

國立交通大學

電機資訊國際學位學程

碩 士 論 文

以實驗為依據對隨機線性網路編碼非均等抹除保護能力之探討

**Unequal Erasure Protection Capability of Randomized
Linear Network Codes - an Empirical Study.**

研究生：柯子東

指導教授：邵家健 博士

中華民國一百年五月

Unequal Erasure Protection Capability of Randomized Linear Network Codes - an Empirical Study

研究生：柯子東
指導教授：邵家健 博士

Student: Nicolas Claude
Advisor: Dr. John Kar-kin Zao

國立交通大學
電機資訊國際學位學程

碩士論文



Submitted to EECS International Graduate Program

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of

Master

May 2011

Hsinchu, Taiwan, Republic of China

中華民國 一 百 年 五 月

以實驗為依據對隨機線性網路編碼非均等抹除保護能力之探討

學生：柯子東

指導教授：邵家健 博士

國立交通大學

電機資訊國際學位學程

摘要

關鍵詞： 實做網路編碼，分離矩陣，非均等抹除保護。

在這篇論文中，我們將探討如何使用線性的隨機網路編碼技術 (Random Linear Network Codes, RLNC) 如何達成非均勻的資料抹除保護機制 (Unequal Erasure Protection, UEP)。在眾多的資料流中，有些子分流與其他的相比之下有較高的優先權。為了對這些相對較重要的分流有比較安全的保護機制，必須要選擇性停止某些網路中的傳輸中繼端點的資料傳輸。這個編碼方式可量化為分離矩陣 (Separation Vector)；並在實驗中更進一步採用一夠大的整數有限域來提升解碼的成功率，而產生並推導這些編碼方式的可行性可透過圖型理論來解決。

除此之外，此研究的最大貢獻是以 Java 語言實做出 UEP-RLNC 的資料傳輸系統。並設計實驗模擬網路中有可能遇到的傳輸問題如封包遺失，觀察此編碼方式如何能面對常遇到的傳輸問題。實驗結果主要為顯示在網路環境不穩定的情景下，網路終端的接收點收錄並成功解碼封包所占的比例。

Unequal Erasure Protection Capability of Randomized Linear Network Codes - an Empirical Study.

Student: Nicolas Claude

Advisor: Dr. John Kar-kin Zao

EECS International Graduate Program

National Chiao Tung University

Abstract

Keywords: Practical Network Coding, Separation Vector, Unequal Erasure Protection.

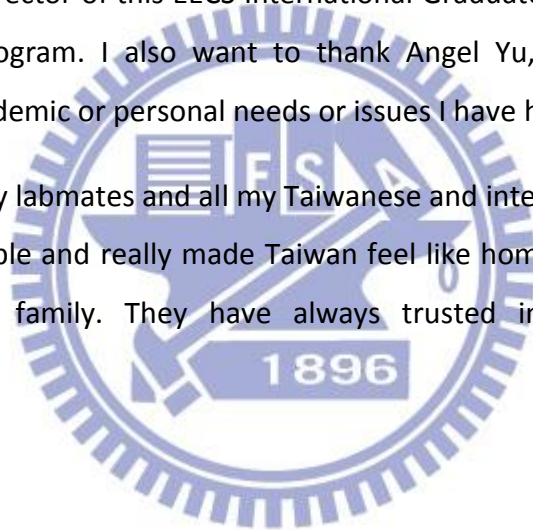
We investigate in this thesis how Random Linear Network Codes can achieve Unequal Erasure Protection. In many data streams, some layers of data have priority over the others. In order to enhance protection of particular layers with RLNC, we need to nullify local encoding coefficients at some specific nodes. The resulting coding schemes can be quantified by using Separation Vector. Furthermore, granted a finite Field large enough to have high theoretical decoding probability, evaluating and generating those coding schemes can be achieved using Graph Theory. Besides this model, our main contribution is an actual implementation of UEP-RLNC in JAVA. Some simulations were performed to verify the validity of our UEP mechanism as well as its quantification. The results obtained show the profiles of retrieval rate with respect to packet error loss for both hotspot losses and background losses scenarios.

Acknowledgements

Above all, I would like to thank my advisor, Prof. John K. Zao, for his guidance, support and patience during my Master study in NCTU. He taught me how to conduct research and always gave me valuable advices about my work. I want to thank also Mr. Yao Chien, for his precious help and concern.

This thesis would not be possible without the support of Pr. Noël Crespi and Pr. John K. Zao, coordinators for this Dual Degree program between Telecom SudParis in France and National Chiao Tung University in Taiwan. And I have to mention Prof. Charles T.M. Choi, director of this EECS International Graduate Program for his work among the EECS program. I also want to thank Angel Yu, for her support and assistance in any academic or personal needs or issues I have had.

I am grateful to all my labmates and all my Taiwanese and international friends. They made my life enjoyable and really made Taiwan feel like home. Last but not least, I want to thank my family. They have always trusted in me and given me encouragement.



List of Figures

Figure 2.1. Example of Packet Combination in Butterfly Network.....	3
Figure 2.2: Graph Transformation from the Topology Graph (a) to Line Graph (b).....	9
Figure 2.3: Subtree Decomposition from Line Graph (a) to Subtree Graph (b)	10
Figure 3.1: Dispatching and Transmission Processes	12
Figure 3.2: Traditionnal Routing vs NC Transmission at Local Scale	15
Figure 4.1: Classes Overall View	22
Figure 4.2: Coefficient Element	23
Figure 4.3: Packet.....	24
Figure 4.4: Buffer	25
Figure 4.5: Buffering at a Node.....	26
Figure 4.6: Decoding Mechanism	28
Figure 4.7: Source Inner Structure.....	29
Figure 4.8: Relay Inner Structure	30
Figure 4.9: Sink Inner Structure	31
Figure 5.1: Simulated Topology	33
Figure 5.2: Sink1 Facing Bottleneck Hotspot (Diagram (1) from Table 5.3)	37
Figure 5.3: Sink2 Facing Bottleneck Hotspot (Diagram (1) from Table 5.3)	38
Figure 5.4: Sink1 Facing Hotspot on Direct Link (Diagram (2) from Table 5.3)	39
Figure 5.5: Sink1 Facing Hotspot on Direct Link (Diagram (2) from Table 5.3)	39
Figure 5.6: Migration #1 from Coding Scheme (a) to Coding Scheme (b)	40
Figure 5.7: Sink1 migration #1	40
Figure 5.8: Migration #2 from Coding Scheme (a) to (c)	41
Figure 5.9: Sink 1 migration #2	41
Figure 5.10: retrieval rate at Sink1 (background losses)	42
Figure 5.11: retrieval rate at Sink2 (background losses)	43
Figure 5.12: Topology (a)	44
Figure 5.13: Sink 1 after background losses	45
Figure 5.14: Sink2 after background losses	45



List of Tables

Table 2.1: Advantages and Drawbacks of RLNC	8
Table 3.1: Exhaustive List of Minimal Subtree Graph for 3-Sources-2-Receiver	19
Table 4.1: Old vs New Features	21
Table 4.2: Summary of Threads Used in Application	25
Table 5.1: Simulation parameters.....	34
Table 5.2: Coding Scheme for Failure Pattern Evaluation	34
Table 5.3: Assessed Failure Patterns	36
Table 5.4: Categories of Links	36

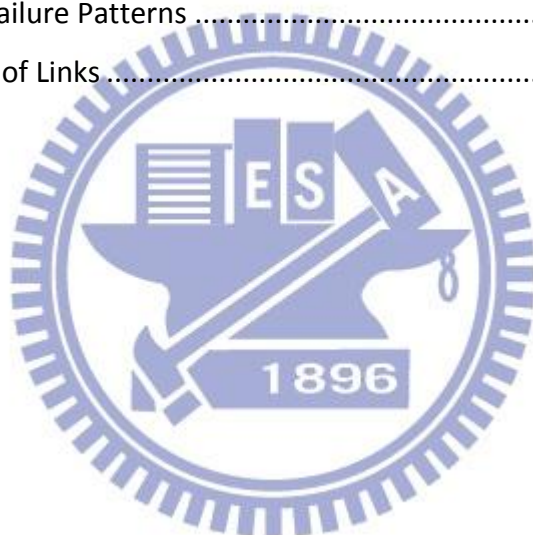


Table of Contents

Abstract.....	ii
Acknowledgements.....	iii
List of Figures	iv
List of Tables	vi
Chapter 1 Problem Statement.....	1
1.1 Motivation.....	1
1.2 Objectives.....	1
1.3 Research Approach	1
1.4 Thesis Outline.....	2
Chapter 2 Background.....	3
2.1 Network Coding.....	3
2.2 Information Flow Decomposition	9
Chapter 3 Achieving UEP Thanks to RLNC.....	12
3.1 General Approach	12
3.2 Unequal Erasure Protection (UEP).....	13
3.3 Coding Rules.....	14
3.4 Assessing UEP Capabilities of Network Codes	16
Chapter 4 Implementation	21
4.1 From Existing Code.....	21
4.2 Galois Library.....	22
4.3 Data Structure Package	23
4.4 MyThreads Package	25
4.5 Network Coding Package	26
4.6 Simulation Package	28
Chapter 5 Experiments	33
5.1 Simulated Topology and Parameters.....	33

5.2	Simulation 1: Evaluation of Coding Schemes.....	34
5.3	Simulation 2: Hotspot.....	36
5.4	Simulation 3: Background Erasures.....	42
Chapter 6	Conclusion.....	47
6.1	Contributions.....	47
6.2	Future Work	47
References	48



Chapter 1 Problem Statement

1.1 Motivation

Unequal Erasure Protection (UEP) is a widely used feature in the field of scalable data multicast. It grants some of the symbols a higher chance to be retrieved at the destinations by prioritizing data layers according to their utility value.

Network Coding (NC), a recent breakthrough in the field of information theory, seems to be particularly well adapted to provide UEP. The idea at the origin of NC is that information shouldn't be treated as a parcel. So far, we were only applying coding mechanisms at the source and the final destination of a packet and treating packets in between following the inherited "store and forward" principle [1]. Actually, it can be processed at any hop between the source and the destination. By doing so, we significantly improve the efficiency of transmission by combining packets together.

1.2 Objectives

The purpose of this research is to enable UEP thanks to Network Coding. By investigating these two cutting edge topics together we aim at producing a practical application that shows both NC advantages and UEP capabilities. In other words, we want to generate, assess, and implement coding schemes that enable UEP in some small networks.

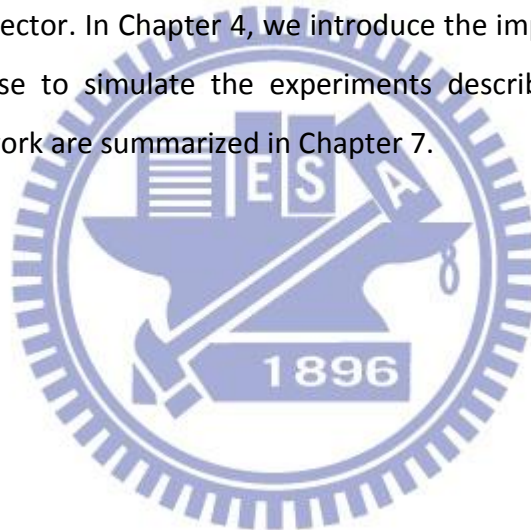
1.3 Research Approach

Combining Random Linear Network Coding (RLNC) and UEP can only be done by introducing specific coding schemes. Some prior work [4] was done to determine the best coding schemes for a specific topology. However, our work differs from that one since we plan to investigate UEP properties of Network Codes. In that sense, we have a broader approach that consists in taking into account unequal survivability of data to failures in addition to the size of the finite field or the rank of the global encoding kernels.

Our work is based on RLNC [2] and its practical implementation [3]. We assume that given a big enough finite field, the issue of finding an optimal coding scheme is equivalent to a graph problem. In order to assess that optimality, we introduce the concept of Separation Vector that is well known for usual channel coding but has not yet been used with Network Coding.

1.4 Thesis Outline

The rest of this thesis is organized as follows: we will review Network Coding, and Information Flow decomposition (a recent breakthrough in this field) in Chapter 2. Chapter 3 presents the definition of UEP, the concept of coding rule, and our mean of quantifying UEP capabilities: separation vector. In Chapter 4, we introduce the implementation of UEP-RLNC and the platform we use to simulate the experiments described in Chapter 5. Finally, conclusions and future work are summarized in Chapter 7.



Chapter 2 Background

We present in this chapter the concepts of NC, RLNC, and Information flow decomposition, one of the latest breakthroughs in this field.

2.1 Network Coding

2.1.1 Basic Concept

The basic concept of Network Coding is that we consider packets that can be combined together at any hop of the path from the source to the destination. The combination of packets can be performed in many ways but in the context of Linear Network Coding, packets are linearly combined together.

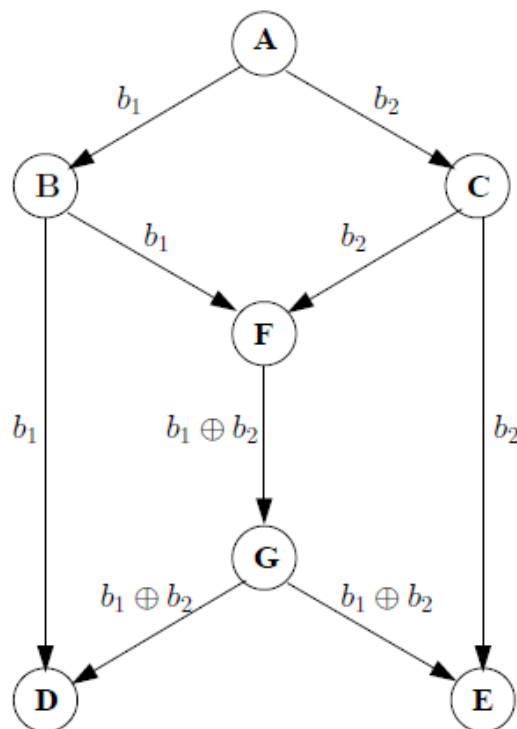


Figure 2.1. Example of Packet Combination in Butterfly Network

In Figure 2.1, the node A (source) tries to send two symbols b_1 and b_2 to the nodes D and E (sinks). The node F mixes the two symbols it just received. The resulting symbol is noted

$b_1 \oplus b_2$. If we assume that each link has a capacity of transmitting one symbol per unit of time, the multicast rate is then two symbols per unit of time. Without network coding, it would have been impossible to transmit the two symbols on the bottleneck between F and G. So the multicast rate would have been only one symbol per unit of time.

2.1.2 Main Theoretical Result

In Figure 2.1, we could achieve the multicast rate of two symbols per unit of time thanks to Network Coding. From a naïve perspective, we could say that Network Coding enables all receivers to use all of their possible paths from the source simultaneously.

This example illustrates the main theoretical result on Network Coding that can be extended to any acyclic network:

For the unicast case, the famous max-flow min-cut theorem [1] states that a source node s can send information through a network (V, E) to a sink node t at a rate ω determined by the min-cut value separating s and t .

In the multicast case, the upper bound for the achievable transmission rate is the minimum of the maximum flow for each sink. We note it

$$\omega = \min_{t \in T} \text{MinCut}(s, t)$$

where:

- T is the set of receiver nodes in the graph
- s is the source node considered
- t is the destination

In general, the multicast rate of ω suggests the existence of ω non-intersecting paths from the source to each sink, which are called edge-disjoint paths in [2], although the paths destined to different receivers may share some edges. It was proved in [3] that reliable multicast at a rate equal to the upper bound ω , can always be achieved in any network using network coding. Therefore, we can always benefit from the possible advantages of Network coding described in [4]: increased bandwidth, robustness to dynamical changes of

topology, minimizing energy per bit for wireless communications, minimizing delay, enhanced security.

In contrast, most of the time traditional routing cannot reach the upper bound ω . Even when it can, working out the multicast trees (Steiner trees) that achieve that rate is a NP-hard problem.

2.1.3 Random Linear Network coding (RLNC)

It was proved in [5] that linear network coding is sufficient to achieve the multicast rate in any acyclic network. Furthermore, deterministic polynomial time algorithms and even faster randomized algorithms were found in [6] for directed acyclic graphs with edges of unit capacity. Some codes that are tolerant to edge failures were designed. These results are very interesting since RLNC can achieve the highest multicast rate with reasonable calculation time.

2.1.3.1 Local and Global Encoding Vectors

Whenever a packet p reaches a specific node e , the packet p is a linear combination of the original packets. The coefficients with which you can express the packet received in function of the original packets from the source constitute the **global encoding vector**. These coefficients are the result of every mixing that occurs at each hop of the packet.

$$y_p(e) = \sum_{k=1}^n a_{p,k} x_k$$

where:

- $\vec{g}_p(e) = [a_{p,1} \ \dots \ a_{p,n}]$ is the global encoding vector at the node e for the packet p .
- $y_p(e)$ is the coded value received at the node e with the packet p .

After receiving a certain number of packets m , the node e generates a **Local Encoding Vector** $\vec{l}_j(e)$. This vector contains the coefficients that this specific node will use to combine the k packets it just received in order to generate a new packet j and to send it to the next hop f .

$$\vec{l}_j(e) = [b_1(e) \quad \dots \quad b_m(e)]$$

So one of the packets (say the j^{th}) received by the next hop (g) will be:

$$y_j(f) = \vec{l}_j(e) \times \begin{bmatrix} y_1(e) \\ \vdots \\ y_m(e) \end{bmatrix} = \sum_{p=1}^m b_p(e) y_p(e)$$

If we try to express $\begin{bmatrix} y_1(e) \\ \vdots \\ y_m(e) \end{bmatrix}$ in function of $\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$, we obtain:

$$\begin{bmatrix} y_1(e) \\ \vdots \\ y_m(e) \end{bmatrix} = \begin{bmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m,1} & a_{m,2} & \dots & a_{m,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

where:

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m,1} & a_{m,2} & \dots & a_{m,n} \end{bmatrix} = \begin{bmatrix} \vec{g}_1(e) \\ \vec{g}_2(e) \\ \vdots \\ \vec{g}_m(e) \end{bmatrix} = G(e)$$

$G(e)$ is the matrix composed of the global encoding vector for each of the m packets that arrived at the node e . This matrix $G(e)$ is composed of rows that come from the different matrices of the predecessors of e . Note that without this matrix, it is impossible to retrieve the initial value \vec{x} .

We also define the **Local encoding kernel** and **global encoding kernel** as the juxtaposition of respectively the local encoding vectors and the global encoding vectors at a node.

2.1.3.2 Packet Tagging

In order to facilitate the decoding of the final GEK matrix, we tag each packet with the coefficients of its own global encoding vector. This procedure was first used in [7], under the name "packet tagging".

At the source, instead of combining only a set of symbols $\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$, we juxtapose it to I_n and send this resulting matrix one line (= one datagram) at a time:

$$M(\text{source}) = \begin{bmatrix} 1 & 0 & \cdots & 0 & x_1 \\ 0 & 1 & \ddots & \vdots & x_2 \\ \vdots & \ddots & \ddots & 0 & \vdots \\ 0 & \cdots & 0 & 1 & x_n \end{bmatrix}$$

After a certain number of hops, m different packets will arrive at the node e . So the content of the buffer at the node e will be:

$$M(e) = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} & y_1(e) \\ a_{2,1} & a_{2,2} & \ddots & a_{2,n} & y_2(e) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} & y_m(e) \end{bmatrix}$$

This is due to the fact that the unit vectors will go through the exact same path that the rest of the packets. Thus, it will undergo the same combinations. After extracting the different information from this last matrix, we obtain the equation:

$$G(e) \times \vec{x} = \vec{y}(e)$$

$$G(e) = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \ddots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{bmatrix}; \vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}; \vec{y}(e) = \begin{bmatrix} y_1(e) \\ y_2(e) \\ \vdots \\ y_m(e) \end{bmatrix}$$

where for any packet i that was received by the destination:

- $y_i(e)$ is the coded value received.
- $\vec{g}_i(e) = [a_{i,1} \ \cdots \ a_{i,n}]$ is the global encoding vector at the destination for the i^{th} packet.

We need to solve the above system for the destination to get the original symbols that the source coded and sent. Therefore, \vec{x} is the unknown. Finding the solution of this linear equation system leads to finding the original values.

2.1.3.3 Random Encoding and Invertibility

The key idea in Random Linear Network Codes is that the Local Encoding Vectors have to be chosen randomly among the finite field, independently between nodes and following a uniform law. This approach enables [4] to provide us with a lower bound on the invertibility probability of the global encoding kernel:

$$P(G_t \text{ is invertible}) \geq 1 - \frac{E}{F}$$

where:

- E is the number of edges in the network,
- F is the size of the Galois field in which you choose the coefficients.
- G_t is the global encoding kernel at the sink t

The finite field has to be big enough for a given G_t to have a high invertibility probability. But on the other hand, bigger Finite field would lead to bigger overhead. Thus, this parameter has to be chosen carefully.

advantages	drawbacks
No need for an optimal coding scheme since this one performs very well in average	Efficient in average only: Certainly can perform better (size of the Finite field, ...)
Decentralized : if you can propagate the coefficients	Doesn't prioritize data
Easy to make use of it	

Table 2.1: Advantages and Drawbacks of RLNC

Although RLNC can already achieve the maximum throughput for a multicast session, very little work has been done on creating optimal network codes with respect to UEP.

2.1.3.4 Data Protection of RLNC

A trivial approach is to achieve UEP through basic redundancy. By sending multiple combinations of the same packets over edge-disjoint paths, you can improve the protection of original packets. This technique is the key idea in the approach in [8]. This protection is a

built-in feature of RLNC but it has two major drawbacks. Firstly, sending redundant packets is very costly in terms of network resource. Secondly, all packets have the same protection level. It is impossible to prioritize certain data stream over the others with such a trivial technique.

2.2 Information Flow Decomposition

Thanks to [9], we can turn an algebraic problem into a graph problem. This result is of utmost importance since we can try new approaches to solve the thorny problem of optimal coding schemes. Hence, we explore here a decomposition model of the information flow spread through the network.

Let $G = (V, E)$ be a given topology graph. The definition of Line Graph is:

$$\gamma = \bigcup_{\substack{1 \leq i \leq h \\ 1 \leq j \leq N}} L(S_i, R_j)$$

where $L(S_i, R_j)$ is the Line Graph of the path from source S_i to the receiver R_j . In other words, $L(S_i, R_j)$ is the graph with vertex set $E(S_i, R_j)$ in which two vertices are joined if and only if they are adjacent as edges in the path (S_i, R_j) .

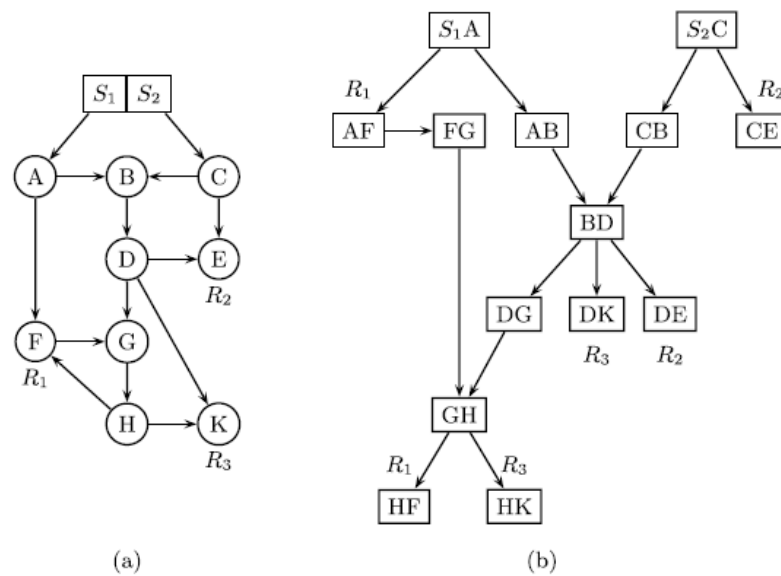


Figure 2.2: Graph Transformation from the Topology Graph (a) to Line Graph (b)

Definition of Subtree Graph:

On the line graph, we can identify some Subtrees inside which the same information will be propagated. Therefore, we can consider the corresponding **Subtree Graph** $\Gamma = (E_\Gamma, V_\Gamma)$.

In Γ , each Subtree corresponds to either a source Subtree or a coding Subtree. The coding Subtrees are the Subtrees where the root receives multiple inputs, the source Subtree is rooted at a source node. These nodes are where the network coding encoding operations actually take place. In fact, all the other nodes merely relay the information they received.

Thus, by considering the Subtree graph instead of the topology graph or the line graph, we can:

- Generalize NC behavior to several different topologies.
- Identify the real amount of coding points needed.
- Reduce the size of the alphabet (Galois Field).

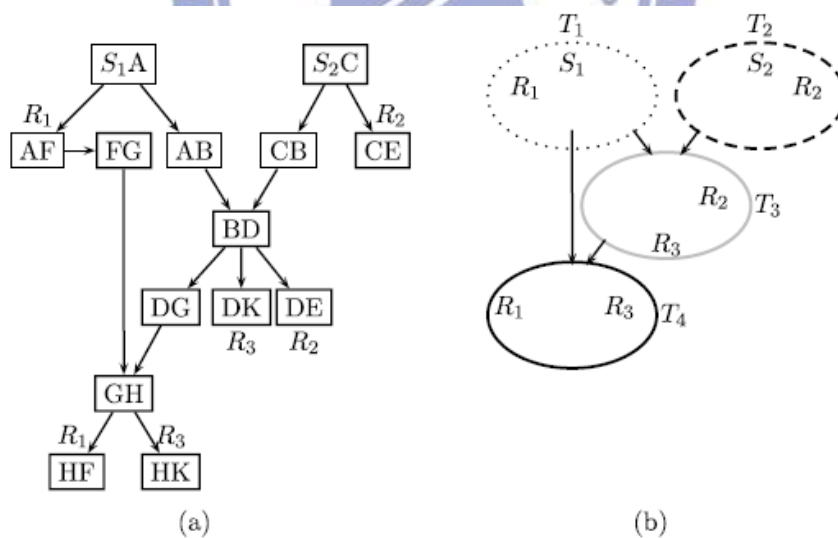


Figure 2.3: Subtree Decomposition from Line Graph (a) to Subtree Graph (b)

Definition of Minimal Subtree Graph:

A Subtree graph is called minimal with the multicast property if removing any edge would violate the multicast property.

Identifying a **Minimal Subtree Graph** before multicasting may allow us to reduce the number of coding points further down and thus to use less network resources. Furthermore, such graphs have some structural properties, which can be exploited to derive several theoretical results.

Properties:

For a Minimal Subtree Graph, the following holds:

- 1) A valid network code where a Subtree is assigned the same coding vector as one of its parents does not exist.
- 2) A valid network code where the vectors assigned to the parents of any given Subtree are linearly dependent does not exist.
- 3) A valid network code where the coding vector assigned to a child belongs to a subspace spanned by a proper subset of the vectors assigned to its parents does not exist.
- 4) Each coding Subtree has at most h parents.
- 5) If a coding Subtree has p parents, then there exist p vertex-disjoint paths from the source nodes to the Subtree.

These properties are very interesting because they impose strong limits on the choice of coding scheme. Hence, they reduce the number of candidate coding schemes that we consider and try to evaluate.

Chapter 3 Achieving UEP Thanks to RLNC

We first clarify what UEP means in the context of RLNC and see how we can achieve it.

3.1 General Approach

The work done for this thesis was part of a common effort among the team composed of Pr. John K. Zao, Pr. Chung-Hsuan Wang, Dr. Yao Chien, Kuo-Kuang and me. This section briefly presents their work which is necessary to explain clearly the issue tackled here.

We divide the problem in two. First we consider the issue of finding optimal repartition of symbols to the physical source nodes (dispatching) and then we consider the issue of transporting and mixing those symbols over the network (transmission).

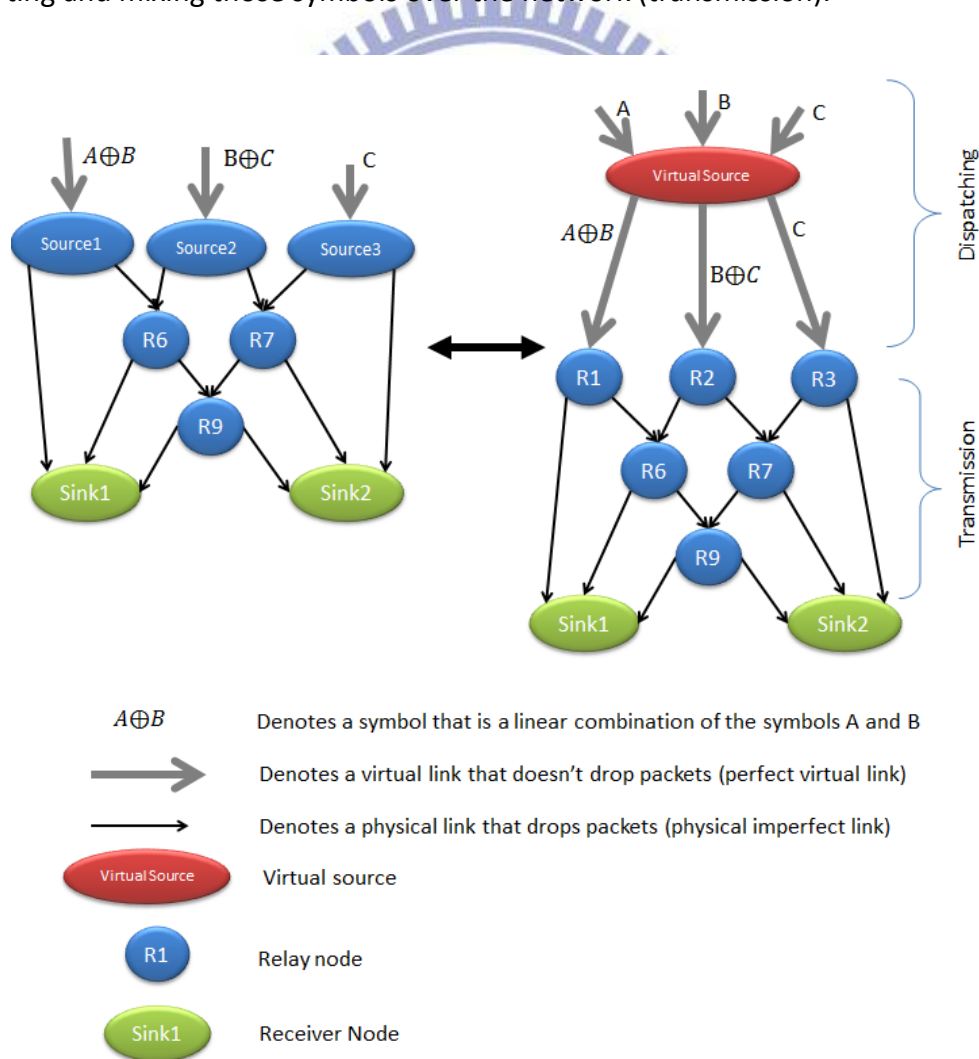


Figure 3.1: Dispatching and Transmission Processes

In Figure 3.1, three sources want to multicast one symbol each to the two sinks. The symbols are A, B and C. These sources are connected to 3 relay nodes as showed on the topology graph on the left. We model the problem as it is depicted in the diagram on the right. The source nodes become relays for the symbols that they receive from a virtual source. The coding rules at that virtual source dictate the initial symbols that will be injected in the network. Since this work is practical, we focus on the transmission problem while some other researchers are now investigating the dispatching one.

For the following section we will consider the dispatch matrix to be the identity matrix. In other words, each source will propagate one original symbol only. This choice allows us to focus on the second problem.

3.2 Unequal Erasure Protection (UEP)

We model the entire problem as a **network topology** made of nodes and imperfect links. The nodes are either sources nodes, relay nodes or receiver nodes (sinks). Source nodes inject in the network a set of specific symbols, or **codewords**. The choice of these codewords is not unique and affects greatly the UEP properties. Then, at a given time, we can identify a **failure pattern**. This failure pattern is merely the set of links in the network that are facing a failure at a given time. All these parameters are basic components of every network environment. On top of that, we have to consider the parameters of Network Coding. These are the **sets of local encoding vectors** in each source and relay nodes.

These parameters are necessary to fully grasp what UEP means. Although they all greatly influence the set of symbols that will be decoded at each sink we can only control the codeword assignment and the local encoding vectors. The others are a network constraint that we have to overcome.

To retrieve prioritized information, we have to produce a coding scheme that will guarantee the crucial information to be multicast while some other users will have supplementary data due to their relative high available resources.

In order to do so, we can then identify at least three different types of UEP.

- **Between symbols:** at a given sink, some symbols will be more protected than the others. Some may even not be accessible from the start (before any failure occurs).
- **Between different sinks:** different receiver nodes can have very different protection levels for the same symbols. In that case, we need to find a tradeoff among all receivers to achieve the maximum overall quality through the network. This issue is particularly well explained in [10].
- **According to different failure patterns:** Due to the topology, all the links do not have the same importance for each receiver and/or codeword. A link that is directly connected to a sink S_i will have a major impact on the performance of that node. But little on the performance at other sinks. On the contrary, if a backbone link fails, most sinks will suffer from it. The distance from each link to every sink is an intuitive parameter of the importance of each link in the network.

3.3 Coding Rules

We introduce the concept of coding rule. We can see this mechanism as a relationship between multiple input (the combining packets), and only one output (the result of the combination). In that case, the problem of designing a routing scheme for NC can be seen as a generalization of routing. Indeed, current routing matches one incoming packet to one outgoing packet (one to one). In the case of UEP-RLNC, we need to make multiple incoming packets combine into only one outgoing one (many to one). Here, routing packets and combining packets are actually the same operations. Thus, combining or routing rules refer to the same operation. While traditional routing only focuses on taking the least costly (often shortest) path, our routing schemes can achieve more by:

- Making the most of the bandwidth available in the network (NC feature)
- Prioritizing packets (for UEP)

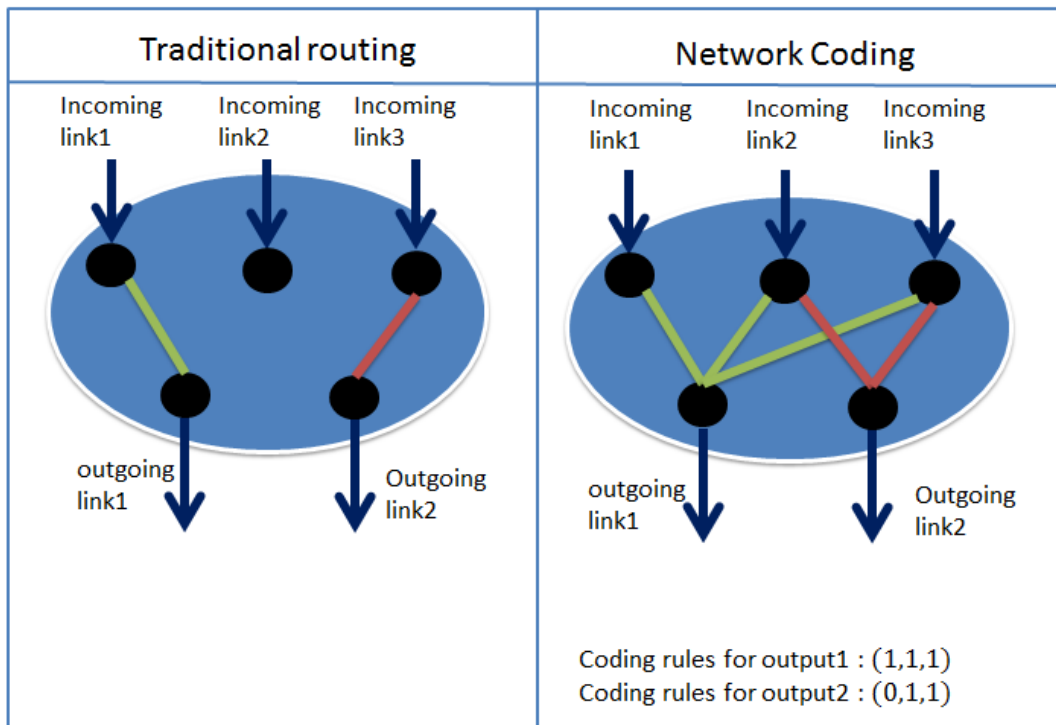


Figure 3.2: Traditional Routing vs NC Transmission at Local Scale

These coding rules define what shall be the input of the combination for each packet transmitted over the outgoing links. Each rule is specific to a unique outgoing link. Thus, we define a set of coding rules at each node. Note that this technique is equivalent to negate existing physical links, or to sacrifice some packets. Indeed, some packets are not propagated further down to the sinks.

The way to achieve this through network coding is for the node to voluntarily turn some of its local coefficients to zero. Consequently, that node makes sure that the most critical piece of information won't be polluted by less important ones in the later part of the network.

Note that unlike in [6], this approach requires a centralized knowledge of the network. This knowledge grants the necessary authority to efficiently set these coefficients to zero. Also note that the procedure of defining what coefficient should be nullified is costly, thus we only consider the case of static network codes. In other words, the coefficients that are non-zero will follow the RLNC rules. They will be randomly chosen at each packet combination. But the nullified coefficients won't change over time.

Methods to generate coding schemes are extremely hard to find. Indeed, finding one combines the difficulties of finding optimal multicast trees (traditional routing) with the complexity of assessing each coding scheme. Consequently, we currently do not know how to generate optimal UEP coding schemes. Nonetheless, we showed that given a sufficiently large finite field, the solvability problem of a coding scheme can be reduced to a graphic problem. Thus, the solutions can be found using well known algorithms.

3.4 Assessing UEP Capabilities of Network Codes

3.4.1 Separation Vector of Network Codes

Definition of Separation Vector with respect to one sink S_i : we note $SV^{(S_i)}$ the Separation Vector of a sink node S_i with respect to the ω source symbols $(X_j)_{1 \leq j \leq \omega}$.

$$SV^{(S_i)} = (SV_1^{(S_i)}, SV_2^{(S_i)}, \dots, SV_\omega^{(S_i)})$$

where $SV_j^{(S_i)}$ is the highest number of link failures that may occur on the network so that the sink S_i can still retrieve the symbol X_j .

In our efforts to assess the UEP capabilities of a Network Code, we use separation vector as a way to quantify UEP. The separation vector guarantees a certain level of performance by always considering the worst case scenario for each erasure it takes into account.

It also perfectly conforms to the fact that, in practice, it is unfeasible to change the coding scheme every time there is a link failure. Thus, we limit our study to static network Codes. And we will use the separation vector to assess the protection of each of the symbols for a given coding scheme.

Working out this separation vector requires a lot of processing power since it involves a combinatorial search but once again, it can be turned to a pure graphic problem where famous algorithms can be used.

3.4.2 Extended Theorem

We propose hereafter an extension of the structural properties in [7].

Proposition:

For a Minimal Subtree Graph that has h source nodes and N receiver nodes, the following holds:

1. Each coding Subtree T in Ω has at most $\alpha = \min(h, N)$ parents.
2. Each coding Subtree T in Ω has at most N children.

Proof:

Let $\Omega = (V, E)$ be a minimal Subtree graph with h source nodes and N receiver nodes. Consider a coding subtree T of Ω , it is shown in [3] that T has at most h parents.

1. Let's prove by contradiction that the number of parents of T is at most N when $N < h$. By definition, Ω can be seen as the union of N sets of paths, f_1, \dots, f_N , where each set consists in h vertex-disjoint paths from the source nodes to a receiver node R_i , where $1 \leq i \leq N$. Suppose that a given T has more than N parents. From **Theorem 3.5 (5)** in [3], there exist at least $N + 1$ vertex-disjoint paths from the source nodes to T . Therefore, at least two of these paths belong to a same set of paths f_j , where $1 \leq j \leq N$. And these two paths share T as a common vertex. That contradicts the definition of Ω . We conclude that T has also at most N parents.

Q.E.D.

2. Let's prove by contradiction that T has at most N children. By definition, Ω can be seen as the union of N set of paths, f_1, \dots, f_N , where each set consists in h vertex-disjoint paths from the source nodes to a receiver node R_i , where $1 \leq i \leq N$. Suppose that T is a coding subtree in Ω that has more than N children. From [3], there exist at least $N + 1$ vertex-disjoint paths from T to the receiver nodes. Therefore at least two of these children belong to a same set of paths f_j , for some $1 \leq j \leq N$. And these two paths share T as a common vertex. That contradicts the definition of Ω . We conclude that T has at most N children.

Q.E.D.

This proposition reduces dramatically the number of Minimal Subtree Graphs that one has to assess while investigating the different minimal coding schemes. Moreover, in the general case, it sets a bound on the size of the local encoding vector.

The above proposition can help us to enumerate Minimal Subtree Graphs that one has to assess while investigating the different minimal coding schemes. Moreover, in the general case, it sets a bound on the size of the local encoding vector.

3.4.3 Illustrative Example: Complete Analysis of Case 3-Sources-2 Receivers

A great number of topologies can be represented by a single Subtree Graph. This innovative approach differs from previous work such as in [11] and [12] since we try to obtain an exhaustive list of Minimal Coding graphs for a given number of sources and receivers. By doing so, we look for equivalent classes of coding schemes with respect to UEP properties regardless of the actual network topology.




We focused our efforts on the simplest example to illustrate UEP multicast. In order to have multicast, we need at least 2 receivers. And in order to have reasonable scalable data, we need to have at least 3 different symbols. We assume that each of these symbols is the result of a dispatching mechanism.

According to the limits on the number of incoming and outgoing edges for each Subtree as well as the structural properties stated in [7], we could narrow down the number of possible Subtree topologies to 4 equivalent classes.

We use these 4 classes as starting points. We iteratively trigger disappearance of links on the Subtree graph which is equivalent to the nullification of local coefficients. This work reflects the possible coding schemes that one may choose by reduction of Minimal Subtree graphs.

	C=1	C=2	C=2	C=3
K=0	 $SV_{S1} = SV_{S2} = (1,1,1)$	 $SV_{S1} = SV_{S2} = (1,1,1)$	 $SV_{S1} = SV_{S2} = (1,1,1)$	 $SV_{S1} = SV_{S2} = (1,1,1)$
K=1	 $SV_{S1} = (1,1,1)$ $SV_{S2} = (2,0,1)$	 $SV_{S1} = (1,1,1)$ $SV_{S2} = (0,2,2)$	 $SV_{S1} = (1,1,1)$ $SV_{S2} = (0,2,2)$	 $SV_{S1} = (2,0,0)$ $SV_{S2} = (1,1,1)$
		 $SV_{S1} = (2,0,0)$ $SV_{S2} = (1,1,1)$	 $SV_{S1} = (2,0,1)$ $SV_{S2} = (1,1,1)$	 $SV_{S1} = (1,1,1)$ $SV_{S2} = (0,1,1)$
			 $SV_{S1} = (1,1,1)$ $SV_{S2} = (0,1,2)$	 $SV_{S1} = (1,1,1)$ $SV_{S2} = (0,2,1)$
			 $SV_{S1} = (1,1,0)$ $SV_{S2} = (1,1,1)$	
K=2		 $SV_{S1} = (2,1,0)$ $SV_{S2} = (1,1,1)$	 $SV_{S1} = (1,1,1)$ $SV_{S2} = (0,0,3)$	 $SV_{S1} = (2,2,0)$ $SV_{S2} = (1,1,1)$
		 $SV_{S1} = (1,1,1)$ $SV_{S2} = (0,1,2)$	 $SV_{S1} = (2,0,0)$ $SV_{S2} = (0,1,2)$	 $SV_{S1} = (2,1,0)$ $SV_{S2} = (1,1,1)$
		 $SV_{S1} = (2,2,0)$ $SV_{S2} = (0,2,1)$	 $SV_{S1} = (2,1,0)$ $SV_{S2} = (1,1,1)$	 $SV_{S1} = (2,0,1)$ $SV_{S2} = (1,0,2)$

Table 3.1: Exhaustive List of Minimal Subtree Graph for 3-Sources-2-Receiver

- Rectangles () represent Subtrees. The three at the top are the Sources Subtrees.
- Triangles () represent physical links to Sink 1.
- Circles () represent physical links to Sink 2.
- K is the number of coefficients that we sequentially set to zero.
- C is the number of Coding Subtrees in the initial Minimal Subtree Graph.
- SV_{S1} and SV_{S2} are the separation vectors associated to Sink 1 and Sink 2 respectively.

We manually generated and assessed all possible coding schemes for a minimal configuration of 3 sources and 2 receivers. Each of them is represented in Table 3.1. Table 3.1 also shows the derived coding schemes after setting K coefficients to zero. Note that the edges on this graph can be seen as the coefficients of the Local Encoding Vector. They are directed from top to bottom.

After setting a coefficient to zero (i.e. deleting an edge on the Minimal Subtree Graph), we merge the Subtrees in order to respect the definition of a Subtree Graph. We observe an evolution from Minimal Subtree Graphs where the multicast capacity is achieved to configurations where a trade-off between performance and protection is found. This illustrates the use of the separation vector to efficiently evaluate these two parameters at once.

Chapter 4 Implementation

This thesis is first and foremost a practical one that aims at investigating concrete and applied aspects of Network Coding. Therefore, the main piece of work consists in a prototype implementation of UEP-RLNC based on [4].

4.1 From Existing Code

We used an implementation of NC coding done for the Project FRANC [13] as a starting point. This open-source application was initially designed to implement a bulletin board (a chat on which every user communicates with every other node). We used it as a framework to implement some missing features (e.g. generations) as well as some more problem-specific functions (simulated packet loss, coding rules...). Table 4.1 summarizes the added functionalities.

Old features	New/modified features
Finite field operation	Packet handling depending on the roles of the nodes
Packet combination methods	Generations (tagging, sorting, updating, flushing)
Decoding methods	Internal structure(multiple input buffers)
	Enabled any topology
	Encoding scheme (coding rules)
	Multiple Communication threads
	Packet loss
	Enabled Inter machine communication
	Improved modularity

Table 4.1: Old vs New Features

The program consists in five packages containing 37 classes in total, illustrated in Figure 4.1.

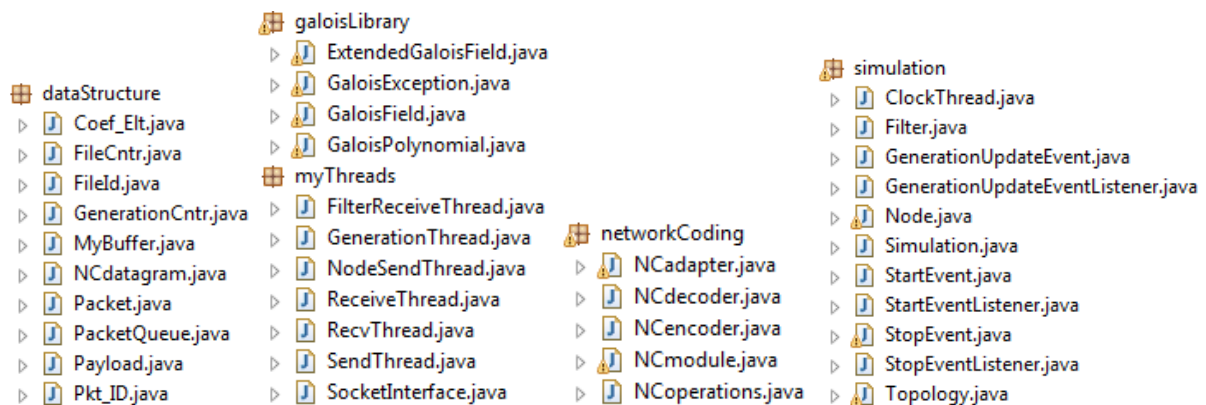


Figure 4.1: Classes Overall View

4.2 Galois Library

The class NCooperation from NetworkCoding package takes care of the packet wise addition, subtraction and multiplication required by the NCencoder and NCdecoder.

This class uses a library for finite field operation: GALOISFIELD

```
This Package is written by Jan Struyf
URL: http://ace.ulyssis.student.kuleuven.ac.be/~jjeans
EMail: jan.struyf@student.kuleuven.ac.be
Post: Jan Struyf, Hoogstraat 47, 3360 Bierbeek, BELGIUM
```

You can do anything with it, if you leave this comment field unmodified.

This library lets the user choose the size of the field and other parameters. As previously seen, the choice of the size of the finite field is of utmost importance when we rely on RLNC. In our implementation, we consider a field of size 2^8 . The main reason of this choice is to find a compromise between linear dependency and processing time.

According to the previous formula in Section 2.1.3.3, we can then simulate networks that around 20 nodes with an average decoding success probability of 0.92. This choice seems realistic in order to run simulations of rather small networks.

4.3 Data Structure Package

In order to combine packets together, the data partition has to be done with great care. Besides the usual partition into packets, we define what we will refer to as blocks. These are the addition of packets that were encoded together.

The size of the block is a key parameter that has to be finely tuned in order to find a tradeoff between large and small blocks:

- Having large blocks leads to enhanced protection but also higher processing cost during the coding and decoding steps.
- Having small blocks leads to the opposite statement.

The original file is fragmented into blocks and packets. These packets can contain a lot of symbols. The numbers of packets per block and the number of blocks are parameters of the system.

4.3.1 Coefficient Element

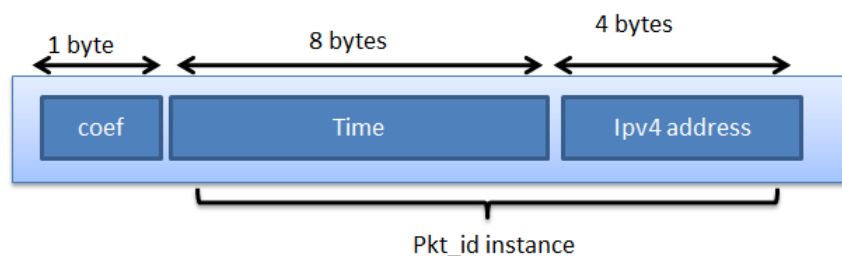


Figure 4.2: Coefficient Element

In Figure 4.2, a coefficient element is composed of a coefficient and a packet identifier. The size of the coefficient is equal to the size of the Finite Field. The packet identifier is necessary to determine whether two coefficients are the same or not. This is critical at the sinks when we try to decode the information with a set of linearly independent coefficient elements.

4.3.2 Packet

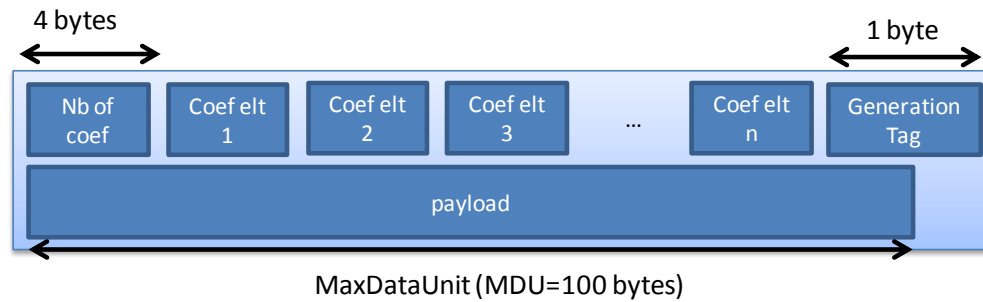


Figure 4.3: Packet

First, we notice in Figure 4.3 that the number of coefficients in a packet may vary. The number of packets in a buffer when a transmission opportunity occurs changes over time. Thus, different amount of packets can be involved in the combination. The maximum number of coefficients involved is a multiple of the size of the generations.

Following the packet tagging mechanism, we store each coefficient element in the packet. These coefficients will be encoded at each hop. We also need to add a generation tag to survive asynchronous transmissions. The packet is tagged at the source and this field will remain the same until the packet reaches its destinations.

The payload consists of the actual data transmitted. Its size (Max Data Unit) can be changed as long as an entire packet can fit in a UDP packet. In our case, we are not interested in transporting a lot of data so the MDU is rather small. And we measure the number of packets transmitted rather than the transferred amount of data.

4.3.3 Buffer

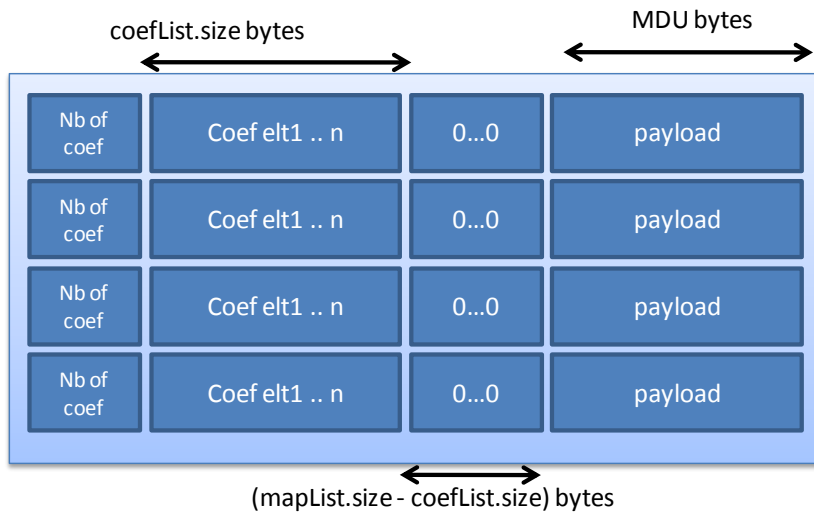


Figure 4.4: Buffer

The buffer, Figure 4.4, is merely a linked list of packets that is sorted according to the generation tags of the packets inside it. A vital feature is to keep track of the received coefficient elements at the sinks. This is performed through the update of a *mapList*. The *mapList* makes sure that we do not mix the coefficients after some step of the Gaussian elimination were performed.

4.4 MyThreads Package

This package contains very simple classes that all inherit from the standard Thread class. They allow asynchronous transmission with UDP datagram sockets. Table 4.2 summarizes each of these classes.

	Location	Number	Goal
Receive Thread	in each node	one per incoming link	trigger packet reception
Send Thread	in each node	one per outgoing link	trigger packet combination and transmission
Clock Thread	in the Topology class	unique	send clock ticks to every node to let them synchronize their generation update

Table 4.2: Summary of Threads Used in Application

To support generations, we need to keep track of the current generation at each node. That is also true for the source. Thus, we have a unique *ClockThread* inside the network. This thread will send ticks to each node in the network through a *GenerationUpdateEvent*. This

event will update the value of the current generation based on the generation tag of the majority of packets stored. It will also trigger the flushing of the input buffer to get rid of packets that belong to old generations. This policy is important in terms of performance and robustness. So we discard some of the packets in the input buffers and increase the current generation value. The new value is the value of the majority of packets in the input buffers.

4.5 Network Coding Package

The most important class of this package is *NCmodule*. It performs the packet handling. Arriving and outgoing datagrams go through the *NCadapter* which is converting datagram packets to *NCdatagram* and vice versa.

4.5.1 Packet Handling

In practice, network coding has to face the problem of asynchronous transmissions over the internet. But buffering techniques can solve this problem.

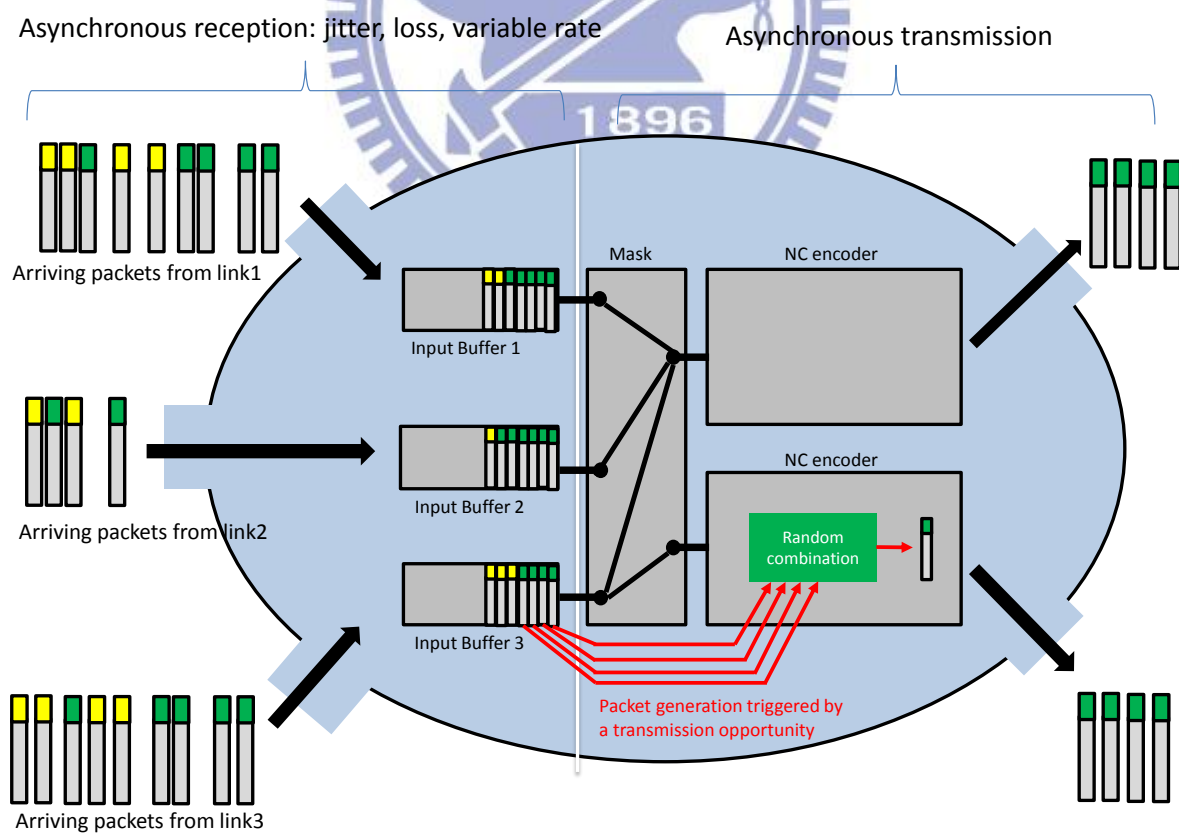


Figure 4.5: Buffering at a Node

First, we need to introduce the concept of generations presented in [4]. These generations consist in a certain numbers of packets. These packets need to be decoded together to retrieve the original block of source symbols. Thus, we tag each packet with a generation number (represented by a color on the above Figure 4.5).

Unlike in [6], we cannot consider a single buffer for all outgoing edges of a given coding point. In our case, coding rules apply to a specific outgoing link. So we now have to differentiate each outgoing link and assign one buffer to it. Therefore, each incoming link has a dedicated input buffer.

Inside each of these buffers, we follow the procedure described in *Practical Network Coding* (sort packets according to their generation number, enforce a flushing policy, etc.).

4.5.2 Encoding

Each outgoing link has a dedicated encoder. When the encoders are created, they follow coding rules. These rules are represented as a binary mask. In Figure 4.5, if the first outgoing link was to combine packets from all input buffers while the second one only cares about the last input buffer. In this case, the corresponding masks would be: "111" and "001".

Whenever a transmission opportunity occurs, the corresponding *NCencoder* generates non zero random coefficients (a local encoding vector) to encode the packets of the current generation inside the authorized input buffers. The operation of combining them according to these random values creates an outgoing packet.

4.5.3 Decoding

Decoding is only performed at the sinks, inside the *NCmodule* class. Instead of input buffers, sinks have a decoding buffer and a decoded buffer.

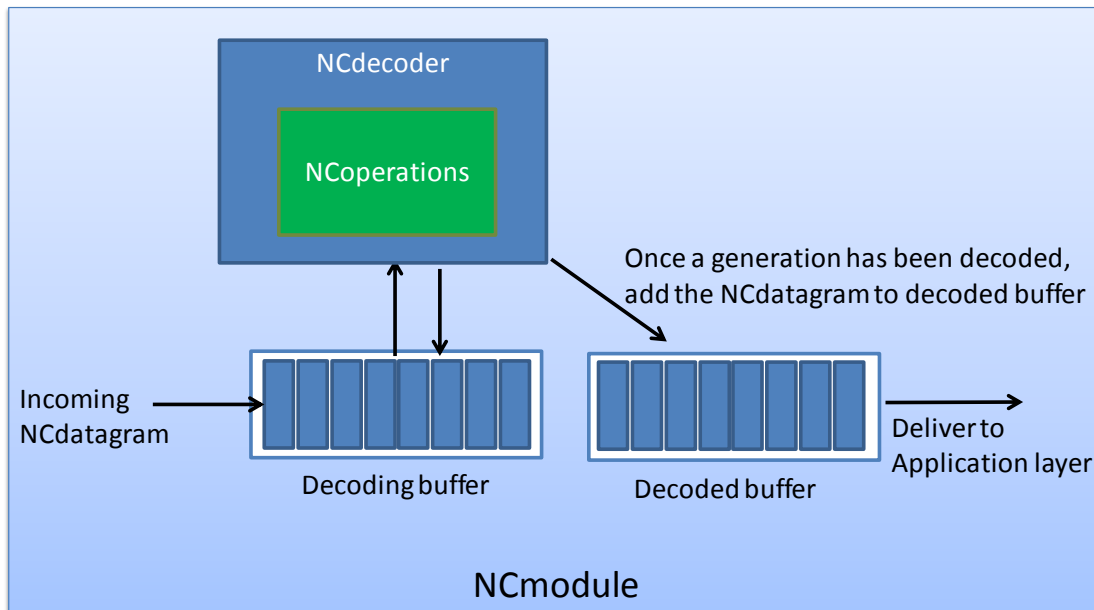


Figure 4.6: Decoding Mechanism

If we see the problem as a linear system to be resolved as follows:

$$\begin{cases} a_{1,1}x_1 + a_{1,2}x_2 + \dots + a_{1,n}x_n = y_1 \\ a_{2,1}x_1 + a_{2,2}x_2 + \dots + a_{2,n}x_n = y_2 \\ \vdots \\ a_{n,1}x_1 + a_{n,2}x_2 + \dots + a_{n,n}x_n = y_n \end{cases}$$

Where n is the number of unknowns which corresponds to the size of a block.

Since each packet carries one line of the final equation system, one step of the Gaussian elimination can be performed each time a packet is received (earliest decoding). Of course, the final system can only be solved when the node received to adequate number of packets so all packets within the same block will be decoded at the same time.

4.6 Simulation Package

4.6.1 Nodes

Each node has a role. It can be a “Source”, a “Relay” or a “Sink”. Source generates data and then sends them out to relay nodes. The relay nodes merely transmit the data without accessing it and the Sink only consumes data. Here is a more specific description of each of these nodes.

4.6.1.1 Source Node

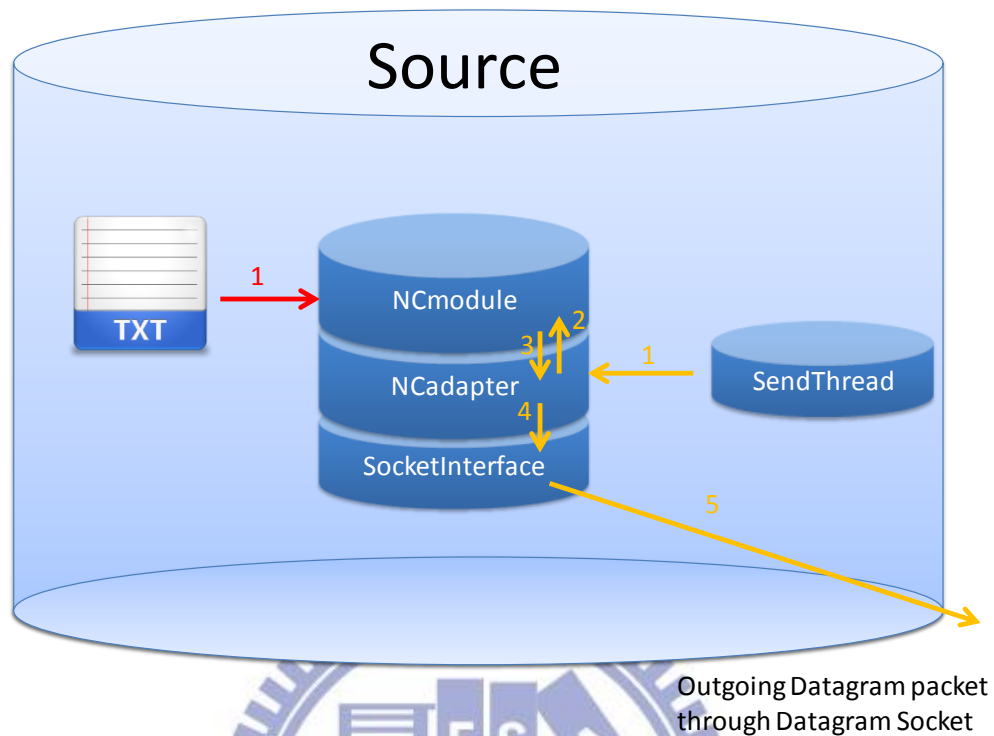


Figure 4.7: Source Inner Structure

Data input (in red): Generate a dummy text file.

1. Pass it the input buffer, one file per buffer.

Sending data (in orange):

1. At a given rate (parameter), the *sendThread* will trigger the generation of a packet to the adapter.
2. *NCadapter* forwards the request to the *NCmodule*.
3. The *NCmodule* generates the new packet by combining the content of its input buffer and passes it the *NCadapter*.
4. The *NCadapter* transforms the *NCdatagram* to a generic datagram packet, and sets the destination field before passing it to the *socketInterface*.
5. The *socketInterface* sends the datagram packet through the datagram socket.

4.6.1.2 Relay Node

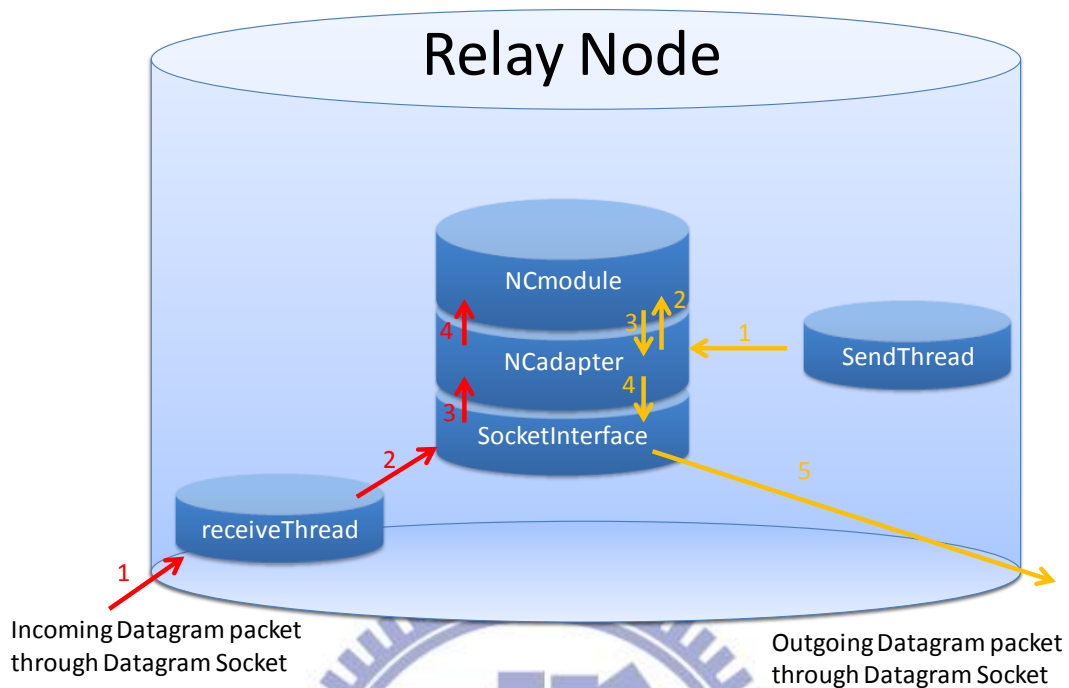


Figure 4.8: Relay Inner Structure

The relay nodes receives the data from a receive thread.

Receive data (in red):

1. A datagram packet arrives through datagram socket.
2. It goes through the *socketInterface*.
3. It is passed to the *NCadapter* that transforms the packet into a *NCdatagram*.
4. The new *NCdatagram* is passed to *NCmodule* and stored in the input buffer that is dedicated to the incoming link from which the packet just arrived.

4.6.1.3 Sink

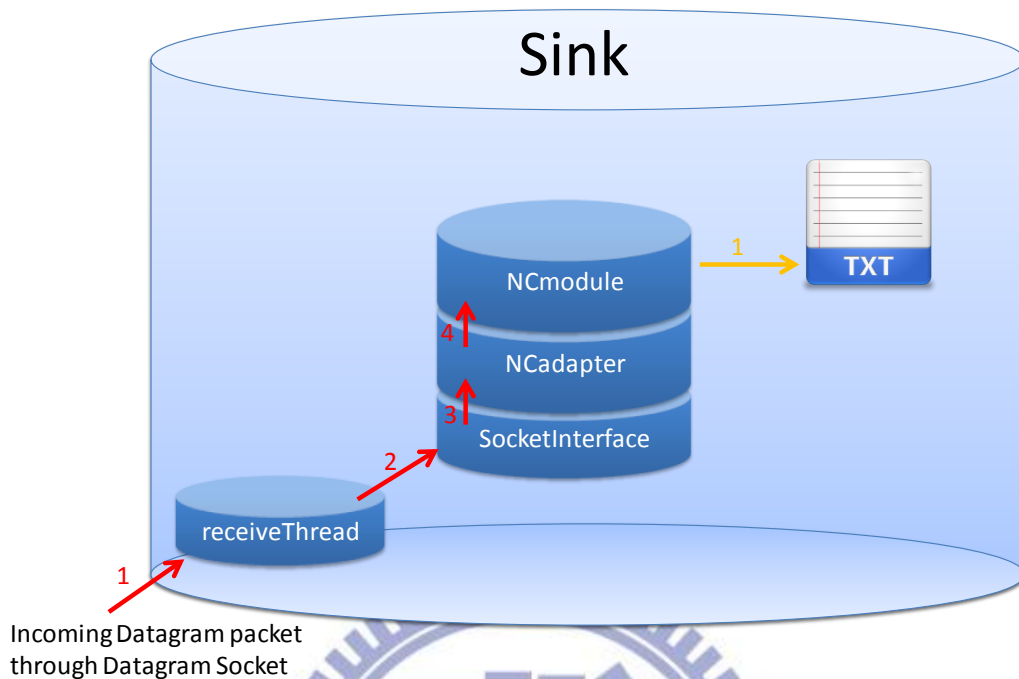


Figure 4.9: Sink Inner Structure

Data retrieval (in orange):

1. The sink receives data, and stores it in its decoding buffer. Whenever a generation is successfully retrieved it is written in a file and stored in decoded buffer.

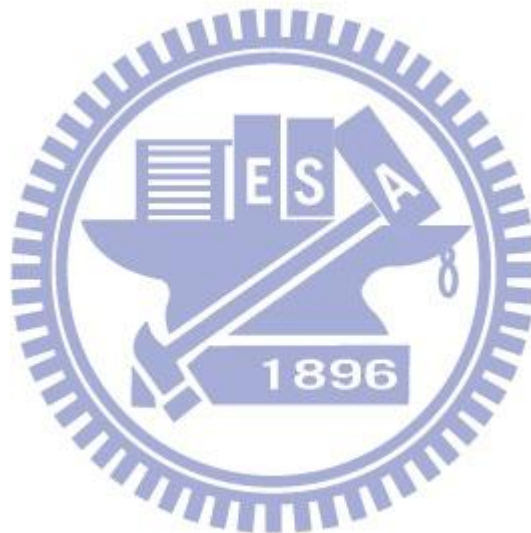
4.6.2 Filters

Unlike most previous work on the subject, we assume each edge in the topology graph has a failure probability. Whenever a failure happens, the symbol that was being transmitted on the link disappears (the corresponding output of the link is nothing). Therefore, it is impossible for a node to detect whether a failure occurred or not. In that sense the physical links are actually intermittently disappearing. In order to simulate these losses on a single machine, we use simple objects called filters.

Since all nodes use datagram sockets to communicate, these filters intercept packets between two datagram sockets and drop packets according to the failure probability of the simulated link. The dropped packets are discarded while the others are forwarded to the node.

These objects are outside the nodes so they can easily be removed if future users want to test the application on different physical nodes. Furthermore, we can easily introduce delay in these filters to make the simulation more realistic.

Filters use Receive threads that are similar to the one inside the nodes.



Chapter 5 Experiments

We conducted different experiments to exhibit the different UEP capabilities of coding schemes found in Chapter 3. To limit the number of devices required for the simulation, we used the basic simulation platform that comes along the application. Therefore, all the experiments were run on a single physical node. The three symbols from the coding schemes are changed to three different files.

5.1 Simulated Topology and Parameters

We consider coding schemes from the 3-sources-2-receivers example. These coding schemes can be implemented using coding rules. In other words, if an arbitrary network topology contains three sources and two receivers, we can always match the topology to one of the Subtree Graphs from Table 3.1. Figure 5.1 represents the network topology that we simulated with the parameters from table 5.1.

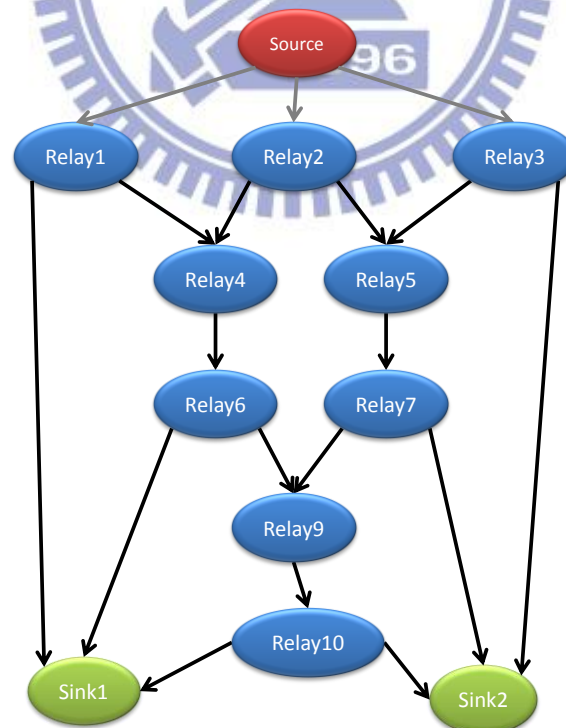


Figure 5.1: Simulated Topology

Parameter	Description	Value
HCL (Hard Coding Coefficient)	Number of packets per generation	10
MCLU	Maximum number of packets mixed together	40
MDU (Maximum Data Unit)	Number of bytes contained in the payload of NCdatagram	100
Rate	Rate of the links throughout the network	30 ms
Failure probability	Packet loss rate on imperfect links	varies
Generation refresh rate	Rate followed by the clock thread to trigger generation update	2500 ms
File length	Number of generations in file	30

Table 5.1: Simulation parameters

5.2 Simulation 1: Evaluation of Coding Schemes

We now consider the coding scheme presented in Table 5.2 which derives from the maximal configuration after we nullified two links by enforcing coding rules.

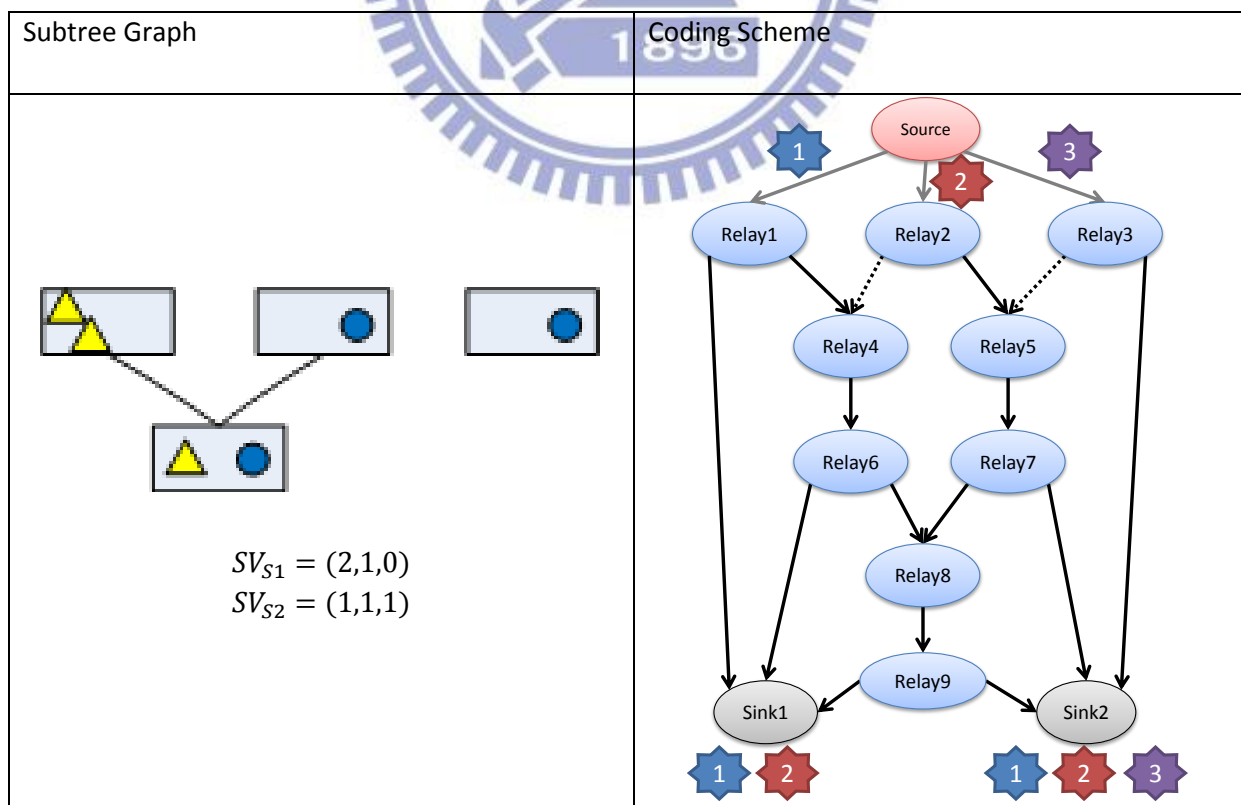
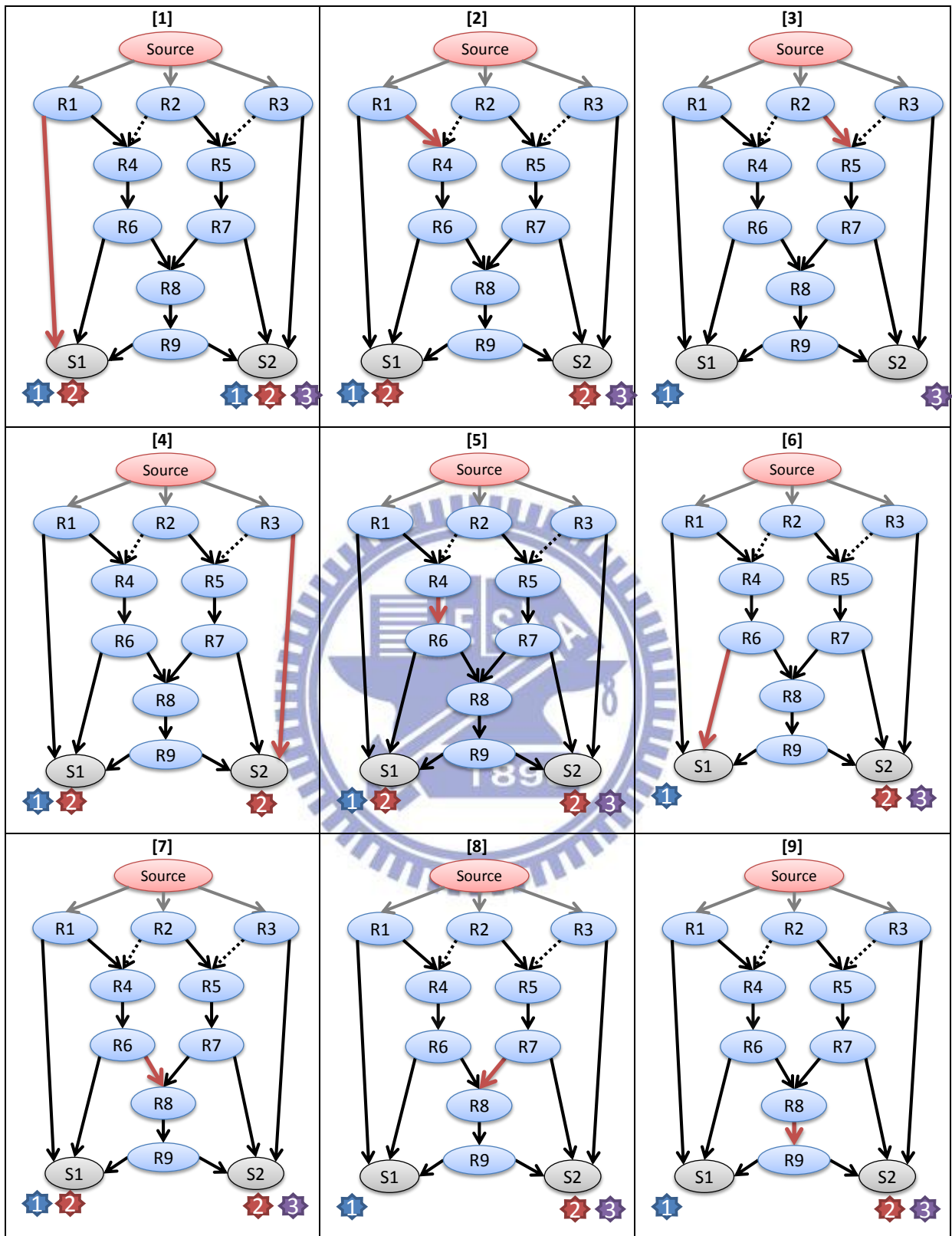


Table 5.2: Coding Scheme for Failure Pattern Evaluation



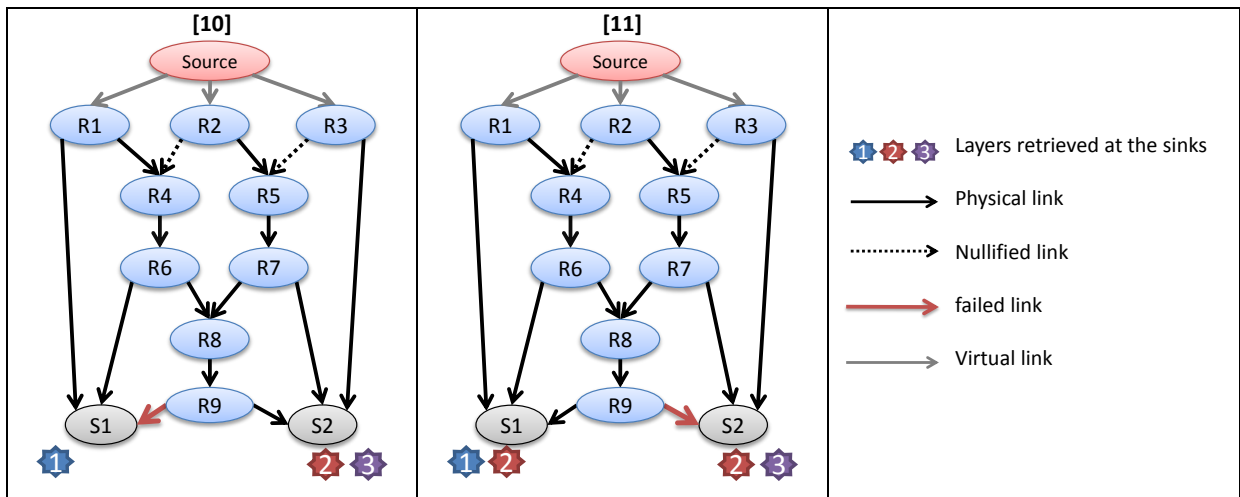


Table 5.3: Assessed Failure Patterns

In Table 5.3, we observe that for any layer received by S2, it is possible to find a link failure that causes the loss of that layer. On the contrary, layer 1 is always retrieved by S1. Note that for an arbitrary layer, we can always find two link failures that would cause the loss of that layer for both Sinks. Thus, this experiment demonstrates the validity of the separation vector values as well as the model that we use for the evaluation of the UEP capabilities for a variety of failure patterns.

Furthermore, we can sort the previous result in order to categorize links depending on their topological location:

Category	Consequence when link fails
Inter Subtree links	Migrate from one coding scheme to another
Intra Subtree bottleneck links	Damage both sinks by cutting one path to each sink
Intra Subtree links connected to a sink	Damage one sink only, but severely

Table 5.4: Categories of Links

5.3 Simulation 2: Hotspot

For the following simulations, we choose the coding scheme from table 5.2 and evaluate its average performance. Unlike the previous experiment where the network was lossless, we

now introduce erasures on it. We specifically want to simulate hotspot situations. In order to do so, we add a packet loss rate to a unique link. We choose to evaluate two hotspot situations corresponding to the diagrams number (9) and (10) from Table 5.3. Diagram (10) belongs to the bottleneck category while Diagram (9) is directly connected to Sink 1.

In the following graphs, we plot the retrieval rate of the layer i at the Sink j which is the ratio of decoded packets from layer i at the sink j over the number of source packets from layer i (simulation parameter).

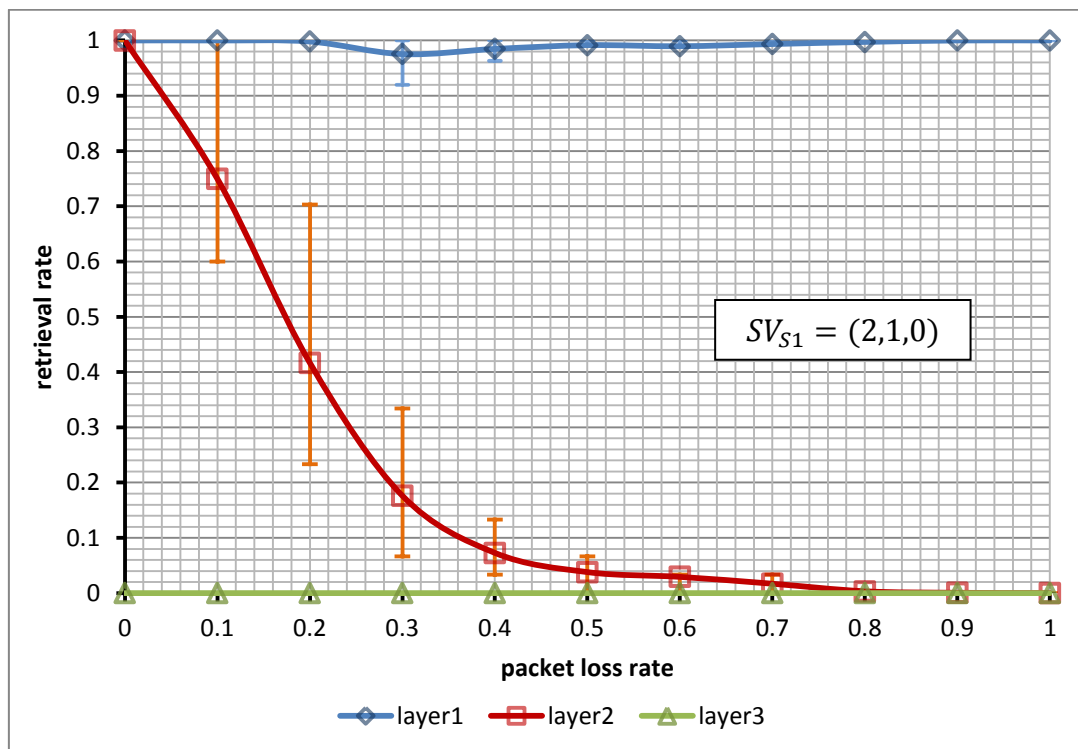


Figure 5.2: Sink1 Facing Bottleneck Hotspot (Diagram (1) from Table 5.3)

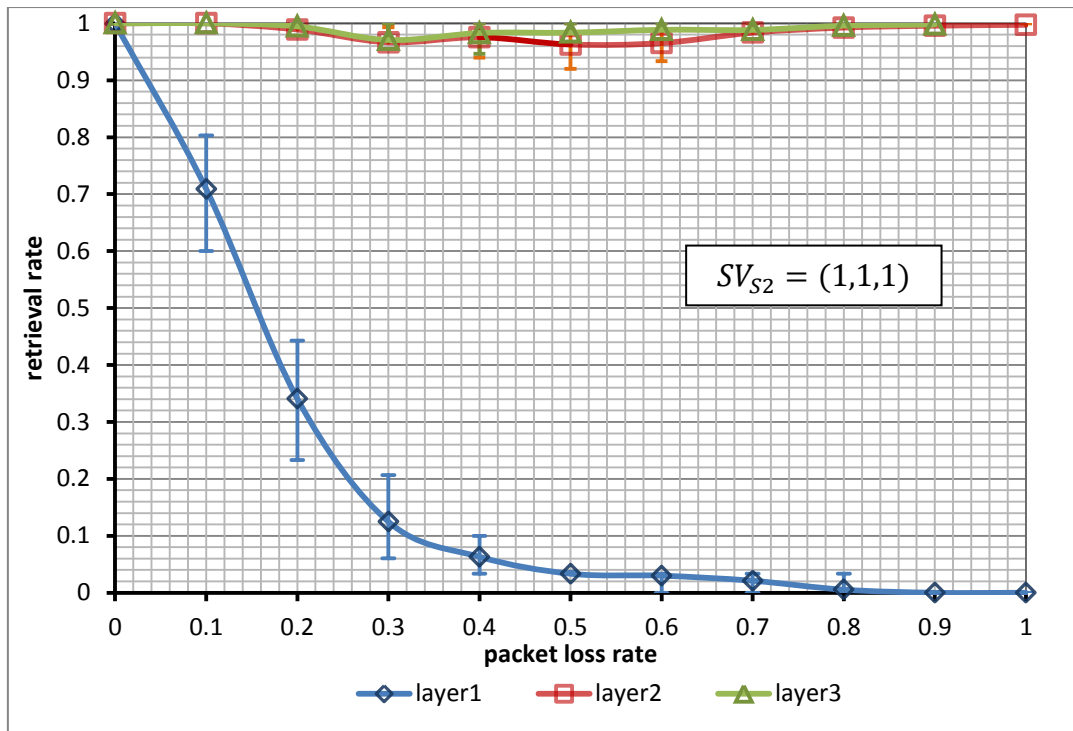


Figure 5.3: Sink2 Facing Bottleneck Hotspot (Diagram (1) from Table 5.3)

Figures 5.2 and 5.3 show that losses over a bottleneck damage both sinks. More accurately, Sink 1 and Sink 2 lose layer 3 and 1 respectively as the packet loss rate increases. The retrieval rates remain unchanged for all other layers.

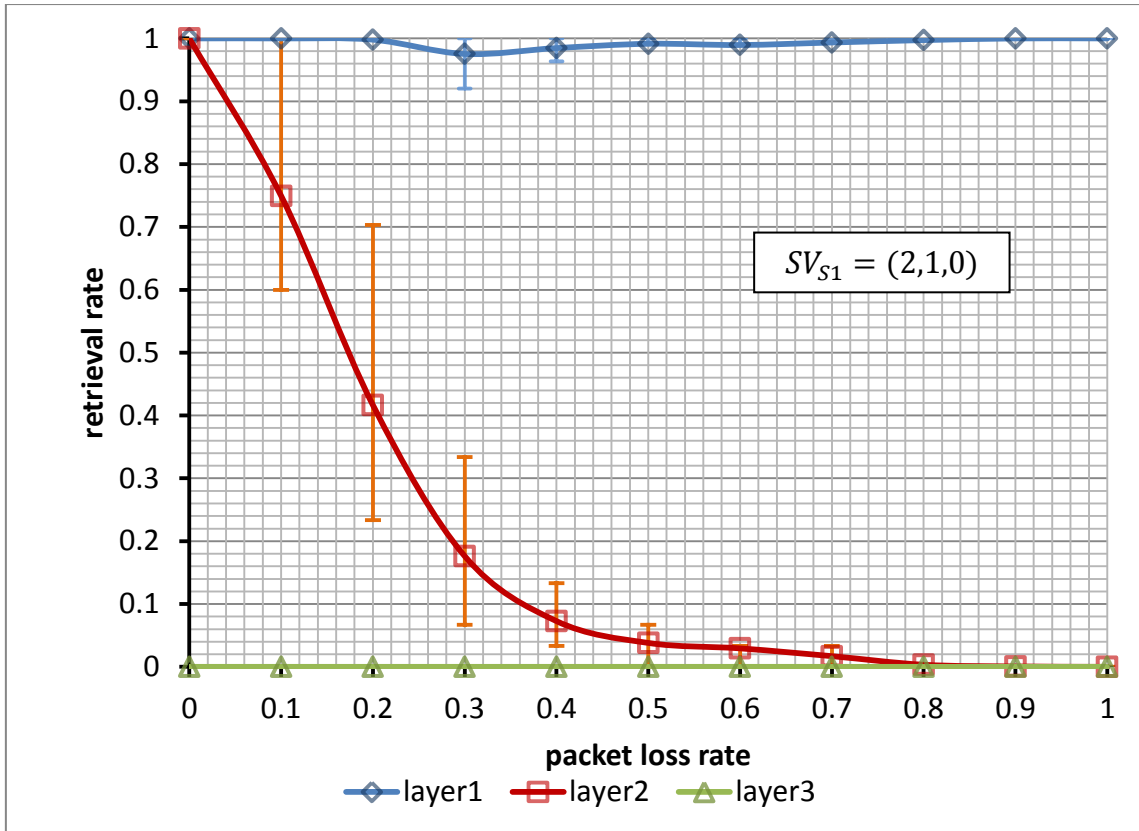


Figure 5.4: Sink1 Facing Hotspot on Direct Link (Diagram (2) from Table 5.3)

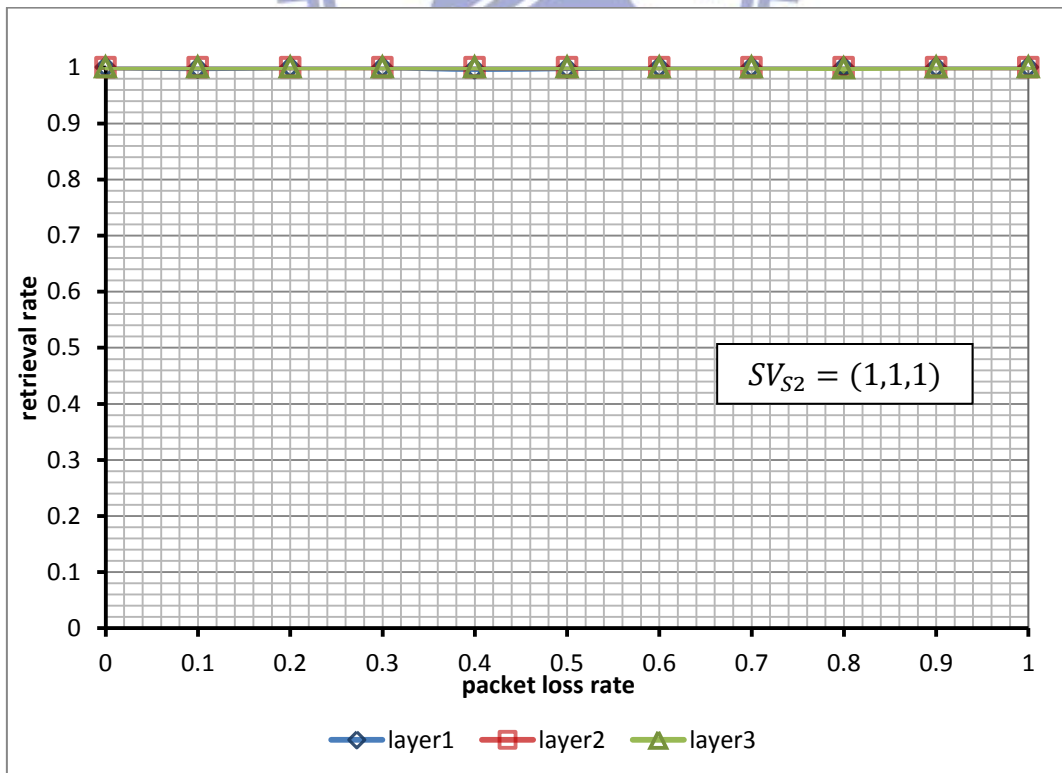


Figure 5.5: Sink1 Facing Hotspot on Direct Link (Diagram (2) from Table 5.3)

Figures 5.4 and 5.5 show that losses over a link directly connected to Sink 1 damages it heavily but has no consequence on the other sink. More accurately, Sink 1 loses layer 2 as the packet loss rate increases. The separation vector for layer 1 guarantees full retrieval of that layer at Sink1. Retrieval rates remain unchanged for Sink 2.

For the last simulation of this section, we choose to change the coding scheme to better illustrate the migration between coding schemes.

Figure 5.6 illustrates migration #1.

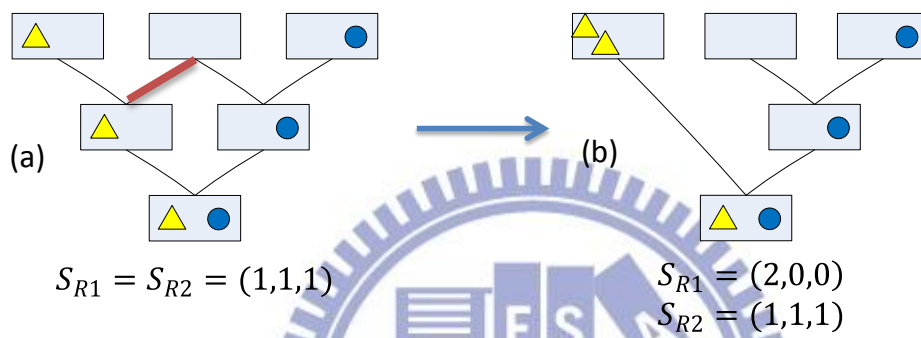


Figure 5.6: Migration #1 from Coding Scheme (a) to Coding Scheme (b)

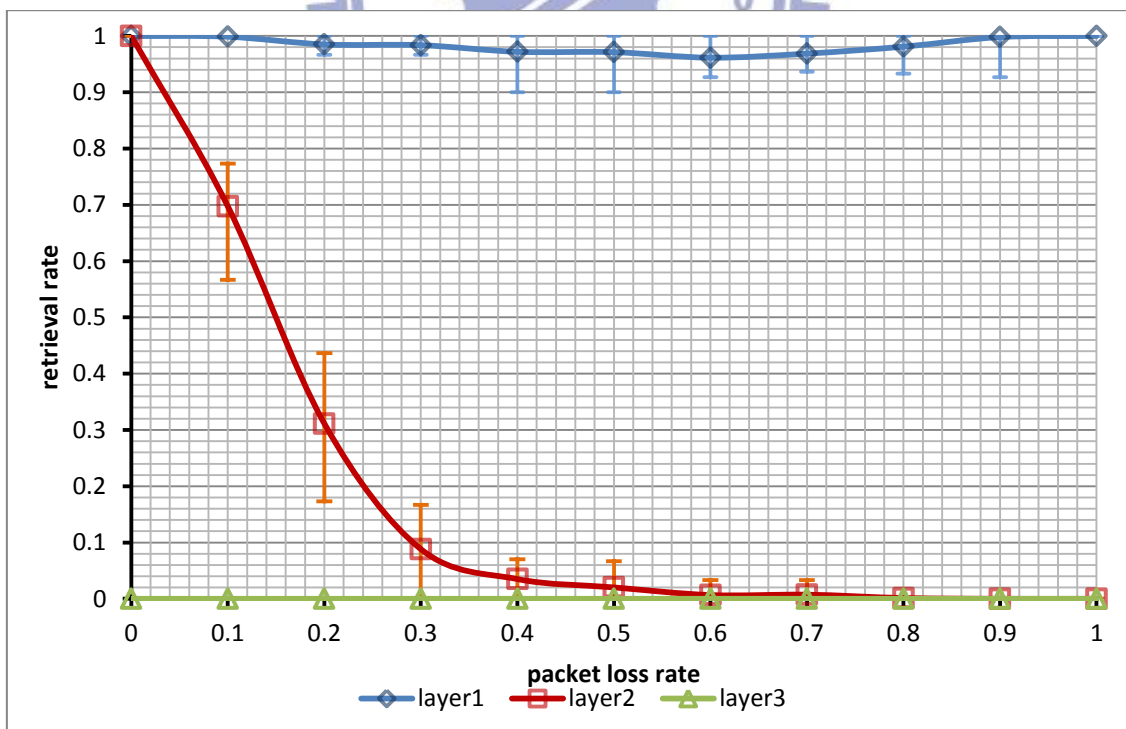


Figure 5.7: Sink1 migration #1

We note that the migration #1 causes Sink1 to lose layer2 while Sink1 is not affected (chart not present here).

Figure 5.8 illustrates migration #2.

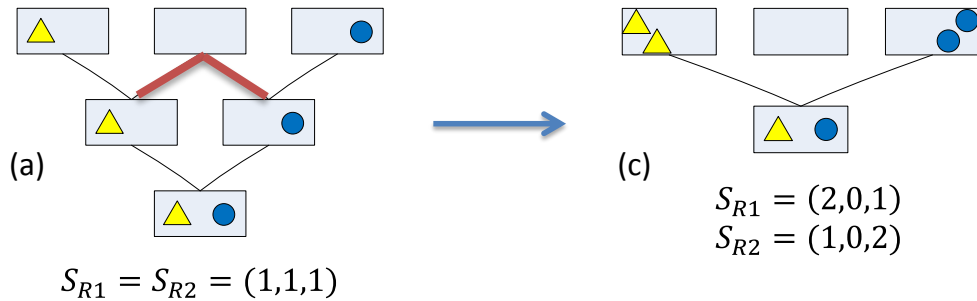


Figure 5.8: Migration #2 from Coding Scheme (a) to (c)

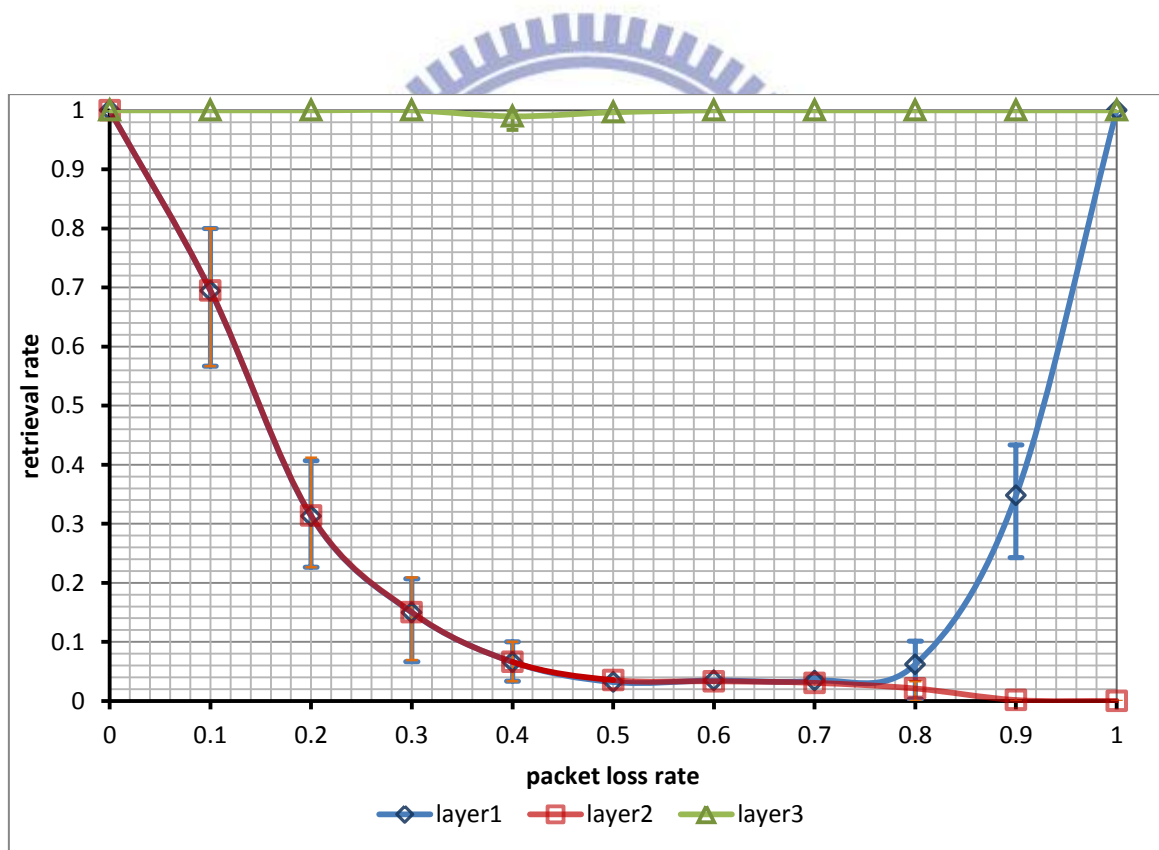


Figure 5.9: Sink 1 migration #2

Figure 20 shows the migration from a coding scheme where the separation vector is (1,1,1) to another one where the separation vector is (2,0,2). The curve corresponding to layer 2 follows the same pattern as the previous experiment. The packet retrieval rate of layer 3

shows a counter intuitive behavior. After an expected decreasing phase, the retrieval rate increases again to finally reach its initial level of 1.

The area corresponding to packet loss rates between 0.4 and 0.8 show extremely bad performances. This area corresponds to the maximum pollution generation. Since the transmission of packets from layers 2 half of the time, the remaining packets represent pollution for the retrieval of packets from layer 3. As the packet loss rate increases more, the number of polluting packets from layer 2 decreases, thus leading to an improvement of the retrieval of the layer 3.

5.4 Simulation 3: Background Erasures

This experiment aims at showing the UEP capabilities of the different layers with respect to background erasures. To simulate these erasures, we add a packet loss rate on every link of the topology. This is the second typical erasure pattern that we can find in reality.

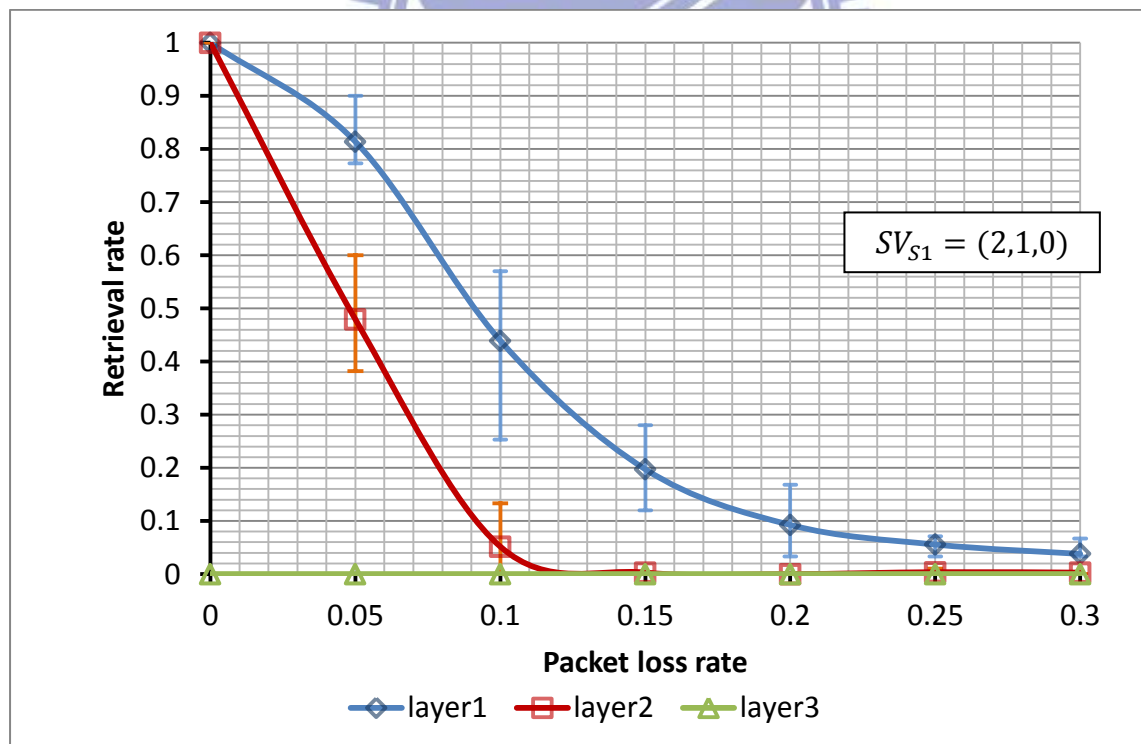


Figure 1010: retrieval rate at Sink1 (background losses)

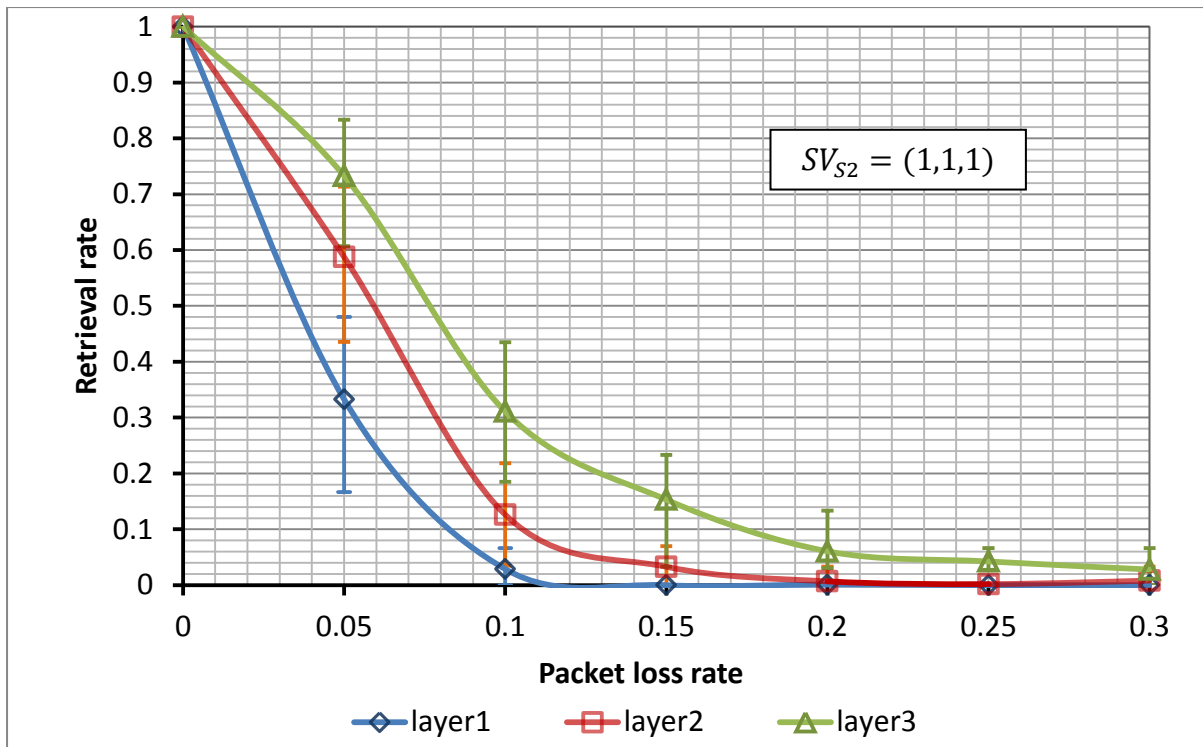


Figure 11.11: retrieval rate at Sink2 (background losses)

We observe that the behavior of the three layers on Figure 5.10 is different than the one on Figure 5.11. Indeed, the three curves in Figure 5.11 tend to have a closer behavior (error bars overlap). This phenomenon follows our prediction since Sink2 has the equal amount of protection on all three layers.

We want to compare the behavior of coding scheme presented in Table 5.2 with topology (a). Topology (a) is such that each source has a single path to each sink. To be fair, we introduce the right amount of links on each path. The number of links on each of these path is equal to the number of links on the shortest path from the source to the destination in the previously assessed topology.

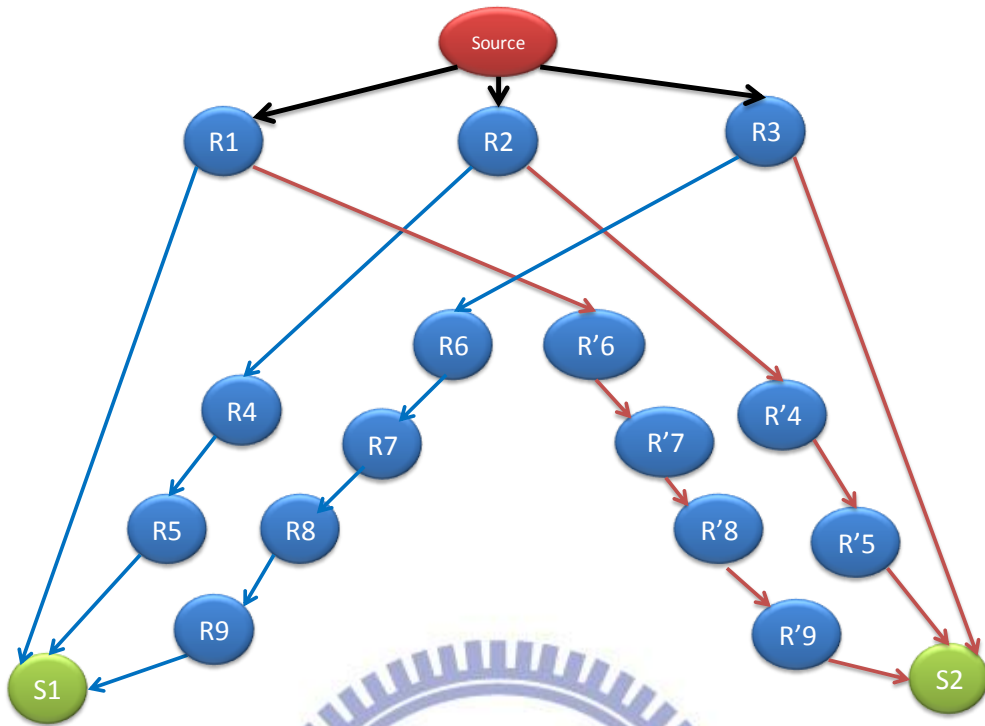


Figure 5.12: Topology (a)

At a given Sink j , for a received layer i , we define $R(i, j)$ as follows:

$$R(i, j) = \frac{r(i, j)}{r'(i, j)}$$

Where:

- $r(i, j)$ is the observed retrieval rate of layer i at Sink j .
- $r'(i, j)$ is the theoretical retrieval rate of layer i at Sink j for topology (a).

$$r'(i, j) = (1 - p)^k$$

- p is the packet loss rate on every link.
- k is the number of links in the path from Source i to Sink j .

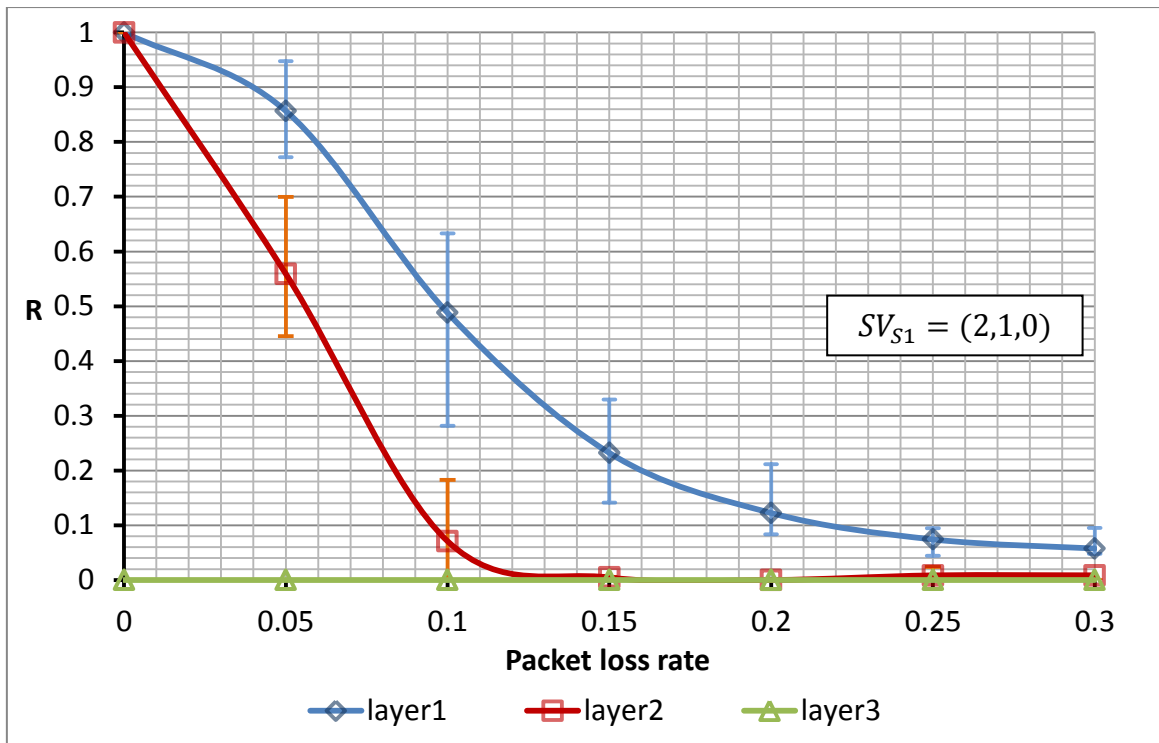


Figure 5.13: Sink 1 after background losses

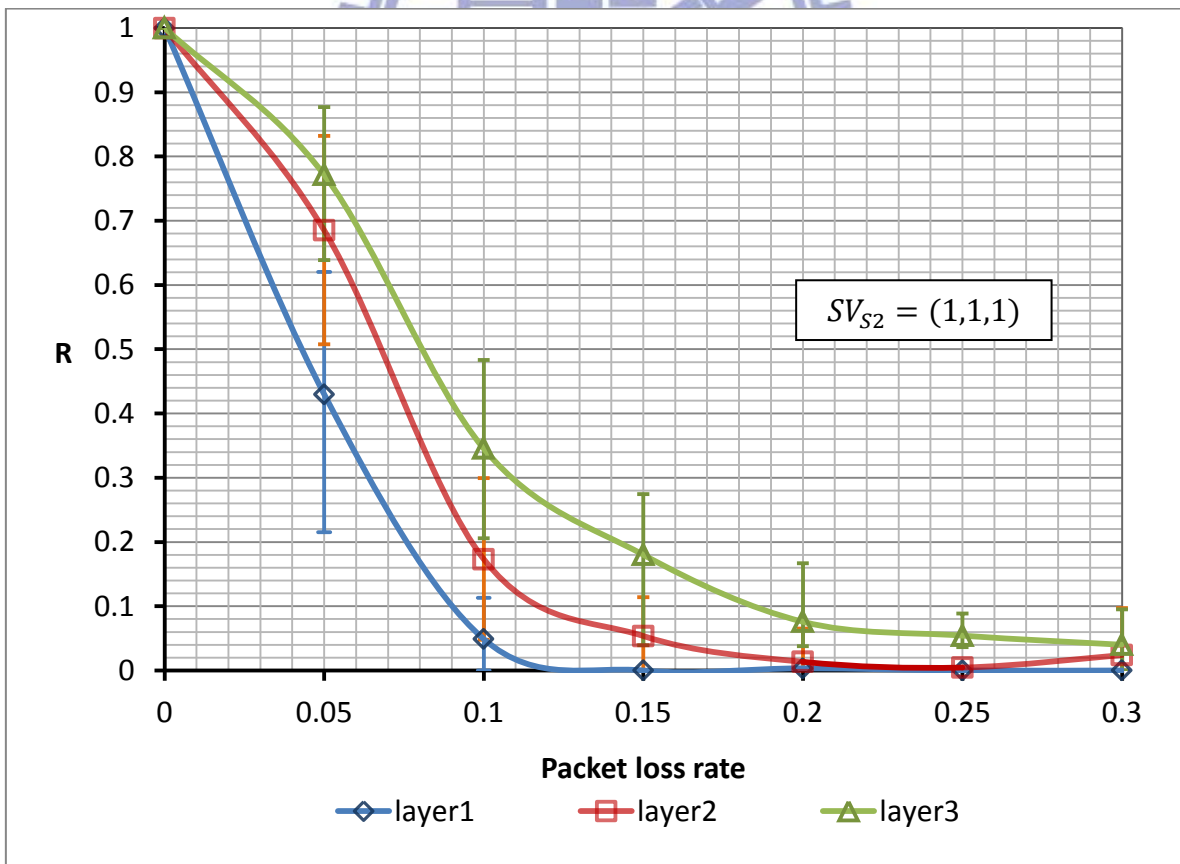


Figure 5.14: Sink 2 after background losses

Figures 5.10 and 5.11 represent the ratios $R(i, j)$ for $i = 1..3$ and $j = 1..2$. We observe that the resistance of coding scheme 1 to background error losses is significantly lower than the resistance in topology (a), since $R(i, j) < 1$.

This attempt to normalize the retrieval rate does not help us to draw any further conclusions since the curves are extremely similar to Figure 5.10 and Figure 5.11.



Chapter 6 Conclusion

6.1 Contributions

First, we used the proposed extension of the structural properties of Minimal Subtree Graphs, to derive an exhaustive list of “minimal” coding schemes for the modest configuration of 3-sources-2-receivers.

Since this work is first and foremost a practical one, the most important contribution is an actual implementation of UEP-RLNC. Along this application, we developed and tested a simple simulation platform that will enable future users to investigate how to achieve high multicast performances for different kinds of bit streams.

These tests confirmed the validity of our prior work and revealed the profiles corresponding to the different packet loss rates. Since these profiles cannot be theoretically obtained yet, this practical approach shows some valuable results that need to be compared with the future theoretical models to come.

6.2 Future Work

The results we obtained can be used to confirm the validity of a theoretical model of UEP-RLNC when it is created. Furthermore, from a practical perspective, our UEP-RLNC application and its associated simulation platform should be tuned and used to run some more experiments on scalable data such as video streams.

The next step of the project should include an algorithm to assess separation vector for any given coding scheme and another algorithm to generate more coding schemes. These results could be used to generate an optimal coding scheme for an arbitrary topology.

References

- [1] *Flows in Networks*, Ford L. R. and Fulkerson, D. R. *Flows in Networks*. Princeton : Princeton University Press, 1962.
- [2] *Network Information Flow*, Alswede, R., et al. s.l. : IEEE, 2000, Trans. Information Theory, Vol. 46.
- [3] *Linear Network Coding*, Li, S.-Y. R., Yeung, R. W. and Cai, N. s.l. : IEEE, 2003, Trans. Information Theory, Vol. 49, pp. 371-381.
- [4] *Network Coding for the Internet and Wireless Networks*, Philip A. Chou and Yunnan Wu June 2007.
- [5] *An Algebraic Approach to Network Coding*, Koetter, R. and Médard, M. 5, s.l. : IEEE/ACM, October 2003, Trans. Networking, Vol. 11, pp. 782-795.
- [6] *Polynomial time algorithms for multicast network code construction*, Jaggi, S.; Sanders, P.; Chou, P.A.; Effros, M.; Egnér, S.; Jain, K.; Tolhuizen, L.M.G.M. Dept. of Electr. Eng., California Inst. of Technol., Pasadena, CA, USA, Information Theory, IEEE Transactions on.
- [7] *Practical network coding*, Philip A. Chou, Yunnan Wu, and Kamal Jain, in Allerton Conference on Communication, Control, and Computing, October 2003.
- [8] *Robust Network Coding in the presence of link failure using multicast rate-diversity trade off*, Hossein Bahramgiri, Mohammad Jabbari and Farshad Lahouti, IEEE information theory workshop, 2007
- [9] *Information Flow decomposition*, Fragouli, Soljanin, and Shokrollahi., Sch. of Comput. & Commun. Sci., EPFL, Lausanne, IEEE transactions on Information Theory, 2006.
- [10] *UEP Network Coding for Scalable Data*, Apirath Limmanee and Werner Henkel, Jacobs University, 5th International Symposium on Turbo Codes, 2008.
- [11] *An Algorithm to Search Minimal Subtree Graphs*, Jing Wang, Ying Li, and Xinmei Wang, State Key Lab of ISN, Xidian University.

[12] *Static network codes using network minimal subgraphs*, Rezagholipour, M.; Ahmadian, M.; Aref, M.R.; Dept. of Electr. Eng., KNT Univ. of Technol., Tehran.

[13] *A framework for network coding in challenged wireless network*, Alaeddine El Fawal, Kave Salamatian, David Cavin, Yoav Sasson and Jean-Yves Le Boudec, School of Computer and Communication Sciences, Ecole Polytechnique Fédérale de Lausanne (EPFL).

