

國立交通大學

資訊學院 資訊學程

碩士論文

於光纖同軸混合型網路上自動救援纜線數據機出廠資料

之方法

An Auto-Recover Method for the Factory Data of Cable
Modem over the HFC Network

研究生：張屹銘

指導教授：袁賢銘 教授


中華民國 九十九年六月

於光纖同軸混合型網路上自動救援纜線數據機出廠資料之方法
An Auto-Recover Method for the Factory Data of Cable Modem over the HFC
Network

研究生：張屹銘
指導教授：袁賢銘

Student : Yi-Ming Chang
Advisor : Shyan-Ming Yuan

國立交通大學
資訊學院 資訊學程
碩士論文

The logo of National Chiao Tung University is a circular emblem. It features a central shield with a book and a lamp, surrounded by the letters 'ES' and 'A'. The year '1896' is inscribed at the bottom of the shield. The entire emblem is set against a background of radiating lines.

A Thesis
Submitted to College of Computer Science
National Chiao Tung University
in Partial Fulfillment of the Requirements
for the Degree of
Master of Science
in
Computer Science
June 2010

Hsinchu, Taiwan, Republic of China

於光纖同軸混合型網路上自動救援纜線數據機出廠資料之方法

學生：張屹銘

指導教授：袁賢銘

國立交通大學資訊學院 資訊學程碩士班

摘要

隨著有線電纜資料服務介面規範（DOCSIS, Data-Over-Cable Service Interface Specifications）產品的普及化，以及大量產品安裝到客戶端的情況下，如何保持產品的品質以及快速修復有問題的量產產品，已經成為企業一個重要的課題。

在實務中某些情況之下，纜線數據機的出廠資料以及數位認證(Digital Certificate)等出廠資料會消失或損壞，根據傳統的手法，若是纜線數據機的這些有獨特性的出廠資料損失因而造成校準表等各個數據機獨有的資料無法修復，就只有送回工廠一途，不只運送時間過久要恢復這些資料也是十分的曠日費時。本論文的目的就是實現一種可以在線上提供資料復原的方法，CAROL (Cable modem Auto-Recovery Online)。透過 Organic Computing 的精神，讓依照 DOCSIS 標準上線的產品，可以在我們的系統以 Network coding 為解碼編碼方式所提供的服務下，順利的恢復重要的資料。我們的目標是能夠做到利用各個在線的裝置所沒有用的到儲存空間，來互相保護重要的資料，因而能降低裝置回到產線來修復的比率。

關於實驗的部份，我們使用業界常用的測試方式，透過有公信力的軟硬體設備，來驗證復原資料的正確性，實驗結果是能完整地重新獲得原本該有的出廠資料。整體而言，經由本論文所提供的方法與系統服務，只要能夠在纜線數據機內實現，並且透過在系統業者端針對需求所下的設定，就能夠有效的提昇纜線數據機的可靠度。

An Auto-Recover Method for the Calibration Data of Cable Modem over the HFC Network

Student : Yi-Ming Chang Advisor : Shyan-Ming Yuan

Degree Program of Computer Science

National Chiao Tung University

Abstract

With the popularization of the DOCSIS (Data-Over-Cable Service Interface Specifications) product, more and more devices are already installed in client side. The topic of how to maintain the quality and the speed of repairing becomes more and more important.

In some practical circumstances, the cable modem may loss the factory data which includes the calibration table and the digital certificates. These are unique and vital for the regular function of cable modem. In the traditional way of repairing this kind of product is shipping it back to the maintenance branch. We would like to present a solution to avoid time waste. It is named CAROL (Cable modem Auto-Recovery Online).

Within this thesis, we focus on applying the concept of Organic Computing and make good use of the advantages of Network Coding, then we could propose a really implementable method based on a common system over DOCSIS network to make the cable modem auto recover their unique calibration tables, digital certifications. Our goal is to make all of the process happen in client side, and simplified the multiple system operator's workload and it will be no necessity to conduct the routine maintenance in low-level system behavior. Thus we can reduce the probability of sending the cable modem back to factory. In our practice, we use the famous utility to measure the throughput of the CM which gets the recovered factory data, and it proved our method in the results.

Acknowledgment

關於這篇論文，最重要的是感謝袁賢銘老師這幾年來的指導，對於論文题目的尋找，老師提供相當大的空間讓我自行思考，同時從旁輔助給我建議，讓我得以順利的訂出一個我喜歡並且能夠發揮的論文題目，並且在論文遇到瓶頸時給我適當的提醒與教導，因而能好好完成這份論文。另外還要感謝蘇俊銘博士的協助，針對這份論文初稿的表現和文字結構做出了許多建議，並且肯花時間聽我對這份研究的看法，這實在是對這份論文在最後的收尾有很大的幫助。還有也要感謝蕭兆鈞熬夜在百忙中，坐在我旁邊與我討論並訂正此論文中有關英文文法與用字的錯誤。同時，謝謝和碩的同事陳信亦，在我設計系統時針對 DOCSIS 相關的問題有許多不懂的地方幫我解惑。最後要感謝一直陪著我的家人以及多年的好友們，謝謝你們的加油打氣，你們一定知道我有多感謝的。

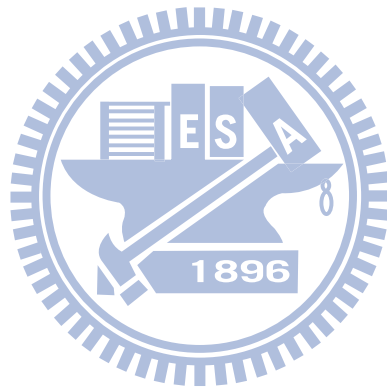


Table of Contents

Acknowledgement.....	IV
Table of contents.....	V
List of Figures.....	VII
List of Tables.....	VIII
1 Introduction.....	1
1.1 Preface and Motivation.....	1
1.2 Research Objectives.....	2
1.3 Research Contribution.....	3
2 Background	6
2.1 Organic Computing.....	6
2.2 Network Coding.....	8
2.3 Factory data.....	9
2.3.1 Calibration table.....	9
2.3.2 Digital Certificates.....	11
2.3.3 The common arrangement for Recovery and Backup.....	13
3 System Architecture.....	16
3.1 Overview.....	16
3.2 The Back-End Services.....	17
3.3 The Head End Devices.....	19
3.4 The End Nodes.....	20
4 Evaluations.....	22
4.1 Nodes Implementation.....	22
4.1.1 Allocate the Jobs.....	22
4.1.2 Encoding and Decoding.....	26
4.2 Program Flow Scenarios.....	30
4.2.1 Node Scenarios.....	30
4.2.2 Program Flow.....	33
4.3 Scenario Demonstration.....	36
4.3.1 Overview.....	36
4.3.2 Implementing Services.....	40
5 Experiment and Results of Test.....	44
5.1 System Performance Test.....	44
5.2 Comparison.....	51
6 Future works and Conclusion.....	54
6.1 Future works.....	54

6.2 Conclusion.....54

7 Reference.....56



List of Figures

Figure 2-1 The core schema of the Organic Computing.....	6
Figure 2-2 The standard example for The Network Coding.....	8
Figure 2-3 Illustrating a protocol stack for a cable modem.....	10
Figure 2-4 Illustrating how a basic authentication works between the CM and the head end.....	13
Figure 3-1 Overall System Architecture.....	16
Figure 3-2 An example for the CM configuration file.....	18
Figure 3-3 TLV data in ASCII.....	19
Figure 3-4 An example of the provision XML file.....	21
Figure 4-1 Using XOR operation to merge different packets into one packet.....	24
Figure 4-2 How to Generate blocks from the source data.....	26
Figure 4-3 Illustrating an example for block generation of each byte.....	26
Figure 4-4 Exponentiation table and Log table using 0xde as the generator.....	28
Figure 4-5 Using the look-up tables to do the multiplication method in the Galois field.....	29
Figure 4-6 Using the look-up tables to do the division method in the Galois field.....	30
Figure 4-7 Illustrating a scenario for backup process.....	31
Figure 4-8 Life cycle of a node.....	32
Figure 4-9 Program flow chart.....	35
Figure 4-10 Topology of the factory data distribution.....	37
Figure 4-11 Off line scheme.....	39
Figure 5-1 Test Topology of test 1.....	45
Figure 5-2 Protocols Configuration of IXIA slot 1 Port 1.....	46
Figure 5-3 Stream Control Configuration of IXIA slot 1 Port 1.....	47
Figure 5-4 IXIA IxRouter configuration.....	47
Figure 5-5 Test Topology of test 2.....	49
Figure 5-6 MRTG Daily Chart.....	51
Figure 5-7 MRTG Weekly Chart.....	52

List of Tables

Table 2-1 The traditional method for maintaining factory data.....	14
Table 4-1 A comparison of information delivery methods of DOCSIS products.....	23
Table 4-2 Task list.....	43
Table 5-1 Comparison of Uni-Directional Downstream.....	48
Table 5-2 Comparison of Uni-Directional Upstream.....	48
Table 5-3 Comparison of UDP Downstream test.....	50
Table 5-4 Comparison of UDP Upstream test.....	50
Table 5-5 Comparison with CAROL system and Traditional method.....	53



1 Introduction

1.1 Preface and Motivation

Even to present days, the competitive strength of the electronic industry has been to concentrate upon reducing the cost and more addictive features to software and the hardware performance enhancement. We always try our best to research and develop newly inventing fields to assure we can develop more powerful project to avoid error within product content. In the past, computer technology usually were used to improve hardware performance and made easy-to-use software. But, applying the great technology to reduce the cost is not a common strategy in business field.

More and more embedded products are equipped with computational power and a little free storage spaces. For such reason, they become more usable and flexible. When multifunction happens, it becomes unreliable in daily life. Since we already have computing power ∙ free storage space device and it is essential for our daily life. We can make good use of these features to deal with some little issue which may consume excessive human resource but lower technique strength. As a result, we can use distributed computing as the main concept to reduce our maintenance cost.

On the other hand, the embedded systems become increasingly unstable and complex. We must handle it with extreme care. In our experience, normally if the cable modem encounter malfunction in client side and the problem can not be simply solved by manual reset. What we can do is just replace it with a new cable modem, or recall it to factory side and try to find out the problem after return to the manufacture facility. In some cases, one of these cases is power cycle suddenly, the cable modem just lost it's calibration table and digital certificates, so that malfunction occurs. What we need is to re-calibrate it or find a method to preserve it's calibration tables. In this general practice, we can not avoid the increasing cost of maintenance. Because each tuner has their different features. Even if we keep the general calibration data, it can not totally meet the requirement of

every tuner. And it is almost impossible to keep each identical calibration data for every cable modem. The digital certificates is a kind of the company property, it's not allowed to store in the client side. Based on the budgeting control, the multiple system operator (MSO) and the factory would not willing to preserve this kind of unique data. They have to create a huge data base 、 equip complex hardware and assign human resource to maintain. It means they have to give those extra costs to assure the integrity of the calibration data. The whole process is contrary with the economic benefits. In our point of view, the cable modems have sufficient computing power 、 storage spaces and the ample band width to maintain the calibration data for each other. Since it is almost essential in our daily life, and cable modems bare be power off. We can restored the calibration data without the presence of every cable modems that hold part of backup data. This concept is similar to a kind of application of Network Coding. The other feature of DOCSIS (Data Over Cable Service Interface Specification) 3.0 is high transmission speed, there is 4 channels bounding for upstream can provide about one hundred Mbps. It can be conducted under the network computing. The whole process of factory data recovery or data distributing can be quickly completed.

1.2 Research Objectives

As we mentioned in the previous section, we try to make all the process could be resolved inside the cable modems, and make sure the cable modem can self-handling each necessary processes without or with the minimum management from the MSO. It is called the CAROL (Cable modem Auto-Recovery Online) system. In this thesis, we focus on applying the concept of Organic Computing and make good use of the advantages of Network Coding. However, a system could be named organic if all of its components and subsystems are well coordinated in a purposeful manner. Organic structures realize themselves as hierarchically nested processes, and structured to be able to meet upcoming challenges by goal-oriented reactions [1]. The Organic Computing is capable to achieve complex distributed systems with addictive abilities, such as self-configuration 、 self-organization 、 self-repair 、 self-optimization 、 self-protection [2] or adaptation. So that the

embedded systems become more life-like and can satisfy what we need. The embedded systems become free employees with no wage. Therefore we believe the concept and principles of the Organic Computing are capable to satisfy our demands. Then we could propose a really implementable method which is based on a common system over DOCSIS network to make the cable modem auto recover their unique calibration tables, digital certifications and default settings. Our goal is to make all of the process happen in client side, and simplified the multiple system operator's workload. So that there will be no necessity to conduct the routine maintenance in low-level system behavior.

1.3 Research Contribution

This thesis will discuss how to make a automatic recovery method in the cable modem. This is a proposed and completed method which includes the provision method for the MSO, mechanism that distributes the factory data, action flows which can retrieve the original data, and how to add customized log for tracing the life cycle of each cable modem. Additionally, we proposed equations as the tips for the developer who can customize the original method into an more efficient method based on their specific and unique demand. And we also provide the test results which were testified by some common tools. Such authorized tools are commonly used in industry. We can easily compare the traditional method and our system by these test reports. There are four merits of this thesis can be summarized as the following parts:

- **The factory data is auto recovered and auto distributed**

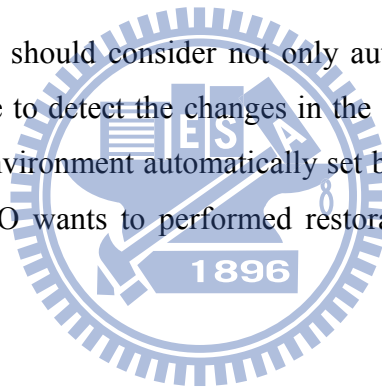
This is the most important part. We design some appropriate rules and patterns for internal behavior in the Hybrid Fiber-Cable network of the cable modems which should deliver some requested fairish functionality for protecting calibration data. To actually elevate the necessary degrees of freedom for adaptive behavior, we will definitely need the self-reconfigurable system for

this task. Choosing the recovery data distributing timing is the key of this system. In this thesis, we give a general introduction to this life-like method, and propose a generic calibration data exchanging architecture [3].

- **Flexibility and multiple methods of recovery**

According to the actual usage of the cable modems over the Hybrid Fiber-Cable network, we design a flexible method to fit any possible case. The life-like system must be prepare an environment for the self-organization between cable modems, whether the amounts of CM is large or small. The method must be designed as which could be easily resumed original status after being stretched or expanded.

As for the recovery plan, we should consider not only automatic but also manual operation. The cable modems should be able to detect the changes in the surrounding environment, which in accordance with changes in the environment automatically set based on user's demand or behavior. If the user, for example, the MSO wants to performed restoration manually, we should provide theses selective options for users.

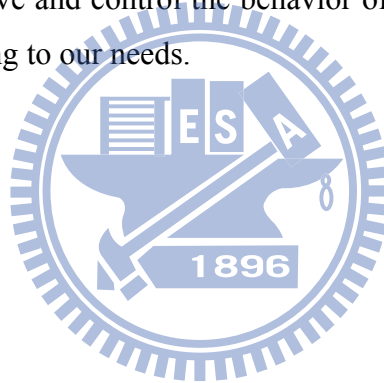


- **Less management**

This system is a model of CM-CM interaction in which the calibration data processing has been thoroughly integrated into each objects and activities. In other words, this is an active protection system. Whether the activity is distribution or recovery, it should be achieved between the cable modems. This thesis provides a method to make all of the clients can cover each other without prejudice to common operations. After integration of all the cable modems into a group or many independent groups, this specific system becomes increasingly complex. This feature raises a new challenge for designing and implement .

- **Traceable logs**

There should be some logs for history inquiry. As we mentioned before, this system must be flexible. We have to trace the process flow and make some notes, then we can handle any case that we might image or not. For example, if increasing number of cable modems join this system, we can study the operations and then evolve the system by the logs. To achieve this goal, we should apply the second research objective and try to design the log generator. The principle is to make the log generator simple and effective, and integrated into each important point in this system. Thus, it is important that input should be limited and simplified as much as possible. The application generates logs automatically, and the log is used only for research and recording. This demonstrate the need for new methods to evolve and control the behavior of this flexible systems and to create trustworthy systems which adapting to our needs.



2 Background

2.1 Organic Computing

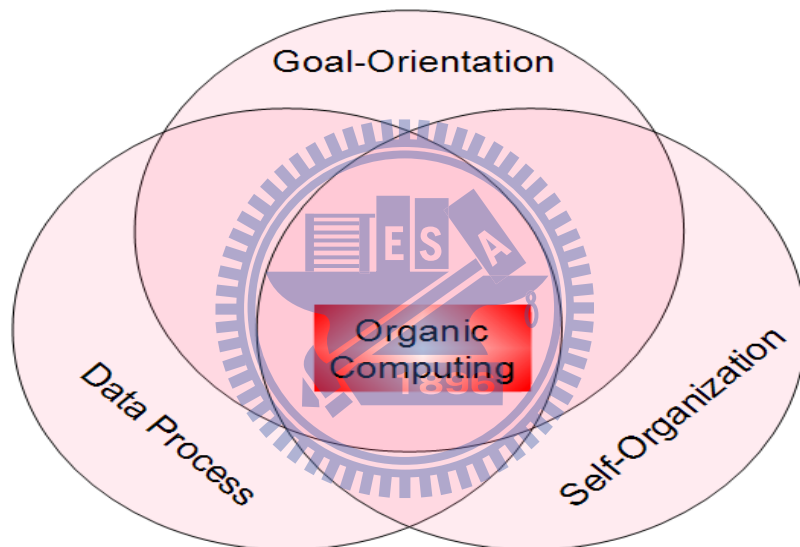


Figure 2-1 The core schema of the Organic Computing

As we illustrated in **Figure 2-1**, we demonstrate our most interesting field by drawing the core schema of the Organic Computing. The Organic Computing is a paradigm for the design a complex system. It was proposed by a department in University of Hanover that is SRA (System – und RechnerArchitektur)[9] . It mainly involved in the launch of the Organic Computing Initiative. [8]Also, the Organic Computing is a new vision for distributing embedded systems. In fact, when we try to design a method that really can help us to protect the factory data, the Organic Computing

is the first idea that attracts our attention. In the beginning, there is only a concept in our imagination. The whole concept is to make the embedded system can take care of each other, and the concept of the Organic Computing seems to be helpful in building our system.

The extended applications and ideas from the Organic Computing are already been published many times. It is similar to the Ubiquitous Computing. The initial incarnation of ubiquitous computing was in the form of "tabs", "pads", and "boards" which was built on Xerox PARC, since 1988. And it was called the Calm Computing. They advocate that the computers usage should be catered for the human behavior which involving people's daily environment and autonomous interaction whatever user-generated.

As we know, there is a continuing climate in Information Technology to distribute and parallel both data accessibility and data operating. Certainly, all of the operating systems are smarter and more capable but also gradually become complicated. The undesirable error-proneness and colossal costs of host management and maintenance in famous distributed machines may demonstrate the increasing complexity of up-to-date software systems. There are already some admirable endeavors who made this science field progress to confront this complexity with contemporary technology. The traditional approach of configuring software in specific hierarchical regulated layers and modules with standardized and customized interfaces is unable to satisfy our need. The researcher who focus on computer science demanded to build new ideas. New challenge is to configure systems of interacting actions and modules in a specific method, which can really stand for the desired manners of the whole system. The modern development of the demanding features in computer systems achieve the cornerstone of self-controlled computing and the prototype of Organic Computing. These features are generally named as famous "self-x properties".

The Organic Computing system was developed as a autonomic computing system and this concept will not require an algorithmic division of labor, and is under the proceeding evolution, development, self-organization, adaptation, learning, teaching, and goal-orientation. Thus, we want to build up a system that facilitate this core concept to satisfy user demands for trustworthy cable modems which act life-likely and can perform self-recovery without human guidance. In order to avoid repair the cable modem one by one manually, a life-like device can achieve this goal avoid

repair the cable modem one by one manually, a life-like device can achieve this goal perfectly.

2.2 Network Coding

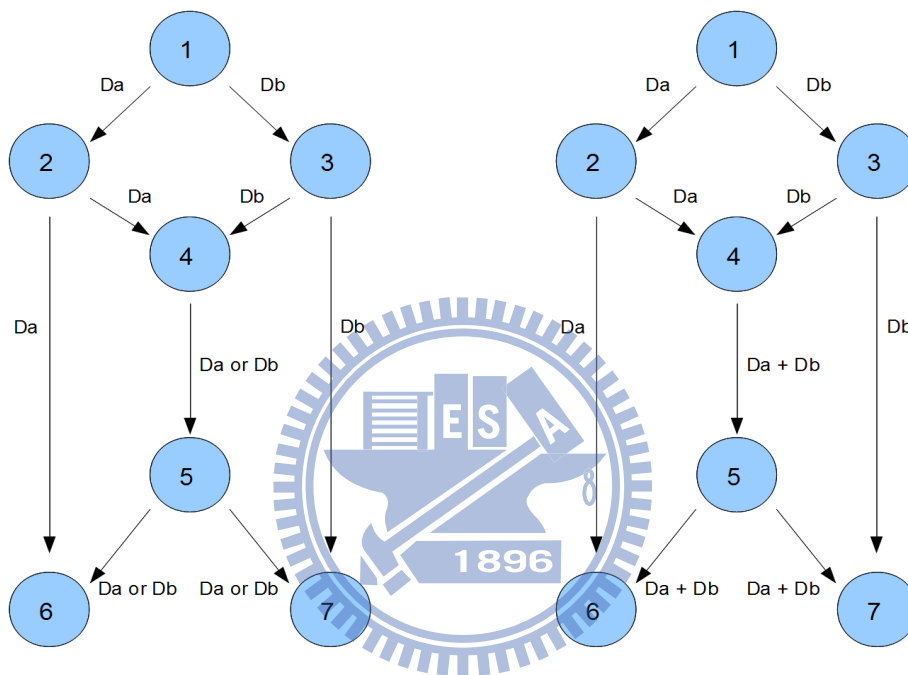


Figure 2-2 The standard example for The Network Coding

In the last few years, there is a method named Network Coding that arouse a lot of relative researches. Because this method can improve the efficiency of network transmission and the reliability by a wide margin. We can say that all the research about the Network Coding are derived from the seminal journal of [11], where it was modified that the maximum flow with minimum cut capacity in a common multicast network can only be accomplished by allowing intermediate nodes to merge different data flows, and it's demonstrated by a number of studies in network coding research (e.g. [12], [13] and [14].) To disclosed its conceivable means to support higher throughput and robustness, specially where highly volatile networks such as sensor networks, mobile ad-hoc

networks and P2P networks are concerned. Since we want to implement a method which can be applied into the peer-to-peer network, surely we have to draw on this advantage.

The basic idea behind the Network Coding is demonstrated in **Figure 2-2**. Let us assume that the node #1 wants to send data bits D_a and D_b at the same time (to say multicast) to the nodes #6 and #7. It is very clear to find out that the link between the nodes #4 and #5 will form a bottleneck. That is, in the sense that not only bit D_a is forwarded (in that case the node #6 does not get bit D_b), but also bit D_b is sent (in that case node #7 will receive incomplete information). Because of the minimum cut to each destination equals 2, this one follows that even if the capacity of the network is 2 bits per transmission, this capacity is impossible to achieve unless the node #4 jointly encodes D_a and D_b , for instance, we can do it through an XOR operation that allows perfectly recovery at the sinks.

2.3 Factory Data

There are many specifications and works for CM data recovery implementation. This Section introduces the target data – calibration table and digital certificates which are very important for CM working properly. Also, the cable architecture and simple protocol stack for a cable modem are sketched in this section.

2.3.1 Digital Certificates

At beginning, in the **Figure 2-3 Illustrating a protocol stack for a cable modem**, we like to introduce this protocol stack to catch the point why we have to protect calibration table. And referencing to the Data-Over-Cable Reference Architecture [6], we can see that, RF tuner handles the Downstream (DS) and Upstream (US) RF Network in the PHY 3.0 interface. The calibration table is an individualized data table for the modem tuner. For example, about the Downstream calibration table, it provides the basic calibration of the technical possible power levels when the cable modem is receiving Downstream signals. In the other hand, Upstream (US) calibration must

compensate for frequency response of the upstream chain. According to the specification that provided by the chip maker [7], all standard requirements related to US transmitting accuracy are calibrated as follows:

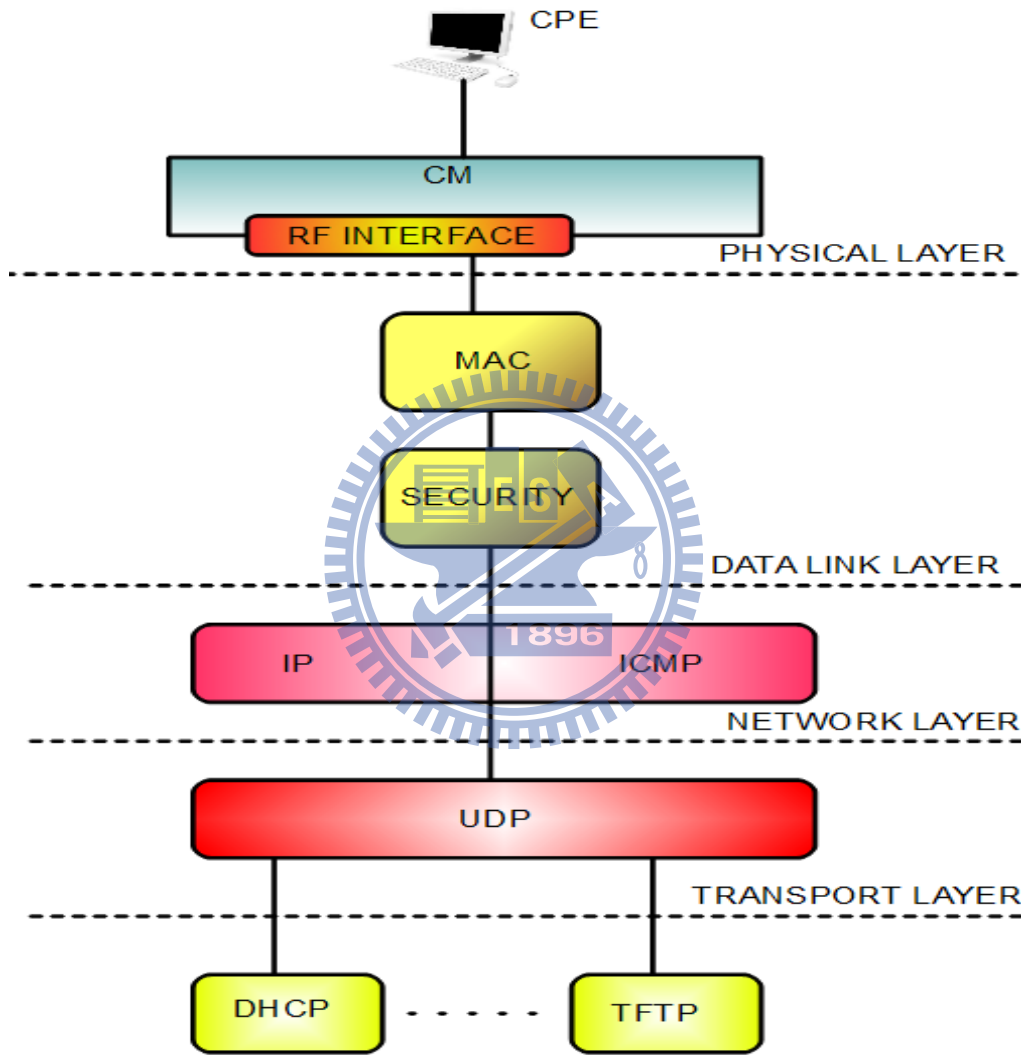


Figure 2-3 Illustrating a protocol stack for a cable modem

- The CM is required to report its output power accurately and limited within 2dB.

- The step resolution in transmit power for each channel must be 0.5dB or less.
- The step size accuracy must be within 0.4dB.
- All active channels without a power change command must change no more than 0.1db during a burst even when a power change command is used on any subset of other active channels.

Once we get the calibration data, it will affect the CM signals. The nominal power level of the upstream CM signal(s) will be as low as possible to achieve the required margin above noise and interference. Uniform power loading per unit bandwidth is commonly followed in setting upstream signal levels, under the specific levels which is established by the cable network operator to achieve the required carrier-to-noise and carrier-to-interference ratios [6].

In power-on or reset the provisioning procedures and initializations reveal that a Euro-Modem or Network Interface Unit (NIU) is capable of tuning to the correct channel in the upstream and downstream directions and then it can receive the following basic network parameters. Then, the sign-on and calibration are performed in order to adjust the internal clock and the transmission power of the NIU according to the specific transmission delay and cable attenuation [5]. In the other words, if we lost the calibration tables, the cable modem may encounter malfunction to get link or perform worse throughput than it's own unique calibration tables.

2.3.2 Digital Certificates

According to International Data Corporation (IDC), worldwide Cable Modem Services Market to Reach 69.4 Million Subscribers in 2008, the number of installation is doubled in 2003. As the amount of cable modems being subscribed continues to soar, cable-service piracy through cloning the cable modems is gaining more concern. The CableLabs has created the DOCSIS specification, this specification demands that all of the cable modems should be authenticated with digital certificates, and this requirement is made for address this problem. The digital certificates usually are installed into cable modems during manufacture. All manufacturers who want to continue to sell it's own cable modems, must comply with this specification.

The Cable Modems do not require any user-name or password verification in the same way just like what the other dial up connections do. As we mentioned before, the certificates are hard coded into the Cable Modems when they are built. It is the X.509 digital certificate and it is built up of the some items:

- * A serial number
- * Cryptographic public key
- * Ethernet MAC address
- * The Manufacture's Identification

The X.509 is verified by the head end equipments, usually it is the Cable modem termination system (CMTS). Once this certificate has been verified, the following data which will sent to user is encrypted in public key. At beginning, the cable modem sends its own X.509 certificate and manufactures certificate. What the verification is checking expiration date, and confirming the name of issuer. And finally that the X.509 signature will be valid under the manufactures certificate public key. In the other hand, the CMTS also uses DOCSIS root public key to test signature and verify the manufacturer certificate. The CMTS responds to ensure the certificate owner is actually the correct owner when the head end has verified the x.509 certificate. To do this, the head end will encrypt the cable modem public key with the authorization key.

In Figure 2-4 Illustrating how a basic authentication works between the CM and the head end, we can see the Cable modem must uses it own private key to get the authorization key (if it was an fake one, it won't have the correct private key). Using this it will generate a Hash-based Message Authentication Code (HMAC) key and reply to the CMTS with this. And the verification of this Hash-based Message Authentication Code key proves the CM has the private key to match the public key.

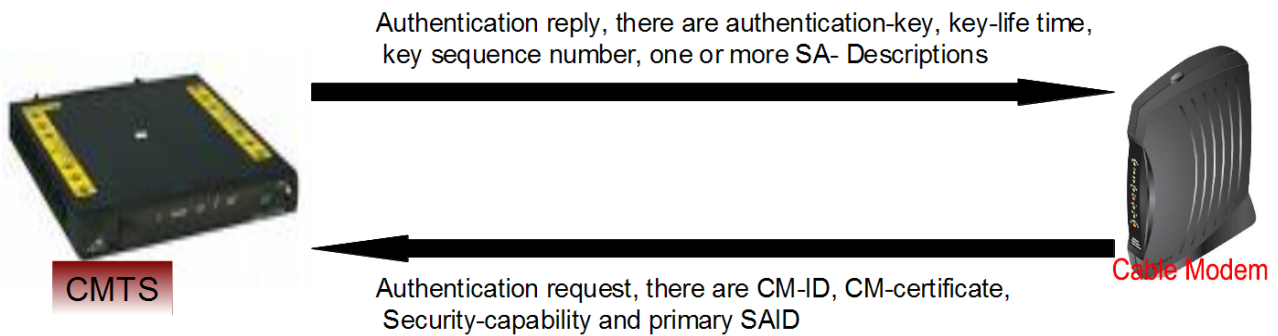


Figure 2-4 Illustrating how a basic authentication works between the CM and the head end

For the reason that the Cable Modems it would be possible for anyone to acquire the flow data , whatever sending or receiving, with some special packet sniffer or some similar tool like that. The cable operators (to say, MSO) always use a system which is known as Baseline Privacy Plus (BPI+) to solve this kind of problems. The BPI+ encrypts all the packets to ensure that no one can snoop on any transmissions. Surely it also stops the illegal use from someone trying to make any connection to gain free access. The Baseline Privacy Plus encryption must travel with the information where ever it goes. In the **Figure 2-4**, the CM certificate is the X.509 certificate and the CM-ID is the cluster of the serial number, manufacture ID, MAC address and RSA public key. In this manner, the available maximum encryption is 128 bit. To have a data exchange, first key exchange will be performed. The key exchange uses triple DES as its encryption. This is quite a strong encryption and provides a satisfactory protection to the key exchange. What the algorithm used is a public key exchange. As it used to be, the United States has laws on the level of encryption that is allowed to be exported, thus it is not very easy to get the unique digital certificate keys for our factory to produce our cable modems. That is the point why we have to protect the digital certificates of any our cable modems.

2.3.3 The common arrangement for Recovery and Backup

In our journal mining and research, we did not find any completed solution for implement of

the Network Coding on any kind of embedded system product. But there are some ideas of p2p data sharing based on Network Coding we can refer to. One of these research, Zhu, Li and Guo [21] have generally provide an algorithm which takes full advantage of application-layer overlay networks. In the conventional view, the data can only be replicated and forwarded by overlay nodes. But in their algorithm, linear network coding was applied to application layer multicast. They constructed a multicast graph, which is a subgraph of the rudimentary mesh and a supergraph of the rudimentary tree. It constructed a 2-redundant multicast graph, which means that each node has two disjoint paths to the source. They designed their algorithm such that the costs of link stress and stretch are explicitly considered as constraints and minimized[21]. We refer to this idea, and give the full capability of encoding and decoding data in CM nodes. However, while dynamic membership is a common phenomenon in peer-to-peer networks, this paper did not discuss about how to process peers with dynamic leaving or joining. Although they stand for the point of view that their algorithm essentially doubling the end-to-end throughput in most cases, but the property of 2-redundancy limits the minimum cut of the multicast graph, which in turn limits network throughput. In the other hand, Dimakis, Godfrey, Wainwright and Ramchandran [19] provide a method of fundamental tradeoff between storage and repair bandwidth. They present a regenerating codes method which can significantly reduce the repair bandwidth. Before we can present our solution, we have to put my thinking cap on this paper.

	Backup	Recovery
Process	Manually process	Manually process
Implement	Store data in Factory	Recall and Replace DUT
Advantage	Sheer institution	Sheer institution
Disadvantage	Can not do it immediately	Outside data base, it takes risk
Feature	Follow SOP, but only few Factory wants to do this	Low entrance hurdle, but only few MSO wants to do this

Table 2-1 The traditional method for maintaining factory data

In this industry, we list the methods for maintaining factory data in Table 2-1. With the previous practical experience, in the backup part, we saved the unique factory data after the product is ready. When any one of these product lost it's original data, the MSO sent it back to factory and give the client another good DUT (device under test) which is been repaired under the traditional recovery method. This arrangement works on the premise that the Factory is capable to maintain these original data files, and the MSO is willing to coordinate with these arrangement. Even if we meets all of the conditions, it still makes the production costs growing. For giving up these disadvantages and enhancing production yield rate, we made this CAROL system and we will demonstrate it in next chapters.



3 System Architecture

In this chapter, we introduce the scheme of the system in detail. In the section **3.1 Overview**, we present the system architecture for most important concepts of this system. In section **3.2 The Back-End Services**, we discuss about the necessity of these servers. In section **3.3 The Head End Devices**, we describe the main target we want for this system. In section **3.4 The End Nodes**, we portray the way we design the mechanism and what methods we build on each node to make this system working well.

3.1 Overview

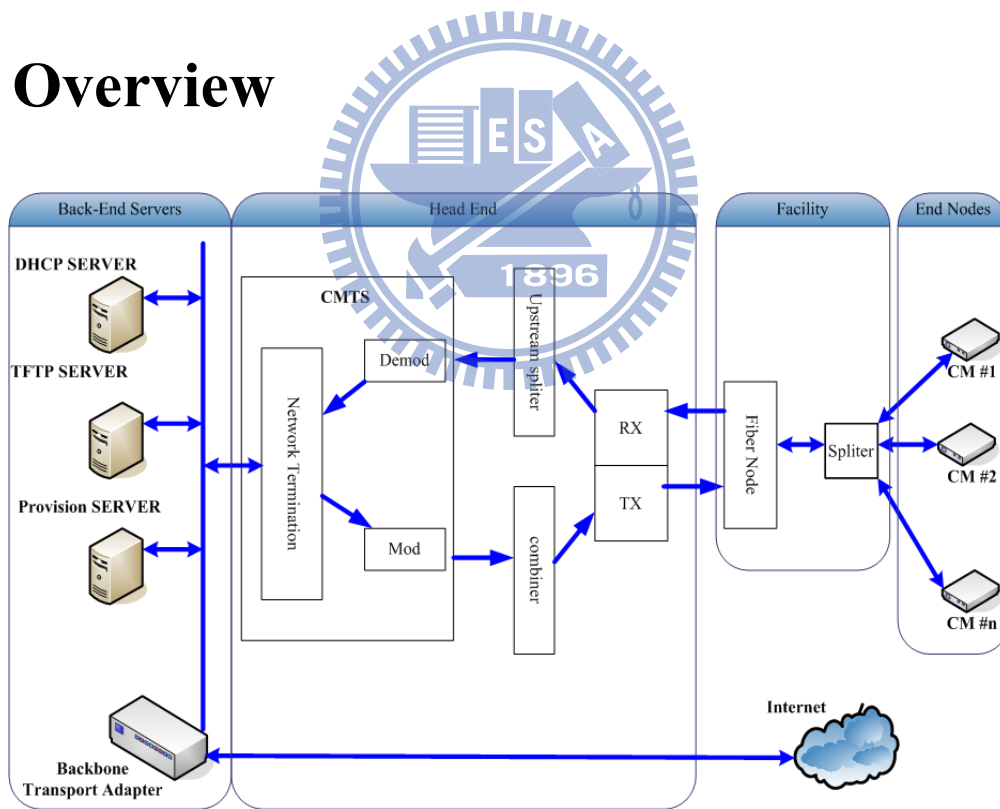


Figure 3-1 Overall System Architecture

With the rapid economic growth , the networking products vendors pay more attention to fast integration for increasing client demands. For this case, the most serious issue is the length of time for a product to be conceived til its being finished and ready for sale. So that the vendor always consider that the shipment date than the quality and reliability of product. The traditional method to recovery factory data is to insert a set of hard code data into the product, but this method will be suitable while the data can be applied to each same kind of product. If the data is being the only one of it's kind (or been individualized), the conventional ways of re-installment (e.g. digital certificates), readjustment (e.g. the calibration table) and backing up with the prepared or the common data of golden board is not suitable. The disadvantages of these traditional methods are time wasting and losing the client trust in the product consistency and quality, thus we want to follow the standard DOCSIS working model and build a system to do the factory data maintenance automatically. With the concept for this system, the client is preferring to fix this issue at their hands. We offer a solution that can meet this condition without giving any instruction to the client.

As we present on Figure 3-1, it includes three main parts for this system: (1) The back-end services. (2) The head end device. (3) The end nodes. We explain the roles for each part in below sections.

3.2 The Back-End Services

In this part includes TFTP server , DHCP server, TOD server and provision server. It makes standard provision as what [6] DOCSIS spec detailed description of design criteria for the end nodes, which can be precede normal online status and provision the vendor data for this system. This was been adapted in this thesis. The end nodes can obtain the related information of recovery system from the provision servers. For DOCSIS spec, the cable modems have to parse the configuration file to get the standard information that helps the cable modems to communicate with the head-end (that is CMTS) and lock the provisioned frequency. And then the cable modems can also get the extended information. After a CM completes it's ranging adjustment, it establishes IP

connectivity through using of a DHCP provision. A DHCP server provides the IP services that contain the necessary IP information for the modem to establish IP connectivity, which including its IP address, the IP addresses of the TFTP server for download of the CM configuration file, and other parameters such as the fields must be present in the DHCP request from the CM and MUST be set. In this system, we design all the necessary information in a XML (extensive makeup language) file. This is also a famous method for MSO and vendors to provide their information for their customized services. In **Figure 3-2 An example of the CM configuration file**, we provision the URL (Uniform Resource Locator) of the XML file as TLV 150 in the CM configuration file.

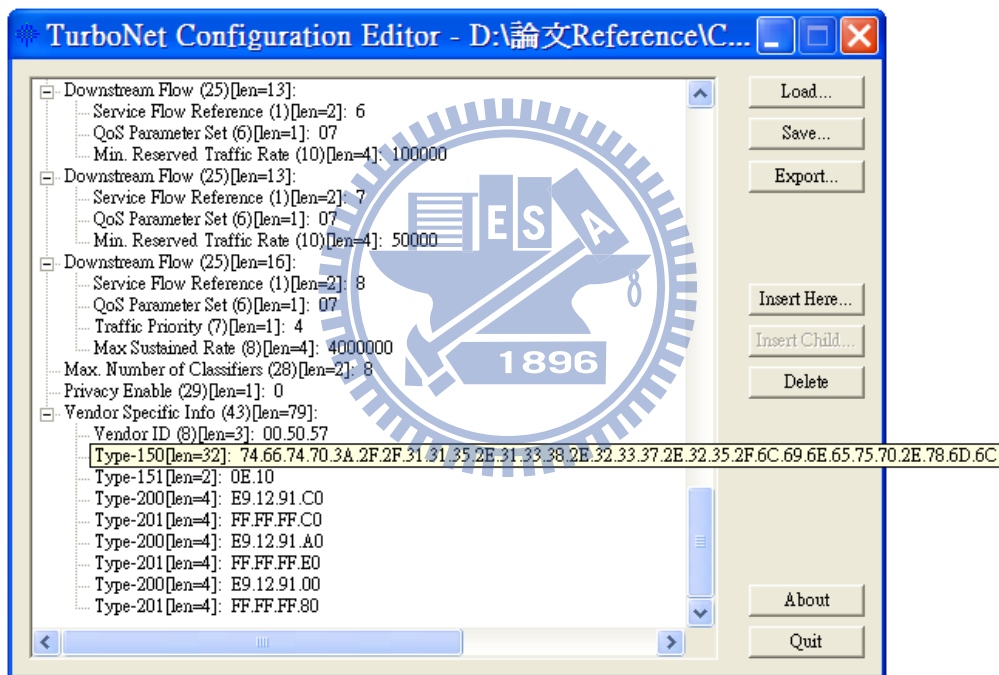


Figure 3-2 An example for the CM configuration file

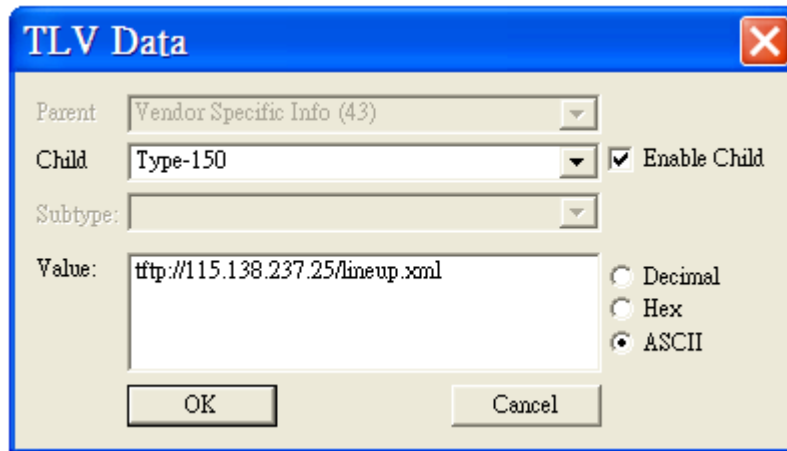


Figure 3-3 TLV data in ASCII

Usually, the configuration file in its final form is a binary file, and typically a configuration tool is used to create the file. There are many public available tools to assist in the creation of DOCSIS CM configuration files. As this tool TurboNet Configuration Editor which we use in this thesis, it is very clear that the CM will receive the TLV 150 value in Hex form. In **Figure 3-3 TLV data in ASCII**, the TLV 150 must be translated into ASCII form by the XML parser of each cable modem. After the CM obtains the necessary information of this system, the CM could comply with ordered operation mechanism of this system. We will discuss this part later.

3.3 The Head End Devices

This part includes the CMTS and the operator aggregation network. The main target we want in this part for our system is setting a basic CMTS system running configuration for the cable modems to enter standard DOCSIS online status. For implementation, we have to setup some basic policies, such as the CMTS interface for the CM to communicate to, private IP address that is used for the CM to talk to the CMTS, the public IP address which is used for CPE (Customer Premises

Equipment) to talk to the CMTS and this IP address is also assigned so that the CPE can go to the internet, the IP address of DHCP server, type of annex (there are different scan plans between DOCSIS, Euro DOCSIS and Hybrid scan plans), modulation type (e.g. 64qam), the default downstream frequency that must match the up-converter for the CMTS platforms, the power-level of upstream, the default upstream frequency that is set to a clean part of the spectrum and after careful analysis, and the last provision is to enable the cable DHCP functionality so that primary addresses are signed for cable modems and secondary addresses are assigned to remote hosts.

3.4 The End Nodes

The end nodes are the cable modems. Their job is simple, the main target is just make sure their factory data is safe. That is the CM have to follow our indispensable rules of backup processes to reach this target. After the cable modem establishes IP connectivity through use of a DHCP that is specified by the CMTS. A DHCP server provides the DHCP policy and the IP services that contain the necessary IP information for the cable modem to establish IP connectivity, including its IP address, the IP address of TOD server to synchronize the operation time for the cable modem and the IP addresses of the TFTP server for download of the CM configuration file that includes Network Access Configuration Setting, Class of Service Configuration Setting and End Configuration Setting such as vendor provision data (e.g. TLV 150). Then, the cable modem acquires the XML file by parsing this configuration file and access the URL of this XML file. Thus the cable modem can get our provision data to start the backup process. There is an example of the provision XML file in **Figure 3-4**, what we can provide are the IP address of provision server, the id of the provision server, the scale size and the block number. In some cases, there are several provision servers in the operator aggregation network. For extenuating the burden of the provision server and increasing the efficiency of recovery, we can counterpoise the load by allocate the cable modems into different provision servers. About the scale size, we provision this to the cable modem as how many end nodes will backup its factory data. The block number is the order that shows how

many blocks of the factory data should be divided and generated. After the cable modems collect these information, the cable modems have the basic rules to comply with this system, and we present the more detail implementation in the next chapter.

```

<?xml version="1.0" encoding="big5" ?> <!-- declare -->
<catalog> <!-- root start -->
<FactoryDataRecovery Version="1.0000"> <!-- version -->
  <provisionServer>
    <ip>233.18.145.195</ip> <!-- server ip address -->
    <id>A0</id> <!-- server id -->
    <scale>7</scale> <!-- member counts -->
    <blocks>3</blocks> <!-- blocks of org data -->
  </provisionServer>
  <hr />
  <provisionServer>
    <ip>233.18.145.196</ip> <!-- server ip address -->
    <id>A1</id> <!-- server id -->
    <scale>13</scale> <!-- member counts -->
    <blocks>6</blocks> <!-- blocks of org data -->
  </provisionServer>
</FactoryDataRecovery>
</catalog> <!-- END -->

```

Figure 3-4 An example of the provision XML file

4 Evaluations

In this chapter, we start with proposing how we design this system with clear environment for demonstration. In section **4.1 Nodes Implementation**, we portray the way we implement the mechanism and what methods we build on each node to make this system working well. In section **4.2 Program Flow Scenarios**, we describe the necessary scenario and flow of this system in detail. In the section **4.3 Scenario Demonstration**, we present the demonstration architecture and how to construct the main architecture. There are some services of this structured recovery system consist in basic distribution and recovery operations, we define overall structure and fail-over mechanisms. And then list the task of services in order to present these important tasks we must implement in this system.

4.1 Nodes Implementation

4.1.1 Allocate the Jobs

According to the above introduction, the cable modems were in charge of the backup process. In order to play an appropriate role and showed their usefulness, we allocate 3 parts of main assignments for each nodes, these implementations should be helpful to anyone who wants to put them into practice on their DOCSIS product:

1. **XML parsing**. In normal case, the power-up operation of any DOCSIS product can be seen as a step-by-step flow. For the purpose of this part, in our design, we assume that the vendor (operator, also know as MSO) expects to provide the information of environment settings and guiding the nodes to start the process all over the Hybrid Fiber-Cable network by the XML configuration file, just after the step of receiving the DHCP policy.

2. **Posting requirements and Receiving requests.** In table 4-1 A comparison of information delivery methods of DOCSIS products, there are some famous selections of implementing this part: Broadcasting User Datagram Protocol (UDP) packets, accessing configuration file (in which a central TFTP server provides a configuration file to a number of cable modems connected together in a Hybrid Fiber-Cable network), and accessing data base server. Because CM certificates is a sensitive data, we can not use this kind of method due to the security issue. About the broadcasting, we have to open one or more ports to listen the special packet which vendor can defined. In this case, the cable modem just accepts the packets that include useful information by filtering the packet contents and discard others. This mechanism acts to prevent performance losing or memory leak. About accessing configuration file, we have a real example. In South Korea, the LG-Powercom as a supplier of high-speed communications access services, adopted the commercial deployment of BigBand's video IP service and IPTV service, it uses the configuration file to provision the On-Screen Display (OSD) information of channels and primary frequencies to lock IPTV channels. For fetching the advantages of these two methods, we make the cable modems parse the XML file to get necessary information. Thus the cable modems were guided to broadcast their requirements and communicate to other nodes, and then transmit the data in our design format based on Network Coding way.

Method	Advantage	Disadvantage
Broadcasting	Fast data transmission	Security Issue
Configuration file	Easy to maintain	System Performance, and Storage issue
Accessing Data Base	Easy to build DB server	Maintenance and Security Issue

Table 4-1 A comparison of information delivery methods of DOCSIS products

3. **Backup and Recovery.** On the basis of the above introduction, we adapt the essence of Network Coding to perform data backup and recovery processes. The data that we delivered, either backup or recovery process, are all underwent encoding and decoding sequence. The idea that how

a system working with Network Coding is well established, it depends on 4 necessary components: **store**, **forward**, **encoding** and **decoding** [15][16][17] . There are a lot discussion about the idea of encoding and decoding processed, but in our journal mining and search, we did not find a completed solution for implement of Network Coding on an exact distributed system. So that we referred these ideas in our own project and offer our implementation below.

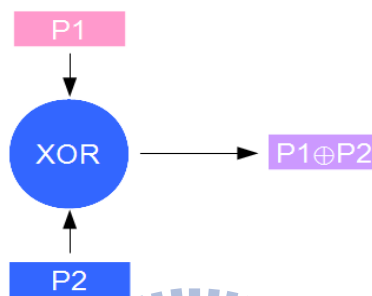


Figure 4-1 Using XOR operation to merge different packets into one packet

About the **store** operation for this system, each cable modem snoops on the HFC network and follow a certain rule that is provisioned by the MSO to accumulate specific amount of overhead packets from other nodes. The overhead packet includes Media Access Control address (MAC address), IP address and backup data of the owner. The MAC address and IP address make the receiver acknowledge that this overhead packet belongs to which cable modem, and this receiver can follow the guidance to prevent from excessive overhead packet inflow from the same sender. That is, the vendor can provision a maximum limit of the overhead packet from same sender as a guideline for the receiver. And it is periodic property of snoops. Of course, the vendor provides a certain period to guide each node snooping.

Concerning the **forward** part, while a node receives recovery request that includes the index of overhead packet, MAC address and IP address of the requester. Then this node should search it's "stock" to find out is there any overhead packet belong to this requester by matching the MAC address and the index of overhead packet. If there is any available overhead packet for this requester, receiver node should announce the owner and make a tunnel to send it back. Therefore, it

can be conducted through conventional routing protocol.

In **Figure 4-1**, we can see how a node uses XOR operation to merge different packets into a single packet. In **Figure 4-2**, there is an idea that we can divide the source data into 3 parts and generate 4 blocks originally from those parts, that is **encoding** method. Therefore we can **decode** it by using Gaussian Elimination method and solving these equations. This is a famous idea but there are not much literature discussions about how to implement it actually. Since we must put it into practice, we have to present a executable method to implement this part. It is easy to perform addition and subtraction. Addition and subtraction can be simply performed by XOR operation. In the binary scale, we can just combine two bytes into a single byte, and then calculate any single byte of those two bytes by XOR combination with another byte from other equation. The more difficult parts are performing division and multiplication.

In the beginning, it is confusing that there is harmful in multiplying any packet by the constant. About the encoding operation, we can not acquire the result of division and multiplication. Thus we can not do decoding operation. After searching and designing, we use what is well-known as a Galois Field (GF) to implement this part:

$$GF(p^k) = f(x) / h(x) \quad (1)$$

p is a prime number.

$$f(x) = \sum_{i=0}^{k-1} a_i x^i, a_i \in \{0, 1, \dots, p-1\}; \quad (2)$$

The degree of $h(x)$ must be k , and it is an irreducible polynomial in this set. This is a special mathematical construct where addition, subtraction, multiplication, and division are redefined, and where there are a limited number of integers in the field. In binary scale, the overhead packets are regarded as continuous bytes. As we know, the size of an unsigned integer is one byte, and the

specified range is from number zero to 255. We can let the $p=2$ and $k=8$ to fit in it, then we can get a limited result after addition, subtraction, multiplication, or division in an 8 bit number. In more detail, it is presented in next section about how to perform addition, subtraction, multiplication, and division of Galois Field with Gaussian Elimination methods.

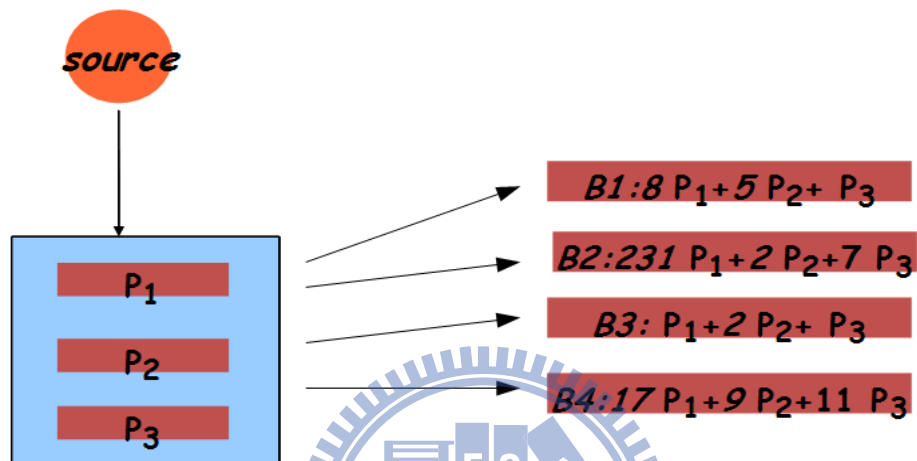


Figure 4-2 How to Generate blocks from the source data

4.1.2 Encoding and Decoding

$$\begin{array}{r}
 a1 \quad P1 \quad + \quad b1 \quad P2 \quad + \quad c1 \quad P3 \quad = \quad B1 \\
 a2 \quad P1 \quad + \quad b2 \quad P2 \quad + \quad c2 \quad P3 \quad = \quad B2 \\
 a3 \quad P1 \quad + \quad b3 \quad P2 \quad + \quad c3 \quad P3 \quad = \quad B3 \\
 \text{Byte} \quad \quad \quad \text{Byte} \quad \quad \quad \text{Byte} \quad \quad \quad \text{Byte}
 \end{array}$$

Figure 4-3 Illustrating an example for block generation of each byte

About the encoding operation, we simply multiply the packets by the constants according to the above introduction. There is an illustration in **Figure 4-3** which demonstrates an example for block generation of each byte, each byte of any packet is regard as an single unit. After we multiply an 8 bit number constant by the single byte from any packet, the cable modems just process these result with addition and then we have the byte from single block.

In the case of multiplication, it is a trifle more complicated [22]. In Galois field for this system[26][27], we take coefficient C and parameter P as two 8 bit numbers, and we take B as an 8 bit product. We can say that, the symbol C is a constant that defined by this system, the symbol P is the single byte of the packet and the symbol B is the the single byte of the block. In the beginning, we set the symbol B to zero. And then make a copy of C and P , which we will simply call C and P in the rest of this algorithm. After that, we do the following five operations (A) to (E) 8 times as a loop. (A) If the low bit of the symbol P is set, XOR the product B by the value of the symbol C . (B) And then Keep tracking of whether the high bit (8th from left) of C is set to one. (C) Rotating C one bit to the left, then discarding the high bit, and making the value of zero to the low bit. (D) In this step, we check if the high bit of C had a value of one prior to this rotation, then XOR C with the hexadecimal number 0x1b. (E) Now we Rotating P one bit to the right, discarding the low bit, and making the 8th left most bit have a value of zero. After processing this loop, now has the product of C and P as the product B .

Log table:																
	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00	00	ff	28	e2	50	c5	0b	7b	78	5e	ed	2c	33	ee	a3	a8
10	a0	8b	86	68	16	f6	54	d1	5b	26	17	41	cb	0f	d0	0d
20	c8	ef	b3	6a	ae	f5	90	f1	3e	12	1f	09	7c	24	f9	5a
30	83	a7	4e	6e	3f	4b	69	e7	f3	b4	37	27	f8	48	35	d9
40	f0	bc	18	81	db	58	92	2b	d6	89	1e	97	b8	0a	1a	a1
50	66	dd	3a	8a	47	51	31	da	a4	ca	4c	b6	22	88	82	2e
60	ab	7d	cf	d2	76	4d	96	7e	67	d4	73	72	91	df	10	d8
70	1c	3d	dc	9c	5f	ea	4f	07	21	e3	70	f4	5d	eb	02	1b
80	19	fd	e4	e6	40	63	a9	ce	04	36	80	c6	ba	d7	53	9d
90	fe	8c	b1	20	46	7f	bf	23	e0	e9	32	fb	42	c7	c9	cd
a0	8e	a2	06	60	62	b5	b2	29	6f	61	79	e1	59	94	03	30
b0	cc	bb	f2	c3	74	15	de	c2	4a	ac	b0	b7	aa	55	56	25
c0	d3	c1	a5	9f	f7	64	fa	57	9e	0e	75	af	be	d5	a6	3b
d0	8f	84	fc	2d	9b	93	9a	ec	b9	39	08	6c	38	7a	01	8d
e0	44	11	65	52	05	5c	c4	6b	87	e5	13	ad	77	99	2f	3c
f0	49	95	0c	c0	98	6d	1d	45	85	bd	14	71	2a	e8	43	34

Anti-log table:																
	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00	01	de	7e	ae	88	e4	a2	77	da	2b	4d	06	f2	1f	c9	1d
10	6e	e1	29	ea	fa	b5	14	1a	42	80	4e	7f	70	f6	4a	2a
20	93	78	5c	97	2d	bf	19	3b	02	a7	fc	47	0b	d3	5f	ee
30	af	56	9a	0c	ff	3e	89	3a	dc	d9	52	cf	ef	71	28	34
40	84	1b	9c	fe	e0	f7	94	54	3d	f0	b8	35	5a	65	32	76
50	04	55	e3	8e	16	bd	be	c7	45	ac	2f	18	e5	7c	09	74
60	a3	a9	a4	85	c5	e2	50	68	13	36	23	e7	db	f5	33	a8
70	7a	fb	6b	6a	b4	ca	64	ec	08	aa	dd	07	2c	61	67	95
80	8a	43	5e	30	d1	f8	12	e8	5d	49	53	11	91	df	a0	d0
90	26	6c	46	d5	ad	f1	66	4b	f4	ed	d6	d4	73	8f	c8	c3
a0	10	4f	a1	0e	58	c2	ce	31	0f	86	bc	60	b9	eb	24	cb
b0	ba	92	a6	22	39	a5	5b	bb	4c	d8	8c	b1	41	f9	cc	96
c0	f3	c1	b7	b3	e6	05	8b	9d	20	9e	59	1c	b0	9f	87	62
d0	1e	17	63	c0	69	cd	48	8d	6f	3f	57	44	72	51	b6	6d
e0	98	ab	03	79	82	e9	83	37	fd	99	75	7d	d7	0a	0d	21
f0	40	27	b2	38	7b	25	15	c4	3c	2e	c6	9b	d2	81	90	01

Figure 4-4 Exponentiation table and Log table using 0xde as the generator

In **Figure 4-4 Exponentiation table and Log table using 0xde as the generator**, we can do the multiplication more quickly with a log table and anti-log table. That is a basic concept of how to

do multiplication [26][27]. But we have to find out a faster method because we want to implement this in an embedded product. It must save more memory and the NC need to be in real time. There are some numbers in the Galois field, traversing all possible value in the Galois field beside zero that is possible via exponentiation which is defined as repeating multiplication of the same number. These numbers in the Galois field as known as generators. And the multiplication can be done more quickly with 256 byte exponentiation table and 256 byte log table which is produced by one of these numbers as the generator. So that only needs 512 bytes of memory cost in this way of multiplication. In **Figure 4-5** using the look-up tables to do the multiplication method in the Galois field, we present how to do the multiplication.

```

unsigned char Gal_Multi(unsigned char C, unsigned char P) {
    int B;
    int z = 0;
    //This is because zero is a special case;
    //anything multiplied by zero is always zero.
    //The number zero, consequently, does not appear on the table
    if ((C == z) || (P == z))
        return z;
    //we define antiTable = Anti-log table as an Unsigned char array
    //we define logTable = Log table as an Unsigned char array
    //All with 256 elements
    B = antiTable[ (logTable[C] + logTable[P]) %255 ];
    return B;
}

```

Figure 4-5 Using the look-up tables to do the multiplication method in the Galois field

In normal case, we do not need the division method to do encoding. Therefore, we use Gaussian elimination in $GF(2^8)$ for decoding so that it is necessary to use division method. We taking the logarithm of C and subtracting the logarithm of P from it, modulo 255 to perform Dividing C by P in the Galois field. Especially, division is done by taking the logarithm of the numerator when the numerator is one, which can be represented as the number 255, and subtracting the logarithm of denominator from 255. In **Figure 4-6** using the look-up tables to do the division method in the Galois field, we present how to do the multiplication. As for the implementation of

the Gaussian Elimination method, it is very simple. All we have to do is just replacing the operators (addition, subtraction, multiplication and division) with these methods. And then we can substitute the elements for the parameters.

```
// Dividing a by b
unsigned char Gal_Div(unsigned char a, unsigned char b) {
    unsigned char s;
    if(b ==0)
        return 0;
    if(ltable[a] > ltable[b])
        s = atable[ltable[a] - ltable[b]];
    else
        { //we have to do this for where there are a
          //limited number of integers in the Galois field.
          s = atable[ltable[a] - ltable[b]+255];
        }
    return s;
}
```

Figure 4-6 Using the look-up tables to do the division method in the Galois field

4.2 Program Flow Scenarios

In this section, we present the implemented scenarios and system flows in detail. In section **4.2.1 Node Scenarios**, we portray all the required scenarios of each node. In the section **4.2.2 Program Flow**, we introduce the program flow about main operations.

4.2.1 Node Scenarios

According to the motivation of this system, we want to pass the maintenance tasks to the end nodes. So that we don't have to recall the products which have problems. We design a protocol for the nodes to communicate and behave. There is an diagram in **Figure 4-7** illustrating a scenario for

backup process. We can see clearly how the interaction works between the node B, which helps to backup the blocks, and the node A, which sends its backup block out. First of all, the node A broadcasts its claim for backup demand to each receiver. If any node is ready to intake the block from sender node which needs for help, the helper node (in this case, node B) gets this request. That means there is some free physical space within this helper node. It sends a signal that presents its will of receiving to the node A over a socket connection directly. When the node A gets this signal, the sender must check the status to figure out if it is necessary to have more backup node. If there are not sufficient backup nodes, the node A sends a chosen blocks with established rule for the receiver. If it's not necessary, the node A just ignore this signal, and after a certain time the node B will terminate this socket for backup need diminished. After sender block has been transmitted and the node B confirms the integrity of this block. The node B must notify the node A that this block has been backup successfully. After that, the node A arranges the owner list which records the relationship between the blocks and the nodes, and then the node A notified the node B this relation is created. Thus the node B arranges the stock list and these blocks from the node A is officially protected by the node B.

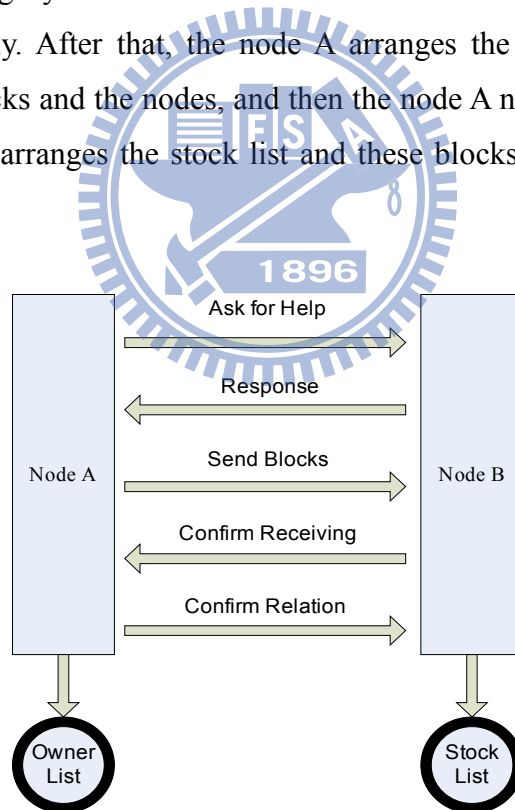


Figure 4-7 Illustrating a scenario for backup process

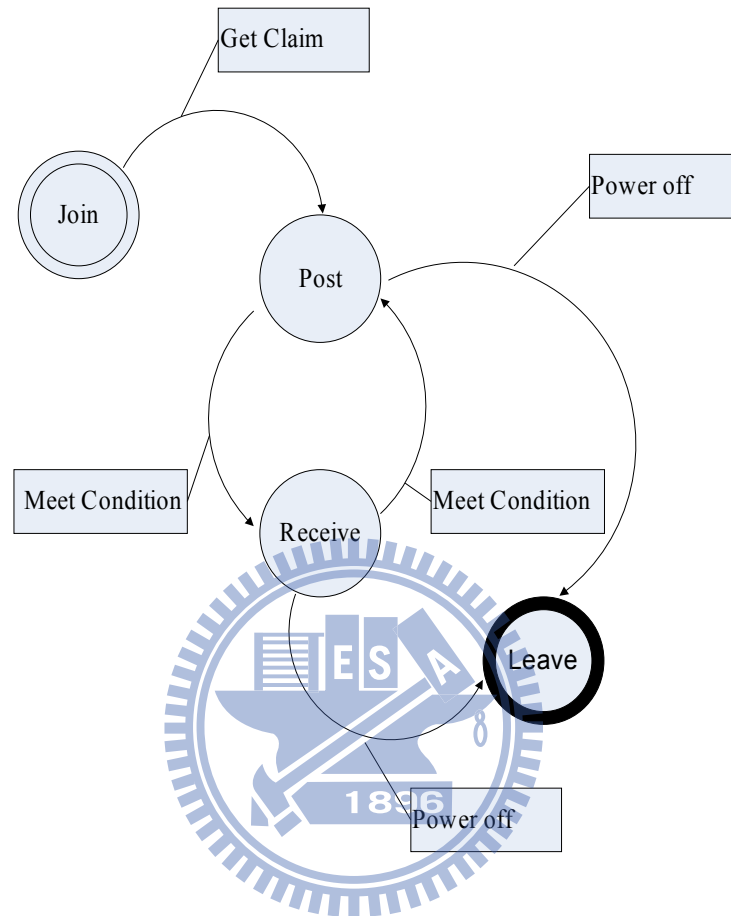


Figure 4-8 Life cycle of a node

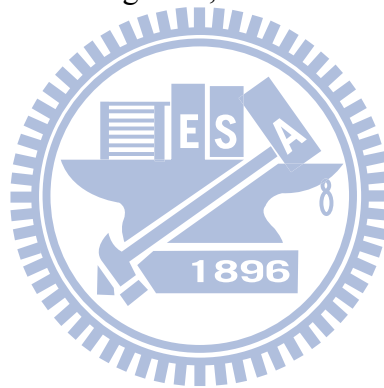
Figure 4-8 shows the Life cycle of a node and there are four states: **join**, **post**, **receive** and **leave**. In the beginning, the node **join** the system and as we mentioned in previous section, the node posts it's claim for searching it's own block within network. When it meets certain condition, it enters the **receiving** states and then goes to finish the backup process. After that, the node keeps listening for any demands from other mode until receives certain signal and then enter **post** status. It depends on the purpose of the signal, this node will send the correct data out corresponding to the request. We can say that, after the node join this system, the node should backup it's factory data and helps the other nodes in the same network to backup their own factory data until it leave this system. Because of each node has to maintain the owner list periodically, when any node leaves this

system (power off), the other nodes should update their list by starting new processes. Thus, depending on the practical implementation, those state transformations may perform the distribution of the factory data. And any node is supposed can retrieve their factory data by following the guidance of these special lists.

4.2.2 Program Flow

We can see the program process of a node from join this system and help other nodes to saving backup blocks to accomplish the backup task of this node clearly, as shown in **Figure 4-9**. From previous sections, we introduce that the node is provisioned with some necessary information includes backup condition which are made by the vendor. After acquiring these information and joining the system, the node (the new comer) announces there is one more available node in this system to the other nodes in this system. And this new node also posts it's demand about counting how many it's own backup blocks in this system. If any node in this system has it's backup block, this new node must gets a response with information of the blocks. This searching process continues for a length of time which was provided by the vendor. After that, this node determines if it is possible to recover to the original factory data by currently existent backup blocks or not. If there are not sufficient backup blocks to recover to the original factory data, this node must generate new backup blocks by current factory data, because in this moment maybe the first time this node join this system. If there are enough backup blocks, this node must asks for these online backup blocks holders and have to regenerates whole new backup blocks. And therefore, this node prepares the backup blocks and ready to separate these blocks into parts and gives a share to each nodes over this network. After this node posts it's demand out, this node creates a circulating thread and opens a sock port for listening to any other node which can help this node to backup it's block. After waiting for a certain time, this node starts to help the other nodes to backup. In the meantime, if there is any node respond to this backup demand, this node keeps sending it's backup blocks out and making a list about which node owns what backup blocks. This thread works until the backup task is accomplished.

In the other side, the node services other node before it's stock is full. Without a doubt, the size of the stock is provisioned by the vendor. This condition makes this system more scalable. After that, this node has to maintain it's list that records what nodes have the backup blocks. This polling operation working periodically. In our design, we set a longer time period for this. Because of the cable modems usually do not leave the HFC network frequently. Base on this property, the cable modems do not have to check the completeness of this list frequently. And as this result, we can have better performance of this cable modem. Of course, we let the vendor provide this property before the nodes join the network. This makes this system suitable for other kind of environment. In the way of traditional cable industry, we can decide the maintenance period time by acquiring the MRTG (Multi Router Traffic Graph). For example, if the using habit in a certain region is not keeping the cable modem online for a long time, we can shorten the length of the maintenance period time.



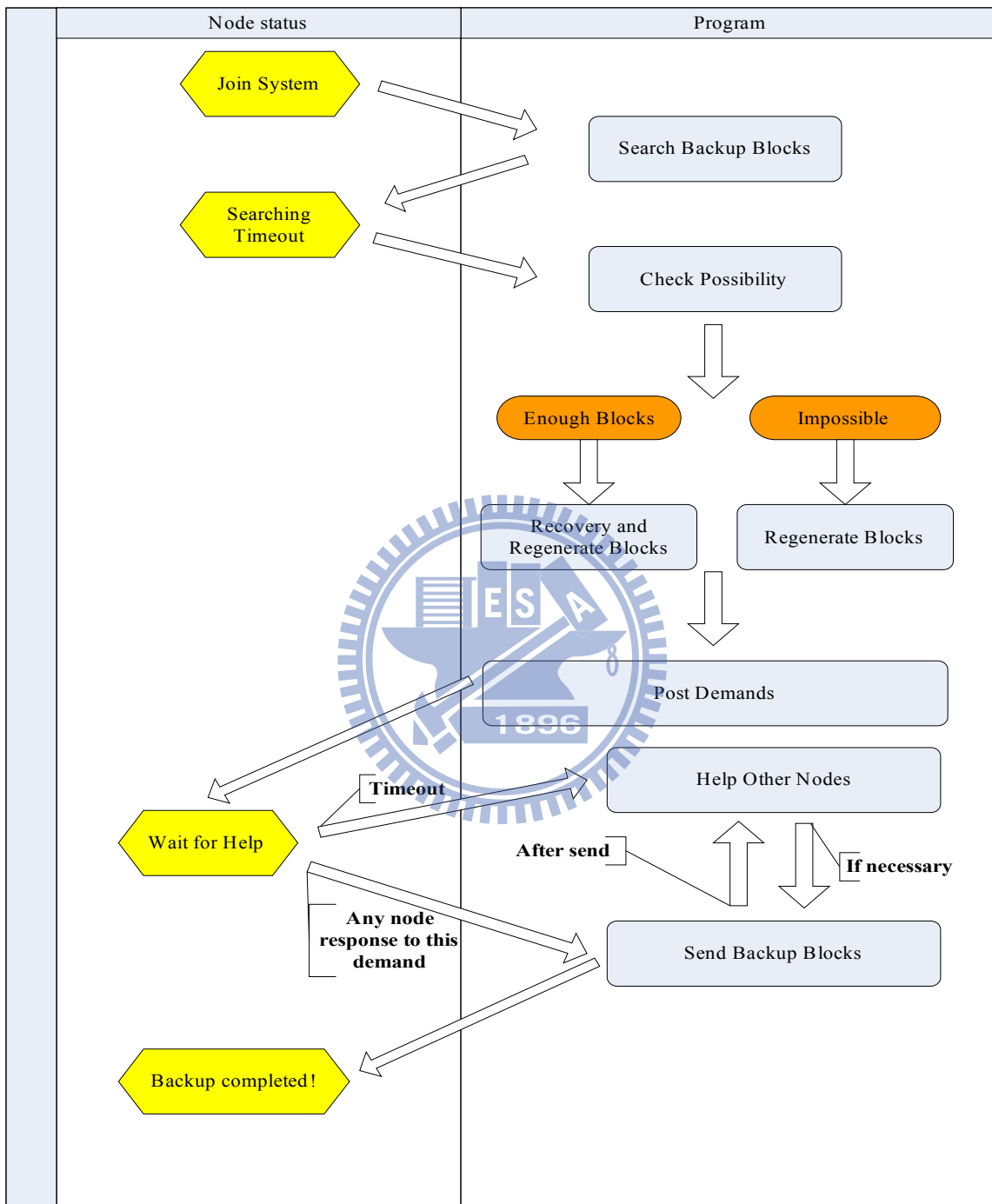


Figure 4-9 Program flow chart

4.3 Scenario Demonstration

4.3.1 Overview

The main purpose of this demonstration is about **testing** and **comparing the traffic throughput** before the recovery operation and after factory recovery operation. The expected result is the unique factory data of our target cable modem which is fully recovered from our system quickly and flawlessly, so that this cable modem can achieve the ideal throughput.

We take the advantage of a special type network with the idea of a structure which has a normal inflected form called combination network. The combination network was presented in [18] "Network coding gain of combination networks". And this type of network, when the size of the network increases, can reach the unbounded NC gain which is calculated by the network throughput ratio with NC to without NC. As we know, the combination network is presented as a multicast network with a topology that conforming to a certain principle. For taking this advantage of the combination network, we make the topology of this system as a regular graph which is presented in **Figure 4-10 Topology of the factory data distribution**. This topology contains three kinds of nodes: (1) Factory Data, (2) Generated Blocks and (3) Receiver CM. We explain these nodes in the following descriptions.

(1) Factory Data: In our design, we tar (derived from tape archive and commonly referred to as "tarball") the factory data files into one compressed file (named FactoryData.tgz). And then this factory data can be divided into the continuous parts. So that we can easily get back the original factory data when the source cable modem gets the continuous parts.

(2) Generated Blocks: As we discussed in chapter 3, this system gets the continuous parts by using Gaussian Elimination to solve linear equation method. Since we get the continuous parts from the tared factory data, the source cable modem can generated blocks by linear coding.

(3) Receiver CM: The nodes at the bottom of this topology are the receivers. These receiver

cable modems which is used to receive certain generated blocks from the source cable modem. In our design, the edge vector of a receiver cable modem's incoming edges must be linear independent.

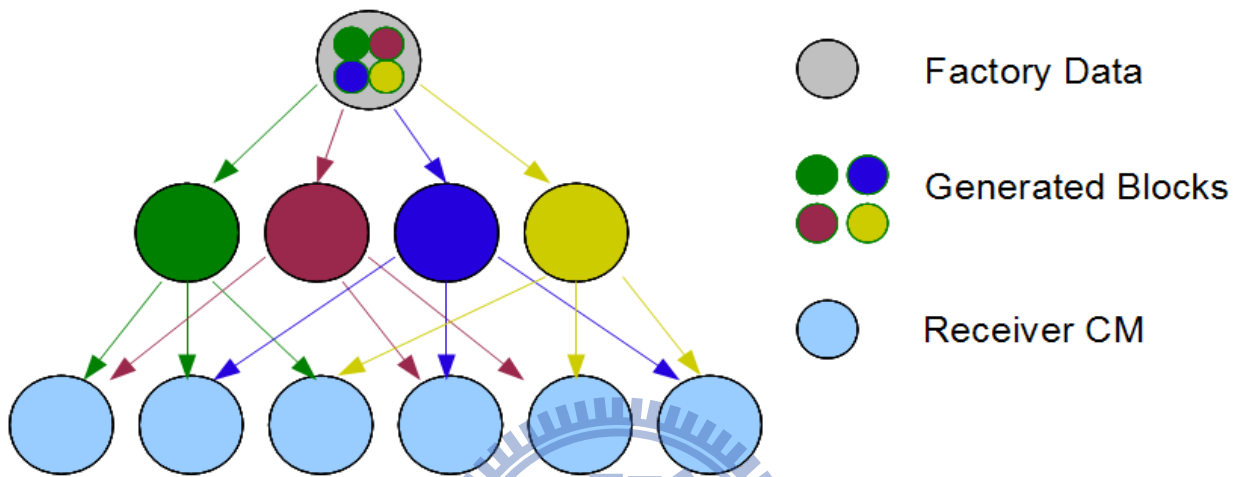


Figure 4-10 Topology of the factory data distribution

As we know, in the random network coding, because of the nodes encode the received data with the random coefficients. That is, the nodes generate random coefficients for the edge functions to determining the outgoing edges' function independently. This kind of feature makes the random network coding maintains a linear coding scheme among the nodes without any control overhead, and become the main advantage [15]. In the other hand, the receiver cable modem's edge vectors of the incoming edges may not be linear independent. But we can not make this system at the risk of fail in factory data recovery.

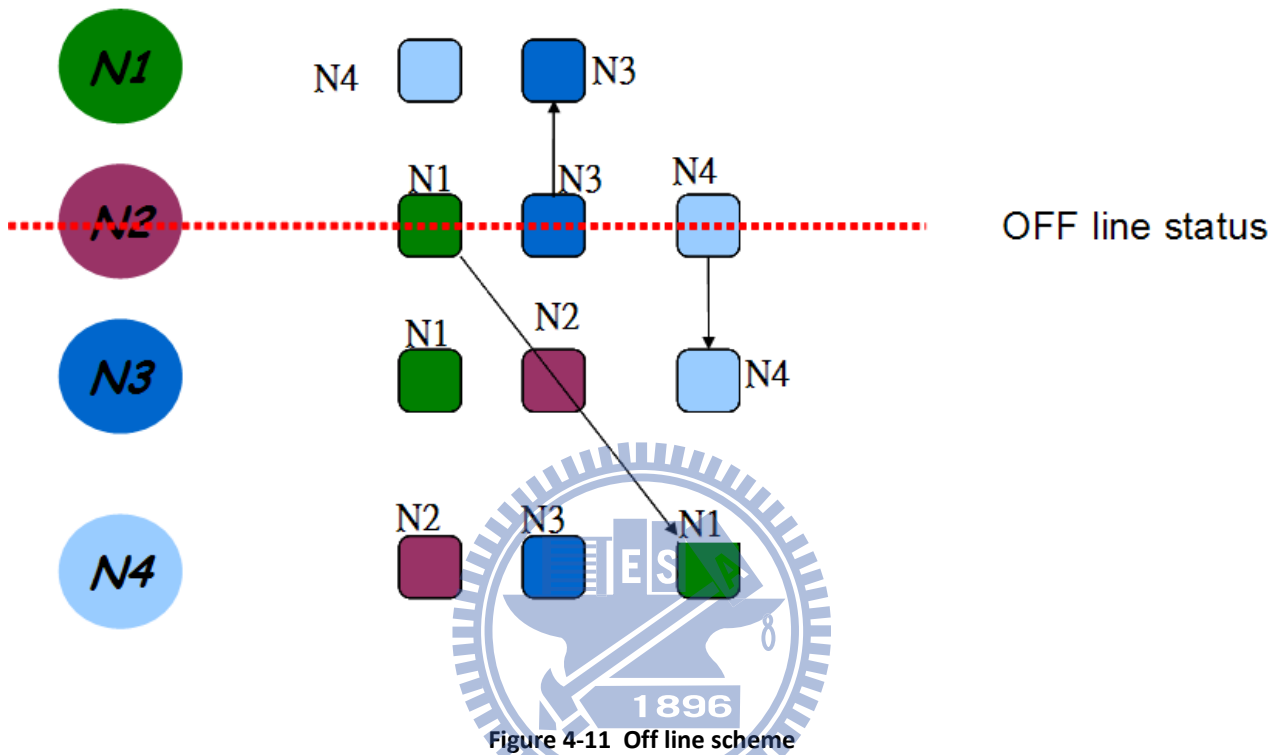
Here is our solution, we make the coefficients that were generated like the way what the deterministic network coding dose. As we present in the last chapter, the cable modem which is going to backup it's factory data, it receives the coefficients that were assigned by the system. In **Figure 4-10** , the source cable modem can generates the blocks with these assigned coefficients for it's factory data. There are n (in **Figure 4-10**, the number is **4**) blocks which were generated by the source cable modem with these known coefficients, and relay these to the receiver cable modems.

There are k (in **Figure 4-10**, the number is **2**) multicast capacities of the network, in another word, any one of the receiver cable modems which receives k generated blocks from the source cable modem. Thus, we can see that there are C_k^n receiver cable modems which receive the generated blocks from the source cable modem.

For this demonstration, the source factory data is divided into three equal continuous parts. The source cable modem generate four blocks with these equal continuous parts by the assigned coefficient. In **Figure 4-10**, any receiver cable modem gets two generated blocks from the source cable modem. As our design and the rules we set to receiver node, any receiver node can only have one block which is the same as another node. In **Figure 4-10**, the green block is **G1**, the purple block is **G2**, the blue block is **G3** and the yellow block is **G4**. We can see that the leftmost receiver node gets **G1** and **G2**, the second on the left node gets **G1** and **G3**, the third on the left node gets **G1** and **G4**, the 4th on the left node gets **G2** and **G3**, the second on the right node gets **G2** and **G4** and the last node gets **G3** and **G4**. For this designed model, there are 7 node available cases: all available, 5 available nodes, 4 available nodes, 3 available nodes, 2 available nodes, 1 available node and no available node. It is easy to understand that if the available nodes more than 4, there are more than 4 kinds of encoded block we can get. Obviously, the source cable modem only needs to get the generated blocks from any two receiver cable modems of these six ones successfully, and then the source cable modem must have three to four encoded blocks. These received blocks are sufficient to return to the original factory data by **Gaussian Elimination**. We can say that, in this demonstration system, we only need one third of the receiver cable modems which are available to response for the recovery operation, so that $1/3$ is our required availability.

Since we consider the issue of the receiver cable modem is unavailable, it is taken for granted to think over the status, when the source cable modem is out of this system in some cases, there should have a method to ensure the distributed blocks available in the network. Let us consider about this case, we have to guarantee the online nodes have enough distributed blocks for maintaining the factory data. In our concept, we must shift the missed blocks to any available

stocks. For implement details, there is a synoptic and schematic diagram in the **Figure 4-11 Off line scheme**. And we are going to present the solution below.



We let the nodes N1, N2, N3 and N4 be the cable modems in the network, and all nodes distribute the source factory data in this network already. These are either source nodes or receivers. In some cases, N2 enters the off line status. For keep enough numbers of receiver cable modems, as we present in past chapter, the source cable modems N1, N3 and N4 sending maintain message and detect the generated blocks in N2 is unavailable for recovery operation. To this step, all of these source cable modem enter the post status, broadcast it's requirement to look for a appropriate receiver cable modem to take the place of the node who leaves. Base on the maintain list, the source cable modems have to give a judgment on the selected receiver cable modem which been percolated through the responded message from all the available nodes in this network.

4.3.2 Implementing Services

There are some services of this structured recovery system consist in basic distribution and recovery operations, defining overall structure and fail-over mechanisms. For arriving the concepts we purposed earlier, some tasks must be accomplished. We like to define these tasks for implementing this system, and then present a table lists all the necessary tasks below for quick reference. It is all in the following Table 4-2 Task list.

- Define size: This service is helpful for reaching the research objectives which is mentioned before. According to different conditions of network, we design a service to accept the setting that provisioned from head-end server.
- Timer: Because of any end nodes must ensure the integrity of the distributed blocks, that is the node must keep the owner list traceable. We implementing a method named seeker for this purpose. Any node can trace the distributed block by this service. And the periodic confirmation time of tracing must use the setting assigned from provision server.
- Define topology: The speed of distribution and recovery is affected by the topology. For determining the replication set and the neighbor set, the node interface with routing information must be implemented.
- Node initialization: Responding to the system design, we implement a method named Init-Handler to take care of the join operation. The node can retrieve available environment configuration with this method. After that the node can really comply this topology and play it's role good.
- Join operation: When any node accomplished the initialization, the node is ready to join this network. We implement the corresponding actions of the working flow, and we design the format of correlative messages for node-to-node communication.
- Distributing operation : In this case, the node creates appropriate amounts of blocks and

maintains the lists we mentioned in previous chapter. Actual allocation is due to the information from the node initialization.

- Recovery operation: In this case, the node follows the list to recovery distributed blocks. We implement corresponding actions according to the scenarios mentioned in chapter 3.
- Detect operation: For preventing in sending and receiving errors, we make an agent to monitor and make sure the delivery process is successful.
- Re-distributing operation: As the seeker find any distributed block is unavailable, the node must send the distributed block out for ensuring the distributed blocks is sufficient for recovery.
- Remove operation: We define the removing actions to remove the blocks out of the maintaining list, and we define the message format for removing request and response.
- Leave operation: If the node is not leaving violently, for example power off. We offer this operation to notify the other related nodes. So that the other related nodes can save time from detecting this node. We define the working flow for the node while leave the system, and we design the communication format.

<i>Task</i>	<i>Description</i>
Define size	Define the size of the distribution operation. Implement <i>node</i> interface to handle the scalable network, to suit different requirement.
Timer	Define the detecting time for maintaining the relation between sender node and receiver node. Implement <i>seeker</i> method for this task.
Define topology	Determine the replication set and the neighbor set.

	Implement <i>node</i> interface with routing information.
Node initialization	Define all the TLV of necessary parameters for boot up configuration. Implement the <i>Init-Handler</i> for environment configuration.
Join operation	Define the working flow for the node while join the system, and define the format of the communication. Implement corresponding actions in the case of <i>JoinInit</i> method and design correlative message sending and receiving actions in <i>MessageComunicator</i> method. Change the status of the node after this operation finished.
Distributing operation	Define the message format for distributing request and response. Implement <i>olistHandler</i> and <i>stockManager</i> for recording the generated blocks and maintaining the received blocks. Implement <i>dataSender</i> and <i>datareceiver</i> for block transferring. Implement corresponding actions in <i>MessageComunicator</i> method.
Recovery operation	Define the broadcasting message format for blocks look-up request and response. Implement corresponding actions in <i>MessageComunicator</i> method.
Detect operation	Define the detecting actions to prevent fail in get back blocks. Implement <i>seeker</i> method for detecting.
Re-Distributing operation	Define the necessary actions to distribute the blocks while the receiver node is unavailable. Define the message format for distributing request and response.

	Implement corresponding actions in <i>MessageComunicator</i> method.
Remove operation	Define the removing actions to remove the blocks out of the maintaining list. Define the message format for removing request and response. Implement corresponding actions in <i>MessageComunicator</i> and <i>selfMaintain</i> method.
Leave operation	Define the working flow for the node while leave the system, and define the format of the communication. Implement corresponding actions in the case of <i>SendLeave</i> method and design correlative message sending and receiving actions in <i>MessageComunicator</i> method. Notify the related nodes of changing status.



5 Experiment and Results of Test

In this chapter, we introduce our experiment and the results of our test in detail. In the section **5.1 System performance test**, We provide our solution about constructing the special structure of this system over HFC network. And then list the result of throughput test in order to present that the recovery of the factory data is doable. For giving more supports to our point of view, we follow the trail of practical usage, it is based on the MRTG (Multi Router Traffic Graph) of a certain MSO in Hsinchu city. In section **5.2 Comparison**, we gives a detailed result of our recovery method in comparison with the traditional method. According to these results, it shows the factory data is recovered from CAROL system.

5.1 System Performance Test

According to our product line experience, there are two features present when the cable modem which lost it's calibration table: lower throughput and higher packet loss rate. So that the main goal of this experiment is using the IXIA 400T to measure the UDP Ethernet throughput of the node which gets the recovered calibration table, and then comparing it's throughput with the throughput of the node without losing the calibration table. The expected result is these two throughput values are really close or the same. We also tested the system performance by IPERF for UDP throughput test, the point is we can monitor and compare the packet loss rates, to make sure the cable modem can work properly with this system.

In our experiment, we design two scenarios: (1) LAN to RF Cable UDP Performance Test by Using IXIA 400T and (2) LAN to RF Cable UDP Performance test by using IPERF. The IXIA 400T is a common facility we use in this industry, and the IPERF is the common soft ware we use.

■ LAN to RF Cable UDP Performance Test by Using IXIA 400T

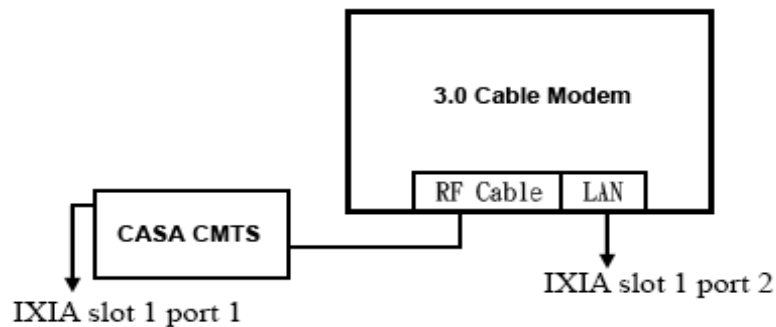


Figure 5-1. Test Topology of test 1

It's easy to understand the topology of this test by **Figure 5-1**. As what it illustrated, we connect one port (of IXIA 400T) to CASA-CMTS, and we connect another port to the LAN port of the CM. So that the data path is clear and no noise data. This is a general test to make sure that we can measure the Ethernet throughput from a clean Environment setting. And there is a list for the special equipments and the test configuration below.

■ Equipments

- CASA CMTS, software version: 5.0.22.11
- IXIA 400T - Chassis, 4-Slot standard form, Includes integrated PC controller, IxExplorer software, and IxScriptMate RFC 2544 & 2889 Test Suites (928-0011)
- LM1000STXS4 - Gigabit Ethernet Load Module, 4-port Dual-PHY (RJ45 and SFP) 10/100/1000 Mbps, Full L2-7 Support; Fiber Ports REQUIRE SFP transceivers

■ Test Configuration

The **Figure 5-2**, and **Figure 5-3** are the most important configurations for IXIA slot 1 Port 1 to slot 1 Port 2. In this case. we set the **frame size** from short to long to ensure the CM can have the best throughput value in every stage. The **no error** configuration and **UDP** setting are set for

improving CPU utilization. We also have to make the stream use continuous packet. After we finished the environment definition, as we can see in **Figure 5-4** we set the routing information and start this test. And then we fill in the interface by the environment condition we setup for this experiment.

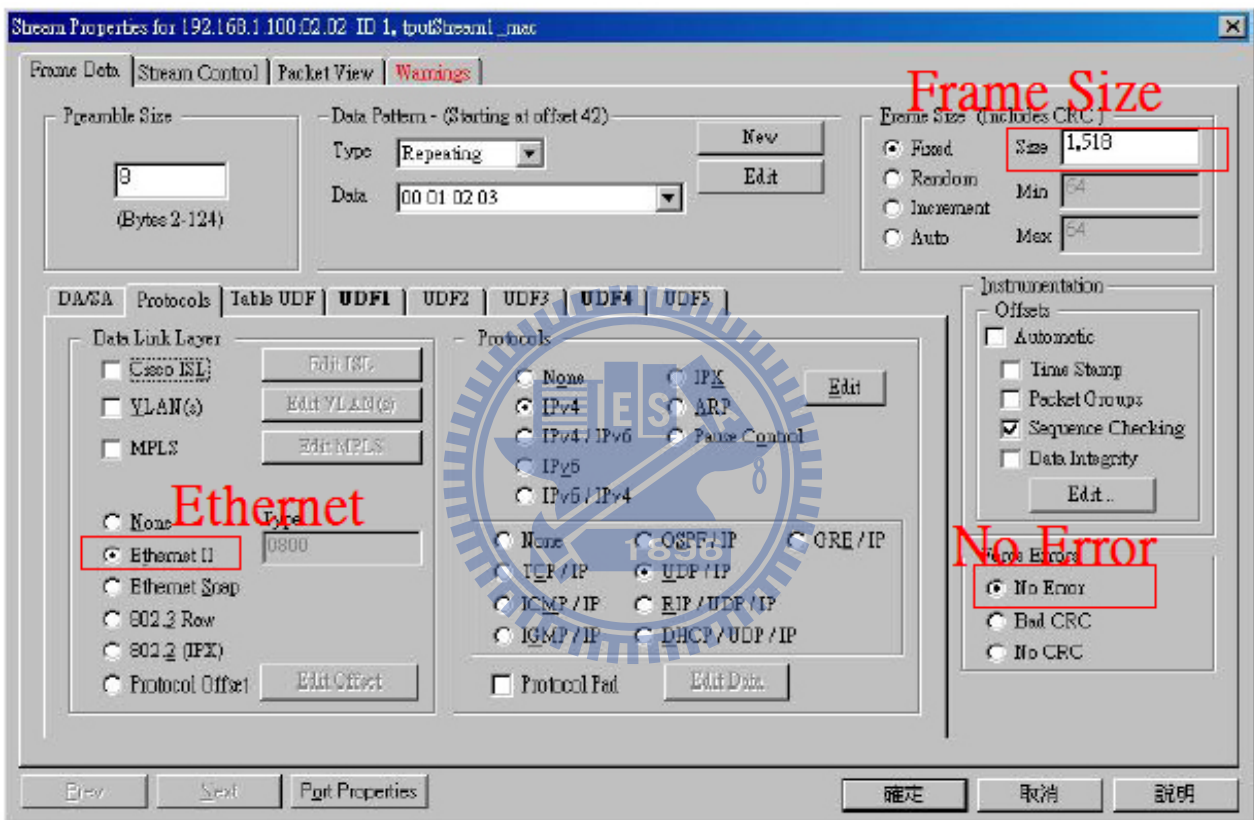


Figure 5-2 Protocols Configuration of IXIA slot 1 Port 1

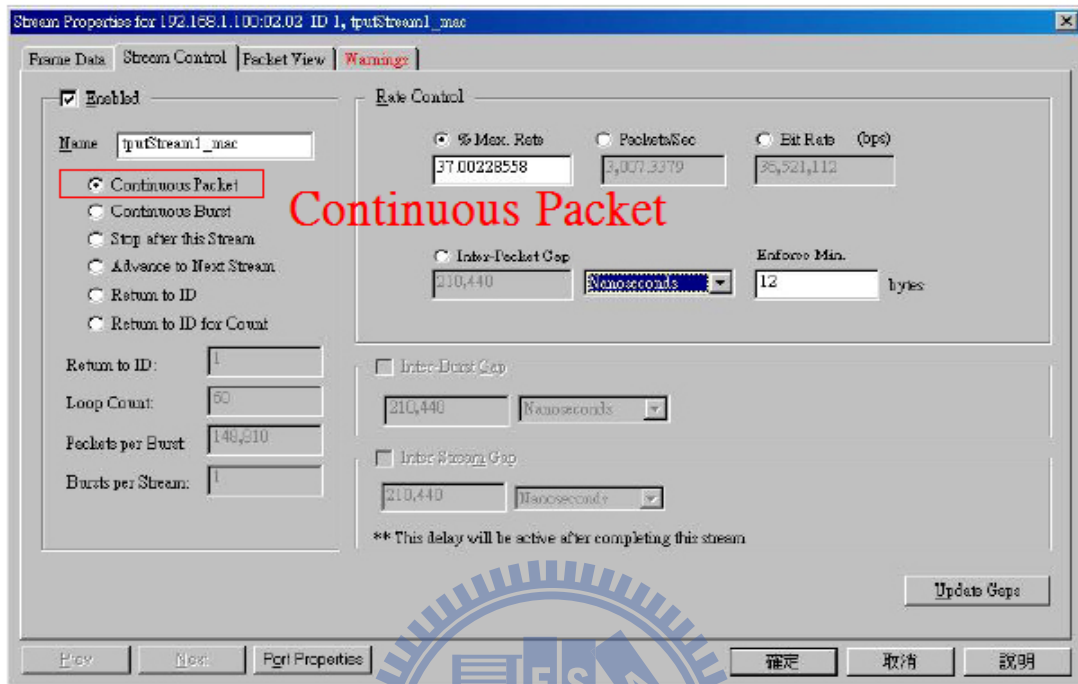


Figure 5-3 Stream Control Configuration of IXIA slot 1 Port 1

Protocol Interfaces							
Unconnected Interfaces		GRE Tunnels		Discovered Neighbors		Interface Addresses	
						DHCPv4 Discovered Information	
						DHCPv6 Discovered Information	
Interface Description							
	Port Description	Port Link	IP Address	Mask	IPv4 Mask Width	Gateway	IPv6
1	192.168.1.100:02.02.01 - 10/100/1000 Base T	Up	11.26.252.1029 - 02.01 - 2	24	11.26.252.10	11.26.252.254	
2	192.168.1.100:02.02 - 10/100/1000 Base T	Up	10.26.252.1024 - 02.02 - 6	24	10.26.252.10	10.26.252.1	

Figure 5-4 IXIA IxRouter configuration

■ Test Result

We list the test results of the DUT with recovered calibration table and original calibration table in **Table 5-1** and **Table 5-2** below. As we mentioned earlier, the throughput value is an important point for proving that the calibration data is recovered. We can see the throughput values of these two subjects are almost the same in each test. That is because of the calibration data is fully

recovered from this system, the DUT with recovered data can reach its best throughput value. By reason of the tuner is calibrated, the CPU does not need to do over work, this DUT is able to have better performance.

Frame Length	Node with recovered data Throughput / CPU Utilization	Node with original data Throughput / CPU Utilization
64	17 Mbps / 100%	17 Mbps / 99%
128	29 Mbps / 93%	29 Mbps / 93%
256	55 Mbps / 97%	54 Mbps / 96%
512	103 Mbps / 99%	103 Mbps / 99%
1024	156 Mbps / 70%	156 Mbps / 70%
1280	154 Mbps / 65%	155 Mbps / 67%
1518	156 Mbps / 60%	155 Mbps / 59%

Table 5-1 Comparison of Uni-Directional Downstream

Frame Length	Node with recovered data Throughput / CPU Utilization	Node with original data Throughput / CPU Utilization
64	9 Mbps / 67%	9 Mbps / 67%
128	15 Mbps / 60%	15 Mbps / 60%
256	28 Mbps / 71%	27 Mbps / 69%
512	56 Mbps / 70%	56 Mbps / 70%
1024	56 Mbps / 41%	56 Mbps / 41%
1280	56 Mbps / 30%	56 Mbps / 30%
1518	56 Mbps / 30%	56 Mbps / 33%

Table 5-2 Comparison of Uni-Directional Upstream

- **LAN to RF Cable UDP Performance test by using IPERF**

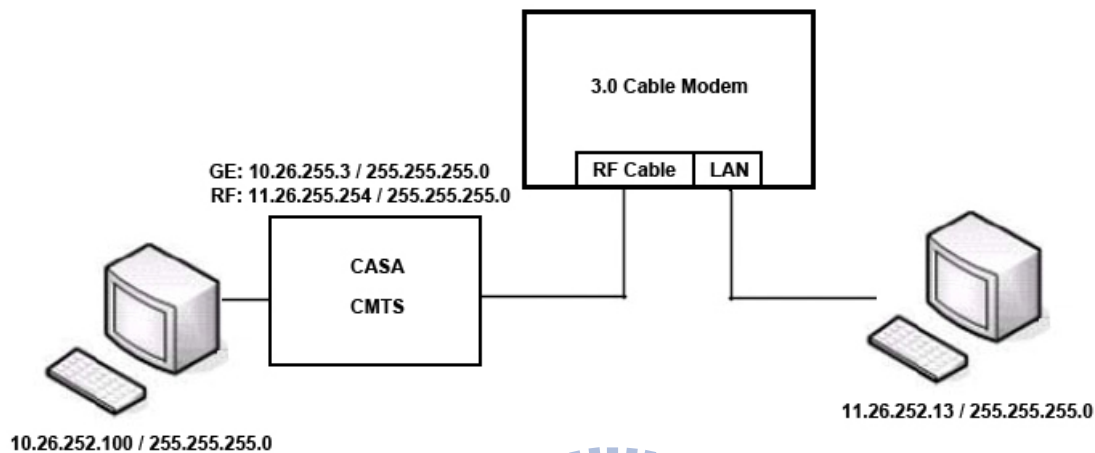


Figure 5-5 Test Topology of test 2

The **Figure 5-5** presents a clear illustration for this experiment. We use both this simple environment and the software IPERF to verify the system performance. Owing to this experiment includes throughput values of downstream test and upstream test, these two PCs can be server side or client side, it depends on the test requirement. In this topology, we can see the clear data path. No matter what the data send from the PC in the CMTS side or from the LAN port side, there are no interference in this topology. After all, by comparing the packet loss rate of the DUT (which has original factory data) with the DUT with recovered data , we can say that the calibration table is recovered from this system because of the cable modems have the same performance.

- **Test Commands**

After we setup the static IP address for **the PC behind the CMTS** and the **PC behind the Cable modem**. We use the following command to do this test.

Client: iperf -c [server ip] -i 10 -b [bandwidth] -t 60

Server: iperf -s -i 10 -u -f M

- **Test Result**

There are the results which we present in **Table 5-3 Comparison of UDP Downstream test** and in **Table 5-4 Comparison of UDP Upstream test**, this is a two-way test. It presents the packet loss rates are very close no matter what the data flow direction is. These bandwidth settings are in order to prove that the node with recovered calibration data has good ability of tuner. It can work fine just as never lost the calibration data. The packet loss rate is not increased, the subject reaches the best throughput whatever the limit of bandwidth is.

Bandwidth	Node with recovered data Packet Loss Rate	Node with original data Packet Loss Rate
11 Mbps	0.0053%	0.0053%
20 Mbps	0.013%	0.012%
40 Mbps	0.093%	0.091%
52 Mbps	0.045%	0.045%
54 Mbps	0.14%	0.13%

Table 5-3 Comparison of UDP Downstream test

Bandwidth	Node with recovered data Packet Loss Rate	Node with original data Packet Loss Rate
11 Mbps	0.0018%	0.0018%
50 Mbps	0%	0%
52 Mbps	0%	0%
54 Mbps	0.51%	0.50%

Table 5-4 Comparison of UDP Upstream test

5.2 Comparison

In this industry, the common method of handling a mass-produced cable modem which lost the unique data is simply recall the problematic product or just replace a new one in client side. Recently, we do not find any solution for recovery cable modem system which is comparable with our system. The system we proposed is based on event-driven methodology over traditional HFC network. The original purpose in our beginning idea is to enhance the stability of mass-produced cable modem. Moreover, this system can monitor the specified events to do distribute and recover. This kind of methodology reduce the recall rate of the mass production cable modem. In order to be more suitable to the practical usage of the cable modem, we investigate the cable modem use habit in reality.

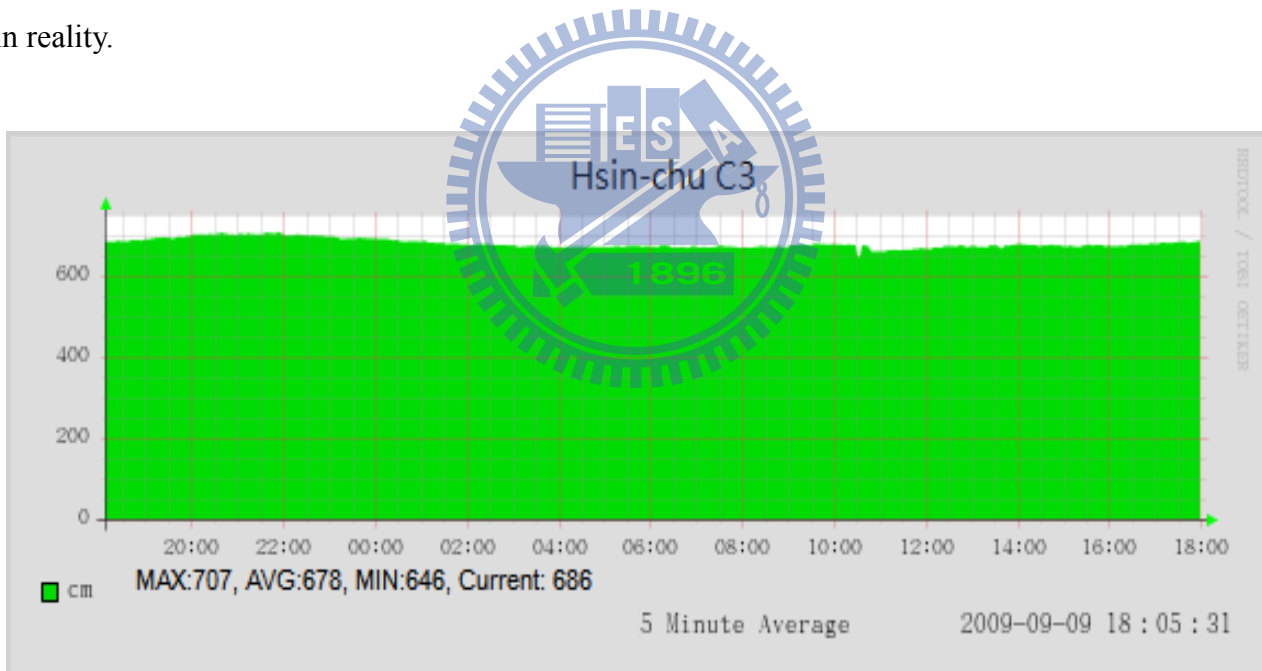


Figure 5-6 MRTG Daily Chart

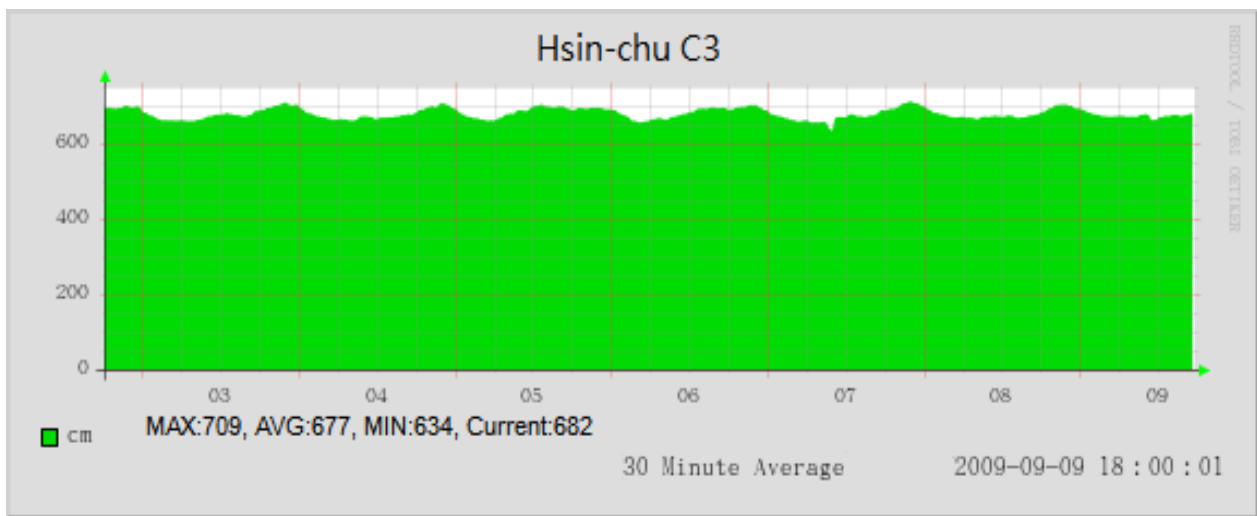


Figure 5-7 MRTG Weekly Chart

For business issue, we erase the name of the MSO on the **Figure 5-6** and **Figure 5-7**. We can see a smooth curve in **Figure 5-6**, that is the statistics for official cable modem online numbers, and it is counted per 5 minutes. According to this curve, we can say that there is no big change of online amount. About the **Figure 5-7**, this is an official cable modem online statistics too, and the time range is seven months. The total numbers of the sample is 709 , and the minimum online cable modem amount is 634. The average number is 677. And then we have the number larger than 0.89 as the probability for each cable modem which is in online status.

As we mentioned in above section, the generated blocks are distributed to six receiver cable modems. According to our demonstration, the source cable modem needs to receive it's generated blocks from any two receiver cable modems of these six DUT successfully. In our design, the source cable modem must has three to four linear independent blocks, which are enough for the source cable modem to restore the original copy of the factory data. In the other word, the recovery operation for the source cable modem which had already distributed the generated blocks into the

receiver DUT, the failure rate is $(0.11)^5$. However, according to practical experiment, the probability of a cable modem that lose the factory data can up to 2% in some case. We believe this system can be an effective method of enhancing the stability. For easy comprehension, we arrange the following table about our system to compare with the traditional method which we used in this industry.

As the end of this section, in **Table 5-5**, we compared the CAROL system with the tradition method which we used to maintain the factory data. There is no doubt the CAROL system will succeed in online maintain job. We do not have to re-install the product for this kind of issue. Of course, online process always faster then recall. We believe this system can reduces the cost effectively and makes the product more reliable.

	CAROL system	Traditional method
Way to Recover Data	Online process	Manually process
Velocity	Fast	Slow
Architecture	Based on NC	Client / MSO
Software	Customized by demands	None
Feature	Reducing the number of recall product	Low entrance hurdle, everyone know how to do

Table 5-5 Comparison with CAROL system and Traditional method

6 Future works and Conclusion

6.1 Future works

In this scheme, for theoretical improvement, we shall settle a intelligent diagnostic tool into our system. The future work can add the condition filter and the node tracer. About the idea of the condition filter, because of the difference between one environment and another, we can make this system more efficiently by add correct variable to let the cable modem avoids unnecessary duplication. About the idea of the node tracer, we like to build a recorder for the node behavior. In this case, the system can trace the online time of the each cable modem. In the other hand, we shall build a reporter for MSO. So that the MSO can classify the cable modems by the behavior. The MSO can mark the cable modem that always do power cycle or not usually online, these kind of nodes were marked as low priority. We can install this information for each cable modem while the cable modem trying to get online. By learning this information, the cable modems had to detour the low priority nodes. And then the successful recover rate might be increased.

6.2 Conclusion

The thesis proposes the CAROL system to solve the problem that the cable modems suffered from loss their unique factory data occasionally, and it's hard to restore the factory data back. We have established the recover service for the HFC network and submit a new solution for inserting this system into the cable modem. In order to preserve the advantage of protecting the factory data of the cable modem, building this system via P2P architecture is a good solution for distributing the factory data of cable modems from vendor to clients. The features of the Network Coding that we

used just meet our demands that reduce the repairing time for cable modem which lost it's unique factory data and making this kind of mass-produced product become more reliable. Although we do not have much devices for the experiment and we know the P2P network needs numerous nodes to keep the network operation steadily. However, we design the scenario demonstration in previous chapters to present the theoretical possibility. As the result we can see in past chapter, it is a good solution for this issue.



7 References

- [1] The Organic Computing page. Retrieved April 25, 2009, from <http://www.organic-computing.org/index.html>
- [2] Wikipedia(n. d.). Organic Computing Retrieved April 25, 2009, from http://en.wikipedia.org/wiki/Organic_computing
- [3] Hartmut Schmeck : Organic Computing – A New Vision for Distributed Embedded Systems , Institute AIFB, University of Karlsruhe, Germany , Computer Society, IEEE.
- [4] Wikipedia(n. d.). Ubiquitous computing. Retrieved May 2, 2009, from http://en.wikipedia.org/wiki/Ubiquitous_computing
- [5] V Rangel and R Edwards : Performance Analysis and Optimisation of the Digital Video Broadcasting/Digital Audio Visual Council Cable Modem Protocol for the Delivery of Isochronous Streams , IEEE Global Telecommunications Conference, 2001.
- [6] DOCSIS® Specifications — DOCSIS 3.0 Interface. Retrieved May 1, 2009, from <http://www.cablemodem.com/specifications/specifications30.html>
- [7] Upstream Calibration and Power Scheme Application Note for TNET950 v1.0 : Texas Instruments, September , 2008
- [8] Hartmut Schmeck, "Organic Computing - A New Vision for Distributed Embedded Systems," isorc, pp.201-203, Eighth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'05), 2005
- [9] C. Müller-Schloer. Organic Computing - on the feasibility of controlled emergence. In CODES + ISSS 2004 Proceedings, pages 2–5. ACM Press, September 8-10 2004.ISE
- [10] Alexander Sinsel, Juliane Claus, Thomas Ludwig: The Semanto-Pragmatic Perspective of Information Processing - a Dynamical System Approach to System Design. (submitted to SASO 2009, San Francisco, California, September 14-18, 2009)

- [11] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Inform. Theory*, vol. IT-46, pp. 1204–1216, 2000.
- [12] DS Lun, M. M' edard, R. Koetter, and M. Effros, "Further results on coding for reliable communication over packet networks," *IEEE International Symposium on Information Theory (ISIT)*, pp. 1848–1852, 2005.
- [13] T. Ho, M. M' edard, R. Koetter, D.R. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *IEEE Transactions on Information Theory*, vol. 52, no. 10, pp. 4413–4430, October 2004.
- [14] Ralf Koetter and Muriel M' edard, "An algebraic approach to network coding," *IEEE/ACM Trans. Netw.*, vol. 11, no. 5, pp. 782–795, 2003.
- [15] Christos Gkantsidis and Pablo Rodriguez Rodriguez, "Network Coding for Large Scale Content Distribution", *IEEE INFOCOM*, 2005
- [16] Philip A. Chou, Yunnan Wue, and Kamal Jain, "Practical Network Coding", *Proceedings of the Annual Allerton Conference*, 2003
- [17] Sachin Katti, Hariharan Rahul, Wenjun Hu, Dina Katabi, Muriel Me'dard and Jon Crowcroft, "XORs in the air: practical wireless network coding", *IEEE/ACM Transactions on Networking (TON)*, 2008
- [18] C.K. Ngai and R.W. Yeung, "Network coding gain of combination networks" , *IEEE Information Theory Workshop*, Oct. 2004.
- [19] A.G. Dimakis, P.B. Godfrey, M.J.Wainwright and K. Ramchandran, "Network coding for distributed storage systems," *Proc. IEEE INFOCOM 2007*, May. 2007.
- [20] NJA Harvey, DR Karger, K Murota, "Deterministic network coding by matrix completion" - *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, 2005 – portal.acm.org.
- [21] Y. Zhu, B.C. Li and J. Guo, "Multicast with network coding in application-layer overlay networks" *IEEE Journal on Selected Areas in Communication*, Sep. 2004
- [22] Jesus Garcia and Michael J. Schulte, "A COMBINED 16-BIT BINARY AND DUAL GALOIS FIELD MULTIPLIER" *Proceedings of the IEEE Workshop*, 2002

- [23] Modeling Topology of Large Internetworks. Retrieved May 17, 2010, from <http://www.cc.gatech.edu/projects/gtitm/>
- [24] K. Bhattad, N. Ratnakar, R. Koetter and K.R. Narayanan, “Minimal network coding for multicast,” Proc. IEEE International Symposium on Information Theory , 2005.
- [25] Y. Wu, A. G. Dimakis, and K. Ramchandran, “Deterministic regenerating codes for distributed storage,” in Allerton Conference on Control, Computing, and Communication, (Urbana-Champaign, IL), September 2007.
- [26] Finite field arithmetic, From Wikipedia. Retrieved May 27, 2009, from http://en.wikipedia.org/wiki/Finite_field_arithmetic
- [27] AES' Galois field. Retrieved May 27, 2009, from <http://www.samiam.org/>

