



ICT - Information and Communication Technologies



Network Coding for Robust Architectures in Volatile Environments
Collaborative Project

Grant Agreement Number 215252

**APPLICATION LAYER ASPECTS OF NETWORK
CODING: REPORT ON STATE-OF-THE-ART**

Due Date of Deliverable: 31/12/08

Actual Submission Date: 23/01/09

Revision: Final

Start date of project: January 1st 2008

Duration: 36 months

Organization name of lead contractor for this deliverable: TELEFONICA

Authors: Juan Lara Ambel, Alberto Lopez Toledo

Contributors: Steluta Gheorghiu, Tobias Lutz, Paulo F. Oliveira, Christina Fragouli, Leandros Tassioulas, Savvas Gitzenis, Leonidas Georgiadis

Project Information

PROJECT	
Project name:	Network Coding for Robust Architectures in Volatile Environments
Project acronym:	N-CRAVE
Project start date:	01/01/2008
Project duration:	36 months
Contract number:	215252
Project coordinator:	Leandros Tassiulas – CERTH
Instrument:	STREP
Activity:	Theme Challenge 1: Pervasive and Trusted Network and Service Infrastructures
DOCUMENT	
Document title:	Application Layer Aspects of Network Coding: Report on State-of-the-Art
Document type:	Report
Deliverable number:	D3.1
Contractual date of delivery:	31/12/08
Calendar date of delivery:	22/01/09
Editor:	Juan Lara Ambel, Alberto Lopez Toledo
Authors:	Steluta Gheorghiu, Tobias Lutz, Paulo F. Oliveira, Christina Fragouli, Leandros Tassiulas, Savvas Gitzenis, Leonidas Georgiadis
Workpackage number:	WP3
Workpackage title:	Network Coding Experimentation
Lead partner:	TID
Dissemination level:	Public
Date created:	15/09/2008
Updated:	22/01/2009
Version:	v1
Total number of Pages:	45
Document status:	final

Contents

1	Introduction	4
2	Distributed Information Storage and Retrieval	6
2.1	Erasur Coding Distributed Storage Systems	7
2.2	Random Linear Coding Distributed Storage Systems	8
2.3	Network Coding based Distributed Storage Systems	10
3	Content Distribution	13
3.1	P2P Based Content Distribution	14
3.2	Content Distribution with Scalable and Multiple Description Source Coding	19
3.2.1	Layered Coding Approaches	20
3.2.2	Non-Layered Coding Approaches	21
4	Network Coding Based Security	23
4.1	Securing Network Coding Protocols	24
4.1.1	Countering Eavesdropping Attacks	24
4.1.2	Countering Byzantine Attacks	25
4.2	Combating Byzantine Attacks with Network Coding and Rank- metric error correcting codes	28
4.2.1	Network Model	28
4.2.2	Principles of Error Control in Random Network Coding	29
4.2.3	A Concrete Coding/Decoding Scheme	31
5	Network Monitoring and Management	32
5.1	Network Monitoring	33
5.2	Network Management	35
6	Network Coding Applications in Sensor Networks	36
7	Conclusions and Guidelines for Future Work	38

Abstract

We survey a representative subset of applications that use network coding techniques in different environments. We identified two major scenarios in which such applications are used: distributed information storage and retrieval, and content distribution. Moreover, we survey how the inherent properties of network coding can be applied to security and network monitoring, and study existing applications that use network coding in sensor networks. Last, we conclude with directions for future research.

1 Introduction

The objective of this working package is to explore the benefits that network coding may provide in order to optimize performance of applications. In particular, we will be addressing three main broad applications: distributed storage, content distribution, network management and monitoring and security.

In this document, we survey the state-of-the-art and present the main results in the literature regarding the above applications. While the ultimate objective is to deploy such applications in wireless environments, it is important to understand how existing works use network coding in order to enhance security, distributed storage and content distribution in any type of networks. We also explore how network coding has been used in different applications over *sensor networks*. Existing work in sensor networks can provide invaluable insight on the advantages that network coding provides in wireless environments, and how it takes advantage of the broadcasting capabilities of the wireless medium. The insight obtained by such exploration, together with the basic network coding body of knowledge in [1], and the network coding based protocols for wireless networks described in [2], provide the basis of the future work of this working package.

In **distributed storage systems**, our objective is to study the impact of network coding in distributed storage, video and other content for providing maximum throughput and minimum latency. Existing work show that traditional distributed storage techniques such as erasure codes achieve good reliability, optimizing the trade-off between redundancy and reliability. However, [3] shows that in practical distributed storage systems other considerations enter into play, such as network bandwidth consumption. In such settings, [4] shows that traditional schemes do not suffice alone. However, they achieve an optimal trade-off curve between storage and bandwidth overhead caused by management (i.e. repair) operations when paired with network coding.

This result is particularly promising, as it is expected that in volatile wireless mobile environments, more repair operations will be triggered in an

already bandwidth-limited context. Hence the benefit that network coding can offer to distributed storage in such scenarios is even greater.

In **content distribution**, our objective is to explore how network coding can provide benefits to existing applications in terms of performance, efficiency and resilience. We also emphasize hostile environments with frequent node mobility and communication errors such as wireless mesh networks and isolated mobile internet worlds. Existing state-of-the-art employing network coding for content distribution is abundant, particularly in peer-to-peer networks (P2P). The nature of such networks and their ability to operate in a distributed fashion, scale, and self-organize in the presence of a highly transient population of nodes, network and computer failures, make it particularly suitable for network coding. Notably, [5] shows that under realistic and noisy settings in which the network changes over time (peers join and leave at any time, nodes have only local information), network coding does provide an advantage for various network topologies and sizes. The Avalanche protocol ([6], [7] and [8]) is a living example of this, and is described in detail in Section 3.1. The question that remains, however, is if the results in these dynamic P2P environment can be extrapolated (and adapted) to N-CRAVE's scenarios of interest. A more detailed joint design between the protocols described in [2] and the network coding P2P protocol may be needed in order to achieve reasonable performance results.

Furthermore, this working package will study the inherent **security** properties of network coding to fortify information transport against data corruption and other malicious actions. Byzantine attacks such as jamming attacks by malicious nodes, i.e. injection of corrupt packets in the network, are especially fatal in the context of network coding since a single malicious packet has the potential to infect many other packets within a short period of time. Hence, the performance of a (cooperative) system, which uses network coding for efficient information dissemination, is strongly dependent on the incorporated ability to detect malicious packets. Also, information flowing through other peers is susceptible to eavesdropping, in which an adversary can gain unauthorized access to a node communications. We show that there are literature results covering those scenarios. The question, again, is how these schemes would perform in highly hostile and volatile environments, and what advantages and disadvantages the wireless medium introduces from the security standpoint.

Finally, we explore **network management and monitoring**, a crucial tool to monitor the health of a network where network coding has been extensively used with great success. Being able to control and to infer network characteristics efficiently and accurately, without incurring in significant use of costly network infrastructure, is critical. This is particularly true in volatile wireless environments where the available resources and the infrastructure support is very limited.

The rest of this document is organized as follows. Section 2 presents the

state-of-the-art techniques for distributed information storage and retrieval, and how network coding can improve their performance and efficiency. We show that network coding techniques outperform state-of-the-art techniques based on erasure coding and random linear coding. In Section 3 we explore content distribution networks. In particular we explore the effect of network coding on P2P distributed systems and some applications specific to wireless environments such as ad-hoc, vehicular and sensor networks. The combination of network coding with the distribution of content source-coded using the scalable and multiple description source coding methods is also studied. Section 4 reviews existing mechanisms for securing network coding protocols, with details of the specific characteristics of network coding that can be leveraged to counter some of the threats posed by eavesdroppers and Byzantine attackers. In Section 5, we discuss network coding protocols for network monitoring and management, including network tomography and network recovery management. Finally, Section 6 explores other type of applications that use network coding over sensor networks. The document is concluded in Section 7, with a brief discussion on the main research directions for future work.

2 Distributed Information Storage and Retrieval

Distributed storage systems are becoming the de-facto method of data storage for the new generation of applications. As an example, modern cloud applications developed by companies like Google, Amazon and Yahoo!, that require terabytes, and even petabytes of data, need to rely on distributed computing and storage in order to meet scalability, availability and performance demands. Compared to traditional relational database systems, distributed storage networks increase storage efficiency and data availability by providing shared storage access to computers and servers in multiple locations [9]. Distributed storage systems provide reliable access to data through redundancy spread over individually unreliable nodes. Their geographic distribution of resources also results in a lower observed latency, and resources can be placed closer to the clients.

A distributed storage system consists of a set of storage elements placed into nodes¹, which function independently of each other and thus exhibit independent failure patterns. These nodes are often connected through a network with arbitrary topology, and the information objects are stored in specific nodes according to a mapping function.

Typically, one wants to reliably recover the information in a distributed storage system with the lowest communication cost possible, and by requiring the lowest possible aggregated storage. Several performance metrics of

¹Here node is any device that stores information and can be independently addressed, such as servers, hard disk drives or sensors.

interest are defined in this scenario, for which we present the most relevant:

- *Communication cost*: it represents the overhead incurred by the system in terms of messages needed to retrieve the desired object.
- *Storage cost*: it represents the actual storage used by the information objects relative to their actual size. These include management information, metadata and data replication for reliability.
- *Object availability*: it represents the probability to successfully retrieving the distributed stored object.
- *Node availability*: the probability that a node can be successfully reached, caused by simple communication errors to more serious node failures.
- *Repair cost*: a repair takes place when nodes enter and leave the system, and other nodes must regenerate the substrate of information across the network after a node entry/departure in order to attain the same level of reliability. That repair operation requires an exchange of information across the network, incurring into a transmission cost.

A distributed storage system can be easily conceived as the process of dissemination and recovery of information taking place across several nodes, and thus it is a good candidate to employ some form of coding. By encoding information, distributed storage systems increase reliability over former, simpler non-coded methods. Two main coded alternatives were proposed: Erasure coding-based distributed storage systems and Random Linear Coding storage systems [10], [11], [12].

Erasure coding-based distributed storage systems were the first to handle information pieces as algebraic entities, conceiving however the dissemination and recovery as a one-shot process. In Random Linear Coding (RLC) based storage systems, the objects are also treated as algebraic, but are randomly generated. These random generated objects represent an intermediate step towards application of network coding.

Many of the literature that uses network coding in such systems, use the well-known capacity-achieving results and simply apply network coding to the distributed network implicitly defined by the process of distributing and recovering the information among the participant nodes. The network coding inspired methods basically outperform erasure coding in terms of downloading time for an object availability target, as well as in repair bandwidth needs.

2.1 Erasure Coding Distributed Storage Systems

In a distributed storage system employing a (N, M) erasure code [10], each information object is divided into M distinct pieces which are encoded using

appropriate erasure codes. Two popular examples of erasure-codes are Reed-Solomon codes [13] and Tornado codes [14].

Encoding of the M original pieces generates $N \geq M$ encoded pieces, while any different $M\gamma$ out of the total N pieces, where $\gamma \geq 1$, will suffice to recover the entire object². Here, $R = \frac{M}{N} < 1$ is the code-rate, and the introduced relative redundancy equals $\frac{N-M}{M}$. These N pieces are then placed, either deterministically or randomly, at different nodes within the distributed storage system.

The increased resilience of erasure coding comes from the statistical diversity obtained when spreading information pieces over several nodes. If an error occurs, a common practice to repair from a node failure is for a new node to download subsets of data stored at a number of surviving nodes, reconstruct a lost coded piece using the downloaded data. Only when a number of nodes greater than the introduced redundancy fail, the data would not be able to be recovered. The amount of storage needed to obtain a similar resilience by means of simple replication would be prohibitive [15].

Erasure coding and in general coding-based methods yield their best performance in scenarios with medium/low node availability, as in mobile mesh or ad-hoc networks. In [16] the redundancy required for a given object availability target is calculated. This redundancy depends on node availability. The paper benchmarks erasure coding using node arrival-departure traces and availability datasets from three real wired networks. Only in one out of the three wired networks the availability per node is low enough so that the erasure coding storage system outperforms the non-coded system in storage and bandwidth needs.

As a drawback, erasure coding adds complexity and constraints to system design [16],[15]. In order for erasure coding to be practical, the number of nodes must be large, at least equal to N . Yet, the main disadvantage of erasure coding is the need for gathering $M\gamma$ pieces for repair. Gathering these coded pieces is costly: in non-coded storage, the transmission latency when recovering an object is bounded by the closest replica; in coded storage it is bounded by the M -th closest piece.

2.2 Random Linear Coding Distributed Storage Systems

In a distributed storage system employing a Random Linear Code (RLC), each information object is divided into M distinct pieces $\{m_1, \dots, m_M\}$, elements of \mathcal{F}_q^s , i.e., a vector of size s in a finite field of size q . Then each node stores k random linear combinations of the M pieces, i.e., each stored piece of content f_i at a node has the form [10]

²If $\gamma = 1$, the erasure code is said to be Maximum Distance Separable (MDS).

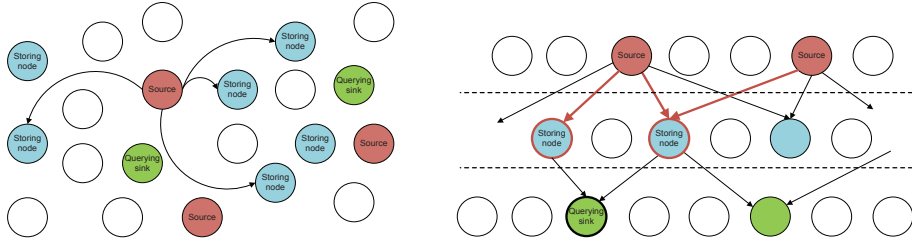


Fig. 1: On the left, Fig. 1a shows the dissemination of the code, whereas on the right, Fig. 1b shows the induced connectivity and bipartite graph superimposed in red.

$$f_i = \sum_{i=1}^M \beta_i m_i, \quad \forall \beta \in \mathcal{F}_q, \quad (1)$$

where the elements β_i are selected independently and uniformly among the q elements of F_q .

Also each node additionally stores the associated *code vector* $\{\beta_1, \beta_2, \dots, \beta_m\}$ for each of the k pieces. It is easy to observe that RLC codes presented in [10] are very close to network codes, but where the encoding operations are performed only at the sources.

In [10], it is shown that RLC greatly outperforms erasure codes for distributed storage. For typical values of k, M, N the RLC method outperforms erasure coding up to 30% in the average time to recover the necessary pieces of information (mean availability), which broadly represents the average download time for the object. They also show that RLC based systems need to query less nodes in order to find an object, and also the amount of bandwidth used for the recovery is decreased.

This holds for a large enough size of the finite field from where the random elements are drawn. The role played by the field size is similar to the redundancy introduced by an erasure code. Notice however that the redundancy in erasure codes leads to an increase of storage requirements up to 10 times larger than the one required by RLC for comparable performance [10].

The proposals in [11],[12] present a variation of the RLC-based distributed storage with applications to large scale wireless sensor networks. The authors focus on the tradeoff among reliability and the bandwidth required at dissemination stage. In their system, k sources out of the total N sensor nodes independently transmit their object to a number of other nodes picked uniformly from the network. Every node generates random linear combinations of all the received information over a finite field and

stores the result. The resulting connectivity matrix G is aimed at being kept sparse and the associated communication cost at this dissemination stage low (Fig.1a).

In order to retrieve the data, it is enough to select k nodes randomly. The resulting $k \times k$ matrix is invertible with high probability. In fact, [11] shows that recovery from the combinations available at just k out of the N nodes is granted asymptotically under certain conditions. However, the proposed method is only claimed to be cost-optimal as long as coding and routing are jointly performed.

The main theoretical contribution of [11] is in quantifying how sparse the matrix G must be in order to asymptotically guarantee the recovery. The authors find out that the output degree of the source nodes is enough to be as large as $5\frac{N}{k} \ln(k)$ in order for the recovery probability to approach to one. Moreover they bound the recovery probability of matrix G with the probability that the bipartite graph represented by the source to storage nodes edges (non-zero elements in G) has a perfect matching (Fig.1b).

The method in [11],[12] is different from previous erasure coding as it is performed in a decentralized fashion. The method differs also from typical RLC methods in that it does not multicast to the whole network; only a subset of nodes is fed (sparsity), and the resulting connectivity graph is unknown a priori (random). It is observed that the decoding is not as fast as in erasure coding based methods using low-density parity-check (LDPC) codes or Luby-transform (LT) codes. The former requires k^2 operations, instead of the k^3 operations required for a matrix inversion in the finite field.

2.3 Network Coding based Distributed Storage Systems

Erasur codes achieve good reliability by storing the data redundantly in a large collection of unreliable storage nodes, such that when a small subset of nodes fail, the data can still be recovered from the surviving nodes. In this context, erasure coding presents a good trade-off between redundancy and reliability [3].

However, in practical distributed storage systems, there are more considerations than achieving a good redundancy-reliability tradeoff. In particular, the codes must be designed optimally and they have to be properly repaired in the presence of node failures. In practical storage systems, network bandwidth is also a critical resource. Hence an important consideration is to design the code as to minimize the network bandwidth needed by repair operations.

Existing literature, notably [4],[3], extends the concept of RLC beyond the one-relay mode, and introduce network coding concepts into the distribution network in order to tackle the above issues.

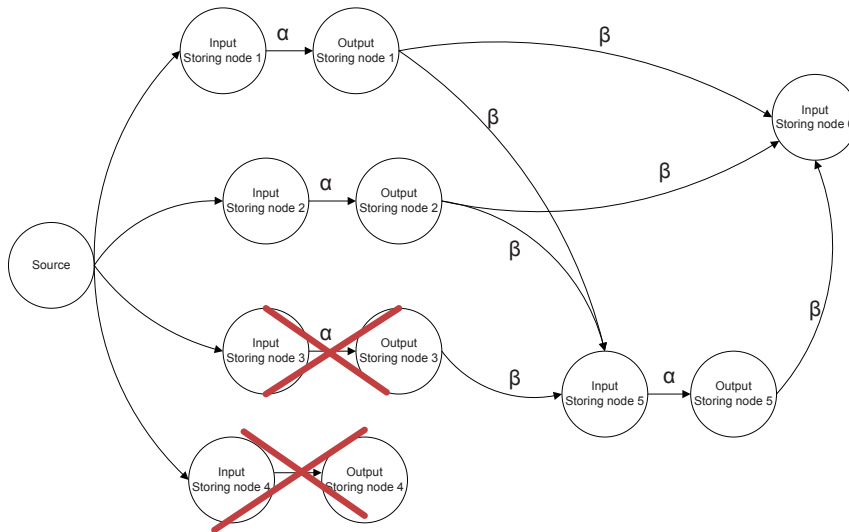


Fig. 2: Information flow graph for $d = 3$ and $n = 4$, where d is the maximum allowed number of connections for a joining node and n is the minimum number of living nodes in the system. Node departures are depicted in red and repair is exemplified. α is the storage capacity per node and β the repair bandwidth per connection

Repair Operations

Consider the scenario in which a distributed storage system is composed by n nodes with successive node departures and arrivals, where each node joining the network is able to connect to a number d of existing nodes. The connectivity graph resulting of the *join* and *leave* events of the nodes is referred as the *information flow graph* [4], representing the evolution of information flow as nodes join and leave. This information flow graph casts the original storage problem as a network communication problem where the source s multicasts the file to the set of all possible data collectors.

With that setting in mind, many existing networking techniques can be employed to design and optimize the codes over the resulting information flow graph. For example, fundamental performance bounds about codes can be derived, or we can employ network coding over the resulting multicast graph to achieve the multicast max-flow of the network as described in deliverable [1].

In particular, the **repair operations** can be posed as a max-flow/min-cut problem in the induced connectivity graph between source and any arriving node (e.g. Fig.2). Based on that, bound conditions on both storage

and repair bandwidth requirements can be derived so that the min-cut of the graph would suffice to eventually recover the stored objects by just querying k storage nodes. For example, if the sum of min-cuts between a node s and a node i and j is less than the size of original file, then we can conclude that it is impossible for the data collector to reconstruct the original file by only accessing storage nodes i and j , regardless the code used. Then, as min-cut capacity suffices, well-known network coding results can be used that guarantee the existence of a code achieving it.

Code Maintenance under Repair Operations

Let assume that we want to achieve a redundancy that requires n active storage nodes, each storing α bits. Whenever a node fails, a newcomer downloads β bits each from any d surviving nodes. Therefore the total repair bandwidth is $\gamma = d\beta$. Following the above reasoning, [4] shows the optimal trade-off curve between storage and repair bandwidth, and introduces regenerating codes that achieve any point in this curve. In particular the authors show the bandwidth and storage requirements for two relevant codes, namely the minimum-storage regenerating (MSR) with

$$(\alpha_{MSR}, \gamma_{MSR}) = \left(\frac{\mathcal{M}}{k}, \frac{\mathcal{M}d}{k(d-k+1)} \right). \quad (2)$$

and the minimum-bandwidth regenerating (MBR) code, with

$$(\alpha_{MBR}, \gamma_{MBR}) = \left(\frac{2\mathcal{M}d}{2kd - k^2 + k}, \frac{2\mathcal{M}d}{2kd - k^2 + k} \right), \quad (3)$$

where \mathcal{M} is the size of the object.

Using a set of trace datasets, the authors in [4] compare the performance of MSR, MBR regenerating codes with replication, ideal erasure codes and the hybrid scheme of [16] in terms of expected repair bandwidth and expected storage need. These two performance indices are computed based on the fraction of nodes that permanently fail per unit time and the probability that a node is temporarily available. Both parameters are specific to each trace dataset. Compared to hybrid, MSR codes offer slightly less maintenance bandwidth and storage and simpler system architecture since only one type of redundancy needs to be maintained.

Code Design

In [4], the authors characterize the codes achievable by the use of network coding and their bandwidth and storage requirements. However, from existing network coding results, [4] can only conclude that there exists a valid storage code that can tolerate an arbitrary number of failures/repairs, without proposing actual codes.

To that effect, [3] raises the question of the existence of such codes, i.e., whether codes can be designed so that they can work under an unbounded number of failures/repairs, assuming only the population of active nodes at any time is bounded. In fact, the authors show that if the incoming node has access to all the remaining storage nodes after one node departure ($d = n - 1$), there is a linear code requiring a repair bandwidth for an amount much lower than the original object size.

Further, a class of codes is constructed, the so-called Optimally Maintained Maximum Distance Separable (OMMDS) for the aforementioned case of $d = n - 1$. Assuming that a set of (n, k) -MDS linear codes is available at the outset, the authors show that the minimum-distance separation property is conserved after repair, so that future recovery is guaranteed with the same reliability redundancy trade-off, and a necessary lower bound on the finite field size depending on (n, k) is given.

Explicit codes, however, are not found in [3]. The paper either suggests using random linear network codes (RLNC) over a finite field in order to get randomly generated MDS, with low expected probability of failure, or to construct deterministically the code using standard methods in polynomial time [17].

3 Content Distribution

Content distribution solutions provide a mechanism for distributing content on the Internet in order to maximize bandwidth usage and improve accessibility and reliability [18], [19]. The typical content distribution solution relies on placing dedicated equipment at certain places inside or at the edge of the Internet. The best example of such solutions is Akamai, which runs several tens of thousands of servers all over the world.

These content distributed networks (CDNs) improve network performance by maximizing the resource utilization through content replication, caching, server-load balancing, request routing etc [20]. While they have the advantage of good maintainability of the system, they are often vulnerable to single points of failure. Techniques such as content replication, mirroring, caching are widely used by centralized CDNs to alleviate these problems [19].

In recent years, however, a new paradigm for Content Distribution has emerged based on a fully distributed architecture where commodity PCs are used to form a cooperative network and share their resources (storage, CPU, bandwidth). These “peer-to-peer” (P2P) networks facilitate the formation of autonomous networks by being more flexible to the dynamic changes in the network. The network can thus spontaneously adapt to the demand by taking advantage of the resources provided by every end-node. The systems capacity grows at the same rate as the demand, creating limitless scalability

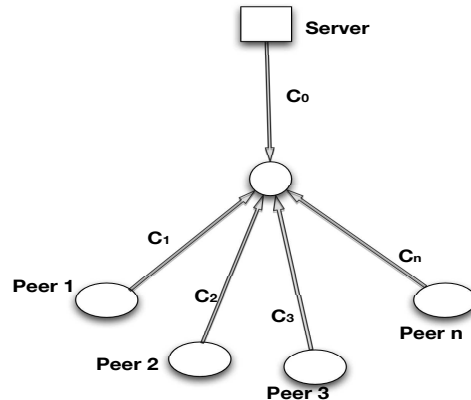


Fig. 3: Star model of a P2P network, consisting of a server and n peers. c_i represents the uplink capacity for nodes $i, i = 1..n$ and c_0 represents the downlink capacity of the server.

for a fixed cost [6].

However, these distributed content distribution systems suffer from a number of inefficiencies which decrease their overall performance [6]. Such inefficiencies are more pronounced for large and heterogeneous populations, during flash crowds, in environments with high churn, or when cooperative incentive mechanisms are in place. It is in those varying environments in which network coding provides a clear benefit, and several solutions have been proposed in the literature that will be explored in Section 3.1.

Finally, another common problem in content distribution when dealing with multiple heterogeneous receivers is that single-rate network coding multicast is not enough to achieve the min-cut of a particular network [21]. In that situation, some form of layered coding is employed in order to deliver different rates to the receivers. Their coexistence with the network coding techniques has also received attention in the literature as we will see in Section 3.2.

3.1 P2P Based Content Distribution

Distributed computer architectures labeled as P2P are designed for the sharing of computer resources (content, storage, CPU cycles) by direct exchange, rather than requiring the support of a centralized server or authority. P2P architectures are characterized by their ability to adapt to failures and accommodate transient populations of nodes while maintaining acceptable connectivity and performance.

In a traditional P2P scheme, the server splits the file of interest into blocks; next, peers download the blocks from the server and also collaborate to distribute downloaded blocks among themselves. Each node can recover

the original file after downloading all the components. Such approaches are sensitive to sudden departures of nodes, and their performance is affected by the policy used to decide which block to forward. Nonetheless, using network coding in a P2P system reduces these problems and increases the robustness to losses.

Network Coding Does Not Provide Benefit in the Ideal Case

The theoretical work in [5] shows that network coding does not provide any benefit over routing in terms of maximum throughput bound, in an ideal scenario. They consider the star network depicted in Fig. 3, which consists of n peers and one server. The metric used to analyze the performance is the *throughput of the system*, which is the sum of the content received by all peers per unit time.

Let c_0 be the uplink capacity of the server, and $\{c_1, \dots, c_n\}$ the download capacity of the n peers. Also let us assume that every peer has a perfect global knowledge of the system and the correspondent c_i values. The theoretic results are summarized in the following theorems:

Theorem 3.1 *Given the star network and the fluid workload, the maximum system throughput is $R = \min \{c_0, \frac{c_0 + \sum_j c_j}{n}\}$ and there exists a two-hop strategy that achieves this throughput.*

Theorem 3.2 *Given the star network with no coding or multicast (replication) in the network, then any coding applied at the peers cannot improve the throughput bound given by Theorem 3.1.*

The authors claim that, while there is usually an advantage obtained by using network coding (often referred to as the *coding advantage*), this advantage is usually small for the single source case in many practical networks [22]. Moreover these results apply always in the case of multicast-capable nodes. In a P2P system, where coding is applied to peers and there is no multicast in the network, the authors prove that there is no coding advantage.

Network Coding in Realistic P2P Systems

The above results hold under the assumption that the network environment does not change and that nodes have perfect knowledge and are able to compute an optimal suite of spanning trees. In a real P2P deployment, the network changes over time, as peers may join or leave at any time, and nodes usually have local information. In practical P2P systems such as BitTorrent, the actual obtainable maximum throughput may be far from the achievable bound shown by [5] due to scheduling difficulties, and the fact that the P2P network is continuously changing. In those situations random network

coding does help to deal with these problems for various network topologies and sizes. In those situations, a P2P system that uses network coding does indeed achieve a better performance.

One of such practical systems is Avalanche, proposed in [6]. With Avalanche, the server and the peers do not send original blocks, but linear combinations of all the available information. The encoding operations take place over a finite field and the coefficients are drawn uniformly from the field. Once enough linear combinations are obtained for which the corresponding coefficient vectors are linearly independent, a peer can solve the linear system and reconstruct the whole file.

As a first advantage of this network coding scheme, note that the system becomes more robust to situations when the server and/or the nodes leave the system. Due to the inherent redundancy of the linear combinations, all nodes are able to finish the download, even in extreme situations. A second advantage is given when scheduling the blocks for propagation. In a large scale network, optimal scheduling of packets is difficult to achieve, and nodes usually have to make a decision based on local information only, which can be suboptimal. Using network coding, this decision is greatly simplified. A node can determine if it can provide an innovative packet to a neighbor by comparing their coefficient matrices. However, if this information is not available to the sender, it can generate a linear combination of all the coefficient vectors previously received and send the resulting vector; next the receiver can check if the received vector is linearly independent with its own coefficient vectors and thus it can determine if the sender can provide new blocks.

The authors compare Avalanche with schemes where unencoded information is sent and to schemes where only the server generates and transmits encoded packets. In clustered topologies, network coding shows a clear benefit. In such topologies, without coding, some packets travel several times over the links that connect the clusters, thus wasting capacity. In heterogeneous networks, with nodes having different upload and download capacities, the performance achieved by the fast peers is degraded without coding. This effect is caused by slow nodes that may spend their bandwidth downloading from the server blocks that are not useful to fast peers.

Practical Implementation of Network Coded P2P System

In [7] and [8], the authors analyze the performance of an Avalanche prototype implemented in C#. In this system, there can be three types of participants: *peers*, *registrars* and *loggers*. A *peer* is a source or sink for content data; if it sends content into the system, then it is called *server*. If a peer remains in the system after it finished downloading, then it becomes a *seed*. A *registrar* enables peer discovery and provides nodes with a set of active peers. A *logger* represents an aggregation point for peer and

registrar trace messages. Every peer reports to the registrar when it needs more neighbors and also reports detailed statistics to the logger. Each peer maintains 4-8 connections to other peers, out of which they drop periodically a neighbor at random to prevent formation of isolated clusters. A file is divided into 1000-2000 blocks that are further grouped into generations. Peers generate and forward linear combinations of packets from the same generation.

The download efficiency of a user is defined as the ratio of the average download rate to the maximum. In this P2P filecasting system, the use of network coding results in users having a similar download efficiency, no matter at what time they joined the system. Network coding allows peers to use the available network resources efficiently while incurring in a small overhead in terms of CPU processing and I/O activity. Of the total download time, at most 6% is spent on decoding, while around 42% of the time is spent seeding the file.

The authors show that the throughput is low when more than 85-90% peers are behind NATs and firewalls (unreachable nodes). However, the system is robust in the presence of at most 60% unreachable peers and the achieved throughput is close to that obtained with full connectivity in the network.

In order to provide secure transmissions, peers verify encoded packets on the fly using a scheme based on masks and mask-based hashes, *Secure Random Checksums (SRC)*. With this approach, the server produces as many random elements as the number of symbol elements in each block. For example, for a block B_i with symbol elements $B_i = [b_{i,1}, b_{i,2}, \dots, b_{i,n}]$ and the random numbers $r = [r_1, \dots, r_n]$, the SRC of block i is $\sum_{j=1}^n b_{i,j} r_j$. A new client that joins the system, first contacts the server which computes a new set of SRCs and it communicates it to the client over a secure channel.

In [8], the authors are also concerned with the server efficiency, which they define as the number of useful (i.e. unique) blocks that are sent by the server in time sufficient to serve one full copy of the file with optimal scheduling. With network coding, the server efficiency is greater than 95%, because every block generated by the server can be used in place of any other to reconstruct the original file.

Analysis of Avalanche

Avalanche is analyzed from the network coding point of view in [23]. There, the authors assume that any transmission between any two neighbors needs an integer number of time units to finish. Next, the graph G , which represents the network in discrete time is changed to graph $G^* = (V^*, E^*)$, with node set $V^* = \{(i, t) : i \in V \text{ and } t \geq 0\}$, where the node $(i, t) \in V^*$ corresponds to node $i \in V$ at time t . The edge set E^* depends on the strategy used by the server and the nodes to upload blocks to the neighbors.

There is an edge of capacity m from node (i, t) to node (j, t') , where $t < t'$, if m blocks are transmitted from node i to node j , starting at time t and finishing at time t' . Moreover, for each $i \in V$ and $t \geq 0$, there is an edge with capacity k from node (i, t) to node $(i, t+1)$, which means that the blocks generated/received at a node are retained in that node indefinitely over time; also, all blocks received by nodes $(i, l), l \leq t$ are transmitted on the edge from (i, t) to $(i, t+1)$.

Let node $(s, 0)$ denote the source that multicasts a file consisting of k data blocks to all nodes in G^* and let $\text{maxflow}(v)$ represent the maximum flow from $(s, 0)$ to a node $v \in G^*$. Based on the results shown in [24], those nodes (i, t) for which $\text{maxflow}((i, t)) \geq k$ can receive the whole file with probability close to 1, if the base field is large enough. Therefore, with Avalanche a node $i \in V$ receives the whole file after the minimum t such that $\text{maxflow}((i, t)) \geq k$ (**lower bound**). If a node i cannot decode at the minimum time t , it will eventually decode after downloading some more coded blocks from its neighbors. For the case when some subset of nodes $U^c \in V$ leave the system, the remaining subset of users $U = V \setminus U^c$ can recover the whole file with high probability if $\text{maxflow}(U(t)) \geq k$, where $\text{maxflow}(U(t))$ represents the maximum flow from $(s, 0)$ to the set of nodes $U(t) = \{(u, t) : u \in U\}$.

Network Coding on Vehicular P2P Networks

In the area of vehicular networks, [25] introduces CodeTorrent, a swarming protocol based on single-hop communications. Each of the nodes broadcasts the descriptions of the files that they have. If some neighbor is interested in a file, it broadcasts a request that contains the file id, which is assumed to be unique. When a neighbor receives a request, if it has a piece of the requested file, it generates and sends a newly coded frame, $c = \sum_{k=1}^n e_k \mathbf{p}_k$ where e_k is a certain element in a certain finite field and \mathbf{p}_k is a piece of the original file. A node keeps requesting coded frames until it receives n linearly independent frames and can decode the original file.

As an optimization, a request may contain the *nullspace vector* (a vector in the nullspace spanned by all encoding vectors of the frames stored at the requesting node), which can be used by the neighbors to determine if they can provide innovative information or not.

Network Coding on Live P2P Streaming

For live P2P streaming, network coding can be used to ensure error control [26]. Such a system system is described in [27]. The authors use a C++ implementation of random network coding to run extensive experiments in different scenarios. They compare the network coding approach to naive broadcast, considered to be representative of the worst case. Although the

paper does not include any results for naive broadcast, the authors conclude that network coding does not provide any advantage over naive broadcast!

However, they reconsider this conclusion in a following work [28]. *Lava* is an experimental testbed consisting of a cluster of 44 dedicated dual-CPU servers. In *Lava*, a live session consists of a multimedia stream with a specific multimedia rate and such a stream is divided into *segments*, which are further divided into *blocks*, if network coding is used. The architecture of a *Lava* node is based on two functionalities: a *network* part, that deals with all the connections and emulates the upload and download capacities, and an *algorithm* part, that implements the algorithms and protocols for peer-to-peer live streaming. The authors also implement *Vanilla*, a standard peer-to-peer protocol, where network coding is added as a plugin component. This protocol keeps a *playback buffer*, where segments for a streaming session are stored in order. A segment is removed from the playback buffer after it is played. If a segment is not available in time for playing, it will be skipped. A peer starts to produce and serve new coded blocks after it receives $a \times n$ coded blocks, where n represents the number of blocks from a segment and $0 < a \leq 1$ is called *aggressiveness*.

The metrics used for evaluation are the percentage of playback skips and the percentage of discarded blocks due to linear dependence or obsolescence. The experimental results show that network coding can support a wide range of streaming rates, from 100 KB per second to 8 MB per second. However, the decoding process represents the bottleneck of network coding in the streaming process. When the demand is close to the bandwidth supply, network coding performs better than *Vanilla*, because in the first case, the streaming is made with a finer granularity. As the number of peers in the network increases, network coding does not scale with respect to playback skips, because of the computational overhead. Nonetheless, when it comes to redundancy, network coding outperforms *Vanilla*. For the flash crowd scenario, network coding shows a higher percentage of initial playback skips, but is more resilient to peer departures.

3.2 Content Distribution with Scalable and Multiple Description Source Coding

Scalable source coding and multiple description source coding (MDC) aim at coding a source into respectively layers and descriptors, where each of these provides a quality enhancement at source decoding. The larger the number of different layers or descriptors available, the higher the quality of decoded source signal [29].

The scalable variant poses a number of constraints in the order these layers are to be received, whereas the MDC variant is flexible to that respect. This sort of source encoding is appropriate in applications where variable quality is admissible and perceptible (e.g. video or audio), and transmission

is non-reliable, as in wireless scenarios or congested wired networks. It is also sensible to use this type of coding in applications where the receive bandwidth may be disparate across receivers, as in broadcasting or multicasting, in order to take advantage of the downlink bandwidth of the most favoured destinations while satisfying a lower but acceptable reception quality at the less favoured ones.

The target for these source coding schemes is to get a smooth quality degradation curve as packet loss probability increases, and to avoid as much as possible the *cliff effect*, where there is a point in the number of reception erasures beyond which perceptual quality of received media reduces drastically [29]. This idea is quantified in the so-called intermediate performance-metric: the ratio of the number of received packets contributing to source decoding to the number of received source packets.

The applicability of these source coding techniques is conditioned to the existence of a *multirate* network coding multicast scheme, it being able to satisfy an elastic rate demand among sinks. The min-cut -based multicast solution traditionally provided by network coding was conceived as a single-rate multicast. There, a common rate that matched the lowest attainable rate among the sinks in a session was set to all the sinks, regardless of their actual available min-cut capacity from source and thus leaving no room for the application of scalable/ MDC source coding.

Recently, some proposals have tried to tie up multirate multicast with network coding. They aim at realizing the individual sinks min-cuts while maximizing the aggregated delivered rate over all considered sinks as well, and thus extend the application domain of scalable/ MDC source coding. These proposals mostly led into two directions: the so-called layered coding and the non-layered coding variant, that are presented next.

3.2.1 Layered Coding Approaches

In layered coding, the content is divided into a number of layers. For each layer, a multicast session is set up. Each receiver subscribes to the number of sessions it is able to accommodate depending on its receive bandwidth. In this scenario, a popular (though sub-optimal) approach is to perform network coding independently for each individual multicast session.

Layered coding proposals aim at achieving the individual min-cuts of the receivers while maximizing the aggregated delivered rate over all receivers [30] [21], [31] [32]. Sinks are grouped by the largest layer index they can successfully receive given their corresponding min-cut. These proposals assume in all cases that per-layer rates can be fine-grain tuned as some recent source codecs (*FGS*, *PFGS*) allow [21]. They mainly differ in the optimization approach followed to find the rates per layer and to accommodate the multiple sessions (and induced meshes, the so-called *coding subgraphs*) into the network.

Most of the layered coding approaches consist in an optimization problem to find the so-called *capacitated* or *non-capacitated* coding subgraphs, representing a session/layer. A capacitated subgraph is the set of per-session and edge utilization factors minimizing a given cost function. A non-capacitated subgraph is a plain enumeration of the multiple paths a session flow will traverse based on a cost criterion independent of flow rate. Non-capacitated subgraphs are more practical to compute as they require no information on the actual rapidly changing link capacities in the network [31]. In general, once a coding subgraph is constructed, transport layer adjusts to the delivered rate per session and network layer performs network coding within the subgraph.

The authors of [32] find capacitated subgraphs while maximizing delivered aggregated rate. They also show the network code construction for each coding subgraph, based on the standard LIF algorithm for single-rate multicast [17]. Alternatively, [30] finds optimal capacitated subgraphs by explicitly leaving the largest possible residual bandwidth per edge in each subgraph in order to accommodate as many other subgraphs as possible. The authors also present a distributed heuristic algorithm that implements this technique.

Finally, in [31], an optimal per-layer rate control algorithm is obtained when non-capacitated coding subgraphs are given.

3.2.2 Non-Layered Coding Approaches

In the non-layered coding variant, the multirate multicast network code construction is jointly considered for all layers/descriptions. Thus there is no distinction or consideration on layers and subgraphs.

This variant is not as well-understood as the layered coding, and the joint network code construction leads to decodability issues at sinks. The number of delivered linear combinations to the sinks may be larger than their respective min-cuts, and non-decodability happens whenever the dimension of a message set received into a sink exceeds its rank.

For example, the authors in [33] show some multicast examples where all min-cut capacities to all sinks are indeed met, but not all the MDC descriptors that arrived to the sinks can be decoded with traditional linear broadcast ([34]) network codes.

To resolve the non-decodability issues, the Rainbow Network Coding (RNC) problem [35],[33] was introduced. In RNC, instead of maximizing the number of packets sent to a set of sinks, the target is maximizing the number of *distinct* packets arriving at each sink in the set, each packet being distinctly coloured at source and hence the name “Rainbow Problem” (RP). The original RP is presented in [35] without network coding, and it is shown to be generally intractable except for a few cases. The authors show RP to be NP-hard since the NP-hard problem of Edge-Disjoint Path is shown to

be a reduction of the RP. Interestingly, [35] shows that for the single sink case RNP boils down to a Max-Flow calculation.

Authors in [33] focus on the single-source RNC, which tries to determine the actual network codes to use in the RP. RNC is also shown to be NP-hard, by reducing the Coloring Graph problem to an instance of the RP. Because of NP-hardness, [33] eventually puts forward heuristics resembling to the layered coding approaches.

In a more particular scenario, [36] provides achievability results for network coding multicast, where a constant demand d_0 is requested by a majority of sinks but two sinks with a different demand each: $d_1, d_2 < d_0$. Assuming the min-cut from source to sink is higher or equal to the respective demand for any sink, every of them can get d_0 linearly independent combinations and the two sinks in consideration can still disambiguate d_1 and d_2 messages from less than these d_0 linearly independent combinations. This is accomplished via a precoding at source that allows gaussian decomposition at each of two sinks to rank respectively d_1 and d_2 without mutually interfering their zero column spaces. No results are provided in [36] for an arbitrary population of sinks with a different demand each.

Circumventing non-decodability

In [37] a potential solution to the non-decodability problem is proposed by designing a packetization of source-coded layers using erasure codes. The source node embeds up to W layers of equal size L across W packets, where W is the maximum min-cut value among all sinks. The i -th layer is split into i equally sized pieces. The pieces of the i -th source layer are protected by a (W, i) Reed-Solomon code. The encoding of every layer is represented with a generator matrix of size $W \times i$. Further, packets are formed across the resulting W codewords. While the proposal is not claimed to be optimal in terms of transmission cost, it guarantees the recovery of the first k layers from any k packets.

Moreover, [37] shows the conditions for decodability of a layer at the sinks under the above packetization scheme. The paper shows that the rank of the matrix resulting from the product of the network code decoding matrix and the erasure code generator matrix has to be larger than every layer index up to the min-cut at the sink. These additional requirement only pose marginal additional constraints to network code construction. A variant of the LIF algorithm is presented in [37] that realizes a non-random network code holding to be full rank when multiplied with the former identity-truncated generator matrices.

Alternatively, [38], building on top of [39], identifies the multi-rate content multicast as a particular case of multisource network coding, where all the sources, each representing a layer and session, collapse over the same node.

In [39] the necessity and sufficiency conditions on achievability for the problem of multiple independent sources are given, each feeding the network with one layer per multicast session. Mixing in the network among some sessions is allowed, while preventing unintended cross-mixing among others, the so-called *pollution*. Network graph is extended by substituting each network node and edge with a number of input-output pipes equal to the number of allowed session mixtures, the so-called *piped graph*. Each session mixture is identified with one tag and these tags are stored in the incoming packet headers. Tags are subsumed into supersets, i.e. an output packet tagged as belonging to a source mixture requiring a given set of sources, may get mixed at a node in any outgoing packet from any other source mixture that includes the same or more sources. E.g. flows conveying sources s_1, s_2 can be mixed out within packets conveying sources s_1, s_2, s_6 but not viceversa. [39] shows that a solution for the piped graph case is a solution for the non-piped graph since the former is more constrained. Thus, by respecting a proper tag labelling across nodes in the network, high min-cut destinations get as much dimensions as needed while lower min-cut ones not get polluted.

The session/layer relation gets tighter in the scalable content multicast case, since source layers are useful at the receiver only in if delivered in a given order. For the decoding of layer M , only layers $1, \dots, M - 1$ are relevant. In [38], the approach from [39] is exploited to construct mutually non-polluting tags representing these subsets. Namely, a node can encode together into output source layer M incoming packets tagged as containing only up to the M -th output source layer; that is, tags subsets $S \subseteq \{1, \dots, N\}$, $N \leq M$.

4 Network Coding Based Security

Although the prevalent design methodology for network protocols views security as something of an add-on to be included after the main communications tasks have been addressed, we shall contend that the special characteristics of network coding warrant a more comprehensive approach, namely one which gives equal importance to security concerns. Motivated by the recent surge in network coding research and its applications, in deliverable *D.1.1* we discussed the security implications of stateless and state-aware network coding protocols in comparison with common routing solutions. By exploiting the intrinsic properties of network coding, most notably the algebraic dependency between packets and the possibility of information dissemination along multiple paths, it is possible to find natural lightweight solutions for distributing keys and ensuring confidentiality against eavesdroppers.

The main goal now is to present and discuss the most salient aspects of network coding from the point of view of network security at the application layer, where for instance practical examples show that network coding can

be combined with classical cryptography for secure communication. Some emphasis shall also be given to active attacks, which can lead to severe degradation of network coded information flows.

Open problems include how to construct and maintain network topologies that are particularly robust against Byzantine attacks on network coding (as pursued e.g. in [40] via limited broadcast), how to combine network coding and cryptographic primitives in the most effective way, how to exploit the properties of network coding for anonymous communication (e.g. along the lines of the *information slicing* approach proposed in [41]), and how to leverage network coding for secure multi-party computation. As network coding protocols continue to flourish and thrive, addressing the aforementioned concerns is likely to yield a considerable number of research opportunities in network coding security.

4.1 Securing Network Coding Protocols

In [1] we have provided a security taxonomy highlighting the specific vulnerabilities of network coding. We now focus on finding appropriate mechanisms for securing network coding protocols and on showing how the specific characteristics of network coding can be exploited to counter some of the threats posed by eavesdroppers and Byzantine attackers

4.1.1 Countering Eavesdropping Attacks

We shall start by presenting countermeasures against passive attacks. [1] gave special emphasis to *nice but curious nodes*, which do not break any of the established rules except for not ignoring the data for which they are not the intended receivers. In the same document, we addressed a second instance of eavesdropping attacks, in which the eavesdropper is able to wire-tap a subset of network links. The crux of the problem is then to find code constructions capable of splitting the data among different links in such a way that reconstruction by the attackers is either very difficult or impossible. Under this assumption, it was shown in [42], that there exist secure linear network codes that achieve perfect information-theoretic secrecy for single source multicast. In [43], these results are generalized to multi-source linear network codes by using the algebraic structure of such codes to derive necessary and sufficient conditions for their security. Furthermore, [44] goes on to prove the optimality of a secure network code proposed in [42]. More specifically, it is shown that the constructed code maximizes the amount of secure multicasted information while minimizing the necessary amount of randomness. The contribution of [45] consists of a coding scheme that can achieve the maximum possible rate of $n - \mu$ information theoretically secure packets, where n is the number of packets from source to each receiver and μ is the number of links that the wiretapper can observe. This can be applied

on top of any communication network without requiring any knowledge of the underlying network code and without imposing any coding constraints. The basic idea is to use a “nonlinear” outer code, which is linear over an extension field \mathbb{F}_{q^m} , and to exploit the benefits of this extension field.

The third type of attacker, which is the main target of this section, is a worst-case eavesdropper who is given full access to all the traffic in the network. In this case, the threat model is one in which the attacker has access to all the packets traversing the network but not to the secret keys shared among legitimate communicating parties. SPOC (Secure Practical Network Coding) [46] is a lightweight security scheme for confidentiality in RLNC which provides a simple yet powerful way to exploit the inherent security of RLNC in order to reduce the number of cryptographic operations required for confidential communication. This is achieved by protecting (or “locking”) only the source coefficients required to decode the linearly encoded data, while allowing intermediate nodes to run their network coding operations by means of “unlocked” coefficients that provably do not compromise the hidden data. The latter set of coefficients stores the operations performed along the network upon the packet.

Seeking to evaluate the level of security provided by SPOC, [47] analyzes the mutual information between the encoded data and the two components that can lead to information leakage, namely the matrices of random coefficients and the original data itself. This analysis, which is independent of any particular cypher used for locking the coefficients, assumes that the encoding matrices are based on variants of random linear network coding and can only be accessed by the source and sinks. The results, some of which hold even with finite block lengths, prove that information-theoretic security is achievable for any field size without loss in terms of decoding probability. In other words, since correlation attacks based on the encoded data become impossible, protecting the encoding matrix is generally sufficient to ensure the confidentiality of network coded data.

4.1.2 Countering Byzantine Attacks

The most serious security challenges posed by network coding thus seem to come from various types of Byzantine attacks, which target either the control traffic or the network code itself and demand more sophisticated security solutions than those currently in place. Although Byzantine attacks have already been addressed in [1], here we provide a deeper description on the specific properties of linear network codes that can be used effectively to counteract the impairments caused by traffic relay refusal or injection of erroneous packets.

In particular, RLNC (explained in [1]) has been shown to be very robust to packet losses induced by node misbehavior [48]. More sophisticated countermeasures, which modify the format of coded packets, can be sub-

divided into two main categories: (1) end-to-end error correction, and (2) misbehavior detection, which can be carried out either packet by packet or in generation based fashion.

End-to-end error correction.

The main advantage of *end-to-end error-correcting codes* is that the burden of applying error control techniques is left entirely to the source and the destinations, such that intermediate nodes are not required to change their mode of operation. The typical transmission model for end-to-end network coding is well described by a matrix channel $\mathbf{Y} = \mathbf{A}\mathbf{X} + \mathbf{Z}$, where \mathbf{X} corresponds to the matrix whose rows are the transmitted packets, \mathbf{Y} is the matrix whose rows are the received packets, \mathbf{Z} denotes the matrix corresponding to the injected error packets after propagation over the network, and \mathbf{A} describes the transfer matrix, which corresponds to the global linear transformation operated on packets as they traverse the network. In terms of performance, error-correction schemes can correct up to the min-cut between the source and the destinations. Rank-metric error-correcting codes in RLNC under this setting appear in [49] and the results are extended in [50, 51] for the scenario in which the channel may supply partial information about erasures and deviations from the sent information flow. Still under the same setting, [52] considers a probabilistic error model for random network coding, provides bounds on capacity and presents a simple coding scheme with polynomial complexity that achieves capacity with exponentially low probability of failure with respect to both the packet length and the field size. [53] extends [49] and provides an error-correction scheme based on random sparse graphs and a low-complexity decoding algorithm for a probabilistic error model. Bounds on the maximum achievable rate in an adversarial setting are provided by [54], which derives generalizations of the Hamming and Gilbert-Varshamov bounds.

The work proposed in [55] and explained in [1] is extended in [56], which introduces three additional adversarial models and gives optimal rates $C - z$ for each, where C is the network capacity and z is the number of links controlled by an attacker.

Misbehavior Detection

Generation-based detection schemes generally offer similar advantages as network error-correcting codes in that the often computationally expensive task of detecting the modifications introduced by Byzantine attackers is carried out by the destination nodes. The main disadvantage of generation-based detection schemes is that only nodes with enough packets from a generation are able to detect malicious modifications, and thus, usage of such detection schemes can result in large end-to-end delays. [57] proposes

an information-theoretic framework for detecting Byzantine attackers. The underlying assumption is that the attacker cannot see the full rank of the packets in the network. It is shown that a hash scheme with polynomial complexity can be used without the need for secret key distribution. However, the use of a block code forces an a priori decision on the coding rate.

The key idea of *packet-based detection schemes* is that some of the intermediate nodes in the network can detect polluted data on the fly and drop the corresponding packets, thus retransmitting only valid data. However, packet-based detection schemes require active participation of intermediate nodes and are dependent on hash functions, which are generally computationally expensive. Alternatively, this type of attacks can be mitigated by signature schemes based on homomorphic hash functions. The use of homomorphic hash functions is specifically tailored for network coding schemes, since the hash of a coded packet can be easily derived from the hashes of previously encoded packets, thus enabling intermediate nodes to verify the validity of encoded packets prior to mixing them algebraically. Unfortunately, homomorphic hash functions are also computationally expensive.

[58] proposes a homomorphic signature scheme for network coding which is based on Weil pairing in elliptic curve cryptography. Homomorphic hash functions are also considered in the context of peer-to-peer content distribution with rateless erasure codes for multicast transfers in [59]. With the goal of preventing both the waste of large amounts of bandwidth and the pollution of download caches of network clients, each file is compressed to a smaller hash value, with which receivers can check the integrity of downloaded blocks. Beyond its independence from the coding rate, the main advantage of this process is that it is less computationally expensive for large files than traditional forward error correction codes (such as Reed-Solomon codes referenced in Section 2 of this deliverable).

As explained in [1], one of the contributions of [60] is a cooperative security scheme for on-the-fly detection of malicious blocks injected in network coding based peer-to-peer networks, such as Avalanche, described in Section 3 of this deliverable. In order to reduce the cost of verifying information on-the-fly while efficiently preventing the propagation of malicious blocks, a distributed mechanism is proposed where every node performs block checks with a certain probability and alerts its neighbors when a suspicious block is found. The authors also included in [60] techniques to prevent denial of service attacks due to the dissemination of alarms.

In [61] the basic idea is to take advantage of the fact that in linear network coding any valid packet transmitted belongs to the subspace spanned by the original set of vectors. A signature scheme is thus used to check that a given packet belongs to the original subspace. Generating a signature that is not in the subspace yet passes the check is shown to be hard.

A comparison of the bandwidth overhead required by Byzantine error correction and detection schemes is provided in [62]. The intermediate nodes

are divided into regular nodes and trusted nodes, and only the latter are given access to the public key of the Byzantine detection scheme in use. Under these assumptions, it is shown that packet-based detection is most competitive when the probability of attack is high, whereas a generation-based approach is the more bandwidth efficient when the probability of attack is low.

4.2 Combating Byzantine Attacks with Network Coding and Rank-metric error correcting codes

Motivated by the outstanding performance gains of network coding in recent experimental setups combined with the intrinsic need to detect malicious packets efficiently, we focus on a new random linear network coding approach in the following [49, 50]. This approach allows to detect and remove a certain number of malicious packets per generation by applying error-control techniques at the sending and receiving nodes whereas intermediate nodes encode information in the usual manner of random linear network coding.

4.2.1 Network Model

The communication network is represented by a finite directed graph $G = (V, E, \delta : E \rightarrow V \times V)$. The elements of V and E are the nodes and edges of G respectively. The mapping δ assigns to each edge $e \in E$ a unique ordered pair of nodes $v \times \tilde{v}$ where \tilde{v} is the head of e while v corresponds to the tail. All edges incident to a node $v \in V$ can be partitioned according to

- (i) $E_v^+ = \{e \in E \mid \exists \tilde{v} \in V \text{ with } \delta(e) = v \times \tilde{v}\}$
- (ii) $E_v^- = \{e \in E \mid \exists \tilde{v} \in V \text{ with } \delta(e) = \tilde{v} \times v\}$.

The capacity of a single edge equals one q -ary symbol per use.

During each generation, source $s \in V$ injects n packets $\{x_1, x_2, \dots, x_n\}$ into the network. Each packet consists of a fixed number of, say, m symbols taken from the finite field \mathbb{F}_q and, therefore, a single packet can be interpreted as an element from the vector space \mathbb{F}_q^m . We consider also a Byzantine attacker who intends to pollute the communication process by injecting corrupt packets $\{z_1, z_2, \dots, z_t\} \subseteq \mathbb{F}_q^m$ into the network.

Linear coding is done as follows. Each intermediate network node $v \in V$ creates a packet $y(e)$ to be sent on link $e \in E_v^+$ by a \mathbb{F}_q -linear combination of packets on links $d \in E_v^-$ followed by a modulo q sum with potential error packets z_i , i. e.

$$y(e) = \sum_{d \in E_v^-} m_e(d)y(d) + \sum_{i=1}^t \mathbf{1}_e(z_i), \quad \text{with } e \in E_v^+. \quad (4)$$

The indicator function $\mathbf{1}_e(z_i)$ equals z_i , if z_i is injected into edge e and 0 otherwise. The local encoding coefficients $m_e(d)$ are randomly chosen from the field \mathbb{F}_q but assumed to be fixed after initialization.

Since all coding operations in the network are linear, we can relate each packet to a linear combination of source packets and corrupt packets according to

$$y(e) = \sum_{i=1}^n h_i(e)x_i + \sum_{i=1}^t g_i(e)z_i, \quad (5)$$

where the global encoding coefficients $h_i(e)$ and $g_i(e)$ can be determined recursively from the local encoding coefficients.

The destination node collects packets on its say n incoming edges $\{e_1, e_2, \dots, e_n\}$ and tries to infer the original packets $\{x_1, x_2, \dots, x_n\}$. The overall input-output behavior can be described as

$$\begin{bmatrix} y(e_1) \\ \vdots \\ y(e_n) \end{bmatrix} = \begin{bmatrix} h_1(e_1) & \cdots & h_n(e_1) \\ \vdots & \ddots & \vdots \\ h_1(e_n) & \cdots & h_n(e_n) \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} g_1(e_1) & \cdots & g_t(e_1) \\ \vdots & \ddots & \vdots \\ g_1(e_n) & \cdots & g_t(e_n) \end{bmatrix} \begin{bmatrix} z_1 \\ \vdots \\ z_t \end{bmatrix} \quad (6)$$

or, in short form, as

$$\begin{aligned} Y &= HX + GZ \\ &= HX + N. \end{aligned} \quad (7)$$

Note, that the rows of matrices X and Z are the source packets and the error packets, respectively, while the rows of Y correspond to the received packets at the sink.

4.2.2 Principles of Error Control in Random Network Coding

In [49], a coding theoretic approach for random linear network coding was introduced. Compared to the standard network coding approach [63], the new strategy is able to cope with lost and/or maliciously injected packets. The main idea is to interpret the packets of each generation, i. e. the rows of X , as generating system of a subspace³ in \mathbb{F}_q^m . Hence, each information sequence is represented by a unique vector space as opposed to conventional linear block codes where each information sequence is represented by a unique element from a single vector space.

In order to obtain a feeling of how the maximum achievable rate of a constant-dimension subspace code behaves compared to the maximum achievable rate of a linear block code whose packets (vectors) have the same length than the packets of the constant-dimension subspace code, we ask

³For the sake of performance, the set of injected packets should form a basis, i. e. the row space of X should be a n -dimensional vector space.

following question. Is the number of distinct n -dimensional subspaces in \mathbb{F}_q^m of the same magnitude as the number of vectors in a single n -dimensional subspace⁴? The answer is yes, provided that the packet length m is much larger than the number of packets n per generation, which is the case in most practical scenarios. In order to verify this, note that a n -dimensional subspace of \mathbb{F}_q^m consists of q^{nm} elements, while the cardinality of the set of all subspaces that have dimension n (such a set is denoted as Grassmannian) is equal to the Gaussian coefficient $\begin{bmatrix} m \\ n \end{bmatrix}_q$ (for a definition see e.g. [64]). Then the conclusion follows from following inequalities [49]

$$q^{n(m-n)} < \begin{bmatrix} m \\ n \end{bmatrix}_q < 4q^{n(m-n)}. \quad (8)$$

Hence, the focus on subspaces instead of vectors does not a priori yield any degradation in terms of transmission rate.

The row space of X , which represents innovative information, is vulnerable to two different kinds of distortions. First, the propagation of the packets from the source via the network to the sink results in a linear mapping of the row space of X to the row space of HX (see (7)). This is harmless if H is nonsingular. However, if H is singular, the row space of X is projected on a lower dimensional space, i. e. the row space of HX , what results in *deletion* of dimensions. Second, the injection of corrupt packets through a Byzantine attacker potentially adds extra dimensions to the row space of X . This malicious effect, denoted as *insertion*, is captured by matrix Z in (7). Ironically, a Byzantine attacker can sometimes be helpful if he compensates for (a part of) those dimensions which have been deleted due to a singular network H .

In [49], it has been shown that a code \mathcal{C} consisting of an ensemble of subspaces is able to correct ρ deleted dimensions and t inserted⁵ dimensions if the minimum distance between the subspaces is greater than $2(\rho + t)$. In this context, the metric

$$d(U, V) := \dim(U + V) - \dim(U \cap V) \quad (9)$$

was used in order to determine the distance between two vector spaces U and V .

An upper bound on the information theoretic capacity of the network stated in (7) is given [52] by

$$C \leq (m - n)(n - t) + \log_q 4(1 + n)(1 + t) \quad (10)$$

⁴Note that the two codes do not possess any redundancy and, thus, no capabilities regarding error detection or correction. Here, we are merely interested in the maximum possible codebook size.

⁵Inserted dimensions are dimensions, which are added by malicious nodes. However, these dimensions are only harmful if they are not contained in the row space of X .

where matrices H and Z are chosen uniformly at random. The derivation of (10) relies on the assumption that H is a nonsingular square matrix while Z has rank t , i. e. the communication process is only affected by insertions but not by deletions.

4.2.3 A Concrete Coding/Decoding Scheme

In this section, we review a coding scheme which achieves (10) for large field size q . The scheme was derived in [52] and is based on the assumption that the network is not singular, i. e. H is invertible, and suffers from at most t corrupt dimensions per generation, i.e. $\text{rank}(N) \leq t$. Successful decoding is guaranteed with high probability whereas decoding relies on following two results from linear algebra:

1. Let $W \in \mathbb{F}_q^{n \times m}$. The rank of W is the smallest number t for which there exists matrices $B \in \mathbb{F}_q^{n \times t}$ and $E \in \mathbb{F}_q^{t \times m}$ such that $W = BE$.
2. Let $H \in \mathbb{F}_q^{n \times n}$ be an invertible matrix and let $X \in \mathbb{F}_q^{n \times m}$. The reduced row echelon (RRE) forms of X and HX are equal.

In the remainder of this section, we will use following decomposition of the transmission model

$$\begin{aligned} Y &= H(X + H^{-1}N) \\ &= H(X + W), \quad \text{where } W = H^{-1}N. \end{aligned} \quad (11)$$

During each generation, the source injects n packets of length m into the network whereas each packet corresponds to a single row of

$$X = \begin{bmatrix} 0_{v \times v} & 0_{v \times (n-v)} & 0_{v \times (m-n)} \\ 0_{(n-v) \times v} & I_{(n-v) \times (n-v)} & D \end{bmatrix}. \quad (12)$$

Variable v is required to be at least as large as t and the precise value influences the probability of a decoding failure (see below). Matrix $D \in \mathbb{F}_q^{(n-v) \times (m-n)}$ contains the information sequence to be transmitted and is “protected” by surrounding matrices.

According to property 1, W can be written as

$$W = BE = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} [E_1 \quad E_2 \quad E_3] \quad (13)$$

where $B_1 \in \mathbb{F}_q^{v \times t}$, $B_2 \in \mathbb{F}_q^{(n-v) \times t}$, $E_1 \in \mathbb{F}_q^{t \times v}$, $E_2 \in \mathbb{F}_q^{t \times (n-v)}$ and $E_3 \in \mathbb{F}_q^{t \times (m-n)}$. Hence, the polluted source packets equal

$$X + W = \begin{bmatrix} B_1 E_1 & B_1 E_2 & B_1 E_3 \\ B_2 E_1 & I + B_2 E_2 & D + B_2 E_3 \end{bmatrix}. \quad (14)$$

The decoding procedure is constructed such that it fails if the rank of B_1E_1 is smaller than t . However, the probability P_f that this event occurs can be made arbitrarily small and an upper bound on it is given by [52]

$$P_f < \frac{2t}{q^{1+v-t}}. \quad (15)$$

Hence, we assume that $\text{rank } B_1E_1 = t$ or, equivalently, $\text{rank } B_1 = \text{rank } E_1 = t$. This, in turn, implies that the rows of B_2 in (13) are contained in the row span of B_1 and, therefore, an invertible matrix $\bar{T} \in \mathbb{F}_q^{v \times v}$ can be found such that $\bar{T}B_1 + B_2 = 0$. Taking into account the last equality, data matrix D can be recovered from (14) according to

$$\begin{aligned} T(X + W) &= \begin{bmatrix} I & 0 \\ \bar{T} & I \end{bmatrix} (X + W) \\ &= \begin{bmatrix} B_1E_1 & B_1E_2 & B_1E_3 \\ 0 & I & D \end{bmatrix} \end{aligned} \quad (16)$$

or, by bringing $T(X + W)$ in reduced row echelon⁶ (RRE) form what yields

$$\text{RRE}(T(X + W)) = \begin{bmatrix} I & 0 & \tilde{B}_1\tilde{E}_3 \\ 0 & I & D \end{bmatrix}. \quad (17)$$

Now, by uniqueness of the reduced row echelon form, we have

$$\text{RRE}(X + W) = \text{RRE}(T(X + W)) = \text{RRE}(H(X + W)) = \text{RRE}(Y) \quad (18)$$

since T and H are invertible matrices.

The coding strategy can be summarized as follows. The encoder writes the incoming data in a matrix whose structure is determined by (12). Next, the rows of the resulting matrix are injected into the network. After having propagated through the network, n packets are collected by the destination and are written into a matrix Y . Then, Gauss-Jordan elimination is performed with respect to Y and, as a consequence, the original data can readily be obtained from the reduced row echelon form.

As a concluding remark, we want to emphasize that the strategy tolerates up to t malicious dimensions (i. e. t or more corrupt packets) per generation while having a decoding complexity of $\mathcal{O}(n^2m)$.

5 Network Monitoring and Management

A critical network engineering functionality is to monitor the health of a network and adapt its use to the demands of the end users. In order to perform this *network monitoring*, we need control mechanisms to infer network

⁶By Gauss-Jordan Elimination.

characteristics efficiently and accurately, without significant use of costly network measurement infrastructure: a process referred to as network tomography.

Active network tomography aims at inferring internal network characteristics by sending and collecting probe packets from the edge of the network. *Passive* tomography infers network information from passive observations of regular network traffic. Such information is an important input to various control and traffic engineering decisions, both at the network and at the application layers, and therefore is a crucial aspect of network coded systems.

The more general *Network management* includes operation, administration, maintenance and provisioning of networks, and involves network monitoring.

5.1 Network Monitoring

We focus on two aspects of network monitoring where network coding has been more successfully employed, namely link-loss inference and topology estimation.

Link-loss inference

In networks with network coding capability, a crucial aspect of network monitoring is to estimate the link-loss rate through end-to-end measurements. This new aspect was proposed in [65], where the advantages of network coding for link-loss rate estimation was demonstrated.

In [65], the authors infer link loss rates based on end-to-end measurements) in the overlay networks based on sending probes. In order to decrease the bandwidth used by probes, the authors suggest the use of network coding techniques. In an overlay network, the network designer has control over intermediate nodes and therefore network coding would be easily applied in such networks.

In this approach, an overlay network is represented as a directed graph $G = (V, E)$, with each link $e \in E$ having loss probability p_e . Given a set $S \subseteq V$ of source nodes, a set $R \subseteq V$ of receivers and a set of links $L \subseteq E$, the goal is to estimate the link loss probabilities $\{p_e, e \in L\}$ with a desired accuracy. The authors use an algorithm of maximum likelihood estimation, and the performance measure is a cost function proportional to the link utilization required for the estimation.

This proposal has several benefits. First, network coding eliminates the overlap between the paths needed to cover the whole network, therefore the complexity in choosing which paths to monitor is reduced. Second, less probes are needed to achieve a certain estimation accuracy, because the content of the received probes can also be used.

Another link-loss estimation was introduced in [66]. They state that losses on different links affect the coefficient vectors differently and thus one can infer possible locations of link failures or losses in the network. The paper is concerned with how many failure patterns can be distinguished and provides bounds on the required field size of a network code that distinguishes among a given set of failure events.

For a given transmission problem, specified by a network G , source and sink locations, and source rates, a network code is *valid* if every sink is able to reconstruct all the source information without error.

For a given set C of distinguishable events and a given failure event $c \in C$, a *monitoring ambiguity* is said to exist for a network code if the corresponding coefficient vectors on the terminal links are identical to those for some other failure event in C . The probability to have a monitoring ambiguity is given by the following theorem:

Theorem 5.1 *For a given set C of distinguishable favorite events, and a given failure event $c \in C$, the probability of a monitoring ambiguity in a random linear network code is at most $1 - (1 - \frac{|C|-1}{q})^L$, where L is the maximum number of logical links, and $q = 2^u$ is the field size.*

The field size for a network code that is valid under all failure events in a set C and distinguishes among them without any monitoring ambiguities is bounded by:

- **Lower bound:** $q \geq |C|^{\frac{1}{rt}}$, where r is the number of sources and t is the number of terminal links.
- **Upper bound:** $q \leq |C|(\frac{|C|-1}{2} + d)$, where d is the number of sinks.

These bounds are generally pessimistic except for the worst-case networks. For a detailed proof of the results, we refer the reader to [67].

The advantages of the above approaches over traditional methods include better identifiability of links, the trade-off between accuracy of estimation and bandwidth efficiency, and the complexity of probe path selection. This was further investigated in [68] and extended to arbit topologies in [69].

Topology Inference

Another aspect of network tomography is to infer the topology of a network. Active topology inference in network coded systems was proposed in [65], [70]. Passive inference techniques which use normal traffic to infer topology was proposed in [71] and these properties were used in peer-to-peer systems for bottleneck discovery in [72].

5.2 Network Management

In the area of network management, a significant body of work has been carried on [73], [74], [75]. In these papers, the network is represented by the directed graph $G = (V, E)$ where V is the vertex set and E is the edge set. In this network, there may be several receiver nodes, β_i and there are one or more source nodes, at which one or more discrete independent random processes X_i are observable.

[73] discusses the case of multi-transmitter single-receiver for delay-free acyclic networks. A triple (A, F, B) or (A, G, B) is called a *code*, where matrices A, F, G, B are called transfer matrices. In particular,

- matrix A specifies how the source processes $X_i, i = 1 \cdots r$ are represented on the source nodes' incident outgoing links, where r is the total number of source processes;
- matrix B specifies how the receiver outputs Z_j are obtained;
- matrix F specifies how signals are transmitted between incident links;
- matrix $G = I + F + F^2 + \cdots = (I - F)^{-1}$ sums the gains along all paths between each pair of links.

A code (A, G, B) recovers the failure of a link h if $AG^h B^T = I$. In this context, recovery codes can be classified as *receiver-based* (matrix B changes), *transmitter-based* (matrix A changes) or *network-wide* (any combination of A, F and B may change). The authors provide two formulations to quantify the management information, a *centralized* approach and a *node-based* approach. For the first case, the network behavior is given by a single code that determines the behavior of every node. If n_c different codes are needed to achieve the recovery from different failure scenarios, then the network management requirement is given by $\log_2[n_c]$. The node-based formulation needs a higher management overhead and takes into account the number of nodes that change their behavior. In this case, the management requirement is given by the sum over all nodes of the log of the number of different behaviors among which each node switches.

The case of multi-transmitter multicast is analyzed in [74], where results for failures of links adjacent to the receiver nodes are presented. [75] proposes an information theoretic framework for network management, that attempts to give a starting point to the theory of network management for non-ergodic failures. This paper summarizes the results from the previous work in [73] and [74]. Moreover, it shows that if non-linear processing is permitted, then a single code can be enough (*non-linear receiver-based recovery*) and recovery can be done without network management, in some cases.

6 Network Coding Applications in Sensor Networks

Ideas from network coding have also recently started to be applied in sensor networks. These work seek to take advantage of the broadcasting capabilities of the wireless medium, implement intelligent in-network storage [76], and provide resilience in lossy environments [77], [78].

Ad-hoc wireless sensor networks is an area where network coding promises to have have impact, for the following reasons:

- This is an environment that is currently in the design stage, and thus has the flexibility to accommodate new protocols.
- Network coding requires intermediate nodes to perform some sort of packet processing. In sensor networks, node processing has always been advocated, albeit for different reasons: for example, for information aggregation. Thus, unlike for example Internet routers where packet processing would need the addition of new functionalities, in sensor nodes these functionalities are already in place.
- Sensor networks offer a challenging environment, due to the inherent challenges of the wireless medium, and the ad-hoc structure of the network that needs to be maintained. Additionally, sensor nodes are simple devices with very limited resources, and often deployed in hard-to-reach environments where maintenance is too costly. Thus design properties such as energy efficiency, load balancing, and robustness to node failures become not only desirable but of critical importance.

These are problems network coding promises to help with, thus motivating the further study and development of network coding techniques for sensor networks.

The application of network coding to sensor networks is very much in the research stage – some approaches have been proposed, but the theory and practice of the field are not mature. We here review some existing approaches.

An application of the distributed storage problem to sensor networks is the design of networks that can sustain information in the presence of disasters, such as earthquakes, floods, fires, where a significant percentage (or even the vast majority) of the sensor nodes are destroyed. This is a distributed storage problem where we want the random surviving nodes to rescue as much information as possible, and eventually forward this information to the sink. To achieve that, we can take advantage of the fact that, although there is limited bandwidth to send data to the sink, there still remains available bandwidth for nodes to exchange each – others information – so that, if a node fails, its information is not lost. Thus we can have each sensor node act as a storage point. One such approach was proposed in [79].

A more mature approach that aims to save computational resources, uses what are termed “Growth Codes” [76]. The intuition for these codes can also be traced back to a variation of the coupons collector problem. Simulation results show good performance of the proposed approach in TinyOs sensor networks [76].

Another work looks at use of untuned radios in sensor networks. Massive deployment of sensor networks requires that the nodes are easy to manufacture and thus have very low cost, while at the same time the radio component of the node enables communication with as low energy as possible. Narrow-band radios consume less power than spread-spectrum or other wide-band techniques and thus achieve the later goal. However, the analog components needed for the narrowband radios are more costly as compared to wideband radios, because an expensive and bulky off-chip quartz crystal needs to be used: this crystal provides the same low frequency reference at both transmitter and receiver and ensures that the transmitters carrier frequency and the receivers detection frequency are well matched.

The approach proposed in [80], is to eliminate the quartz crystal, and replace it with an on-chip resonator. This architecture allows the sensor node to be developed entirely of thin-film technologies. However, the variations in manufacturing process result in untuned nodes: each sensor transmits at a randomly chosen (from a finite set) frequency, and receives from a randomly chosen set of the frequencies. The observation in [80] is that, if we densely enough deploy these cheap sensors, then we can still create a connected network. In the traditional sensor network, a node would connect to all nodes within its transmission radius (neighbors). The only difference now is that, a node will connect to all neighbors that additionally have a matching reception frequency. The randomness in the transmit and receive frequencies of the components of the network simply lead to a different random network configuration.

The challenging part in this new network is how, without centralized knowledge of the node frequencies and topology, we can route the information from the source to the destination. This is where network coding becomes useful. If we know the min-cut, we can simply have all sensor nodes perform the same functionality, randomized network coding, and deliver any rate below the min-cut to the receiver.

The use of network coding allows to leverage the broadcast capability of the wireless medium and thus achieve increased reliability was explored in [81], [77]. This point was also discussed in detail in [1] were related work was provided.

Finally random linear combinations can be used to facilitate information delivery (network coding), to offer redundancy (channel coding) as well as compress redundant information (source coding). Thus all these three functionalities can be potentially combined in a sensor network.

7 Conclusions and Guidelines for Future Work

In this report we have provided a literature review of network coding techniques applied to applications, focusing on distributed storage, content distribution, network management and monitoring, and security.

We have seen that for all of them there exist results which indicate that network coding can be successfully used on a variety of scenarios in order to improve performance, increase efficiency and reduce overhead cost (such as repair bandwidth). In particular,

1. In distributed storage, various proposals show that network coding can effectively be used to construct a class of codes achieving the optimal storage-bandwidth trade-off.
2. For P2P content distribution, results show that network coding does indeed provide better performance in realistic scenarios where nodes join and leave at any time, and each node has partial or outdated information.
3. Regarding layered source coding for content distribution, existing work indicate that by combining it with network coding it is possible to obtain better performance when multicasting information to heterogeneous receivers with different network capacities.
4. Finally, there exist results showing that efficient protection against both eavesdropping and byzantine attacks can be designed by using different network codes.

In general, for the case of content distribution and storage, network coding shows the most notable performance gain in realistic scenarios in which there are communication errors, frequent topology changes or nodes joins and leaves. The question arises, then, whether that performance can be extrapolated to the type of scenario considered in the project, namely wireless networks in resource-challenged environments such as highly mobile ad-hoc and mesh networks. A second question is if new codes have to be designed in order to operate efficiently in such hostile environments.

Most of the existing state-of-the art considers performance in terms of throughput, and signaling overhead (such as repair bandwidth or storage-bandwidth trade-off). However, other important metrics such as delay or fairness are often ignored. Further questions also arise:

- Can an optimal trade-off curve be calculated under heavy node join and departures for distributed storage systems? Can we design efficient codes for volatile environments with reasonable repair bandwidth consumption and still provide a low replication factor? Can we efficiently store content in a dynamic, volatile group?

- Can we design a P2P network for error-prone wireless mobile clouds with low delay and fair operation? Can network coding approach the multi-rate equivalent multicast min-cut in such environments? And can those codes be transformed in algorithms capable of running in mobile devices with a small memory and CPU capacity?
- And finally, what consequences does the wireless medium have on the design of security measures for network coding protocols?

In order to further understand those questions we explored how network coding is used in other applications that run on sensor networks. The techniques used on those networks is relevant because sensor networks are volatile wireless environments with limited storage and power, and hence are of primary interest to this project.

As future work, we will try to answer these question, looking into extending these promising results to the scenarios contemplated in N-CRAVE, or advancing on the design of new techniques that, incorporating the results of WP2, can provide comparable levels of performance in wireless mobile environments.

References

- [1] N-Crave D1.1: Report on State of the Art. Technical report, 2008.
- [2] N-Crave D2.1: Report on State of the Art. Technical report, 2009.
- [3] Y. Wu, A. Dimakis, and K. Ramchandran. Deterministic regenerating codes for distributed storage. *in Proc. 45th Allerton Conf. Comm. Control and Computing*, Sept. 2007.
- [4] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. Wainwright, and K. Ramchandran. Network coding for distributed storage systems. *submitted for publication (Preliminary versions appeared in Infocom 2007 and Allerton 2007)*, 2007.
- [5] Dah Ming Chiu, Raymond W. Yeung, Jiaqing Huang, and Bin Fan. Can network coding help in p2p networks? In *4th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, 2006.
- [6] Christos Gkantsidis and Pablo Rodriguez. Network coding for large scale content distribution. In *INFOCOM*, 2005.
- [7] Christos Gkantsidis, John Miller, and Pablo Rodriguez. Anatomy of a p2p content distribution system with network coding. In *International Workshop on Peer-To-Peer Systems*, 2006.

- [8] Christos Gkantsidis, John Miller, and Pablo Rodriguez. Comprehensive view of a live network coding p2p system. In *Internet Measurement Conference*, 2006.
- [9] T.C. Jepson. The basics of reliable distributed storage networks. *IT Professional*, 6(3):18–24, May-June 2004.
- [10] S. Acedanski, S. Deb, M. Medard, and R. Koetter. How good is random linear coding based distributed networked storage? in *NetCod Workshop*, April 2005.
- [11] A. G. Dimakis, V. Prabhakaran, and K. Ramchandran. Decentralized erasure codes for distributed networked storage. *IEEE Transactions on Information Theory*, June 2006.
- [12] A.G.Dimakis, V.Prabhakaran, and K.Ramchandran. Ubiquitous access to distributed data in large-scale sensor networks through decentralized erasure codes. *Proc. of IEEE/ACM Int. Symposium on Information Processing in Sensor Networks (IPSN)*, 2005.
- [13] Shu Lin and Daniel J. Costello. *Error Control Coding, Second Edition*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2004.
- [14] Michael Mitzenmacher Michael G. Luby and M. Amin Shokrollahi. Efficient erasure correcting codes. *IEEE Transactions on Information Theory*, 47(2):569–584, 2001.
- [15] H.Weatherspoon and J.Kubiatowicz. Erasure coding vs. replication:a quantitative comparison. in *Proc. IPTPS*, 2002.
- [16] R. Rodrigues and B. Liskov. High availability in dhds- erasure coding vs replication. in *Proc. IPTPS*, 2005.
- [17] S. Sanders, S. Egner, and L. Tolhuizen. Polynomial time algorithms for network information flow. In in *Proc. of SPAA '03*, June 2003.
- [18] Balachander Krishnamurthy, Craig Wills, and Yin Zhang. On the use and performance of content distribution networks. In *IMW '01: Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, pages 169–182, New York, NY, USA, 2001. ACM.
- [19] George Pallis and Athena Vakali. Insight and perspectives for content delivery networks. *Commun. ACM*, 49(1):101–106, 2006.
- [20] A. M. K. Pathan and R. Buyya. A taxonomy and survey of cdns. Technical Report GRIDS-TR-2007-4, The University of Melbourne, Australia, Feb 2007.

- [21] S. Jingjing, Z. Bojin, and C. Anni. Optimal layered multicast using network coding. In *Proceedings of the 4th Network Coding conference*, January 2008.
- [22] Yunnan Wu. A comparison of network coding and tree packing. In *in Proc. 2004 IEEE International Symposium on Information Theory (ISIT 2004)*, page 143, 2004.
- [23] R.W. Yeung. Avalanche: A network coding analysis. preprint 2005.
- [24] Tracey Ho, Ralf Koetter, Muriel Medard, D. R. Karger, and Michelle Effros. The benefits of coding over routing in a randomized setting. In *IEEE International Symposium on Information Theory*, 2003.
- [25] Uichin Lee, Joon-Sang Park, Joseph Yeh, Giovanni Pau, and Mario Gerla. Codetorrent: Content distribution using network coding in vanet. In *MobiShare*, 2006.
- [26] Viktoria Fodor and Gyorgy Dan. Resilience in live peer-to-peer streaming. *IEEE Communications Magazine*, pages 116–123, 2007.
- [27] Mea Wang and Baochun Li. How practical is network coding? In *Proceedings of the 14th IEEE International Workshop on Quality of Service (IWQoS)*, pages 274–278, 2006.
- [28] Mea Wang and Baochun Li. Lava: A reality check of network coding in peer-to-peer live streaming. In *Proceedings of IEEE Infocom*, 2007.
- [29] V.K. Goyal. Multiple description coding: compression meets the network. *IEEE Signal Processing Magazine*, Volume 18, Issue 5:74 – 93, Sep 2001.
- [30] J. Zhao, F. Yang, Q. Zhang, Z. Zhang, and F. Zhang. Lion: Layered overlay multicast with network coding. *IEEE Transactions on Multimedia*, Volume 8, Issue 5, Oct. 2006.
- [31] L. Chen, T. Ho, S. H. Low, M. Chiang, and J. C. Doyle. Optimization based rate control for multicast with network coding. In *Proceedings of INFOCOM 2007*, May 2007.
- [32] N. Sundaram, P. Ramanathan, and S. Banerjee. Multirate media streaming using network coding. In *Proc. 43rd Allerton Conference on Communication, Control, and Computing*, Sep. 2005.
- [33] M.Shao, X.Wu, and N.Sarshar. Rainbow network flow with network coding. In *Proceedings of 4th Network Coding Conference 2008*, January 2008.

- [34] R. W. Yeung, S. R. Li, N. Cai, and Z. Zhang. *Network Coding Theory*. Now Publishers, 2005.
- [35] X.Wu, B.Ma, and N.Sarshar. Rainbow network problems and multiple description coding. In *Proceedings of ISIT 2005*, pp. 268-272, 2005.
- [36] Y. Cassuto and J. Bruck. Network coding for nonuniform demands. *ISIT*, 2005.
- [37] M.Shao, S.Dumitrescu, and X.Wu. Toward the optimal multirate multicast for lossy packet network. In *16th ACM international conference on Multimedia*, 2008.
- [38] Y. Wu. Distributing layered content using network coding. *IEEE International Workshop on Wireless Network Coding (invited paper)*, June 2008.
- [39] Y. Wu. On constructive multi-source network coding. in *Proc. IEEE Int'l. Symp. Information Theory*, July 2006.
- [40] D. Silva D. Wang and F. R. Kschischang. Constricting the Adversary: A Broadcast Transformation for Network Coding. *Allerton Conference on Communication, Control, and Computing, Monticello, IL, September 26-28, 2007*, 2007.
- [41] Sachin Katti, Jeffery Cohen, and Dina Katabi. Information Slicing: Anonymity Using Unreliable Overlays. In *Proc. of the 4th USENIX Symposium on Network Systems Design and Implementation (NSDI)*, April 2007.
- [42] N. Cai and RW Yeung. Secure network coding. In *Proceedings of the IEEE International Symposium on Information Theory*, Lausanne, Switzerland, July 2002.
- [43] Ning Cai and Raymond W. Yeung. A Security Condition for Multi-Source Linear Network Coding. In *IEEE International Symposium on Information Theory (ISIT)*, Nice, France, June 2007.
- [44] Raymond W. Yeung and Ning Cai. On the Optimality of a Construction of Secure Network Codes. In *IEEE International Symposium on Information Theory (ISIT)*, July 2008.
- [45] Danilo Silva and Frank R. Kschischang. Security for Wiretap Networks via Rank-Metric Codes. In *IEEE International Symposium on Information Theory (ISIT)*, July 2008.
- [46] J. P. Vilela, L. Lima, and J. Barros. Lightweight security for network coding. *IEEE International Conference on Communications, 2008*, pages 1750–1754, May 2008.

- [47] L. Lima, J. P. Vilela, J. Barros, and M. Médard. An Information-Theoretic Cryptanalysis of Network Coding – is protecting the code enough? *International Symposium on Information Theory and its Applications (ISITA)*, 2008.
- [48] D.S. Lun, M. Médard, R. Koetter, and M. Effros. On Coding for Reliable Communication over Packet Networks. *Proc. 42nd Annual Allerton Conference on Communication, Control, and Computing*, 2005.
- [49] R. Kötter and F. R. Kschischang. Coding for Errors and Erasures in Random Network Coding. *IEEE Trans. Inf. Theory*, 54:3579–3591, Aug. 2008.
- [50] D. Silva, F. R. Kschischang, and R. Kötter. A Rank-Metric Approach to Error Control in Random Network Coding. *IEEE Trans. Inf. Theory*, 54:3951–3967, Sep. 2008.
- [51] D. Silva and FR Kschischang. Using rank-metric codes for error correction in random network coding. *Proc. of the 2007 IEEE International Symposium on Information Theory*, 2007.
- [52] D. Silva, F. R. Kschischang, and R. Kötter. Capacity of Random Network Coding under a Probabilistic Error Model. In *24th Bien. Sym. on Comm.*, pages 9–12, Kingston, ON, Canada, Jun. 2008.
- [53] A. Montanari and R. Urbanke. Coding for Network Coding. *Arxiv preprint arXiv:0711.3935*, 2007.
- [54] N. Cai and R.W. Yeung. Network error correction. *Proceedings of the IEEE International Symposium on Information Theory, Yokohama, Japan, July*, July 2003.
- [55] S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, and M. Médard. Resilient Network Coding In the Presence of Byzantine Adversaries. In *Proceedings of INFOCOM 2007*, Anchorage, Alaska, May 2007.
- [56] Leah Nutman and Michael Langberg. Adversarial Models and Resilient Schemes for Network Coding. In *IEEE International Symposium on Information Theory (ISIT)*, Toronto, Canada, July 2008.
- [57] T. Ho, B. Leong, R. Koetter, M. Médard, M. Effros, and DR Karger. Byzantine modification detection in multicast networks using randomized network coding. *IEEE International Symposium on Information Theory (ISIT)*, 2004.
- [58] D. Charles, Kamal Jain, and K. Lauter. Signatures for network coding. *Proceedings of the 40th Annual Conference on Information Sciences and Systems, Princeton, USA, March*, pages 857–863, March 2006.

- [59] M. N. Krohn, M. J. Freedman, and D. Mazieres. On-the-fly verification of rateless erasure codes for efficient content distribution. In *Security and Privacy, 2004. Proceedings. 2004 IEEE Symposium on*, pages 226–240, 2004.
- [60] C. Gkantsidis and P. R. Rodriguez. Cooperative security for network coding file distribution. In *Proceedings of IEEE Infocom*, Barcelona, Spain, April 2006.
- [61] F. Zhao, T. Kalker, M. Medard, and K. J. Han. Signatures for content distribution with network coding. In *International Symposium on Information Theory (ISIT)*, Nice, France, 2007.
- [62] MinJi Kim, Muriel Médard, and João Barros. Counteracting byzantine adversaries with network coding: An overhead analysis. *MILCOM*, 2008.
- [63] P. A. Chou and Y. Wu and K. Jain. Practical Network Coding. In *Proc. 2003 Allerton Conf. on Commun., Control and Computing*, Monticello, IL, Oct. 2003.
- [64] J. H. van Lint and R. M. Wilson. *A Course in Combinatorics*. Cambridge University Press, 1992.
- [65] Christina Fragouli and Athina Markopoulou. A network coding approach to overlay network monitoring. In *43rd Allerton Conference on Communication, Control, and Computing*, 2005.
- [66] Tracey Ho, Ben Leong, Yu-Han Chang, Yonggang Wen, and Ralf Koetter. Network monitoring in multicast networks using network coding. In *IEEE International Symposium on Information Theory*, 2005.
- [67] Tracey Ho, Muriel Medard, Ralf Koetter, David R. Karger, Michelle Efros, Jun Shi, and Ben Leong. Toward a random operation of networks. *IEEE Transactions on Information Theory*, submitted, 2004.
- [68] C. Fragouli, A. Markopoulou, R. Srinivasan, and S N. Diggavi. Network monitoring: It depends on your points of view. In *Information Theory and its applications workshop '07*, San Diego, CA, January 2007.
- [69] M. Gjoka, C. Fragouli, P. Sattari, and A. Markopoulou. Loss tomography in general topologies with network coding. In *in Proc. of Globecom'07*, 2007.
- [70] S. Sanders, S. Egner, and L. Tolhuizen. Topology inference using network coding. In *in Proc. of Allerton Conference on Communication, Control, and Computing*, Sept. 2006.

- [71] M. Jafarisiavoshani, C. Fragouli, , and S. Diggavi. Subspace properties of randomized network coding. In *in Proc. of Information Theory Workshop (ITW'07)*, Bergen, Germany, June 2007.
- [72] M. Jafarisiavoshani, C. Fragouli, S. Diggavi, and C. Gkantsidis. “bottleneck discovery and management in network coded peer-to-peer systems”. In *ACM SIGCOMM Workshop on Internet Network Management*, Kyoyo, Japan, Aug 2007.
- [73] Tracey Ho, Muriel Medard, and Ralf Koetter. A coding view of network recovery and management for single-receiver communications. In *Conference on Information Sciences and Systems*, 2002.
- [74] Tracey Ho, Muriel Medard, and Ralf Koetter. A coding view of network capacity, recovery and management. In *IEEE International Symposium on Information Theory*, 2002.
- [75] Tracey Ho, Muriel Medard, and Ralf Koetter. An information theoretic view of network management. In *INFOCOM*, 2003.
- [76] A. Kamra, V. Misra, J. Feldman, and D. Rubenstein. Growth codes: maximizing sensor network data persistence. *ACM SIGCOMM*, 36, 2006.
- [77] Z. Guo, P. Xie, J.-H. Cui, and B. Wang. On applying network coding to underwater sensor networks. In *in Proc. of the 1st ACM International Workshop on Underwater Networks*, Sept 2006.
- [78] M. Ghaderi, D. Towsley, and J. Kurose. Network coding performance for reliable multicast. *Military Communication Conference (Milcom)*, 2007.
- [79] C. Fragouli, J. Widmer, and J.-Y. Le Boudec. On the benefits of network coding for wireless applications. In *Network Coding Workshop*, Boston, 2006.
- [80] Dragan Petrović, Kannan Ramchandran, and Jan Rabaey. Overcoming untuned radios in wireless networks with network coding. *IEEE/ACM Trans. Netw.*, 14(SI):2649–2657, 2006.
- [81] C. Adjih, S. Y. Cho, and P. Jacquet. Near optimal broadcast with network coding in large sensor networks. In *in Proc. of the 1st International Workshop on Information Theory for Sensor Networks (WITS)*, June 2007.