)$ as,

$$U_{ii}(t) = \begin{cases} u(t - \delta_i) & \text{if} \quad \delta_i > l_i, \\ 1 & \textit{otherwise}, \end{cases}$$

where l_i is the shortest path from source to arbitrary edge i.

In our model the nodes have memory. Using the above definitions, the set of vectors that each node v receives in every time instant t can be written as follows

$$\begin{cases} Y(t+1) = AX(t+1) + U(t+1)\sum_{i=0}^{t} F_i^{(t)} Y(t-i), \\ \\ Z_v(t+1) = B_v Y(t), \end{cases} \tag{2.1}$$

where A is a $\xi \times |\mathrm{Out}(S)|$ matrix which denotes the connection of node S to the rest of the network. In the same way matrix B_v defines the connection of node v to the set of edges in network. It is also important to note that B_v is a $|\mathrm{In}(v)| \times \xi$ matrix.

Suppose we are interested in finding the output of such a system up to some predefined time T. We can rewrite the above equation by defining new matrices as follows. For the input vectors let us define

$$\boldsymbol{X} = \begin{bmatrix} X(0) \\ \vdots \\ X(T-1) \end{bmatrix}_{T|\mathrm{Out}(S)| \times n}.$$

For the states of system we define

$$\boldsymbol{Y} = \begin{bmatrix} Y(0) \\ \vdots \\ Y(T-1) \end{bmatrix}_{\xi T \times n},$$

and for the output we can write

$$\boldsymbol{Z}_v = \begin{bmatrix} Z_v(1) \\ \vdots \\ Z_v(T) \end{bmatrix}_{T|\mathrm{In}(v)| \times n}.$$

We also define a new set of matrices which represent the input-output relation. Using matrix A we define the following matrix

$$\boldsymbol{A} = I_T \otimes A = \begin{bmatrix} A & & \\ & \ddots & \\ & & A \end{bmatrix}_{\xi T \times T|\mathrm{Out}(S)|} .$$

We do the same thing for matrix B_v,

$$\boldsymbol{B}_v = I_T \otimes B_v = \begin{bmatrix} B_v & & \\ & \ddots & \\ & & B_v \end{bmatrix}_{T|\mathrm{In}(v)| \times \xi T} .$$

We define matrix $\boldsymbol{F}$ which represent how the states are related to each other:

$$\boldsymbol{F} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\ F_0^{(0)} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots \\ F_1^{(1)} & F_0^{(1)} & \mathbf{0} & \mathbf{0} & \cdots \\ F_2^{(2)} & F_1^{(2)} & F_0^{(2)} & \mathbf{0} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}_{\xi T \times \xi T} .$$

At the end, we introduce matrix $\boldsymbol{U}$ that represent the starting transmission time for each edge,

$$\boldsymbol{U} = \begin{bmatrix} I_{\xi \times \xi} & & & \\ & U(1) & & \\ & & \ddots & \\ & & & U(T-1) \end{bmatrix}_{\xi T \times \xi T} .$$

Using the above definitions we can rewrite Equation 2.1 up to time T as follows

$$\begin{cases} \boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X} + \boldsymbol{U}\boldsymbol{F}\boldsymbol{Y}, \\ \boldsymbol{Z}_v = \boldsymbol{B}_v \boldsymbol{Y}. \end{cases} \tag{2.2}$$

This equation can be solved to find the input-output transfer matrix up to time T which results in

$$\boldsymbol{Z}_v = \boldsymbol{B}_v (I - \boldsymbol{U}\boldsymbol{F})^{-1} \boldsymbol{A}\boldsymbol{X}. \tag{2.3}$$

From the definition of matrix $\boldsymbol{F}$, we know that it is a "strictly lower triangular matrix" which means $\boldsymbol{F}$ is nilpotent and we have $\boldsymbol{F}^T = 0$. The same applies for the matrix $\boldsymbol{U}\boldsymbol{F}$, which means we have $(\boldsymbol{U}\boldsymbol{F})^T$. Then the input-output relation can be written as

$$\boldsymbol{Z}_v = \boldsymbol{B}_v \left(I + \boldsymbol{U}\boldsymbol{F} + (\boldsymbol{U}\boldsymbol{F})^2 + \cdots + (\boldsymbol{U}\boldsymbol{F})^T \right) \boldsymbol{A}\boldsymbol{X}. \tag{2.4}$$

It should be noted that if all nodes operate synchronously from the start of transmission, which equivalently means that if δ_e equals the shortest path length from the source to the edge e, we will have $\boldsymbol{U} = I_{\xi T \times \xi T}$.

The source S has n messages which are represented using n vectors forming a set of basis for the space $\mathbb{F}_q^n$. In every time slot, the source node sends different uniformly at random selected vectors from its space to the outgoing edges. The middle nodes in the network, after the waiting time, also do the same. Each node v which is interested in receiving the set of source's messages, gathers all received vectors up to some time $T \geq \frac{n}{\mathrm{In}(v)}$ in $\boldsymbol{Z}_v$ and checks whether it has rank equal to n or not. If the rank of $\boldsymbol{Z}_v$ becomes n there exist n linearly independent row vectors in $\boldsymbol{Z}_v$ forming a set of basis for $\mathbb{F}_q^n$ that enable node v to recover the source messages.

Chapter 3

Some Notes on Vector Spaces over a Finite Field $\mathbb{F}_q^n$

We will here state and prove basic properties and results that we will exploit towards various applications in the following chapters. In particular, we will investigate the properties of random sampling from vector spaces over a finite field. These properties are the core of our work in the following chapters. Moreover we will study the rate of receiving innovative packets in a synchronous multicast problem setting.

3.1 Sampling Subspaces over $\mathbb{F}_q^n$

The notion of vectors spaces over a finite field $\mathbb{F}_q$ is a very important tool for the problems we are going to consider in the following chapters. In particular, the random sampling of subspaces of $\mathbb{F}_q^n$ and finding properties play an important role in our work. In fact these properties give us a better insight and understanding of randomized network coding. In this section we study some of these basic properties.

Lemma 3.1. *Suppose we choose m vectors from n-dimensional space $W = \mathbb{F}_q^n$ uniformly at random to construct space Π_A. The probability that Π_A will be a k-dimensional subspaces of W is bounded as*

$$\begin{aligned} \Pr(\dim(\Pi_A) = k) &\leq \binom{m}{k} \prod_{i=0}^{k-1} \frac{q^n - q^i}{q^n} \left(\frac{q^k}{q^n}\right)^{m-k} \\ &= \binom{m}{k} \prod_{i=0}^{k-1} (1 - q^{i-n}) q^{(k-n)(m-k)}. \end{aligned}$$

Note that this bound is tight for $k = m$.

Proof. Use union bound. □

Lemma 3.2. *Let us construct Π_A by choosing m vectors uniformly at random from $W = \mathbb{F}_q^n$. The subspace Π_A will be full rank with probability greater than $1 - O(q^{-n+m-1})$.*

Proof. Let us write

$$\begin{aligned}\Pr(\dim(\Pi_A) = m) &= \prod_{i=0}^{m-1}(1 - q^{i-n}) \\ &\geq \left(1 - \sum_{i=0}^{m-1} q^{i-n}\right).\end{aligned}$$

So we are done. □

Lemma 3.3. *Let Π_i and Π_j be subspaces of $\mathbb{F}_q^n$ with dimension d_i and d_j respectively and intersection of dimension d_{ij}. Construct Π_i' by choosing m vectors from Π_i uniformly at random. Then $\Pr(\Pi_i' \subset \Pi_j) \approx 0$, if $\Pi_i \not\subseteq \Pi_j$.*

Proof. The probability that *all* m vectors are in the intersection is

$$\Pr(\Pi_i' \subset \Pi_j) = \left(\frac{q^{d_{ij}}}{q^{d_i}}\right)^m = q^{(d_{ij}-d_i)m},$$

which is of order $O(1/q)$ provided that $d_{ij} < d_i$. □

In the following we are going to show one of the most important properties of randomly selected subspaces. We will show that randomly selected subspaces tend to be "as far as possible". We will clarify and make precise what we mean by "as far as possible". First let us define a subspace in general position with respect to a family of subspaces as stated in [16].

Definition 3.1. *Let W be an n-dimensional space over field $\mathbb{F}$ and for $i = 1, \ldots, m$, let Π_i be a subspace of W, $\dim(\Pi_i) = d_i$. A subspace $\Pi \subseteq W$ of dimension d is in general position with respect to the Π_i if*

$$\dim(\Pi_i \cap \Pi) = \max\{d_i + d - n, 0\} \quad \textit{for } i = 1, \ldots, m. \tag{3.1}$$

It should be noted that $\max\{d_i+d-n, 0\}$ is the minimum possible dimension of $(\Pi_i \cap \Pi)$. So what the above definition says is that the intersection of Π and each Π_i is as small as possible.

Using the above definition let us state the following theorem.

Theorem 3.1. *Suppose $\{\Pi_i\}$, $i = 1, \ldots, m$, are subspaces of $W = \mathbb{F}_q^n$. Let us construct subspace Π by randomly choosing m vectors from W. Then Π will be in general position with respect to the family $\{\Pi_i\}$ with high probability if $q \gg 1$.*

Proof. First of all using Theorem (3.5) in [16], we know that there exists a subspace in general position with respect to all Π_i if q is sufficiently large. So it is sufficient to proove (3.1) for one specific i. The following lemma, Lemma 3.4, completes the proof. □

Lemma 3.4. *Suppose Π_k is a k-dimensional subspace of vector space $W = \mathbb{F}_q^n$. Let us take m vectors uniformly at random from W to construct the subspace Π. Under the assumption that $q \gg 1$ we have*

$$\Pr[\dim(\Pi \cap \Pi_k) = d] \approx \begin{cases} 1 & d = \min\left(k, (m-(n-k))^+\right), \\ 0 & \textit{otherwise.} \end{cases}$$

Proof. Let $\Pi_k^\perp$ be the subspace that has the property $\Pi_k \cup \Pi_k^\perp = W$. Suppose that $U = \{u_1, \ldots, u_k\}$ and $V = \{v_1, \ldots, v_{n-k}\}$ are sets of basis for Π_k and $\Pi_k^\perp$ respectively. Let $\{w_1, \ldots, w_m\}$ denote the vectors that are taken from W. It should be noted that $(U \cup V)$ form a set of basis for W so we can expand the vectors w_i with respect to this basis as follows

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

We put the vectors $\alpha^{(i)}$ with length n in a $n \times m$ matrix columnwise as follows

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

Now let us proceed as follows. Write an arbitrary linear combination of vectors w_i as multiplication of matrix A with an arbitrary $m \times 1$ vector a from the right handside. Consider only the vectorss that are inside Π_k. Then find the probability of this subspace having some specified dimension. Using the defined notation we can write

$$Aa = b,$$

or

$$\begin{bmatrix} \widetilde{U}_{k\times m} \\ \hline \widetilde{V}_{(n-k)\times m} \end{bmatrix} \cdot \begin{bmatrix} a_1 \\ \vdots \\ a_m \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_k \\ \hline 0_{(n-k)\times 1} \end{bmatrix}.$$

Considering the $n-k$ lower equations in the previous matrix equation, we can write

$$D_k = \dim(\mathrm{Kernel}(\widetilde{V})) = m - \mathrm{Rank}(\widetilde{V}).$$

We have assumed that $q \gg 1$ so matrix $\widetilde{V}$ is full rank with high probability and we have

$$\begin{aligned} D_k &= m - \min(m, n-k) \\ &= (m-(n-k))^+, \qquad \text{w.h.p.} \end{aligned}$$

Let $\{x_1, \ldots, x_{D_k}\}$ denotes for the set of basis for the kernel of $\widetilde{V}$. We put these vectors in a $(D_k \times m)$ matrix columnwise as follows

$$X = \begin{bmatrix} | & & | \\ x_1 & \cdots & x_{D_k} \\ | & & | \end{bmatrix}.$$

It should be noted that $D_k \leq m$ so matrix X is full rank. As we take w_i uniformly at random from $\mathbb{F}_q^n$, matrix $\widetilde{U}$ is also full rank. We can conclude that the product $\widetilde{U}X$ is a full rank matrix with rank $\min(k, D_k)$. The rank of matrix $\widetilde{U}X$ is the number of independent columns in it which determines the dimension of the disjoint space of Π and Π_k. So we are done. □

Corollary 3.1. *Suppose Π_1 and Π_2 are two subspace of $\mathbb{F}_q^n$ with dimension d_1 and d_2 respectively and joint dimension d_{12}. Let us take m_1 vectors uniformly at random from Π_1 and m_2 vectors from Π_2 to construct subspaces $\hat{\Pi}_1$ and $\hat{\Pi}_2$. Assuming $q \gg 1$ we have*

$$\Pr[\dim(\hat{\Pi}_1 \cap \hat{\Pi}_2) = d] \approx \begin{cases} 1 & d = \alpha, \\ 0 & \textit{otherwise}, \end{cases}$$

where

$$\alpha = \min \begin{cases} d_{12}, \\ (m_1 - (d_1 - d_{12}))^+, \\ (m_2 - (d_2 - d_{12}))^+, \\ (m_1 + m_2 - (d_1 + d_2 - d_{12}))^+. \end{cases}$$

Proof. Let us write

$$\begin{aligned} & \Pr[\dim(\hat{\Pi}_1 \cap \hat{\Pi}_2) = d] \\ = & \sum_{k=0}^{d_{12}} \Pr[\dim(\hat{\Pi}_1 \cap \hat{\Pi}_2) = d \mid \dim(\hat{\Pi}_1 \cap \Pi_{12}) = k] \cdot \Pr[\dim(\hat{\Pi}_1 \cap \Pi_{12}) = k]. \end{aligned}$$

From Lemma 3.4 we have

$$\begin{aligned} & \Pr[\dim(\hat{\Pi}_1 \cap \hat{\Pi}_2) = d] \\ \approx & \Pr\left[\dim(\hat{\Pi}_1 \cap \hat{\Pi}_2) = d \mid \dim(\hat{\Pi}_1 \cap \Pi_{12}) = \min(d_{12}, (m_1 - (d_1 - d_{12}))^+)\right]. \end{aligned}$$

Let us define $l = \min(d_{12}, (m_1 - (d_1 - d_{12}))^+)$ then we can rewrite the previous equation as follows

$$\begin{aligned} &\Pr[\dim(\hat{\Pi}_1 \cap \hat{\Pi}_2) = d] \\ = &\sum_{\substack{\Pi_l \in \Pi_{12} \\ \dim(\Pi_l) = l}} \Pr\left[\dim(\hat{\Pi}_1 \cap \hat{\Pi}_2) = d \mid \hat{\Pi}_1 \cap \Pi_{12} = \Pi_l\right] \cdot \Pr[\hat{\Pi}_1 \cap \Pi_{12} = \Pi_l]. \end{aligned}$$

Since we choose $\hat{\Pi}_1$ uniformly at random and we know that $\dim(\hat{\Pi}_1 \cap \Pi_{12}) = l$ with high probability. Thus

$$\Pr[\hat{\Pi}_1 \cap \Pi_{12} = \Pi_l] = \frac{1}{\#\text{ of distinct } \Pi_l \in \Pi_{12},\ \dim(\Pi_l) = l},$$

and

$$\Pr[\dim(\hat{\Pi}_1 \cap \hat{\Pi}_2) = d] \approx \Pr\left[\dim(\hat{\Pi}_1 \cap \hat{\Pi}_2) = d \mid \hat{\Pi}_1 \cap \Pi_{12} = \Pi_l\right],$$

where Π_l is an arbitrary subspace of Π_{12} with dimension l. Using the Lemma 3.4 we can write

$$\Pr[\dim(\hat{\Pi}_1 \cap \hat{\Pi}_2) = d] \quad \approx \quad \begin{cases} 1 & d = \min(l, (m_2 - (d_2 - l))^+) = \alpha, \\ 0 & \text{otherwise.} \end{cases}$$

So we are done. □

Corollary 3.2. *Let us construct two subspaces $\hat{\Pi}_1$ and $\hat{\Pi}_2$ by choosing m_1 and m_2 vectors uniformly at random respectively from $\mathbb{F}_q^n$. The subspace $\hat{\Pi}_1$ and $\hat{\Pi}_2$ will be disjoint with probability nearly to 1 if $q \gg 1$ and $m_1 + m_2 \leq n$.*

Proof. In the Corollary 3.1 put $\Pi_1 = \Pi_2 = \mathbb{F}_n^q$. □

3.2 Rate of Innovative Packets

In the following chapters we will need to know the rate of receiving innovative vectors at receivers in a multicast senario. As it is shown in [1], the source can multicast at rate equale to the minimum min-cut of all receivers as stated in the following theorem

Theorem 3.2. *Consider a directed graph $G(V, E)$ with unit capacity edges, s unit rate sources located on the same vertex of the graph and r receivers that are interested to receive all the sources' information. Assume that the value of the min-cut between each receiver and source node is s. Then there exist a multicast transmission scheme over a large enough finite filed $\mathbb{F}_q$, in which the intermediate nodes linearly combine their incoming information symbols over $\mathbb{F}_q$, that delivers the information from the sources to each receiver simultaneously at a rate equal s.*

Proof. For the proof see [1], [3] or [4]. □

For the discussion in the following chapters we will need a variation of Theorem 3.2 that is changed in a way useful for the protocol described in Section 2.5.

Theorem 3.3. *The min-cut from the source S to receiver R_v equals the average rate at which R_v receives innovative packets. In particular, in the protocol described in Section 2.5 (that include the usuall synchronized dissemination protocol), after waiting for enough time, node v receives c_v innovative packet per time slot from its parents if q and n is sufficiently large where $c_v = \text{min-cut}(v)$.*

Proof. Knowing that the min-cut of node v is c_v, let us choose a set of incoming edges to v with size c_v such that there exist c_v edge disjoint paths from S to each edge in the set and find the input-output transfer matrix just for this set of edges. Let us rewrite (2.3) as

$$\begin{aligned} \hat{\boldsymbol{Z}}_v &= \hat{\boldsymbol{B}}_v(I - \boldsymbol{UF})^{-1}\boldsymbol{AX} \\ &= \hat{\boldsymbol{B}}_v(I + \boldsymbol{UF} + (\boldsymbol{UF})^2 + \cdots + (\boldsymbol{UF})^T)\boldsymbol{AX}, \end{aligned} \tag{3.2}$$

where $\hat{\boldsymbol{B}}_v$ is a $c_vT \times \xi T$ matrix. Let $f_{ij}^{t,k}$ denote for the entries of $F_k^{(t)}$ and $x_{ij}^{(t)}$ denote for the entries of $X(t)$. Every node in the network performs random linear network coding so $x_{ij}^{(t)}$ and $f_{ij}^{t,k}$ (those that are not zero) are chosen uniformly at random from $\mathbb{F}_q$.

From (3.2) we know that each entry of $\hat{\boldsymbol{Z}}_v$ is a polynomial of degree at most $T+1$ in variables $x_{ij}^{(t)}$ and $f_{ij}^{t,k}$. Let us define $\hat{\boldsymbol{Z}}_v^i$ be a $n \times n$ submatrix of $\hat{\boldsymbol{Z}}_v$ that starts from ith row of $\hat{\boldsymbol{Z}}_v$ and includes n consecutive rows. Then we can conclude that the determinant of $\hat{\boldsymbol{Z}}_v^i$ is a polynomial of degree at most $(T+1)n$ in variables $x_{ij}^{(t)}$ and $f_{ij}^{t,k}$.

After passing the filling time τ_f, we know that for every $\hat{\boldsymbol{Z}}_v^i$, $i > \tau_f$, there exists a trivial solution for variables $x_{ij}^{(t)}$ and $f_{ij}^{t,k}$ (that connects R_v directly to S and source sends a set of basis at proper time on proper outgoing edges) that lets $\hat{\boldsymbol{Z}}_v^i$ be full rank. This means that $\det(\hat{\boldsymbol{Z}}_v^i)$ is a non-zero polynomial after the transition phase. Using the Schwartz-Zippel lemma we can upper bound the probability that $\hat{\boldsymbol{Z}}_v^i$ is not full rank as follows

$$\Pr[\det(\hat{\boldsymbol{Z}}_v^i) = 0] \le \frac{(T+1)n}{q}.$$

This means that assuming q and n is large enough we will be sure with high probability that each node v receives at least c_v innovative packets per time slot.

Now we will show that there exist a time τ_s where for $t \geq \tau_s$ node v receives exactly c_v innovative packets per time slot. If $|In(v)| = c_v$ we are done because it is not possible to send more than c_v innovative packets in one time slot using c_v edges. If $|In(v)| > c_v$ the minimum cut between S and v takes place somewhere in the middle of network. We can divide the set of nodes into two parts V_S and V_v such that $V_S \cup V_v = V$ and $V_S \cap V_v = \varnothing$. Let also E_{Sv} denote the edges from V_S to V_v. As we stated before E_{Sv} is a minimum cut so $|E_{Sv}| = c_v$. Using a similar argument as stated above it is obvious that for $t > \tau_f$ in every time slot E_{Sv} brings c_v innovative packets from V_S to V_v. Using the above notation let us write the following inequalities for $t > \tau_f$,

$$\begin{aligned} c_v t &\geq \text{\# of independent vectors received by set } V_v \text{ up to time } t \\ &\geq \text{\# of independent vectors received by node } v \text{ up to time } t \\ &= \overline{R}_v(t) \times t, \end{aligned}$$

where $\overline{R}_v(t)$ is the average rate of receiving innovative packets at node v up to time t. Let $R_v(t)$ be the rate of receiving innovative packets to node v at time slot t so we have,

$$\overline{R}_v(t) = \frac{1}{t} \sum_{i=1}^{t} R_v(t).$$

It is also obvious that the average rate $\overline{R}_v(t)$ is equal to c_v as time goes to infinity.

We know that $R_v(t) \geq c_v$ for $t > \tau_f$ and $\overline{R}_v(t) \leq c_v$ for all t. At the start of transmission where $t < \tau_f$, $R_v(t)$ is less than c_v because the network has not been yet filled with the information packets. Since τ_f is a finite time, $R_v(t)$ can be less than c_v just for a finite time so there should be a time τ_s after that $R_v(t) = c_v$. which completes our proof. □

Based on Theorem 3.3, let us state the following definition.

Definition 3.2. *For a specific dissemination protocol in a network, we define the* steady state *phase as the period in which each node v receives exactly c_v innovative packets per time slot and none of the nodes, except source S, have collected n linearly independent packets.*

Chapter 4

Topology Inference

In this chapter, using the ideas developed in Chapter 3, we will investigate the the relation between the network topology and the subspaces collected at the nodes for senario introduced in Section 2.1. We will further explore the conditions that allow us to passively infer the network topology.

4.1 Tree Topologies

Let $G = (V, E)$ be a network that is a directed tree of depth[1] $\mathcal{D}$, rooted at the source node S. We will present (i) necessary and sufficient conditions under which the tree topology can be uniquely identified, and (ii) given that these conditions are satisfied, algorithms that allow us to do so.

We first consider trees where each edge has the same capacity c, and thus the min-cut from the source to each node of the tree equals c. We then briefly discuss the case of undirected trees. Finally we examine the case where edges have different capacities, and thus nodes may have different min-cuts from the source.

4.1.1 Common Min-Cut

Assume that each edge of the tree has capacity c, and consider the following dissemination algorithm, also summarized[2] in Algorithm 4.1.1. Each node i waits until its subspace dimension becomes m_i, *i.e.*, $d_i \geq m_i$ (for this section we will use a common value $m_i = m$). It then starts transmitting to each of its children c random linear combinations per time-slot.

[1]The depth of a tree is the length of the longest path between the root and a leaf of the tree.

[2]Though the Algorithm 4.1.1 is introduced for trees, it will also be used for general topologies in Section 4.2.

```
Algorithm 4.1.1: INPUT($G = (V, E), S, \{m_i\}, n$)

for each $i \in V \setminus \{S\}$
    do $\Pi_i(0) = \emptyset, d_i(0) = 0$
$t \leftarrow 0$
while $\min_i d_i(t) < n$
    do  for each $i \in V$
        if $d_i(t) \geq m_i$
            then node $i$ transmits from $\Pi_i(t)$
        $t \leftarrow t + 1$
        for each $i \in V$ update $\Pi_i(t),\ d_i(t)$
```

The following theorem presents necessary and sufficient conditions that enable us to identify the network topology using a single snapshot of all node's subspaces at a time t.

Theorem 4.1. *Consider a tree of depth $\mathcal{D}$ where each edge has capacity c, and the dissemination algorithm in (4.1.1). A static global view of the network at time t, with $(\mathcal{D}-1)m < t < \frac{n}{c}$, allows to uniquely determine the tree structure, if and only if*

$$c + 1 \leq m. \tag{4.1}$$

Proof. The proof is based on the following simple observations. In a tree there exist a unique path $\mathcal{P} = \{S, i_1, \ldots, i_l, i\}$ from source S to node i. Clearly, in steady-state, for the nodes along the path it holds that

$$\Pi_i \subset \Pi_{i_l} \subset \cdots \subset \Pi_{i_1} \subset \mathbb{F}_q^n = \Pi_S. \tag{4.2}$$

The conditions on t ensure that the network is in steady-state, *i.e.*, all nodes have a non-empty subspace and no node's subspace (apart the source) equals the complete n-dimensional space.

Thus to identify the topology of the tree it is sufficient to show that $\Pi_i \nsubseteq \Pi_j$ for any j that is not in $\mathcal{P}$. But this is what the condition in (4.1) ensures. Indeed, consider a node $u \in V$ in the tree that has k children $u_1, \ldots, u_k$. If (4.1) holds, from Corollary 3.1, then $\Pi_{u_i} \neq \Pi_{u_j}$ for all i, j if and only if $m \geq c + 1$ and $q \gg 1$. $\square$

Thus the condition (4.1) on m ensures that the subspaces of all nodes in the tree are distinct during the steady-state phase. Obviously, if two nodes observe exactly the same subspace at time t, we can never distinguish between them; ensuring distinct subspaces is clearly necessary for identifiability.

The simple network in Figure 4.1 can help us better understand why the conditions on m in Theorem 4.1 are both necessary and sufficient. Assume that the edges have unit capacity ($c = 1$). At time $t = 1$, node A receives a vector y_1 from the source S. If node A starts transmitting to nodes B and C at time $t = 2$, then nodes B and C will both receive the same vector y_1, *i.e.*, $\Pi_B(2) = \Pi_C(2) = \text{span}\{y_1\}$. In fact, at all subsequent times, we will have that $\Pi_B(t) = \Pi_C(t) = \Pi_A(t-1)$. If instead, node A waits to collect

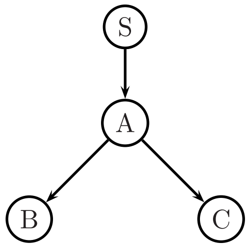

Figure 4.1: Directed tree with four nodes rooted at the source S.

$c + 1 = 2$ vectors, say y_1 and y_2, before starting transmission to nodes B and C, then it will hold that $\Pi_B(t) \neq \Pi_C(t)$, for $2 \leq t \leq n+1$.

Assume now that Theorem 4.1 holds. To determine the tree structure, it is sufficient to determine the unique parent each node has. From the previous arguments, the parent of node i is the unique node j such that Π_j is the minimum dimension subspace that contains Π_i. Then, the parent of node i is the node j such that, $j = \text{argmin}_{k:d_{ik}=d_i} d_k$. Note that to determine the tree topology, we do not need to know exactly which are the node subspaces, but only two "sufficient statistics": the dimension of each subspace $d_i = \dim(\Pi_i)$, $\forall i$, and the dimension of the intersection of every two subspaces $d_{ij} = \dim(\Pi_i \cap \Pi_j)$, $\forall i, j$, as described in Algorithm 4.1.2, assuming that the conditions of Theorem 4.1 hold.

```
Algorithm 4.1.2: TREE($\{d_i\}, \{d_{ij}\}$)

for each $i \in V$
  do { if $d_i = n, i \leftarrow S$
       else node $i$ has parent the node $j$ with
       $j = \text{argmin}_{k:d_{ik}=d_i} d_k$
```

4.1.2 Directed v.s. Undirected Network

In a tree with a single source, since new information can only flow from the source to each node along a single path, whether the network is directed or undirected makes no difference. In other words, from condition 4.2, all vectors that a node will send to its predecessor will belong in the subspace the predecessor already has. Thus Theorem 4.1 still holds for undirected networks with a common min-cut.

4.1.3 Different Min-Cuts

Assume now that the edges of the tree have different capacities. As a result, potentially mincut$(S, i) \neq$ mincut(j, i), for some node j in the path $\mathcal{P}$ that connects node i to the source S. Note that, under Algorithm 4.1.1, for the subspaces of the nodes in the path between S and i, condition (4.2) still holds. However, it is possible that we cannot distinguish between nodes at same level with a common parent. For example, if in the network in Figure 4.1, edge SA has unit capacity, while edge AB and AC have capacity two. In this case it is easy to see that there exists t_0 such that $\Pi_B(t) = \Pi_C(t) = \Pi_A(t-1)$, $\forall t \geq t_0$. Clearly in this case, we cannot distinguish between nodes B and C with this dissemination protocol.

4.2 General Topologies

Consider now an arbitrary network topology, corresponding to a directed graph. An intuition we can get from examining tree structures is that, we can distinguish between two topologies provided all node subspaces are distinct. The following theorem[3] claims that this is in fact a sufficient condition for topology identifiability over general graphs.

Theorem 4.2. *In a synchronous network employing randomized network coding over $\mathbb{F}_q$, a sufficient condition to uniquely identify the topology with high probability as $q \gg 1$, is that*

$$\Pi_i(t) \neq \Pi_j(t) \quad \forall\, i, j \in V, \quad i \neq j, \tag{4.3}$$

for some time t. We can achieve this by collecting global information at times t and $t+1$, i.e., two consecutive static views of the network.

Proof. Assume node i has the p_i parents $P(i) = \{u_1, \ldots, u_{p_i}\}$. Let $\Pi_i^{(u_1)}(t), \ldots, \Pi_i^{(u_{p_i})}(t)$ denote the subspaces node i has received from its parents up to time t, where $\Pi_i(t) = \cup_{j=1}^{p_i} \Pi_i^{(u_j)}(t)$. From construction it is clear that $\Pi_i^{(u_j)}(t+1) \subseteq \Pi_{u_j}(t)$.

To identify the network topology, it is sufficient to decide which node $v \in V$ is the parent that sent the subspace $\Pi_i^{(u_j)}(t)$ to node i for each j, and thus find the p_i parents of node i. We claim that, provided (4.3) holds, node i has as parent the node v which at time t has the smallest dimension subspace containing $\Pi_i^{(u_j)}(t+1)$. Thus we can uniquely identify the network topology, by two static views, at times t and $t+1$, as Algorithm 4.2.1 describes.

Indeed, let $\pi_i^{(u_j)}(t)$ denote the subspace that node i receives from parent u_j at exactly time t, that is, $\Pi_i^{(u_j)}(t+1) = \Pi_i^{(u_j)}(t) \cup \pi_i^{(u_j)}(t+1)$.

• If $\pi_i^{(u_j)}(t+1) \nsubseteq \Pi_v(t)$ for all $v \in V \backslash \{u_j\}$, clearly $\Pi_i^{(u_j)}(t+1) \nsubseteq \Pi_v(t)$ for all $v \in V \backslash \{u_j\}$, and we are done.

[3] Note that if we identify the parents of each node, we know the graph topology.

• Assume now there exist two nodes j and k such that $\Pi_i^{(u_j)} \subseteq \Pi_j \subset \Pi_k$. From Lemma 3.3, node i cannot be a child of node k, because then we would have that $\pi_i^{(u_j)} \nsubseteq \Pi_j$, and as a result, $\Pi_i^{(u_j)} \nsubseteq \Pi_j$. Thus it can only be a child of node j. □

Note that to identify the network topology, we need to know, for all nodes i, the dimension of their observed subspaces at time t, the dimension $\hat{d}_{u_j^{(i)}} \triangleq \dim(\Pi_i^{(u_j)}(t+1))$ for all parents j of node i, and the dimension of the intersection of $\Pi_i^{(u_j)}(t+1)$ with all $\Pi_k(t)$, denoted as $\hat{d}_{ku_j^{(i)}} \triangleq \dim(\Pi_i^{(u_j)}(t+1) \cap \Pi_k(t))$. Algorithm 4.2.1 uses this information to infer the topology.

```
Algorithm 4.2.1: GEN($\{d_i(t)\}, \{\hat{d}_{u_j^{(i)}}\}, \{\hat{d}_{ku_j^{(i)}}\}$)

for each $i \in V$
   do { if $d_i = n, i \leftarrow S$
        else node $i$ has parent the node $j$ with
        $j = \text{argmin}_{k:\hat{d}_{ku_j^{(i)}} = \hat{d}_{u_j^{(i)}}} d_k(t)$
```

The sufficient conditions in (4.3), Theorem 4.2, may or may not hold, depending on the network topology and the information dissemination protocol. Next, we will investigate under what conditions there exist values $\{m_i\}$ for the simple dissemination algorithm 4.1.1 so that (4.3) holds, and the network topology is identifiable.

Lemma 4.1. *Consider two arbitrary nodes i and j, where $P(i) = \{u_1, \ldots, u_{p_i}\}$ and $P(j) = \{v_1, \ldots, v_{p_j}\}$ are the parents of i and j respectively. Let $\Pi_{P(i)}(t-1) = \cup_{l=1}^{p_i} \Pi_{u_l}(t-1)$ and $\Pi_{P(j)}(t-1) = \cup_{l=1}^{p_j} \Pi_{v_l}(t-1)$. The condition $\Pi_{P(i)}(t-1) \neq \Pi_{P(j)}(t-1)$ is sufficient to guarantee that $\Pi_i(t) \neq \Pi_j(t)$.*

Proof. Let us assume that $\Pi_i(t) = \Pi_j(t) = \Pi$. This implies that if $\pi_i(t)$ and $\pi_j(t)$ are subspaces collected at time t then,

$$\pi_i(t) \cup \Pi_i(t-1) \quad = \quad \pi_j(t) \cup \Pi_j(t-1) = \Pi.$$

From construction, $\Pi_i(t-1) \subseteq \Pi_{P(i)}(t-1)$ and $\pi_i(t) \subseteq \Pi_{P(i)}(t-1)$ so we have $\Pi \subseteq \Pi_{P(i)}(t-1)$. The same is true for node j, $\Pi \subseteq \Pi_{P(j)}(t-1)$.

On the other hand, using Lemma 3.3, since we randomly chose $\pi_i(t)$ from $\Pi_{P(i)}(t-1)$ and since $\pi_i(t)$ is a subspace of Π, we should have that $\Pi_{P(i)}(t-1) \subseteq \Pi$, and similarly that $\Pi_{P(j)}(t-1) \subseteq \Pi$. We conclude that

$$\Pi_{P(i)}(t-1) = \Pi_{P(j)}(t-1) = \Pi,$$

which gives us the result. □

Now consider the parents of nodes i and j as supernodes $P(i)$ and $P(j)$. Using a similar argument we can conclude that the parents of $P(i)$ and $P(j)$, denoted as $P^2(i)$ and $P^2(j)$, satisfy

$$\Pi_{P^2(i)}(t-2) = \Pi_{P^2(j)}(t-2) = \Pi,$$

where $\Pi_i(t) = \Pi_j(t) = \Pi$, and $\dim(\Pi) = d < n$. Continuing this procedure, and including at least one new node in the set of parents at each step, we will at some step ℓ, either have $P^\ell(i)$ include the source node S, which leads to a contradiction since the dimension of the subspace $\Pi_{P^\ell(i)}(t-\ell)$ is $d < n$ (similarly if $P^\ell(j)$ includes the source S), or that $P^\ell(i) = P^\ell(j)$. To resolve this last case, we evoke the following theorem.

Theorem 4.3. *Suppose two arbitrary nodes i and j have a common set of parents $P^\ell = P^\ell(i) = P^\ell(j)$ at a level ℓ. The following conditions are sufficient to let exist some $\{m_i\}$ for the Algorithm 4.1.1 such that (4.3) will be satisfied*[4]*:*

$$\hat{c}_i = \textit{min-cut}(P^\ell, i) \le \textit{min-cut}(S, P^\ell) = c_p,$$
$$\hat{c}_j = \textit{min-cut}(P^\ell, j) \le \textit{min-cut}(S, P^\ell) = c_p.$$

Proof. Let us assume that t_0 is the first time that $\dim(\Pi_{P^\ell}) \ge c_p + 1$ and the time after which P^ℓ receives innovative packets at a rate of c_p. Assume that P^ℓ starts transmission after t_0. For t_1 time slots later we can write

$$\dim(\Pi_{P^\ell}(t_0 + t_1)) \ge t_1 c_p + c_p + 1.$$

For node i we can also write

$$\dim(\Pi_i(t_0 + t_1 + l)) \le (t_1 + 1)\hat{c}_i \le t_1 c_p + c_p.$$

The same inequality holds for the dimension of $\Pi_j(t_0 + t_1 + l)$. Thus for $t - l > t_0$ we cannot have $\Pi_{P^\ell}(t-l) = \Pi_i(t)$ and $\Pi_{P^\ell}(t-l) = \Pi_j(t)$. Using Lemma 4.1 we are done. □

Intuitively, what the previous theorems tell us is that, if for a node i there exists a path that does not belong in any cut between the source and another node j, then nodes i and j will definitely have distinct subspaces. The only case where nodes i and j may have the same subspace is, if they have a common set of parents, a common cut. Even then, they would need both of them to receive all the innovative information that flows through the common cut at the same time. Note that the condition of Theorem 4.3 are also necessary for identifiability for the special case of tree topologies, such as the topology in Figure 4.1. We can develop dissemination techniques for general topologies that satisfy the sufficient conditions given in Theorem 4.3 by using a decentralized rate control strategy. This can be done with almost no affect on the dissemination rate.

[4]Note that if $c_i = \text{min-cut}(S, i)$, $c_i = \min\{\hat{c}_i, c_p\}$.

4.3 Conclusion and Discussion

In this chapter we have shown that (for a class of graphs) one could design network coding algorithms which reveal topological structure of the graph while not affecting the dissemination rates. This connection between subspaces of network coded packets and network properties could be useful in other contexts as well. We have only considered the case where the identifiability occurs at *any time* during the steady state, and sufficient conditions for this. If one relaxes this or has further prior information about the network topology, one could also design other schemes.

Chapter 5

On Locating Byzantine Attackers

In a network coded system, the adverserial nodes in the network disrupt the normal operation of the information flow by inserting erroneous packets into the network. This can be done by inserting spurious data packets into their outgoing edges. One way in which these erroneous packets can be prevented from disrupting information flow is by reducing the transmission rate to below the min-cut of the network, and using the redundancy to protect against errors. One such technique, using subspaces to code information was proposed in [15]. In this approach, the source sends a basis of the subspace corresponding to the message. In the absence of errors, the linear operations of the intermediate nodes do not alter the sent subspace, and hence the receiver decodes the message by collecting the basis of the transmitted subspace. A malicious attacker inserts vectors that do not belong in the transmitted subspace. Therefore, if the message codebook uses subspaces that are "far enough" apart (according to an appropriately defined distance measure), then one can correct these errors [15]. Note that in this technique, we do not need any knowledge of the network topology for the error correction mechanism. All that is needed is that the intermediate nodes do not alter the transmitted subspace (which can be done if they do linear operations).

The approach of this chapter to locating adversaries uses the framework developed in the previous chapters, where it was shown that under randomized network coding, the subspaces at the nodes of the network give information about the topology. Therefore, the basic premise in this chapter is to use the structure of the erroneous subspace inserted by the adversary to reveal information about its location, when we already know the network topology.

5.1 Problem Formulation

Consider a network represented as a directed acyclic graph $G = (V, E)$. We have a source, sending information to r receivers, and one (or more) Byzantine adversaries, located at intermediate nodes of the network. We assume complete knowledge of the network topology, and consider the source and the receivers to be trustworthy (authenticated) nodes, that are guaranteed not to be adversaries.

We can restrict the Byzantine attack in several ways, depending on the edges where the attack is launched, the number of corrupted vectors inserted, and the vertices (network nodes) that the adversary has access to. In this chapter we will distinguish between the cases where

I. there is a single Byzantine attacker located in a vertex of the network, and

II. there are multiple independent attackers, located on different vertices, that act without coordinating with each other.

Moreover, we will consider the cases where, an attacker located on a single vertex inserts corrupted packets on

(a) exactly one of the vertex outgoing edges,

(b) all the outgoing edges, or

(c) a subset of the outgoing edges.

We are interested in understanding under what conditions we can uniquely identify the attacker's location (or, up to what uncertainty we can identify the attacker), under the above scenarios.

5.2 Basic Properties

In this section, we will use the subspace model introduced in the Section 2.4 to find the basic properties of subspace in the presence of adversaries in the network.

Suppose source S sends n vectors, that span a n-dimensional subspace Π_S of space W, where W is defined over a finite field $\mathbb{F}_q$, with $q \gg 1$. In particular, Π_S belong to a codebook $\mathcal{C}$, $\Pi_S \in \mathcal{C}$, which is designed to correct network errors [15].

In the absence of any erasures or adversaries in the network each receiver R collects the exact space Π_S. Now assume that there is an adversary who attacks one of the nodes in the network by combining a t-dimensional subspace Π_E with its incoming space and sending the resulting vectors to its children. In addition we will assume that $t < n$. Then receiver R collects $n \leq m \leq n+t$ innovative vectors that span a subspace Π_R. We may write

$$\Pi_R = \mathcal{H}_m(\Pi_S \cup \Pi_E),$$

where $\mathcal{H}_m$ is the erasure operator defined in Section 2.4. The operator $\mathcal{H}_m$ depends on the topology of the network and the code that used in the network. This is a general model which includes all class of linear network coding senario, but as stated in Section 2.4, if nodes in the network perform random network coding, $\mathcal{H}_m$ has some random structure.

We assume that the receiver is able to at least detect that a Byzantine attack is under way. Moreover, we assume that the receiver is able to decode the subspace Π_S that the source has sent. This might be, either because the receiver has correctly decoded the sent message, or, because after detecting the presence of an attack has requested the source subspace through a secure channel from the source node.

Here, based on the model introduce above, we investigate how the insertion of the error subspace Π_E affects the subspaces that the intermediate network nodes observe.

We can write the received subspace at arbitrary node i the same way as we did for the receiver R,

$$\Pi_i = \mathcal{H}_m^i(\Pi_S \cup \Pi_E).$$

Then it is possible to expand Π_i as follows,

$$\begin{aligned} \Pi_i &= \hat{\Pi}_{Si} \oplus \underbrace{(\hat{\Pi}_{Ei} \oplus \hat{\Pi}_i)}_{\Pi_i^\star} \\ &= \hat{\Pi}_{Si} \oplus \Pi_i^\star, \end{aligned} \tag{5.1}$$

where $\oplus$ denotes the direct sum of spaces, $\hat{\Pi}_{Si} \triangleq \Pi_i \cap \Pi_S \subseteq \Pi_S$, $\hat{\Pi}_{Ei} \triangleq \Pi_i \cap \Pi_E \subseteq \Pi_E$ and $\hat{\Pi}_i$ is the rest of Π_i which cannot be represented as just part of Π_S or Π_E. We underline that in general $\Pi_i^\star \nsubseteq \Pi_E$.

If the operator $\mathcal{H}_m^i$ selects Π_i uniformly at random, with high probability we will have

$$\begin{aligned} \dim(\hat{\Pi}_{Si}) &= m-t,\ \dim(\hat{\Pi}_{Ei}) = m-n, \\ \dim(\hat{\Pi}_i) &= t-(m-n). \end{aligned}$$

The above results are a direct consequence of Corollary 3.1, and provide a lower bound for these dimensions in the general case. Using again Corollary 3.1, for two arbitrary nodes i and j, and without any further assumptions, we have the following inequalities

$$\begin{aligned} \dim(\Pi_i \cap \Pi_j) &\geq 2m-(n+t) \\ &= (m-t)+(m-n), \end{aligned}$$

and

$$\begin{aligned}\dim(\hat{\Pi}_{Si} \cap \hat{\Pi}_{Sj}) &\geq 2(m-t) - n \\ &= 2(m-n) + (n-2t).\end{aligned}$$

Lemma 5.1. *In the above scenario, two arbitrary node i and j in the network should gather $m > n + t/2$ innovative vectors to have* $\dim(\Pi_i^\star \cap \Pi_j^\star) > 0$.

Proof. From Corollary 3.1 we can write

$$\dim(\hat{\Pi}_{Ei} \cap \hat{\Pi}_{Ej}) \geq 2(m-n) - t.$$

So if we have $n + t/2 < m$, the two subspaces $\Pi_i^\star = \hat{\Pi}_{Ei} \oplus \hat{\Pi}_i$ and $\Pi_j^\star = \hat{\Pi}_{Ej} \oplus \hat{\Pi}_j$ have nonzero intersection. □

Thus we conclude that, if an adversary introduces Π_E, and intermediate nodes perform randomized network coding, it is not necessary that the nodes collecting corrupted information will collect a subspace of Π_E. Additionally, two nodes that collect corrupted information, may only have as common information a subspace of Π_S, unless they collect a sufficient number of innovative packets.

5.3 The Case of a Single Adversary

In this section we focus on the case where we want to locate a Byzantine adversary controlling a *single* vertex of the network graph. We will develop methods which are suitable for the cases where the adversary corrupts one edge, all edges, or a subset of its out-going edges (cases $(a), (b), (c)$ respectively of Section 5.1.

In Section 5.3.1 we illustrate the limitation of using *only* the information the receivers have observed along with the knowledge of the topology, to locate the adversary. This motivates requiring additional information from the intermediate nodes related to the subspaces observed by them. In Section 5.3.2, we show that such additional information allows us to localize the adversary either uniquely or within an ambiguity of at most two nodes.

5.3.1 Identification using only Topological Information

In order to illustrate the ideas, we will first examine the case where the corrupted packets are inserted on a single edge of the network, say edge e_A. This corresponds to case (a) in Section 5.1. The extension to cases (b) and (c) is straightforward.

Since each receiver R knows the subspaces $\{\Pi_R^{(i)}\}$ it has received from its $\text{In}(R)$ parents, it knows whether what it received is corrupted or not (a subspace of Π_S or not). Using this, we can infer some information regarding topological properties that the edge e_A should satisfy. In particular:

- If R receives corrupted vectors from an incoming edge e then there exists at least one path that connects e_A to e. Let P_e denote the set of paths[1] starting from the source and ending at edge e. Then e_A is part of at least one path in P_e.

- Conversely, if a receiver R does not receive corrupted packets from an incoming edge e, then e_A does not form part of any path in P_e. That is, there does not exist a path that connects e_A to e.

Then, if $\mathcal{E}_1$ is the set of incoming edges to receivers that bring corrupted packets, while $\mathcal{E}_2$ the set of incoming edges to receivers that only bring source information, the edge e_A belongs in the set of edges $\mathcal{E}_A$, with

$$\mathcal{E}_A \triangleq \{\bigcap_{e\in\mathcal{E}_1} P_e - \bigcup_{e\in\mathcal{E}_2} P_e\},$$

where $\mathcal{A} - \mathcal{B}$ denotes the set where elements in $\mathcal{B}$ are removed from the set $\mathcal{A}$. The following example illustrates this approach.

Example 5.1. Consider the network in Figure 5.1, and assume that R_1 receives corrupted packets from edge DR_1 and uncorrupted packets from AR_1, while R_2 receives only uncorrupted packets. Then $\mathcal{E}_A = \{DR_1\}$ and

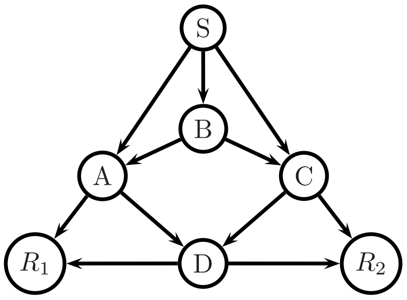

Figure 5.1: The source S distributes packets to receivers R_1 and R_2.

the attacker is located on node D. □

In Example 5.1, we were able to exactly identify the location of the adversary, because the set $\mathcal{E}_A$ contained a single edge, and node R_1 is trustworthy. It is easy to find network configurations where $\mathcal{E}_A$ contains multiple edges, or in fact all the network edges, and thus we can no longer identify the attacker. The following example illustrates one such case.

[1]In the following we are going to equivalently think of P_e as the set of all edges that take part in these paths.

Example 5.2. Consider the line network shown in Figure 5.2. Suppose the attacker is node A. If the receiver R sees a corrupted packet, then using just the topology, the attacker could be *any* of the other nodes in the line network. This illustrates that just the topology and receiver information could lead to large ambiguity in the location of the attacker. $\square$

Therefore, Example 5.2 motivates the ideas examined in Section 5.3.2 which obtain additional information and utilize the structural properties of the subspaces observed.

5.3.2 Identification using Information from all Network Nodes

We will next discuss algorithms where a central authority, which we will call *controller*, requests from all nodes in the network to report some additional information, related to the subspaces they have received from their parents. The adversary could send inaccurate information to the controller, but the other nodes report the information accurately. Our task is to design the question to the nodes such that we can locate the adversary, despite its possible misdirection.

The controller may ask the nodes of the following types of information, listed in decreasing order of complexity:

Information 1: Each node v sends all subspaces $\Pi_v^{(i)}$ it has received from its parents, where $\Pi_v = \cup_{i=1}^{\mathrm{In}(v)} \Pi_v^{(i)}$.

Information 2: Each node v sends a randomly chosen vector from each of the received subspaces $\Pi_v^{(i)}$ ($\mathrm{In}(v)$ vectors in total).

Information 3: Each node v sends one randomly chosen vector from its subspace Π_v.

Information 2 and 3 is motivated by the following well-known observation, see Lemma 3.3: let Π_1 and Π_2 be two subspaces of $\mathbb{F}_q^n$, and assume that we randomly select a vector y from Π_1. Then, for $q \gg 1$, $y \in \Pi_2$ if and only if $\Pi_1 \subseteq \Pi_2$. Thus, a randomly selected vector from Π_v (Information 3) allows to check whether $\Pi_v \subseteq \Pi_S$ or not.

In fact, we will show in this section that for a single adversary it is sufficient to use [2] Information 2, and classify the edges of the network by simply testing whether the information flowing through each edge is a subspace of Π_S or not (i.e., is corrupted or not).

[2]Using Information 2 or 3 these statements are made with high probability, *i.e.,* the probability goes to one as field size $q \to \infty$.

The Line Network

To build the intuition behind our approach, we first examine the case of the line network, depicted in Figure 5.2, that corresponds to a single path connecting the source to the receiver. We saw in Example 5.2, that just topological information was insufficient to reduce ambiguity of the attacker's location. For the line network, cases $(a), (b), (c)$ in Section 5.1 coincide.

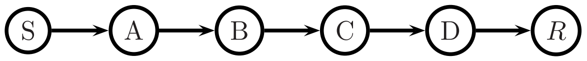

Figure 5.2: The source S sends information to receiver R over a line network.

Assume now that the controller asks for Information 1, *i.e.,* all nodes to report their collected subspaces to the controller. The adversary has two courses of action: it can either correctly report the subspace it received from its parent node, or lie, and claim that it received a corrupted subspace from its parent. We do not know which of the two approaches the adversary has selected. However, in both cases, we can divide the network edges into two sets, the set of edges through which is reported to flow correct information, and the set of edges through which is reported to flow corrupted information.

For example, if the adversary is node C in Figure 5.2, the sets corresponding to the possible adversary actions are depicted in Figure 5.3(i) and 5.3(ii) respectively. It is clear that the adversary is one of the two nodes connecting the edge on the border of the sets, that is, in the set of vertices $\{C,\ D\}$ for the case in Figure 5.3(i), and in the set $\{B,\ C\}$ for the case in Figure 5.3(ii). In particular, the adversary is one of the two adjoining nodes, of the first ancestral reported corrupted edge.

Note that we can divide the edges in these two sets simply using Information 2 or 3 (these coincide for the line network) since it is sufficient to check whether each edge is corrupted.

Also note that networks where all nodes have out-degree one and arbitrary in-degree, can be treated in exactly the same way as the line network. Indeed, the identification in such networks can be, without loss of generality, decomposed in identification along single paths.

General Networks

Consider a directed acyclic graph, and assume that we impose a partial order on the edges of the graph, such that $e_1 > e_2$ if e_1 is an ancestor edge of e_2 (i.e., there exists a path from e_1 to e_2).

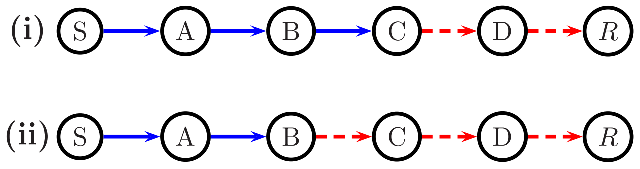

Figure 5.3: Case (i): edge partition if the adversary node C reports the truth, and Case (ii): edge partition if the adversary node C lies. Edges bringing corrupted information are depicted as dashed.

Following a similar approach to Section 5.3.2-1), and using Information 2, we divide the edges of the network into two sets: the set of edges E_C through which are reported to flow corrupted subspaces, and the remaining edges E_S through which the source information flows. Note that all the outgoing edges from the source belong in E_S, while the receiver observes at least one edge in E_C.

Consider case (a), where the adversary corrupts a single edge. Clearly, since there exists a single adversary, E_C forms a connected subgraph. Let e_A be highest order edge in this graph, i.e., $e_A > e$ for all $e \in E_C$. Then, similarly to the case of the line network, the adversary is one of the two nodes adjacent to this edge. We can make similar arguments for cases (b) and (c). This leads to the following lemma.

Lemma 5.2. *Using Information 1 we can narrow the location of the adversary up to a set of at most two nodes. With Information 2, the same result holds with high probability.*

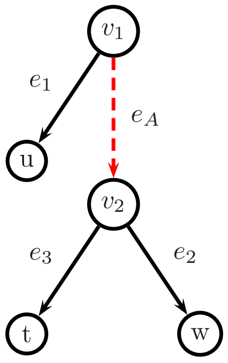

Figure 5.4: Edge e_A and neighboring edges-nodes.

In fact, in some cases, we are able to uniquely identify the malicious attacker, as described by the following lemma.

Lemma 5.3. *If e_A connects vertex v_1 to vertex v_2, and if vertices v_1 and v_2 have outdegree greater or equal to two, then we can uniquely identify*[3] *the attacker in cases (a) and (b) of Section 5.1.*

Proof. The proof is based on the fact that only a single node can lie. Thus we know that every other node, apart from v_1 or v_2, is trustworthy. Let e_1, e_2 and e_3 be outgoing edges of vertex v_1 and v_2 as depicted in Figure 5.4.

- Case (a) (the adversary corrupts a single edge): If e_2 and e_3 are corrupted, the adversary can be only located on vertex v_1, while if only one of them is corrupted, the adversary is located on v_2.
- Case (b) (the adversary corrupts all its outgoing edges): If e_1 is corrupted, the adversary is located on vertex v_1, and otherwise on v_2.

□

5.4 The Case of Multiple Adversaries

In the case of a single adversary, it was sufficient to divide the set of edges into two sets, E_S and E_C, as described in the previous section. In the presence of multiple adversaries, this may no longer be sufficient. An additional dimension is that realistically, we may not know the exact number of adversaries present. In the following, we discuss a number of algorithms, that offer more or less identifiability guarantees.

5.4.1 Identification using Topological Information

The approach in Section 5.3.1 can be directly extended in the case of multiple adversaries, but again, offers no identifiability guarantees.

Example 5.3. Consider again the network in Figure 5.1, and assume that R_1 receives corrupted packets only from edge DR_1 while R_2 receives corrupted packets only from edge DR_2. Then $\mathcal{E}_A = \{AD, CD, DR_1, DR_2\}$ and (depending on our assumptions) we may have,

- a single adversary located on node D,
- two adversaries, located on nodes A and C,
- two adversaries, located on nodes A and D, or nodes C and D, or
- three adversaries, located on nodes A, C, and D.

□

[3] Again, this occurs w.h.p. for Information 2 and 3.

5.4.2 Identification using Information 2

Similarly to Section 5.3.2, we can divide the set of edges into two sets E_S and E_C, depending on whether the information flowing through each edge belongs in Π_S or not. Depending on the network topology, we may be able to uniquely identify the location of the attackers. However, this approach, although it guarantees to find at least one of the attackers (within an uncertainty of at most two nodes), does not necessarily find all the attackers, even if we know their exact number.

Intuitively, this is because an attacker might be "in the shadow" of another attacker, meaning that, it may corrupt only already corrupted vectors and thus not incur a detectable effect. More precisely, we say that node B is in the shadow of node A, if there exists a path that connects every incoming edge of B to a corrupted outgoing edge of A. The following example illustrates these points.

Example 5.4. For the example in Figure 5.1, assume that each attacker corrupts all its outgoing edges, and consider the following two situations:

1. Assume that nodes A and C are attackers. If A reports truthfully while C lies we get $E_C = \{AD, AR_1, DR_1, DR_2, BC, CR_2, CD\}$, which allows to identify the attackers.

2. Assume that nodes B and D are attackers. Then we say that node D is in the shadow of node B, as it corrupts only already packets corrupted by B. Indeed, if $E_C = \{SB, BA, BC, AD, AR_1, DR_1, DR_2, BC, CR_2, CD\}$, knowing that the source is trustworthy, we can infer that node B is an attacker. However, any of the nodes A, C, and D can equally probably be the second attacker. All these nodes are in the shadow of node D.

□

5.4.3 Identification using Clustering and Information 1

In this section, we are going to try to explicitly use the knowledge of all the node's subspaces, and the model that is introduced in Section 5.2.

Suppose that we have two adversaries A_1 and A_2 that inject two spaces Π_{E_1} and Π_{E_2} with dimensions t_1 and t_2 into the network. For simplicity assume that $t_1 = t_2 = t$. Since the adversaries are independent, t relatively small, and $q \gg 1$, we can assume w.h.p. that $\Pi_{E_1} \cap \Pi_{E_2} = 0$.

Based on the received spaces, we can attempt to divide the vertices of the network into four parts, which we will call vertex-clusters. Let us label these four clusters as V_S, V_1, V_2 and V_{12}. Through cluster V_S flows the source information, through cluster V_1 information corrupted by A_1, through cluster V_2 information corrupted A_2 and through cluster V_{12} information corrupted by both A_1 and A_2. Following the notation in Section 5.2, we can decompose each space as follows,

$$\Pi_i = \hat{\Pi}_{Si} \oplus \Pi_i^\star.$$

It is obvious that for node $i \in V_S$ we have

$$\dim(\Pi_i^\star) = 0.$$

From Lemma 5.1, we know that for two vertices i and j both belonging to the same cluster V_1 (or V_2, or V_{12}) we have

$$\dim(\Pi_i^\star \cap \Pi_j^\star) > 0,$$

if the number of gathered innovative vectors is sufficiently large. Moreover, if i belongs to V_1 and j to V_2 then $\hat{\Pi}_i \cap \hat{\Pi}_j = \emptyset$. Finally, if i belongs to V_1 (or V_2) and j to V_{12} we want to have

$$\dim(\Pi_i^\star \cap \Pi_j^\star) > 0.$$

Thus, by examining the intersection of the edge subspaces, we can divide the vertices into two overlapping sets, V_{A_1} and V_{A_2}, where all vertices in V_{A_1} (V_{A_2}) have nonzero intersection. Then, $V_{12} = V_{A_1} \cap V_{A_2}$, $V_1 = V_{A_1} - V_{12}$ and $V_2 = V_{A_2} - V_{12}$. Given the partition, we know that the first attacker belongs in the border of cluster V_1 while the second in the border of cluster V_2. Explicitly examining the subspaces on the incoming edges that cross these two borders allows to identify which of them are corrupted. From the problem definition, exactly the edges adjacent to nodes are corrupted, which allows to identify the adversaries.

Note that this approach works when V_1 and V_2 form distinct clusters (irrespective of whether V_{12} exists or not), but again fails, like the previous algorithm, in the case where the second attacker V_2 "belongs" in V_1, that is, only clusters V_1 (or V_2) and V_{12} exist. This is because, we will not be able to identify cluster V_{12}.

5.4.4 Identification using Subset Relationships

For each node $i \in V$, let $P(i) = \{u_1, \cdots, u_{p_i}\}$ denote the set of parent nodes of i. We are going to treat $P(i)$ as a super node, and use the notation $\Pi_{P(i)} = \cup_{l=1}^{p_i} \Pi_{u_l}$ for the union of the subspaces of all nodes in $P(i)$. Also recall that $\Pi_j^{(i)}$ denotes the subspace received by node j from node i.

Our last algorithm checks, for every node i, whether

$$\Pi_j^{(i)} \overset{?}{\subseteq} \Pi_{P(i)} \quad \text{for node } j \text{ s.t. } e_{ij} \in E.$$

If this relationship is satisfied, we know that node i is not an adversary. If the relationship is not satisfied, that is $\Pi_j^{(i)} \nsubseteq \Pi_{P(i)}$ for at least one of the children of i, we know that maybe node i is an adversary. For sure we know that

$$\Pi_j^{(A)} \nsubseteq \Pi_{P(A)} \quad \text{for node } j \text{ s.t. } e_{Aj} \in E,$$

but depending on the subspace that the adversary reports, the above relation may not be satisfied for other nodes.

If the adversary pretends that it is a trustworthy node (just declares its received subspace from the parents) the above relation also fails for the children of A who receive corrupted subspaces. On the other hand, if the adversary tells the truth and declares its whole subspace, we have

$$\Pi_A^{(i)} \nsubseteq \Pi_{P(i)} \quad \text{for all parents } i \text{ of } A.$$

Thus the ambiguity set we have identified includes the adversary and its parents or children depending on the adversary's report.

Now it is possible to use the topological information to make the ambiguity set smaller. We know that the set contains the adversary and either its parent or its children. So the potential adversaries are nodes that are parents or children of all other nodes in the set. If there exists only one such node in the set we can identify the location of the adversary uniquely.

Repeating this procedure for every node in the network, we can identify sets of potential adversaries. This procedure allows to identify adversaries, even if one is in the shadow of another, and even if we do not know their exact number, provided they are "far enough" in the network to be distinguishable. More precisely, we have the following lemma, where distance refers to the length of the shortest path connecting two nodes.

Lemma 5.4. *If the pairwise distance between adversaries is greater than two, it is possible to find the exact number as well as the location of the attackers (within the described uncertainty of parent-children sets), using the subset method.*

Proof. We know that depending on the adversaries action there exists ambiguity in finding their exact location. In fact in the worst case, the uncertainty is within a set of nodes including the adversary, its parents and its children. So if the distance between adversaries is greater than two, the "uncertainty" sets do not overlap. In this case we can easily distinguish between different adversaries. □

5.5 Conclusion and Discussion

Given a network subject to Byzantine attacks, in this chapter we formulated and examined the problem of locating the adversaries. We showed that in the case of a single adversary, there exist simple algorithms that allow to identify the adversary within an uncertainty of two nodes. For the case of multiple adversaries, we discussed a number of algorithms, and conditions under which we can guarantee identifiability.

Chapter 6

P2P Topology Management

The performance of peer-to-peer (P2P) networks depends critically on the good connectivity of the overlay topology. In this chapter we study P2P networks for content distribution (such as Avalanche) that use randomized network coding techniques. We will use that fact 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. The proposed 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.

6.1 Description and Motivation

6.1.1 Avalanche Topology Management

In a nutshell, Avalanche relies on periodically renewed random selections for the peer neighbors to rewire the employed overlay network [13, 14]. 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.

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,

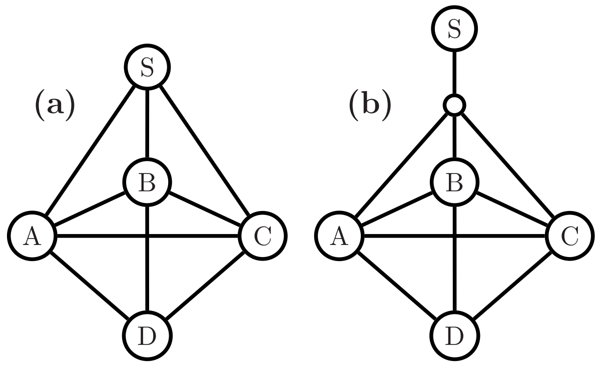

Figure 6.1: The source S distributes packets to the peers A, B, C and D over the overlay network (a), that uses the underlying physical network (b).

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.

6.1.2 The Proposed 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 Figure 6.1(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 Figure 6.1(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 6.2. 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.

6.2 Theoretical Framework

Here we use the same theoretical model that introduced in Section 2.1 and 2.3. 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[1]. 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.

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

In addition to the metric D_{ij} defined in Section 2.3, in some cases 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 pair-wise distance defined as follows

$$D_{\mathcal{AB}} \triangleq \frac{1}{|\mathcal{A}||\mathcal{B}|} \sum_{i\in\mathcal{A},j\in\mathcal{B}} D_{ij}. \tag{6.1}$$

It should be noted that the above relation does not define a metric for the clusters of nodes because the self distance of a cluster with itself is not zero but it satisfies the triangle inequality.

Now we will use the results of Chapter 3 to investigate the information that we can obtain from the local information of a node's subspace. From Section 2.3 we know that for an arbitrary node i we can write

$$\Pi_i(t) = \cup_{j=1}^{p_i} \Pi_i^{(u_j)}(t).$$

We are interested in understanding what information we can infer from these received subspaces $\Pi_i^{(u_1)}, \ldots, \Pi_i^{(u_{p_i})}$. 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.

Let us assume that the subspaces $\Pi_i^{(u_1)}, \ldots, \Pi_i^{(u_{p_i})}$ a node i receives from its set of neighbors $\{u_j\}$ have an intersection of dimension d. This implies that, (i) from Corollary 3.1, the subspaces $\Pi_i^{(u_1)}, \ldots, \Pi_i^{(u_{p_i})}$ of the neighbors have an intersection of size at least d and (ii), from Thorem 3.3, the min-cut between the set of nodes $\{u_j\}$ and the source is smaller than the min-cut between the node i and set $\{u_j\}$. Next we will discuss algorithms that use such observations for topology management.

6.3 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 Theorem 3.3 and Corollary 3.1 and is also illustrated through simulation results in Section 6.4). 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.

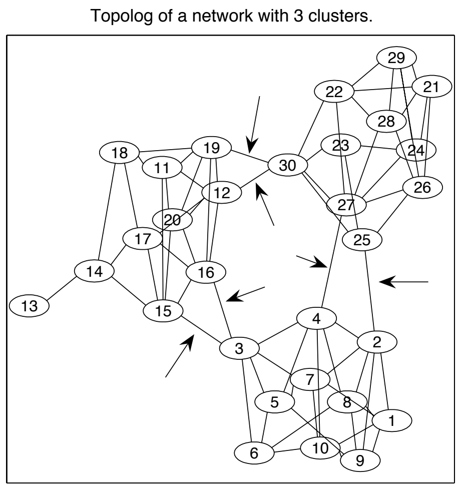

Figure 6.2: Topology with three clusters: cluster 1 contains nodes 1–10, cluster 2 nodes 11–20 and cluster 3 nodes 21–30.

6.4 Simulation Results

For our simulation results we will start from the topology illustrated in Figure 6.2, 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 Figure 6.3 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 (6.1). 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

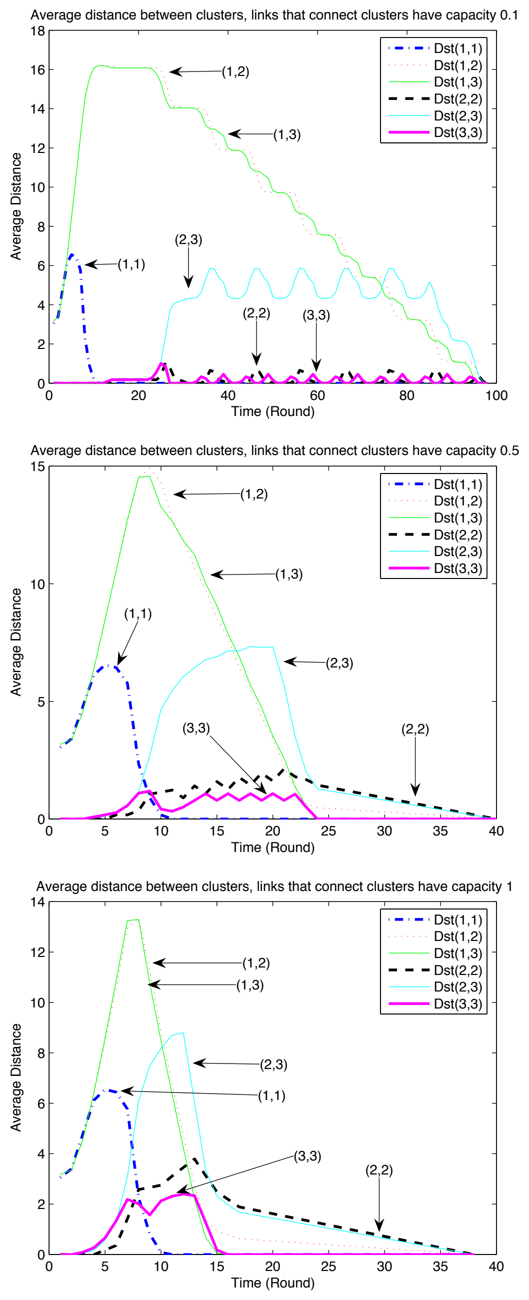

Figure 6.3: Simulation results for the topology in Figure 6.2, with bottleneck link capacity values equal to 0.1 (top), 0.5 (middle) and 1 (bottom).

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 6.3 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 2.3.

Table 6.1 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.

Figure 6.4 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.

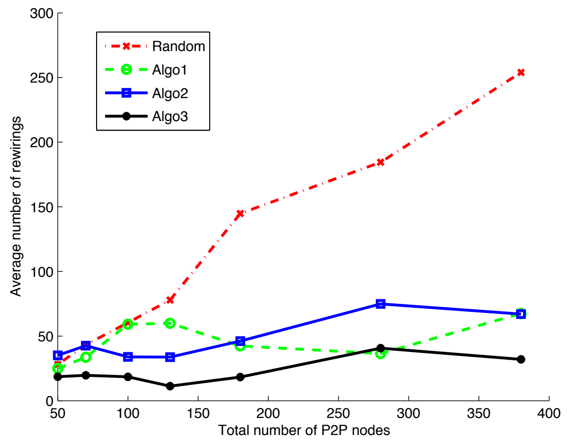

Figure 6.4: Average number of rewirings, for a topology with three clusters: cluster 1 has 15 nodes, cluster 2 has 15 nodes, while the number of nodes in cluster 3 increases from 20 to 250 as described in Table 1.

Table 6.1: 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

6.5 Conclusion and Discussion

In this chapter 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.

Conclusions and Discussion

In this work we explored the properties of subspaces each node collects in networks employ randomized network coding and found that there exists relations between the structure of the network and these properties. This observation led us to use these relations to propose different applications.

As the first application, we studied the conditions a network should satisfy to allow us passively infer the network topology. We showed that these conditions are not very restrictive and hold for a general class of graphs. Our future work includes proposing a disseminating protocol which applies rate control on the information flow that allow us to infer the network topology for all graphs.

Continuing the previous work, we investigated the relation between the bottlenecks in the network and the subspace received at a specific node. Then we emploied this observation and proposed decentralized algorithms for the rewiring between nodes to facilitate to flow of information in the network.

For the last application, we focused on locating the Byzantine attackers in the network. We studied and formulated problem and found that for the single adversary we can identify the adversary within an uncertainty of two nodes. For the case of multiple adversaries, we discussed a number of algorithms, and conditions under which we can guarantee identifiability. For the future work, we are interested in particular in developing decentralized algorithms, where nodes are willing to cooperate (exchange messages or certificates with their neighbors) and identify the Byzantine attacker in a distributed manner (without the use of a centralizer controller).

The applicatins studied in the previous chapters show additional advantages of the randomized network coding. These are just a few examples and we believe that there exists a lot more applications that can use the idea of this work.

Bibliography

[1] R. Ahlswede, N. Cai, S-Y. R. Li, and R. W. Yeung, "Network information flow", *IEEE Transactions on Information Theory*, Vol. 46, pp. 1204–1216, Jul. 2000.

[2] S.-Y. R. Li, R. W. Yeung, and N. Cai., "Linear network coding", *IEEE Transactions on Information Theory*, 49:371-381, 2003.

[3] R. Koetter, M. Medard, "An Algebraic Approach to Network Coding", *Transactions on Networking*, Oct. 2003.

[4] T. Ho, R. Kötter, M. Médard, M. Effros, J. Shi, and D. Karger, "A random linear network coding approach to multicast", *IEEE Transactions on Information Theory*, Vol. 52, pp. 4413-4430, Oct. 2006.

[5] P. A. Chou, Y. Wu, and K. Jain, "Practical network coding", *Allerton*, Monticello, IL, Oct. 2003.

[6] T. Ho, R. Koetter, M. Medard, D. Karger and M. Effros, "The Benefits of Coding over Routing in a Randomized Setting", *IEEE International Symposium on Information Theory*, 2003.

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

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

[9] M. Jafarisiavoshani, C. Fragouli, and S N. Diggavi, "Subspace properties of randomized network coding", *IEEE Information Theory Workshop*, pp 17–21, Bergen, Norway, Jul. 2007.

[10] M. Jafarisiavoshani, C. Fragouli, S N. Diggavi, and C. Gkantsidis "Bottleneck discovery and overlay management in network coded peer-to-peer systems", *ACM SIGCOMM Workshop on Internet Network Management*, Kyoto, Japan, Aug. 2007.

[11] R. Koetter, M. Mdard, "Beyond Routing: An Algebraic Approach to Network Coding", *IEEE Infocom*, 2002.

[12] T. Ho, B. Leong, Y. Chang, Y. Wen and R. Kötter, "Network monitoring in multicast networks using network coding", *IEEE International Symposium on Information Theory*, Jun. 2005.

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

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

[15] R. Koetter and F. Kschischang, "Coding for errors and erasures in random network coding", *IEEE International Symposium on Information Theory*, Jun. 2007.

[16] L. Babai and P. Frankl, "Linear Algebra Methods in Combinatorics", preliminary version, University of Chicago.

[17] S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, and M. Medard, "Resilient network coding in the presence of byzantine adversaries," *IEEE Infocom*, pp. 616–624, 2007.

[18] T. Ho, B. Leong, R. Koetter, M. Medard, M. Effros, and D. Karger, "Byzantine modification detection in multicast networks using randomized network coding", *IEEE International Symposium of Information Theory*, Jun. 2004.

[19] R. W. Yeung and N. Cai, "Network error correction, i: basic concepts and upper bounds," *Communication and Information System*, Vol. 6, pp. 19–35, 2006.

[20] N. Cai and R. W. Yeung, "Network error correction, ii: lower bounds," *Communication and Information System*, Vol. 6, pp. 37–54, 2006.

[21] Z. Zhang, "Network error correction coding in packetized networks," *Information Theory Workshop*, Oct. 2006.

[22] 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 Transactions on Information Theory*, Jul. 2003.

[23] "RON: Resilient Overlay Networks", available at http://nms.csail.mit.edu/ron.