DIPLOMA THESIS

Analyzing the Failover Behavior of an Ethernet Network by the use of Link Aggregation

ausgeführt zum Zwecke der Erlangung des akademischen Grades
eines Diplom-Ingenieurs unter der Leitung von

o. Univ. Prof. Dipl.-Ing. Dr. Tech. DIETMAR DIETRICH
und
Dipl.-Ing. (FH) HEIMO ZEILINGER
als verantwortlich mitwirkendem Universiätsassistenten am
Institutesnummer: 384
Institute of Computer Technology

eingereicht an der Technischen Universität Wien
Fakultät für Elektrotechnik und Informationstechnik

von

ALI REZA SIADAT KHOO
Matrikelnummer: 0227332
Kornhäuselgasse 9/8/26B
1200 Vienna, Austria

Vienna, October 2008 _____

Abstract

Due to the spread of Internet and associated cost reduction of network components, VoIP has found its way to prior areas of circuit switched services. The recent developments show a trend towards VoIP in public safety communication systems. The thesis at hand focuses on the analysis of mechanisms, which improve the availability of the Ethernet networks and a reduction of their convergence time. Therefore, the link aggregation mechanism that has been specified for Layer 2 was chosen from various technologies and mechanisms. Regarding to the asked requirements, the ability of the link aggregation has been explored to utilize in safety critical voice communication systems. Accordingly, this mechanism has been configured in a pre-specified network architecture to investigate the failover behavior of the entire system. The resulted convergence time and the availability of the Ethernet network have been probed and analyzed. The achieved results show a high availability of the network and fast failover recovery in case of link failure. However, due to the restrictions of worst-case scenarios, the convergence time has been located more than one minute.

Kurzfassung

Aufgrund der Ausbreitung des Internets und der damit verbundenen fallenden Kosten von Netzwerkkomponenten nimmt VoIP eine wichtige Rolle, in Bereichen die zuvor von Circuit-Switched Diensten dominiert wurden, ein. Aktuelle Entwicklungen zeigen einen Trend in Richtung der Verwendung von VoIP in sicherheitskritischen Kommunikationssystemen. Der Schwerpunkt dieser Arbeit liegt auf der Untersuchung von Mechanismen, die die Verfügbarkeit von Ethernet-Netzwerken erhöhen und deren Konvergenzzeit verringen. Dafür wird aus verschiedenen Technologien und Mechanismen die Link-Aggregation gewählt. Dieser Mechanismus wird bezüglich der gestellten Anforderungen überprüft um eine Verwendung für sicherheitskritische Sprachkommunikationen zu ermöglichen. Die Link-Aggregation wurde in einer spezifizierten Netzwerkarchitektur konfiguriert und hinsichtlich der Fehlerbehandelung untersucht. Die daraus resultierende Konvergenzzeit und Verfügbarkeit des gesamten Netzes wird überprüft und analysiert. Daraus resultierende Ergebnisse zeigen eine hohe Verfügbarkeit des Netzes. Allerdings gibt es Einschränkungen in Worst-Case Szenarios in denen die Konvergenzzeit über einer Minute liegt.

Acknowledgements

First I would like to take this opportunity to thank everyone who has guided me during the last year to complete this thesis, specially my diploma supervisor Heimo Zeilinger. He was not only my supervisor, but also he has respected my situations and motivated me to do better this job. I would thank also the members of *VOLARE* group Edgar Holleis, Berndt Sevcik and Thomas Turek who were always cooperative and helpful for me in difficult occasions. I would also thank the Institute of Computer Technology that provided me a satisfactory environment and work place to complete this scientific work.

This is an appropriate situation that I thank specially my wife who has always supported me in my whole post-graduate studies. She always believes in me and helps me in happy and sad times. She has served me and prepared me a calm and reposed place to study. I would like to thank also my lovely parents who always supported me, motivated me and encouraged me throughout my academic program. Without my family I could not be successful in my studies and I dedicate this academic degree to them.

Ali Reza Siadat Khoo

October 2008

Table of Contests

Chapter 1 Introduction

The importance of Voice over Internet Protocol (VoIP) in the telecommunication networks has increased in the last decade. Not only the voice quality is relevant for these networks, but also the correctness of the transferred voice data, which influences on the voice quality, can be named as essential point in this area. However, in some cases, the transmitted information is very important, and the data loss can lead to unexpected events. Therefore, the correctness of transmission, the safety and security of this communication network must be guaranteed before building the related telecommunication network. Clearly, safety critical communication systems, such as Air Traffic Control systems (ATC) and/or the emergency call centers, are affected by this working area.

In the new generation of these public safety communication systems, the new technology, VoIP will be used. VoIP is a technology to deliver the voice data not over the typical circuit-switched telecommunication networks, but over Ethernet networks. The technologies, which use Internet Protocol for voice communication, are able to reduce the cost, support the innovations and improve the access of new communication services. VoIP can bypass whole or at least a part of Public Switched Telephone Networks (PSTN). The communication can be occurred computer-to-computer, phone-to-computer and phone-to-phone. Clearly, in traditional ATC systems, there are three networks, one for data, packets and information, one telephone network and finally one network for radio communications. By using VoIP, all these three networks can be aggregated into only one Internet Protocol network and some benefits are served for users and communication providers. People may transmit their voice data over more efficient way with definitely lower cost. [WWD02].

The one of VoIP disadvantages is the lack of a safety mechanism to verify the packet reception. Clearly, the packet loss and transmitted data disordering cannot be performed in this protocol. This protocol like other protocols such as UDP (User Datagram Protocol) and RTP (Real-Time Transport Protocol), which are defined to transfer the data stream over IP-based networks, does not dispose any checksum mechanisms to provide the safety for the packet transmission. The VoIP applications have a real-time nature. Moreover, the voice data cannot be dropped and arrived in a scrambled order. Accordingly, the VoIP applications have to be run on the high quality, very reliable and highly available network.

The Ethernet network, which is standardized in the IEEE 802.3 [IEEE8023], forms the base of the network layer. The big amount of technologies and mechanisms, which are developed for utilizing in Ethernet networks, has made this standard as the favorite technology for the network developers and designers. This work focuses on the convergence time and the availability of the Ethernet networks to approach an appropriate environment for safety critical voice communication systems.

A system will be called available, when the users can request a new appointment, continue existing work and receive the useful answer from the service provider in a specified period of time. When the system cannot answer the requests in a certain time, the system is unavailable. The time that the system is unavailable will be called system downtime. The related definitions to this work will be explained in the next chapter.

Availability in %	Downtime per year	Downtime per week	Downtime per day
99,9	8,75 hours	43,2 minutes	10,1 minutes
99,99	52,6 minutes	4,32 minutes	1,01 minutes
99,999	5,26 minutes	25,9 seconds	6,05 seconds
99,9999	31,5 seconds	2,59 seconds	0,605 seconds

Table 1-1: Classifying the availability of the end-system for 7/24 working time

The availability percentage of Ethernet networks is categorized into different levels. For various services such as ATC the special availability is required. Therefore, it is important for the network developers and service providers to specify the rate of systems and networks availability. Table 1-1 shows such classification of different systems in associating of their specification and expecting system downtimes per a year, a week and a day. This table illustrates that the downtime of a system can be less than one hour per year, where the safety critical voice communication systems will be assumed as the highly available. These values have been obtained from the availability calculating of a system, which provides services 7 day a week and 24 hours a day.

The availability classification of the safety critical voice communication systems such as ATC and emergency call centers must be performed for five nines (99,999%). The safety critical appointments of a VoIP application cannot approve to be down 5 minutes for each downtime, although this downtime occurs once per year. So by a failure occurrence the system must be recovered very quickly after the faulty event. The time difference from the point that a system is located in a faulty state up to the time that a system is available and operable again is called as system convergence time. The voice communication systems in the public safety area have to show a convergence time as low as possible.

In consideration to the use of highly available Ethernet networks for the safety critical communication systems, some technologies and mechanisms are developed to approach the advantages of high performance in such networks. The aim of this project is to investigate on the failover behavior of an Ethernet network, which is used in a VoIP end-system for such safety critical communication networks.

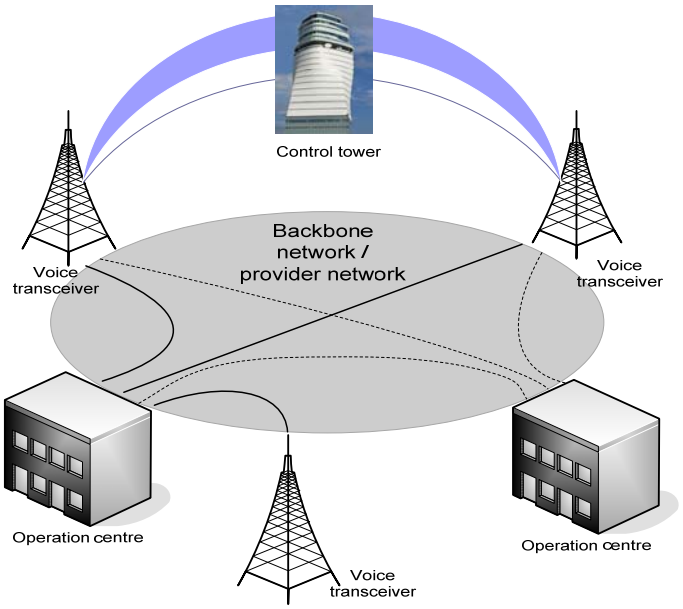


Figure 1-1: General perspective of ATC network

The ATC systems are used to improve the public safety. Figure 1.1 depicts a general perspective of ATC networks that have been previously mentioned. This figure illustrates the located backbone network between various stations. The local IP network is located in control tower or in each operation centre. If the air controller wants to communicate with other control tower over Internet network, the system packetizes the voice and transmits it from the LAN and also the provider network to another end-station. These networks have to provide highly available point-to-point communication between each transceiver, control tower and also the operation center. These point-to-point connections are shown in Figure 1-1. The packet loss can cause critical results. Due to the possibly failure occurrence in this network the convergence time must be kept in appropriate safety area. The availability and the convergence time of a system in this LAN at each end-system is focused and investigated in this thesis.

However, reducing the downtime for system outages helps to approach higher availability. Therefore, the system has to avoid the outages and integrate an alternative way to bypass the failures. It is not always the solution to resolve the system failure but it can be the beginning of the way to decrease the error recovery time. In last decades many technologies,

mechanisms and protocols have been developed, implemented and designed to approach this purpose.

This thesis will explore and compare diverse high availability solutions such as protocols and mechanisms, which are developed to use in Ethernet networks. The important parameter, which will be investigated in this work, is the failover time of these protocols. Due to the facts explained above, the convergence time in networks must be performed in sub-seconds to prevent the reducing of VoIP quality from any deficiencies. The failover time less than a second means that the error detection and error recovery time will be ensured within tens and hundreds of milliseconds. In consideration to the ability of many protocols that are developed and designed for the Ethernet networks, it can be guaranteed that such technologies exist, but they have not probably specified for utilizing in safety critical voice communication systems. Therefore, the assessments of the existed appropriated protocols must be investigated to find, if they can provide the asked requirements.

Furthermore, network architecture must be designed in this work to create an environment to put the proposed solution into the practice and analyze the capability of this mechanism. This network environment must provide the redundancy, either in functionality and ability of the network and also component redundancy. This environment can be used for the end-systems, which used for voice communication systems. The network-level redundancy and also the device-level redundancy will be designed for the end-system. Accordingly, the behavior of each node in case of failure will be probed and analyzed.

Now, it can be said that the analyzing the failover behavior of an Ethernet network is the most important issue in this thesis. However, the measurement of the failover time in the network requires getting knowledge about the operating modes and the capability of the systems and devices used here in the end-system. These cognitions can be achieved through investigating precisely on hardware, software, firmware and also the operating systems. The responds for questions such as, how a packet will be generated and detected in the computer systems and how the packet progression within the network devices and stations is, must be found after reading this document. Clearly, the systems, which are involved in transmitting, forwarding and reception of the transferred information, will be studied adequately to discover the behavior of them to analyze the time delays for a frame in each module and investigate the influences of these delays on the measured convergence time.

Moreover, the failover behavior of the proposed availability solution will be explored and analyzed. The convergence time of the network is measured and also the achieved results will be probed. Regarding to the discussions, it can be explored, which parameters and situations can influence the results and finally in which basic construction the protocol can be performed to use in an Ethernet network. This thesis tries to assess the behavior of the network in general occurring cases.

After introducing the work, the structure of this document is given here. The concepts, which will be used in this thesis and also have the special significations for the safety critical voice

communication system, will be declared. Therefore, they will be explained exact in the way that relates them to this topic. The third chapter comprises the technologies and protocols, which have been implemented and developed to improve the availability of an Ethernet network. These protocols work on the Layer 2 and 3 of the OSI reference model. At the end of this chapter the proposed possibly solution is introduced completely. In Chapter 4 the tools and applications will be described, which are provided and utilized in this thesis to approach the project goals. The entire test cases, which are the main constructions for the analysis, will be explained and discussed in this chapter as well. Finally some prior essential analysis will be given to ease the execution of the tests and also the better realizing of the test results. The results and analysis are given in Chapter 5. The evaluation of the network high availability and the satisfactory convergence time of the proposed solution will be assessed through investigation of achieved results.

Chapter 2 Highly Available Applications and Communication Systems

At the beginning of this diploma thesis, the importance of the availability in an Ethernet network has been declared. Some basic definitions are related to this work and they will be shortly mentioned. Not only these concepts should be explained, but also the related work will be discussed.

2.1 Basic Definition

In consideration to Air Traffic Control (ATC) and Voice Communication Systems (VCS), the concepts of reliability, availability, maintainability, safety, serviceability and redundancy are emphasized according to their relationships to hardware components, software application and operating systems.

The definitions and concepts declared here are basically interesting for this work, because they will be utilized in the further chapters. These concepts are strongly associated to the aim of this project, either direct or indirect. The definitions will be given with the respect to voice communication systems. Therefore, only the required perspective will be explained.

Types of Failure

At first, it is better to distinguish between three definitions- Failure, Fault and Error. The IEEE 610.12-1990 standard defines the term Failure:

> *"Failure is the inability of a system to perform its required functions within specified performance requirements." [Iee90]*

The failure is defined, if the system does not deliver the expected service. Failures are incorrect results or unexpected behavior with respect to the specification. Failure causes a fault. There are two categories of failures- Transient failures and persistent failures. [Sol02, p. 4]

The transient failures are also called temporary failures. They will be found at the hardware environment or starved part of a program in a software application. However these failures occur rarely at software side. [Sol02, p. 4]

The permanent failures occur frequently or more than once. They can be a development or design failure. They are conveniently repairable but in the domain of programming. The system administrator can analyze and detect them. Then a program runs to bypass this failure. [Kop02, p. 120]

An Error is the discrepancy between a measured or computed value or condition and the theoretically correct value or condition. In other words, an error is an incorrect internal state, also an unintended state. If the error exists for a short time and is vanished without any repair action it is called a *transient Error*. If the error is removed by a repair action it is named as *permanent Error*.

A fault or defect is an abnormal condition that may cause the loss of capability of a functional unit to perform a required function. A fault causes an error and thus causes indirectly a failure.

A system fault can be caused by external or internal elements. Specification and design errors can be enumerated as internal factors and electromagnetic interference could be counted as external factors. Fault is referred to as an unusual or uncommon condition in a system, a defect in components or an unacceptable situation that leads to a failure. However the word fault has different definitions in other branches, in this document it is concerned only with Computer Science, Networks and Information technology. [Kop02, p. 121]

An error occurs, when a system fault causes an unwanted or false output. It will occur, when the system reaches to an exceptional condition or an unexpected interference. If the error causes a system crash or reboots then the error becomes a failure. [Var00, AGH01]

System Outage

The time that a system is not up and cannot serve the users called as downtimes. Generally the system outages (*Downtime*) are categorized in two divisions- planned and unplanned system outages. [Sol02, p. 2]

Planned system outages or maintenances are events, such as, backup, repair or upgrade of a system in a scheduled occurrence.

Unplanned system downtimes are uncontrollable events, associated to hardware or software components' faults. These kinds of outages harm the system and investments, sometimes requiring more time and energy to find and remove them than expected, in other words, they cause safety and security problems and are not economical for the system.

Failover

The term 'Failover' is defined in Business and IT continuity as below.

> *"Failover is the capability to switch over automatically to a redundant or standby computer server, system, or network upon the failure or abnormal termination of the previously active server, system, or network. Failover happens without human intervention and generally without warning, unlike switchover."* [ENI08, p. 149]

The time a system needs to execute the failover operation is entitled as "failover time". To improve the failover behavior of an Ethernet network is the subject of this diploma thesis. The concepts "*Convergence and Convergence Time*", which are used in this document, have the similar significance as the failover and failover time.

Availability

The fraction of time a system is available is expressed as availability and is influenced by the system's recovery time and the *Mean Time between Failure* [SV97]. The concept of MTBF describes the time between each two consecutive failures of a system. In [CPR05] the MTBF is written as a general definition, which does not have an especial meaning for air traffic control system. The goal of this project is to decrease the failover time in an Ethernet network.

Mean Time to Repair (MTTR) describes the required time to repair, replace a component and recover a system or restore service after a failure. The modern air traffic communication systems managements have to replace a board or a module within 15 minutes [CPR05]. MTTR depends on the service-friendliness of the entire system, which is determined through system architecture and used components.

Availability is the time, which the system is useful and available. The system downtime that is explained in previous subsection influenced the measurement of availability. In common systems with 8 seconds downtime, the systems with 99,99% availability are categorized as highly available. However, the availability of Voice communication systems must be exceeded to 99,999% [CRP03]. The use of modern technologies makes this rate realistic. The voice communication system developers want to produce a system with 99,99999% availability but such quote cannot be achieved with today's technologies. To achieve the required availability standby and backup systems perform important parts of system design and configuration. [CRP03]

For measuring the availability the designers should define exactly the functional conditions and operable states of components. Another important issue is the setting of a correct and useful time period. Clearly, two systems with different parameters and properties, which are used in different areas, cannot be compared and categorized in a same availability structures. [Sol02, p. 6]

Reliability

Reliability is an important parameter for measuring the availability but it doesn't mean the Availability itself. Reliability shows, how often a system or a component fails.

> *The IEEE standard 1366 defines Reliability as; "The ability of a system or a component to perform its required functions under state conditions for specified period of time."* [Iee90]

Reliability is associated to MTBF and failure rate directly. Reliability is improved when the time interval of spanning separate failures is extended. In the voice communication systems the reliability is associated of the MTBF to the Printed Circuits Boards (PCB). The reliability of a VCS depends on the development of the hardware and also the architecture design. Therefore, the precise requirements implementation and verification of the system design can improve the reliability. [CPR05]

The development of a system with high reliability is typically expensive and difficult. However, in some environment like aerospace and military, designers should achieve the acceptable reliability, because any failures can cause the critical event. Although providing the system with an acceptable continuous operation with repairing the faults and preventing their propagation is more economical. Numerous system designers and developers believe that a system without failures does not exist but they know that there are only systems that have not had failures yet.

The two above explained concepts have two different interpretations and they should not be misunderstood. The difference depends on the costumers' situation. In some cases, reliability has preference to availability and somewhere vice versa. [Var00]

Security

Security is the ability of the system to prevent unauthorized access to information or services [Kop02, p. 120]. Security management provides the required functions to perform adequate security of a VCS by unauthorized users. ATC systems must provide physical-, system- and public network security. All areas, which are used in a safety critical voice communication system, must be protected. The system has to ensure the prevention of unauthorized access and also the functionality of the public network must be restricted to unauthorized users and devices.

Safety

After all these concepts and definitions that have been explained, it is important to define the term "Safety". The main purpose of this work is to increase the safety in critical communication systems. In voice communication systems the safety critical applications and devices must be utilized.

Safety is the state of being safe. This condition is protected from consequences of any errors, damages and failures. In general all physical and electrical menaces are prevented in a safe system. In VCS, the system must achieve all requirements of safety information technology equipments, “EN 60950” [CRP03].

Network Redundancy

The redundancy in a computer system can be defined when the additional components have been integrated into the system and also during the operation, the system can detect the fault and possibly tolerate it. A redundant network can be developed if the redundant links and equipments as well as protocols and techniques have been used to provide this redundancy in the network. [Sol02, p. 6]

A redundant Ethernet network consists of the two redundancy levels. The device-level redundancy, which provides the failover capability in the particular network devices by some defined mechanisms. The second level is the network-level redundancy, which provides the failover capability in the case of network device failure through defined mechanisms and network protocols. [Red04, p.18]

The network redundancy in this thesis is not only for equipments and components, but also for the functionality of the involved components. Clearly, two redundant network devices have to provide the complete services in case of network failure. Therefore, the protocols and technologies provide specified operation redundancy functionality such as dynamic routing and load sharing in the network.

The redundancy will be used to increase availability. All components or the entire system can be made redundant. When the entire system is redundant it is called as Cluster. [Sol02, p. 6]

There are two kinds of Redundancy: [Sol02, p. 6, Kop02, p. 122]

- Active Redundancy: all components of a system will be used and they execute their tasks, but if a component fails another component takes over the failed component’s assignment.
- “Standby”-Redundancy: opposite to the previous kind, this is when the redundant parts are not active, when a component fails; the standby device assumes the assignment of failed component. However, in this case the redundant part can be defect. The standby redundancy is partitioned into cold and hot standby.

Fault Management

The Eurocontrol as below defines the fault management,

> *"Fault management functions enable the detection, isolation and correction of abnormal operation of the telecommunication network and its environment."* [CRP03]

The following set of functions can be named as the fault managements:

- Alarm Surveillance: it is the capability of the system to monitor the failures in near real time. The fault management assigns the fault root, causes and also the fault effects in the system, if a failure occurs.
- Fault location: it is the ability of the system to use the additional failure localization routines and thus the ability to improve the initial failure information.

This concept will not be used directly in this document, but it is important to know how faults can be managed in the safety critical voice communication systems, which are explained above.

2.2 Related Work

In last decades, development of highly available services has been focused. The traditional solutions have designed a complex management application to monitor multiple links. In addition they require multiple IP addresses. If a link fails, the complex management software has to begin the execution of specified tasks to perform a failover function. Such ways cannot guarantee an optimal failover time, especially in cases, where the system high availability is more important than normal and longer failover time can cause critical errors or serious damages [Yu05].

In [Yu05], the developers have investigated on STP and RSTP in this contest and their results on RSTP have been given. Chapter 3.1 investigates and compares STP and RSTP referring to the use for the end system. These protocols are used in a High Available (HA) server with two integrated network interface cards (NIC). The system has built on RSTP to approach 100% transparency to the applications and clients and improve the failover time less than one second. However, these protocols are useful in Ethernet network to improve higher availability, but the main purpose to define them is not to increase failover time. STP and RSTP are specified to prevent loops between network devices; switches and routers. Therefore these two protocols are not appropriate to achieve a sufficient failover time in each scenario. In other words, the highly available systems cannot rely only on these protocols to approach the required convergence time.

Moreover, if a failure has to be detected in the network, one solution is to run a heartbeat application to monitor the link status and also synchronization of ARP (Address Resolution Protocol) tables. However, this job has some deficiencies. Firstly, the most of heartbeat programs are commonly based on Internet Control Message Protocol (ICMP) but not all of them and the time interval can be set only to seconds. Therefore, these programs are not good

enough for applications using VoIP. Furthermore, in updating ARP for new interface, the server must inform all clients to flush the ARP table and to learn new MAC addresses. This process takes a few minutes, which is inapplicable in highly available IP networks. Finally, the last deficiency, if a network interface fails and the new interface operated as active interface, the IP address of this interface must remain as the same. Therefore, it requires binding the IP address for the new interface. Within the bind process the IP address sockets will be flushed and as a result all active application may be lost. [Yu05]

Other attempts in this area are given to reduce failover time in Ethernet networks and also many other researches are working to achieve this purpose. One of these researches is in the network products produced by Hirschmann Electronics GmbH & Co. This company applies a method by installing the HiPER Ring. The Network can be divided into rings and all these rings are interconnected to each other to provide the network redundancy. Every ring has a redundancy manager, which sends watchdog packets to the members of the ring in both directions every 100 ms. The redundancy manager is responsible to avoid the loops and also to activate and deactivate the related members and links. The tests show that this method is much better than RSTP and it is suitable for both industrial and official usage. RSTP is related to switch numbers in the network, which means if the number of switches in the network increases, it affects the failover time of the system. In opposition to this in Hirschmann products used HiPER Ring method, where the failover time is guaranteed less than 500 ms by using 50 switches, over 3000 km. [KS03]

Since the network availability today is more important than other Internet services, dynamically routing methods evolve some adequate results to predict the failover time of the networks. One of the main purposes of using the protocols like Border Gateway Protocol (BGP), Open Shortest Path First, and Enhanced IGRP (EIGRP) is to reduce failover time of Ethernet networks. The complex networks with increasing capability are very difficult to manage. Therefore, these dynamically routing protocols can improve the high availability of the network. In [Wil06], it has been focused on Enhanced Interior Gateway Routing Protocol (EIGRP) and this protocol has been examined. The tuning of dynamically routed Internet protocol networks is explained in the named publication, and a controlled and predictable failover during link instability is achieved.

One of the protocols that are used in networks is Stream Control Transmission Protocol (SCTP) that is defined in RFC 2960, RFC 4960 as well as RFC 3286 [SXS00, Ste07 and OY02]. This protocol is in the same layer as TCP and UDP in abstraction OSI-Model. The SCTP failover mechanism helps the developer to estimate the failover time between two SCTP endpoints. This mechanism indicates the management rules of packet transmission during the failover process. In [BFG07], the results of failover estimation in SCTP multihoming scenarios are given. Due to usability of these works in special scenarios, they are not executed to reduce failover in any Ethernet networks. They are mostly used for the calculation and not to decline the failover time. However, the reducing failover in any type of networks and communication systems is one of the main targets of network developers and

IT specialists. Using redundant network devices such as switches and servers are other approaches. In [BFG07, Wil06 and Yu05] has been shown how two parallel working servers can be used, where one of them is the primary server and the other one is backup server. By using Linux modules such as Operation Main Process (OMP) and Operation Communication Process (OCP) between two primary and backup servers, it is possible to increase service availability (serviceability) of a network and reduce failover time. In this work the UDP is used as data transfer protocol between primary and backup servers, because the TCP is more complicated and sometimes inefficient [WGXX06]. TCP has some complex checking and verifying assignments but the UDP is simpler and also faster. The UDP provides the minimum of protocol mechanism. It is transaction-oriented protocol and delivering and duplication of messages cannot be guaranteed. [Pos80]

Another related work to improve high availability in an Ethernet network is the IEEE standard 802.17. The Resilience Packet Ring (RPR IEEE 802.17) is used in metropolitan and wide area networks [IEEE80217]. It is new ring topology network architecture. A Ring is a multiple point-to-point bidirectional connections between the stations. This protocol is provided the resilience advantage. In other words, a packet can be delivered to the destination even with link failure existence. If a device in the ring receives a packet, which is not recognized to it, it forwards the packet to the next station. Otherwise, if the packet designated to the arrival station it will be received completely. A timeout for each packet in the ring can prevent the circulating of packet endless in the ring. In this protocol only a single MAC address is provided for all connected mediums in the ring. Therefore, the use of interface to this network is similar. Three-level class-based traffic priority scheme are provided in RPR. The low-latency low-jitter class, the predictable latency and jitter and finally the best effort transport class classify these schemes. The so-called fairness algorithm is used for interconnection among the devices in ring. Some other benefits can be named as resilience and bridging. [DYGN04]

Each device in the ring can send to and receive from outside world. In IEEE 802.17 special terms are designed to prevent the latency for sending and receiving packets. The RPR protocol uses the statistical multiplexing to provide the fairness and fast restoration and improve the bandwidth. If the receive removed a packet from the ring, the bandwidth of this frame will be back to the network for the use of other senders. The frame latency in this protocol is less than 1 millisecond for 16 devices, although it depends on participated devices in the ring topology [DYGN04]. By sending a topology discovery operation message to other devices in the ring and providing a database about the device states in the ring, the ability to reduce the failover time is increased. From detecting a new device in the ring to normal operation takes from 128 to 1024 ms [Els04]. This protocol has the same property as the Link Aggregation (see 3.5) to receive the frames. The frame ordering is remained as the same as the sent order. If a link or a station fails a topology is run to recover the stations state. During this algorithm all transmit and receive assignments are stopped. The restoration time is below 50 ms. Big advantage of the RPR is its independence of physical layer. Therefore, this protocol can be used with optical fiber, fast and Gigabit Ethernet. [DYGN04, Els04]

Another technology that has been designed to improve the speed of Internet functions, thus reduce the convergence functions is the SONET (Synchronous Optical Network). SONET is an American published version. Due to the increasing of the network operation and management systems ability and capacity, in last 15 years this standard becomes common technology for optical data transmission. This technology was combined with another technology called SDH (Synchronous Data Hierarchy). Current SONET/SDH interface speeds range from 51 Mbit/s to 10 Gbit/s [MYM02]. However this technology is the main solution to transfer data over fiber links but it has multiple limitations. It has not flexible bandwidth. The fixed bandwidth is defined over a rigid rate hierarchy. In addition to this, a fine granularity to revise the all potential of clients is missed. Finally, the configuring the transport paths takes more time as expected, because SONET/SDH nodes have limited network management functionalities. Therefore various technologies have been developed which are based on SONET/SDH for transmission data over fiber links to revise this standard. [MYM02]

Chapter 3 Protocols and Standards Used to Achieve Highly Available Network

In this chapter, the common protocols and standards will be explained, which are used to increase the availability of an Ethernet network in each Data-Link Layer and Network Layer, also reducing the failover time of a network. These protocols and standards have not only been specified to increase high availability, but also they are assigned for other purposes. However, they can be used to make an Ethernet network redundant and improve the ability of network to reduce the failover time.

3.1 Spanning Tree Algorithms

In Layer-2, switches and bridges use the media access control (MAC) address to make a decision about forwarding the packets. The assignment of defining the active path in case of failure, are accepted in this layer over spanning tree or link aggregation (see 3.5) [Sch03].

Spanning Tree Protocol

The Spanning Tree Protocol defined as the IEEE 802.1d standard on Layer 2 of OSI-Model and CSMA/CD layer architecture. It is a link management mechanism that prevents packet duplication in the network loops. In an Ethernet network it must exists only one active path between two stations. If in a network, there is more than one active path, it is possible to be made a network loop in this network whereas a loop is a potential of duplicated frames that will be forwarded by network components.

The STP provides path redundancy by defining a tree, which spans all switches in an extended network, but these switches must support the STP features and have the option to configure the STP. STP designates the redundant data paths. If a component is not reachable or the status of any components is changed, the spanning tree algorithm redefines the data path and reconfigures the spanning tree structure [Sch03].

All switches in an extended network exchange the data with each other to gather the information of other switches. This message exchange is done through the Bridge Protocol Data Units (BPDU) and results the election of a unique root switch and designated switch for every switched LAN segment and removing loops in switched network. The BPDU comprises information about switch and its ports. The information consists of switch and ports priority, port cost and finally switch and port media access control (MAC) addresses [Cis97].

Each switch has a unique MAC address as switch identifier, and every bridge has a bridge-identifier, as well as all ports of a bridge. If an active path failure has been detected in the network, the spanning tree algorithm blocks all ports and only configuration frames can be transmitted between bridges (BPDU). At the beginning all bridges will be assumed as root-bridges but after exchanging BPDUs the bridge with lowest priority becomes the root-bridge and in case of same priority the bridge with smallest MAC address becomes the root-bridge. The root port will be defined by lowest path cost. In other words the shortest and fastest path becomes the root port. The designate-port will be also determined by path cost, thus the nearest port to root port becomes designated port. After appointment of root port and designated ports all other ports that cause a loop will be blocked. [Sch03, Cis97]

Figure 3-1 depicts how the ports will be determined in this protocol and the ports, which cause a loop, are blocked. The switch “A” is the Root-Bridge. There is one root port in each bridge, which connects that bridge to Root-Bridge.

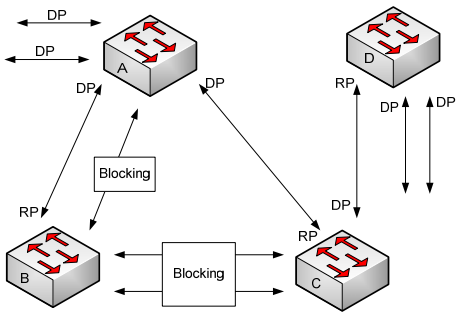


Figure 3-1: Ports appointment in STP

Rapid Spanning Tree Protocol

The STP is not an optimal solution for redundant networks in case of failures. The reconfiguration in STP takes about one minute that is a long time for this work. Therefore, the Cisco implements a new protocol called Rapid Spanning Tree (RSTP), and IEEE standardized it as IEEE 802.1w. It is a further development of STP with an advanced BPDU-Format. Therefore, it has faster convergence by reconfiguration.

In this standard by adding a discarding status, the developers abstained of Blocking, Listening and Disable states. The assignment of root and designated ports remain unchanged, but the Blocking-Port is partitioned to Backup-Port and Alternative-Port. The Alternate-port blocks the BPDUs of other bridges and provide an alternative path to Root-Bridge in case of failure in root port. The Backup-Port blocks the receiving of BPDU from own Bridge [Sch03].

However, after exploring the parameters and investigating on this protocol in the specified network, it became obvious that this protocol cannot provide the appropriated requirements for this work. In consideration to all reasons that is declared here this protocol is named in the related work in Chapter 2.2 as a solution to improve availability of an Ethernet network.

Multiple Spanning Tree Protocol

The Multiple Spanning Tree Protocol (MSTP) is based on RSTP. Up to now the STP is a shared and free from loops protocol on all Virtual Local Network Areas (VLAN). With MSTP an efficiency load balancing between VLANs is achieved, because the assignment of more than one VLAN can be resulted to one instant. Today it is allowed only one tree structure per instance. [Sch03]

Each switch uses its own MSTP configuration, which identified through a name, revision number and elements table. Each element table comprises the classification of VLANs to appropriated instances. If switches have same attributes they belong to the same group. Every switch can determine its allocation by comparing the properties of its BPDU and received BPDUs.

As explained in previous section, RSTP does not have appropriated convergence functionality in this project. Due to the facts explained above, the MSTP cannot provide a better failover time than RSTP, because it is defined for load balancing and only this fact cannot help to achieve lower convergence time. Therefore, MSTP is not analyzed in this work.

3.2 First Hop Redundancy Protocols

The first hop redundancy protocols allow multiple devices working together in a group, by sharing a single virtual IP address. The following protocols can be named as the first hop redundancy protocols.

Virtual Router Redundancy Protocol

Virtual Router Redundancy Protocol (VRRP) specifies an election protocol that dynamically assigns responsibility for a virtual router to one of the VRRP routers on a LAN [KWW04]. This protocol operates on Layer-3 based on IP addresses. The assignment of this protocol is

to control the IP addresses associated to this virtual router and forward the packets that are sent to these IP addresses [KWW04].

There are a master router and a backup router. If the master router fails, assume the backup router the responsibility of the forwarding packets. This process provides a dynamic fail-over in case of unavailability of master router. Therefore, it helps us to approach higher availability and dynamic redundancy by reducing failover time. The big advantage of using VRRP is higher availability of default path without requiring configuration of dynamic routing or router discovery protocols on every end-host. VRRP can be configured on a router if there is more than one router that operating parallel.

According to this factor, which this protocol works on Layer 3 and the assignments on this project are to reduce fail-over time on Layer 2. This protocol has not been analyzed. However it is used in the test environment. VRRP doesn't have any interference to be used parallel with a Layer-2 protocol. Although the fail over time of this protocols is about 2 to 10 seconds and it is not appropriate for this project [KWW04].

Hot Standby Router Protocol

The main purpose of Hot Standby Router Protocol (HSRP) is to support the hosts to use a single router and to maintain connectivity. If the main host fails the other hosts accept the assignment of that host [CLML98]. According to this, there is no difference between this protocol purpose and virtual router redundancy protocol, but these two protocols operate in a different way and provide the failover service in disparate type.

The HSRP protects against the failure of the first hop router when the source host cannot learn the IP address of the first hop router dynamically. HSRP provides failover services for those hosts, which they are capable of configuring the default router. Set of routers cooperates with each other to present a single virtual router to the hosts on the LAN. This set is called standby group. The active router in this group is the responsible router for forward the packets to hosts. Another router, which is ready to assume the forwarding packet, if the active router fails, is called standby router. To minimize the traffic of the network only the active and standby router send the HSRP message among the set of routers in the network. When the standby router fails or it becomes the active router, another router in the group becomes the standby router. Finally, each standby group has a single MAC address. [CLML98]

According to specification of HSRP in RFC 2281, the hello time and hold time in this protocol is in seconds (3 to 10 seconds). Therefore, it cannot be useful in this project because in this work the status of the system must be reported in sub seconds interval or by worst case every one second. Furthermore, the 3 to 10 seconds is approximately in applicable time for status report in this protocol, which is required for this work.

Gateway Load Balancing Protocol

Gateway load balancing protocol is a Cisco designed protocol, like HSRP and VRRP, which have been discussed before. It protects failing routers and prevents circuits in networks, which use Cisco switches. GLBP provides the features to backup routers for IP hosts in an Ethernet network. Multiple routers combine to perform a virtual router. One of these routers act as a forwarding router that forwards the packets and the other GLBP routers replace the forwarding router by occurring the failure in first hop router. There are differences between GLBP and similar protocols like VRRP and HSRP.

In VRRP and HSRP the virtual router group has a unique IP address but only one router is active and this router forwards the packets, all other routers are redundant routers and do not operate until the active router fails. In opposite to this, in GLBP other routers share the load. It means this protocol provides load balancing. All routers in virtual group use the same IP address and multiple virtual MAC address. All routers participate in forwarding packets. The GLBP member routers communicate each other every 3 seconds with broadcasting a Hello message [Cis02].

It is clear that 3-second hello-time is very long to use in this project and if the active router fails within the average of hello message it takes more than 3 seconds to replace it with a redundant router. In addition, one of the main concepts in this project is to use COTS (Commercial Off-the-Shell), inexpensive and standard network devices. As the result, this protocol can not be conformed by the project requirements, where the convergence time must be less than on second.

Border Gateway Protocol

Today the routing problem is more complex than the early days of existing Internet. After increasing the use of Internet and Internet services, the Internet service providers ISP showed their interest to specify the route selection and route propagation. The Internet comprises thousands of autonomous systems, which they are networks owned and operated by single institutions. There are two routing protocols, interior and exterior gateway protocols. Border Gateway Protocol (BGP) is a routing protocol used to exchange reach-ability information across autonomous systems [CR05]. This protocol is developed to solve the problems of previously defined protocols. BGP provides information about controlling packet flows between autonomous systems [CV06].

BGP can provide three types of routing information. The first routing information is an inter-autonomous system routing, which occurs between two and more BGP routers in various autonomous systems. The second type is an Intra-autonomous system routing, which occurs between two and more BGP routes within the same autonomous systems. And the third one is the pass-through autonomous system routing, which occurs between two and more BGP peer routers that exchange packets across an autonomous system that does not run BGP [RL95].

The BGP has been modified in last years many times due to the liking of ISPs to control the routing information of BGP. Many mechanisms and sub-layer protocols has been added to BGP and cause the overlapping them and complexities of using BGP. Any changes in the BGP addressing and update packet design must be informed to other ISPs. Due to these complexities and various policies and mechanisms on this protocol, this thesis has decided not to use and investigate on it.

3.3 Multi Protocol Label Switching

Label switching is a unique label that is attached to a packet and used to switch and route the packet across the network [MS04]. The label is a common header added to a packet and has been used since mid-1990. The technologies such as Frame relay and X.25 use this method. There are some differences between this technology and IP routing. IP routing uses IP of destination to forwarding the packets, but Multi Protocol Label Switching (MPLS) provides a connection-based model. It means the label added to a packet shows the switches how to process and forward the packet from source to destination. This method enables quick switching of packets across the network. This method also is combined with IP technology, because the network devices must find the network-level addresses in the packet. [MS04]

Due to combination of the Layer 2 and Layer 3 capabilities in this mechanism, it has clear advantages, which are explained further. Firstly this technology is very flexible and data can be transferred over any combination of Layer 2 technologies, it supports all Layer 3 protocols and finally it has the possibility to be compatible with technologies that will be developed in the future. [Cis05]

3.4 Dynamic Routing Protocols

In a stable and reliable network, a packet is transmitted from sender to receiver efficiency. It can be provided redundant paths in case of failures. In Wide Area Networks (WAN) with aid of IP addresses on Layer 3, routing protocols like routing information protocol (RIP) or Open Shortest Path First (OSPF) accept the assignments to designate the active path in case of a failure or load balancing and then open an alternative path [Moy98, Hed88, AJY00].

Routing Information Protocol

Router is a device that can forward a packet to its destination whereas there are more than one destination and or path that a packet can go across. This device uses the IP address as the main parameter to choose the path. A routing algorithm makes the decision in which path a packet should travel to destination. Giving a forwarding table and building a routing table makes this decision. The problem is to find the optimal path (low-cost path). To solve this

problem there are some algorithms. One of the famous algorithms is the distance-vector algorithm. Each node constructs a vector containing the distances (costs) to all other nodes and distributes that vector to its next neighbors [Hed88]. Routing Information Protocol (RIP) is an example based on distance-vector algorithms.

RIP is one of the most commonly routing protocols used in routers. By using this protocol, the router sends periodically (every 30 seconds) a message to all other routers to update the routing table, if the routing table changed, the cost of paths would be recalculated and each hop becomes a metric value. A Hop is the path between the source and destination. RIP routers maintain the best route to a destination. After updating the routing table the router sends its new information to all other visible routers. In this protocol the value of the paths is limited to 15 and the value 16 is reserved for unreachable destinations. At the worst-case, which has been given in RFC 1058, it takes about 7,5 minutes that a router reconstructs its routing table and it is absolutely not appropriate for this project. [Hed88]

Open Shortest Path First

Open Shortest Path First (OSPF) is a link-state routing protocol. It is a dynamic routing protocol and run internal to a single autonomous system, which there are multiple autonomous systems in the network environment [Moy98]. Each OSPF router keeps a database, which comprises describing the autonomous system's topology. From this database the router provides a routing table by calculating the construction of the shortest path tree. OSPF is classified as an Interior Gateway Protocol (IGP). OSPF is based on the Shortest Path First (SPF) algorithms [Moy98].

An OSPF router sends a link-state advertisement message to all other routers within the same hierarchical area. After a router collects all information of the receiving link-state advertisements, it calculates the shortest path to each node by using the SPF algorithm. All other routers use the same algorithm. This shortest path constructs as tree that the router is the root of this tree. OSPF allows the networks to be grouped together which called an area and each area has its own topology that is hidden to other areas in autonomous systems. And finally only authenticated and trusted routers can be added to an autonomous system [Moy98].

The OSPF dynamic routing is used to re-route in such networks those require the sub-seconds to reconfigure the routing table. By using this protocol, the SPF algorithm and by detecting the link fails the failover time can be approached in sub-seconds [AJY00].

IS-IS

The routing table and routing path must be always updated, when some links and/or devices removed or added to the network. This process is called as re-routing. If the re-routing process can be operated in tens of milliseconds, the high reliability and availability can be guaranteed and also the application using Voice over IP will be enable for the network.

Intermediate System to Intermediate System (IS-IS) protocol is run on frame-relay network, which can be categorized in two types, point-to-point network and broadcast network, unlike to OSPF that can be run on every type point-to-multipoint and non-broadcast networks. By exchanging Hello message in point-to-point and broadcast networks, the devices, which are running IS-IS, enable to detect the link and device failure. [Cis05]

There are three steps by IS-IS re-routing, detection the link failure and/or repair and also router failure and/or repair by neighbor routers, propagation, whereas the new link state packet shows the new changes to all other routers, and finally the shortest path calculation. [AJY00]

The re-routing time in IS-IS is in tens of second, but by making some changes it is possible to upgrade it in milliseconds. There are some suggestions, whereas the IS-IS has been analyzed [AJY00]. Some of these suggestions are:

- Granulating the hello time interval in milliseconds
- Giving higher priority to link state packet than SPF computation
- Queue hello packets in front of data packets

The OSPF and IS-IS can be used and configured in the network used in this project, although install and test of the Layer-3 protocols is not located in the responsibility of this document. However, the researches and analysis on these protocols show the appropriate results to achieve a highly available Ethernet network environment.

3.5 Link Aggregation

Link aggregation is a mechanism that allows aggregating one or more physical links to form one logical link as a Link Aggregation Group (LAG). This feature is absolutely useful to increase the capacity of system, the system availability and achieve network redundancy.

3.5.1 Port Aggregation Protocol

The Cisco systems use the Port Aggregation Protocol (PAP). It can not be used in other network devices and systems. This protocol allows the multiple ports in Cisco devices to aggregate and be viewed as a single port to all higher levels. Also PAP supports all aggregation methods in the same way as other devices. The aggregated ports form a known AGPort. This AGPort has the same properties like the link aggregation group. This protocol can manage and control the composition of an AGPort and detect the multi-drop connections.

PAP has four parts, the frame forwarding part, which supports the AGPort to forward data, the port state of port aggregation protocol, which runs on every possibly grouped physical

port, the group management, which forms the AGPorts and the flush protocol that sometimes in some applications runs to ensure packet ordering is maintained [Fin98].

3.5.2 IEEE 802.3ad

IEEE has been standardized the link aggregation and also the Link Aaggregation Control Protocol (LACP). It is provided on CSMA/CD architecture layer. The link aggregation is an optional sub-layer between MAC Client sub-layer and MAC Sub-layer. Figure 3-2 depicts the positioning of the link aggregation sub-layer in CSMA/CD architecture and the interrelationships between them. This figure illustrates how numbers of links combine to provide a single MAC interface to MAC Client sub-layer. [SB03, p.192]

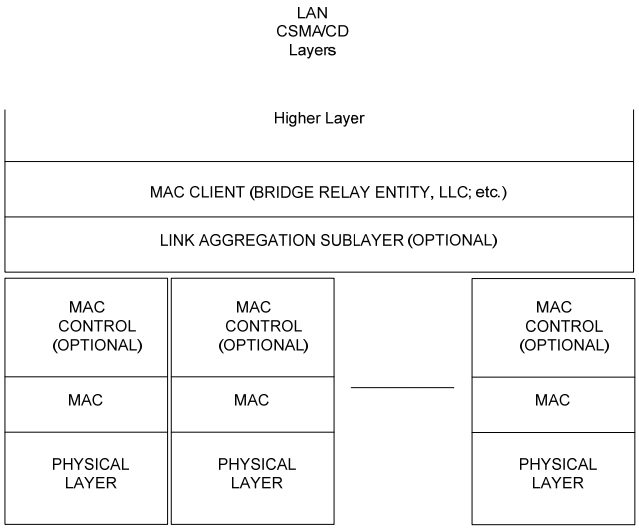


Figure 3-2: Positioning of link aggregation sub-layer [IEEE8023ad]

Link Aggregation is named as port trunking/bonding and channel bundling. The manufacturers of network devices aggregate the links in various ways. Therefore, the IEEE standardized this mechanism in the IEEE 802.3ad Standard and published it as Clause 43. This standard provides a point-to-point connection between two members of a network, whatever these two points are switches and/or servers.

The aggregated links are assembled in a Link Aggregation Group (LAG). All aggregated links must be full duplex and at the same bit transmission rate. In clause 43 of IEEE is noted that the number of aggregated links can be 4 up to 16 links at switches and from 2 up to 32 network interfaces at servers. [IEEE8023ad]

Using link aggregation has many benefits, in [IEEE8023ad] a complete list of these benefits are named but here only the important points in this thesis has been discussed.

- *Increase Availability:* If an aggregated link fails during the operation and packet exchanging, the connection will not be failed and the other aggregated links in the same LAG will continue the communication.
- *Load sharing:* MAC Client traffic may be distributed across multiple links.
- *Automatic configuration:* There is the opportunity to aggregate a set of links automatically in a link aggregation group.
- *Rapid configuration and reconfiguration:* By any changes in the physical connectivity in this mechanism all the changes and new status can be updated in less than one second, because it is possible here to check the port status every on second.
- *Low risk of duplication or disordering:* With the high probability, the link aggregation can keep the frame in the same order that they have been sent. Additionally, they will not be duplicated or disordered during the reconfiguration process.

The link aggregation also does not support the following points:

- *Multiple point aggregation:* This mechanism does not support the link aggregation between more than two systems.
- *Dissimilar MACs:* Only the links using IEEE 802.3 MAC can be aggregated in a link aggregation group.
- *Half-duplex connectivity:* Link aggregation can not be provided on half-duplex connections.
- *Operation across multiple data rate:* All the links allocated in a link aggregation group must operate in the same data rate.

In addition the link aggregation does not provide an extra data frame and layer protocol. It uses the existing MAC client layer architecture and the same frame format like all MAC 802.3 data structure. If a link cannot be aggregated in a LAG because of any reasons like different data rate from other links in LAG and/or the half-duplex operation, it can operate as a simple single link in the system. In conclusion this standard specifies appropriate hardware, configuring, management and control services. The information exchange for controlling the connected stations is pretended by LACP.

Link Aggregation Functionality

The aggregated links are known as a LAG. The link aggregation group can be configured manually and automatic over LACP. Both side of a point-to-point connection must have the similar manually or automatically configuration. By using automatically solution, the Link Aggregation Control (LAC) block as a protocol software manages the aggregation operation between MAC Control sub-layer and MAC client sub-layer, and also control how many links

can be aggregated in a LAG. The positioning of IEEE 802.3ad in CSMA/CD layer model is given in Figure 3-2. All aggregated links cooperate in load balancing. Therefore, they will take all part in forwarding the packets from source to the destination.

Figure 3-3 depicts the block diagram of this mechanism and how the positioning of LACP beside other function blocks is. This figure illustrates the main operation blocks of link aggregation and connectivity of them, that all are explicated further.

If a link aggregation group is configured, the important characteristics requirements of MAC must be satisfied. At first the packet order must be kept at receiver side. Then the configuration and mode of operation of the link aggregation must be determined. Moreover, it is better if the load balancing remains equally and constant.

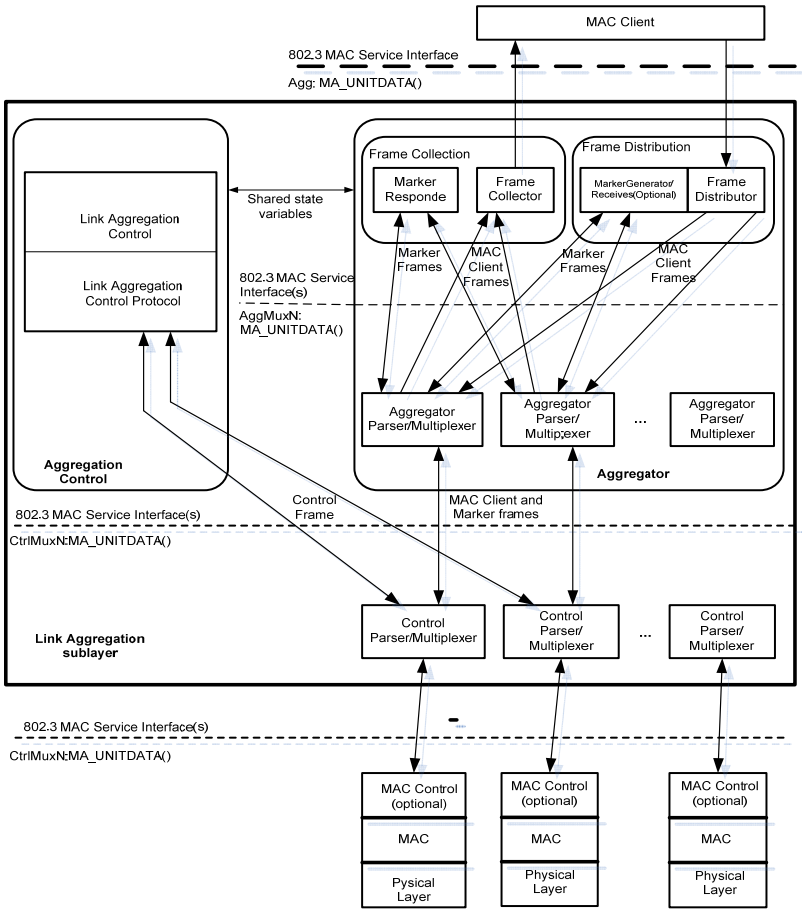


Figure 3-3: Link Aggregation sub-layer block diagram [IEEE8023ad]

As shown in Figure 3-3 the link aggregation sub-layer consists of following functions, which will be involved for the usage and analyzing the LACP in this document [SB03, p. 197]:

Frame distribution: The assignment of this block is to take the frames, which has to be transferred out and transmit them across the aggregated links in LAG. It implements a

distribution algorithm for choosing a link for transmission of any given frame. MAC Client submits the frames and it gives them for transmission up to link aggregation sub-layer. This block comprises the Marker Generator/Receiver and a frame distributor.

Frame collection: The mission of this block is in opposite of frame distributer. The received frames from ports are passed to MAC client through the frame collector. This block consists of a marker responder and a frame collector.

Aggregation Parser/Multiplexer: Each aggregated port has an aggregator parser/multiplexer. On transmission multiplexers have the assignment to pass frame transmission requests from distributer, marker generator and/or marker responder to the appropriate port. On receipt, the parsers decodes frames received from control marker, passes them for marker responder. It passes the MAC client PDUs to appropriate ports.

Aggregator: This block consists of Frame distributer, frame collector and aggregation parser/multiplexers.

Aggregation Control: This block controls the link aggregation processes. Aggregation control block cooperates with the LACP to aggregate the capable links automatically in link aggregation group.

Control Parser/Multiplexer: On transmission, the multiplexers pass the transmission requests to appropriate ports and on receipt, the parsers distinguish LACPDU from other frames and pass them to appropriate sub-layer and to all other frames to the aggregator.

Any distribution and collection algorithms must prevent the frame disordering and frame duplication. The frame distribution algorithm in this mechanism ensure the frames ordering by transmission to be exactly the same as they are produced and transmitted on links and on receipt the frame collector orders or reorders the frames again with respect to transmitted ordering. Therefore, in frame collector it is not required to check and maintain the frame ordering. Only the frame distributor can take the frame ordering and also use the load-balancing feature of this mechanism.

The marker generator/receiver is used for marker protocol that explained detailed in the Clause 43 [IEEE8023ad]. Distribution algorithms request this marker generator/receiver. The marker responder is also used for marker protocol. It receives the marker PDUs and transmits the marker respond PDU through the same port, which the marker PDU was received. The implementing marker generator/receiver is optional unlike the implementing the marker responder that is required.

An aggregator is responsible for standard IEEE 802.3 MAC services for associated MAC client. The MAC client can access the MAC services across the aggregator. Therefore, the aggregator associates to the logical MAC address, which is given to a LAG. If the collector and distributor are enabled and also one or more ports are attached to a group the associated aggregator is available for this MAC address.

A system can have multiple aggregator and also LAG, which each connected to a related MAC client. A given port in the system must bind to an aggregator. This mission is managed by LAC. In addition, the link aggregation function must monitor the condition of a link and LAG to determine the changes. And finally also it must determine which port can be aggregated. LACP achieves the automatic link aggregation, but it is possible to do this job manually through the administration tools and network manager.

The packets are passed through the all ports in a LAG for load balancing and also more availability in the networks in case of failure. However, it can be unequal for various systems and components. In this standard there are no special distribution algorithms. It is possible to use any distribution algorithms, which is supported by MAC client. The MAC address of a port is used as source address for transmitting packets it is unique and individual. However, the single unique MAC address of an aggregator is used as both source and destination address of transmission and receipt the packets through the LAG.

The MAC client and aggregator communicate each other across the standard IEEE 802.3 service interfaces. In addition in a link aggregation standard some special internal interfaces are defined to internal communication of the blocks, which are explained above. As depicted in Figure 3-3 the interfaces, such as *Agg, AggMuxN, CtrlMuxN* and *MacN,* is defined for this standard [IEEE8023ad]. In the main clause has been explained, which methods will be called by using each interface. The N declares the number of each port in these interfaces. In consideration to transmission function and reception function, the methods can be distinguished.

Every port in IEEE 802.3 standard has a single unique MAC address. Each LAG is assigned also a single unique MAC address. This MAC address can be the address of one of aggregated port or a distinct MAC address, which is not used by other parts of the network.

In the link aggregation sub-layer the aggregator takes the unique MAC address, the MAC client also see only the MAC address of the aggregator and by transmitting a packet uses this address as the source address. The other MACs remain hidden in MAC client perspective. Therefore, the MAC client chooses the aggregator's address and if the MAC client does not add any address to the frames, then the aggregator will attach its address to the frame itself as the source address.

Link Aggregation Control

LAC is responsible for configuring and controlling the link aggregation sub-layer. It means it must maintain the configuration information and also exchanges the configuration information with other systems to allocate a link in a LAG. As another assignments the control of the attaching and removing the port into and from the LAG can be named. If the port is allocated in a LAG, the LAC will attach the port to an aggregator and if it should not be a member of the LAG anymore, the LAC also muss remove the port from aggregator. In

case of transmitting and receipt packets, this block has to control and manage the aggregator's collector and distributor. [SB03, p.198]

The LAC is integrated in link aggregation sub-layer, because parameters and factors must be checked in any aggregation process. The link aggregation mechanism requires precisely attention. By aggregating a port in a LAG, some usual cases can be observed:

- It must be checked, if the port is not aggregated in other Link Aggregation Group
- Control the links in system and create a LAG if it is necessary
- Check the validity of a LAG by monitoring the status of a LAG
- Also check the validity of links membership in a LAG

LAG can be performed automatically and without manual configuration. In addition LAC monitors the state of link aggregation continuously. If some changes and updates are given in the system it can perform reconfiguration of LAGs. The configuration can be resolved deterministically. The configuration is suited for devices with different hardware and software. The configuring a link into the LAG doesn't cause the degradation of link in the system. The configuration can perform rapidly in a stable state, the configuration is provided by exchanging only 3 LACPDUs. [IEEE8023ad]

In conclusion to all the characteristics and properties of LAC that explained above, it can be named other three important properties of this block, which were the main purpose of implementing this Standard, the low risk of disordering, the low risk of duplication of frames and also low protocol overhead. [IEEE8023ad]

Principles of Link Aggregation Control Protocol

Link Aggregation Control Protocol provides the ability to exchange the information between two partner systems that aggregated links belongs to them and allows the LAC to reach the agreement of identification of LAG and also move the link to the LAG and enables the transmission and reception functions. [SB03, p. 191]

The most important function in the LACP is the transmission information and port status. The first actor sends LACPDU (Link Aggregation Control Protocol Data Unit) to its partner and they inform each other by exchanging their own states and what action will occur next.

An administrative control entity associated with each port is known as *LACP-Activity*. This can take two values. Active LACP and Passive LACP are these two values. Passive LACP means that the port is must wait and do not transmit any packet until its partner takes the Active Control LACP value. The Active LACP means the ports is participating in the protocol.

The LACPDU is transmitted periodically as long as the both partners have the Active LACP value. The transmission period can be valued in short and fast sequences. The *LACP-Timeout* is given the value Short and Long Timeout.

However the LACPDU will be transmitted periodically but if the protocol needs it to be transmitted then the LACPDU will be sent. In other case if one side of aggregation found that the other side or other partner is not up-to-date, then it will send LACPDU to upgrade the partner side.

All LACPDUs comprise an integral number of octets. The octets are transmitted from top to bottom. The LACP structure is shown in Appendix C of this document. The fields of LACPDU are illustrated in this appendix and all of them are referenced to [IEEE8023ad].

Link Aggregation Identification Factors

If the LAC wants to perform its job correctly, it must determine an Identification number, which participated in Aggregation and LAG and the aggregated ports. The communication between ports and associated aggregators can be providing only by using a valid and specified identification. The correct identification also helps to detect links and LAG failures more quickly.

Single unique system identification belongs to each system that consists of a LAG. The system identification is formed of two parts. The system priority and the MAC address are these two parts. The system identification comprises eight octet unsigned binary number, whereas the two important significant octets are the system priority and the third one delivered to MAC address. In this standard one single unique identification character is delivered to each aggregator. The aggregator's identification is the MAC address. Each aggregator comprises multiple ports and one MAC address is delivered to this aggregator.

The LAC sets port identification to each port aggregated in the LAG. The port identification comprises a port priority and the port number. The port number should be uniquely, and the port number 0 doesn't belong to any ports. The port identification is a four octet unsigned binary number; the two first important octets are the two first important octets of port priority, and the second two important octets are the first two important octets of port number.

The key is a single integer parameter of a Port that shows the ability of the port to aggregate to a LAG. It comprises the physical parameters and properties of a port like duplicity and data rate. Also a key consists of configuration constraints establishment of port by network administrator. In addition to this it shows, if higher layer uses the port. And finally the key comprises the characteristics and limitation of the port implementation.

It is possible to distinguish the keys in two kinds: *Operational Key and Administrative Key*. Each port should be associated to both of these two kinds of keys. If the aggregation will be formed, the operation key is in active uses. The administrative key is used to manipulate the

key values by management. In case of any administrative changes in the key value, it is not possible to reflect the value in the operation immediately. However, if the system supports the dynamic change of the key value, the key manipulation can be reflected in the operation.

The key value is only valid in the system that port allocated in it. Therefore, the relationship between the administrative and operation key are only context of that system. If an administrative key is assigned to a set of ports, it means these ports have the ability to aggregate each other and perform a link aggregation group. If an operation key is assigned to a set of ports, then in the absence of other constraints, the current operational state of the set of ports allows any subset of that set of ports to be aggregated together from the perspective of the system making the assignment. The set of such ports that will actually be aggregated will be those keys that terminate at a common partner system. The set of ports in a given System that share the same operational key value are said to be members of the same key group. All keys are 16 bit identifiers and all value expected null are valid. [IEEE8023ad]

An aggregation group is the set of multiple links, which are aggregated together, or an Individual Link. Each LAG has a unique identification number. This LAG ID is constructed of three parameters:

1. The system identifier
2. The operational key
3. The ports identifier, if the port is an Individual Link

By exchanging protocol information these parameters can be assigned to the system automatically. However, if it is not possible the administrative value can be used for LAG ID. Normally the value Zero cannot be used but in administrative value it is possible to give Zero as a parameter. [SB03, p.199]

Configuring a Link Aggregation Group

At first a port should be selected for membership in the LAG. For this assignment should be checked the LAG ID, this event is executed over LAC. After that, before any frames distributed or collected, the link aggregation control entity of both local and remote peer need to agree on the LAG. By using LACP this process will be executed rapidly, because the peers communicating entities to check current LAG ID.

When the link is selected and the LAC agrees the LAG, the link can be attached to the LAG through attaching to the compatible aggregator. If the aggregator’s operational key is equal to port’s operational key, and all other links currently attached to the aggregator have the same LAG, then the aggregator is known as a compatible aggregator. If there is more than one compatible aggregator, then the link aggregation control determines, to which aggregator the links should be attached. If there is no compatible aggregator the link can executes as an Individual IEEE 802.3 link. Then LAC signals the readiness of the operation to its peer. Every aggregator is responsible for enabling and disabling the distributor and collector. At

first the collector is enabled. Once the received information finds the remote aggregator's collector enable, the distributor is also enabled.

A link is monitored as long as the LAC at both end of link agrees on the configuration of link. If LAC detects any change, the links will be removed from the LAG and moved to new LAG. If any protocol changes occur, or any system constraints are not achieved, then the port will be separated from the Aggregator. By all these changes the frame ordering will be preserved.

3.6 Stacking

Some manageable switches present a feature that can be used to approach higher availability and it is called the Stacking. It is not a technology or a protocol like other previously explained technologies in this chapter. However, some companies offer this mechanism in their developed network components. In this sub-chapter this feature will be explained. Stacking can be used in some similar switches and they are appeared as a single unit. Stack is a surface on network components, which will work all as a single unit. Multiple stacked components can operate as a virtual switch. In other words, stacking provides to manage multiple switches over a single unit. Switches support stacking up to 8 members. [Del05] however, the stacking can be used in some other cases and scenario. Clearly, if the network environment has to be redundant for more than one Layer 2 switches, one of the comfortable solutions is the use of stacking between them. Moreover, the ability to use all ports of stacked switches can perform a switch with more available ports and thus more opportunity. However, the stacking helps to perform higher availability, even in another scenario.

If stacking is set up in a group of switches, one of them is selected as master member and another member can be selected as backup member. All other devices are selected as stack member, each unit in stacking gets a unique member ID and it is used to select their role. All stack members must be running the same switch software version [Del05]. The user of a stacked system can configure the stacked group through the stack master, it means the stack master have the assignment to detect the port status and reconfiguration the ports.

There are four events cause the reconfiguration in a stacking system [Del05]:

- Unit Failure
- Inter-unit Stacking Link Failure
- Unit Insertion
- Removal of a Stacking Unit

If any of the named events occurs then the switch stacking software reconfigures the port status and members characteristics in the stacked system. In the worst-case, if the master stack switch fails the backup master switch is defined as new master and all other members change their roles, for example the next stack member with the higher priority becomes

backup master switch. Member's ID defines the priority of a member. For example if a stack member with ID 1 is a master member and it fails the backup switch with ID 2 becomes master and the member with ID 3 becomes the backup switch. If a switch will be added to a stacking group it gets a member ID but before that the master will check the software version of new switch if it is compatible to all other stack members the switch can be added, otherwise the master will reject the request.

The stacking can be produced in some special topologies like ring and daisy chain topology. A stacked ring topology is defined where all stacked devices are connected to each other like a circle. If a packet is attached to a certain switch and the packet receives to another switch in stacked system first, the packet will be forwarded in the stacking until it reaches to the destination. If a switch in stacked ring topology fails the system automatically call the failover topology to reconfigure the member statuses. [Del05]

Furthermore, a system will be stacked in daisy-chain topology. In this topology the switches get connected to each other forming a chain. In this topology if a member fails, the system will be divided into two sub-segments. Each segment gets a stack master and these two segments operate fully independent without any human interference. Although by various companies and models these topologies and recovering strategies are defined differently. [Del05]

The failover topology in stacking software defines all strategies and mechanisms to recovering the system in case of failure. Each time the system reboots, the Start-up configuration file in the master unit is used to configure the stack. The Dell switch *PowerConnect 3424/P* is used in this work. The stacking process will be explained completely. The failover behavior of this switch will be analyzed.

In conclusion the stacking is the company-defined technology and it depends on the company that has manufactured the switch. It is not a standard or a protocol that would define by the organizations such as IEEE. Each company has its own switch software and different switches made of different manufactures have diverse switch software. Therefore, the convergence time of various switches can be unequal. It means in case of occurring any of above named events, switches made by different companies treat unequally to handle the occurred failure.

Chapter 4 Definition of Test Cases and Proposed Solution

Previously the properties, parameters and functionality of various technologies and protocols in Ethernet networks have been declared. The analysis on these protocols can lead this thesis to select a proposed solution to achieve highly available Ethernet network. This solution can be used in safety critical voice communication systems. After choosing an appropriate protocol and designing the related redundant network environment, some test cases will be defined. These test cases must be executed and help this work to approach its project requirements. The problems and deficiencies during the test cases and some other related analysis would be mentioned as well.

4.1 Proposed Solution

Previously many protocols and standards are mentioned in the last chapter, which are used in Ethernet networks. Some of them are specified for other special attributes and issues of Ethernet networks expecting high availability, but they can be used for this assignment as well.

This thesis consists of several requirements to provide network availability and reduce the convergence time in safety critical voice communication systems. These requirements, such as achievement an optimal failover time and also the achievements of the higher availability, must be approached in this work.

In the scenario of this work, one of the problems is the obvious of node failure or link failure in the network environment. The failure, which can be occurred in this work, will be analyzed and mentioned in next sub-section. However, this issue will be leading this thesis to choose one of the previously explained protocols. As discussed at each section and declared technologies, some of them have inapplicable parameters. The protocols and technologies such as RIP, STP, and etc. cannot help this work to approach its requirements. In another

hand, some other protocols such as LACP, OSPF, VRRP and IS-IS have the absorbing properties, which attract this work to use and integrate them in the network environment.

The proposed solution and protocol must be used in the Ethernet network environment. As explained in Chapter 1 the safety critical voice communication applications will utilize this environment to their assignments. The air traffic control systems and emergency call centers will be use these applications. Therefore, any long downtime causes the lost of important information during the transmission of voice data. Due to the facts, the error detection, error recovery and the fault treatment of the proposed solution must be appropriate for the explained situations.

In consideration to the designed network environment, which will be declared later, all nodes must be redundant to achieve the higher availability and also the optimal convergence time. Whereas, the host must be connected to the backbone network, it is possible to use two switches and two routers to make the connections redundant. Therefore, one node must be connected to two physically and functionally redundant components. Now two various issues can be observed here. The first issue is the functionality redundancy between two components in the same layer of the network, clearly two switches and/or two routers. The second essential point is way to connect this layer to the previous and also the next layers. Figure 4-1 depicts the declaration before to connect a node to two redundant components. The host A in this figure has been connected to the host B with higher availability. Both components A and B can forward the frames to and from a host. If one of these components fails or a connection between component and host fails, the other path is still active and the transaction between these two hosts can be continued without any long downtime. Therefore, the hashed lines illustrate the redundant ways between two hosts, and the continuous line depicts the connection between two redundant components.

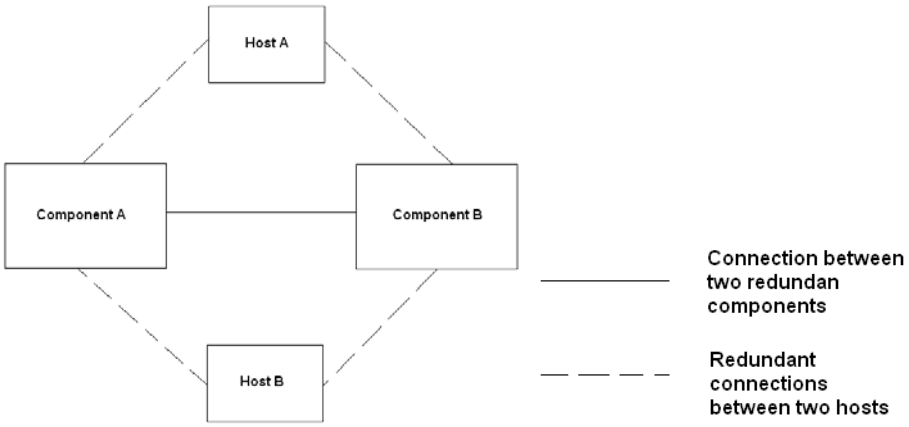


Figure 4-1: Connection between two hosts with higher availability

One of the protocols and mechanisms, which are discussed in Chapter 3, is the link aggregation and LACP. This mechanism performs the ability of a network component to

connect to another component with more than one link. These multiple links provide one virtual single link. As explained in the last chapter, if one link fails, the other links in LAG can continue the transaction with other components. However, the fault treatment and also error recovery of this protocol must be investigated. Due to the facts, this protocol offers a better chance to achieve the higher availability and more network redundancy for the network environment. Therefore, this protocol can be named as the proposed solution in this document, although the capability and compatibility of it must be explored and analyzed.

Furthermore, the LACP can perform the error detection within sub-seconds (see 3.5.2). This attribute can be used in safety critical voice communication systems and provide the better convergence time in case of link failure. The fault treatment, error detection and error recovery of this protocol will be investigated in the next chapter. However, precisely definition of the test cases and also the developing eligible network architecture can approach this thesis to the target.

If this thesis has been suggested LACP and link aggregation as the proposed solution, it does not mean that other protocols are not appropriate for this job in this work. In other words, this document wants only to analyze the properties of one protocol. Therefore, this thesis will only assess the capability of its propose solution and will not suppose this protocol, if it can perform better situation in the network environment.

During this work the failover behavior of the specified network will be investigated. If this solution performs the project goals and also can achieve the specified parameters then it is suggested in the safety critical voice communication networks. If any type of failures occurs, it is significant to note how quickly the system can find an alternative point to replace it. The solution has to provide strong error detection coverage and by executing a few tasks recover the failure, although all these properties will be investigated over test cases.

4.2 Failure Analysis

High availability is closely associated to fault tolerance. Developers and designers classify faults in different classes. They are categorized in failure caused by human, nature or by accident. Finally the last group is internal and external. However, the most failures can be forecasted, repaired or removed.

In this project, it has to be distinguished between some common system failures that occur in Ethernet networks, which provide the connections between various end-stations of safety critical voice communication systems. The aim of this project is to put mechanisms and technologies into practice to decrease the system convergence time under the specified level. Therefore, failure types in respect to these voice communication systems are categorized into different forms.

The first and major failures in the interested purpose are the physical failures. This ensemble of failures covers all physical corruptions of devices, which are used in Ethernet networks. Link breakdown, electrical power outage, inability to achieve Ethernet ports, etc. are categorized in this part.

The second form are failures that happen in the data link layer and the network layer devices, where the MAC address and IP Address are the identification of each device. Software failures cause the breakdowns in these layers and the packets cannot be delivered correctly to the destination. In other cases the network cannot find the device, although it is connected to the network. The failure in producing a routing table can be categorized in this section as well.

Operation errors, mass storage problems and application problems can be named as third group. These failures occur in higher-level applications and higher layers of OSI-model. These failures will not be focused in this work.

There is no clean system, which either no fault is occurred or all failures can be tolerated. The failures can occur in a system but they have not to exceed the specified level. Some failures can be tolerated and some must be accepted by the system. How the failures will be tolerated and which failures are acceptable, are not focused in this project. However, it is only necessary to know which type can occur in the pre-specified network and how quickly they can be detected and recovered.

4.3 Network Architecture

The network architecture that is designed for this project is a model of end systems for critical VoIP communication systems. This device must be suitable for safety critical applications. In other words, this device should be increasing the efficiency by occurrence the failure hence increase the availability in the network. The designed network architecture is the part of an extended network and the end-station in the ATC and emergency call center system. It is not important to know about the capability of the extended network. This work will only concentrate on configuration and properties of that part shown in Figure 4-2.

The end point networks are integrated in the control tower. The ground-to-ground communication will be performed between two control towers located at two different locations. This network depicts how the servers are connected to outside world (Backbone network). This thesis will focus on the availability of network devices and connections between the router and two servers by reducing the convergence time in case of failures.

Whereas the properties of this network are observable in Figure 4-2, it is redundant at any point of connection. If the designed network is partitioned into three parts, it can be seen that each part comprises two exactly the same devices. This architecture approaches the applications to perform higher availability and serviceability.

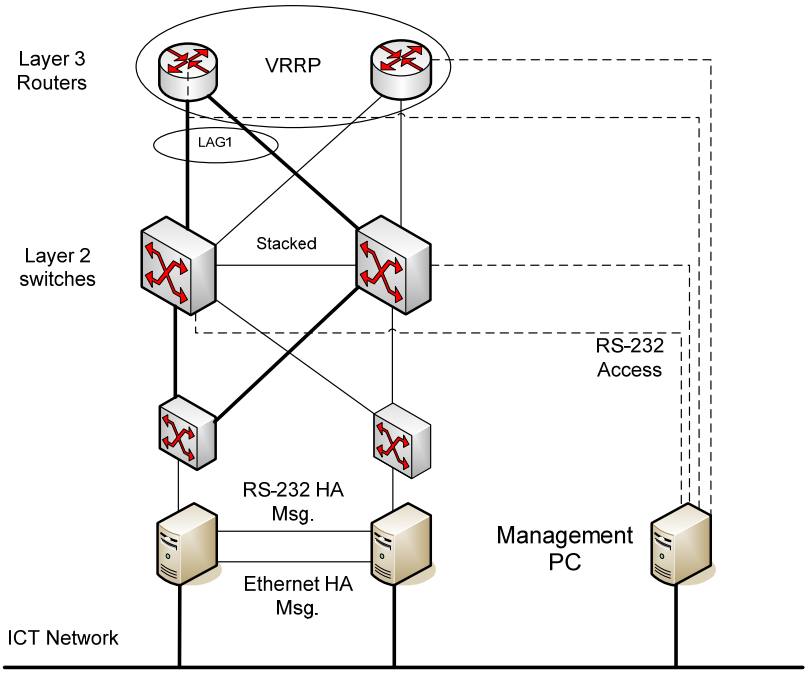


Figure 4-2: Network architecture of entire redundant system

The network is shown in Figure 4-2 is a part of an Ethernet network, IEEE 802.3. This network is based on OSI reference model and Layer-2 and Layer-3 switches and routers are used in it. This network is only as a test environment.

As it seen in the Figure 4-2, this network consists of two routers, two switches and two end computer systems. All these nodes operate redundancy. The redundancy in the nodes will be provided through the protocols and technologies that are explained in the last chapter. The routing and switching mechanisms, which are used in this network, can improve the availability of the specified network. Moreover, the two end-station computer systems are also redundant. They work with the Linux-HA [FM03] to perform the data synchronization. One computer operates as the primary device and the other as the standby backup system. The management PC will simulate the backbone network in this case.

Another issues is the use of a functionally redundancy at routers. These devices will be configured to use of VRRP, which has been declared in Chapter 3. The stacking and LACP are two other technologies and mechanisms used in this environment. This environment will approach higher availability over configured protocols. However, the high availability of devices and physical connections will be investigated in further chapter. “Appendix A” declares each device, which is used in this network architecture and how they will be configured to use in this work.

4.4 Utilized Tools and Applications

This thesis will investigate the capability and the efficiencies of link aggregation and LACP. Therefore, for analyzing the behavior of the system and also the measurement of the convergence time some tools and applications will be used. They have been specified to introduce the way to ease the tests, measurements and analyzing here.

Linux Bonding Driver

Due to the test cases and also the network architecture, two Network Interface Cards (NIC) must be aggregated at computer side. The aggregation process will be done with help of a Linux kernel module called Bonding driver. The Linux Bonding driver provides all aggregation features such as load balancing and link integrity monitoring. The Bonding driver produced from Ronald Becker for Kernel 2.0. However this driver has been updated by any upgrades of Linux Kernel. [Dav00]

The bonding is already available as a Linux kernel module and also a user control program called *ifenslave*. It is required to configure the bonding module in the Linux kernel. The bonding driver is located in *driver/net/boding* subdirectory in the kernel source. The full documentation of configuring and using the bonding driver was explained in Linux Foundation website referenced here as [Dav00]. It is recommended to install the *ifenslave* at first in order to configure the bonding driver.

The bonding options are given as parameters in command line at load time. These options are listed in following. However, some of them are not interesting for this work and some of them are used every time because all interfaces support them. Therefore, only the adequate parameters are described here.

- *lacp_rate*: It specifies the LACP timeout, which is explained in 3.5.2. The value 0 is the slow timeout to transmit LACPDU packet in mode 802.3ad. With this value a request will be transferred every 30 seconds. The value 1 is the fast timeout. It sends the request every 1 second.
- *max_bonds*: It appoints the maximum bonded devices. The default value is 1.
- *miimon*: The MII link will be monitored periodically. This parameter specifies this frequency in milliseconds. The value 0 disables this option, although it is set to 0 by default.
- *mode*: It appoints in which rule and principle operates the bonding driver. There exist 7 operation modes for the bonding driver, which the bonded NICs can support all these operating modes:
    1. *Balance_rr:* In this mode the packets are transmitted sequentially from the first to the last active slave.

2. *Active-backup:* In this mode only one slave is active and the others are backup slaves. If the active slave fails, the backup inherits the duty of active slave.
3. *Broadcast:* All packets are received and transmitted on all slaves.
4. *Balanced-xor*: Transmit based on the selected transmit hash policy.
5. *802.3ad:* This mode supports the IEEE 802.3ad link aggregation. The slaves are aggregated dynamic if this mode are selected. This mode will be used in this work, although the other modes can be supported by this system.
6. *Balanced-tlb:* Adaptive transmit load balancing.
7. *Balanced-alb:* Adaptive load balancing includes balance-tlb plus receive load balancing (rlb) for IPV4 traffic.

- *primary*: By typing a string (eth0 or eth1)in command line, it is possible to select the primary slave.
- *updelay*: In opposite to *downdelay*, if a link up is detected, the *updelay* specifies the time before enabling the link in milliseconds.
- *use_carrier*: To specifying, if the link is active or up, this option allows the *miimon* to use MII or ETHTOOL versus *netif_carrier_ok()*.
- *xmit_hash_policy*: This option specifies the transmit hash policy, if whether the *balance-xor* are used or *802.3ad*.

The cognition of these parameters leads the user to configure the bonding driver for its NICs in the Linux computer to provide the ability that he wants. Clearly, this thesis also wants to improve the availability and to approach higher availability rate. Therefore, the chosen availability protocol will be configured here. The LACP with the short timeout rate will be selected and configured in the network environment.

The capability of bonding driver and what it can perform in this thesis will be discussed in the Chapter 5 where the test results will be analyzed. However, some parameters must be declared in detail.

The mode that is used here as explained before is 802.3ad. Therefore, these parameters will be given when the NICs will be aggregated in a LAG. The *primary* parameter can select the active interface in the communication. However, the link aggregation uses the aggregated ports but in the bonding driver will be used only one interface for communication. Therefore, it is possible to choose the appropriate interface as the active and primary interface.

Moreover, if the *use_carrier* is parameterized in the configuration, the Linux operating system has the ability to observe the interface status by the use of the MII and ETHTOOL. These tools can check the link status of each interface. This document will concentrate on this feature to explore precisely the error detection and error recovery time.

Furthermore, the *miimon* parameter can specify the observation interval of the link status in the system. It can be parameterized as a count for millisecond. In other words, this parameter shows the time period in that the link will be check if it is either up or down. Another important parameter is the *xmit_hash_policy*. This parameter depicts the ability of the bonding driver to select the standby interface to replace with the failed primary device.

In appendix B two script files are given, which has been implemented to configure the bonding driver and also the link aggregation group in this work. These script files are written easily in the shell script. After running the script, the LAG will consist of two NICs and also the parameters and appropriated operating mode will be configured on this LAG.

Frame Detection Tool

To detect the generated and transmitted frames, there exist some applications. These applications can be used to detect the frame structure, protocol and also the frame contents. TCPdump and Wireshark can be named as such applications. This thesis will be use the Wireshark in order to its usage convenience, more user-friendly structure and its GUI. Therefore, in this section the properties and functionality block diagram of this application will be discussed.

Wireshark is a network packet analyzer. This means this software allows to capture the network packets and displays which packet data it is. It is an open source packet analyzer. [LSW04]

This software has the ability to be used on Linux/Unix and Windows operation systems. It captures packets from the NIC. It has the ability to open the packets and save the packet contents. It is possible to import and export saved data from and to other capture programs. The most helpful features that Wireshark provides is the colorization of the displayed packets based on filters. In Wireshark it is possible to filter the packets and capture only the frames in designated protocol.

Wireshark is appropriate software for administration and research assignments. This project needs to analyze the packet receiving and transmitting time. Therefore, by the use of the complete packet information that Wireshark captures it is possible to detect the gap between packets in sub-milliseconds.

Another important point in using this software is to opening the capture files in other editors. The captured files can be exported to a CVS[1] file. In this project all captured files are exported in this type of files to be used with other programs such as excel and word.

[1] Comma Separated Value: the text will be exported as a normal text and the values will be separated with commas.

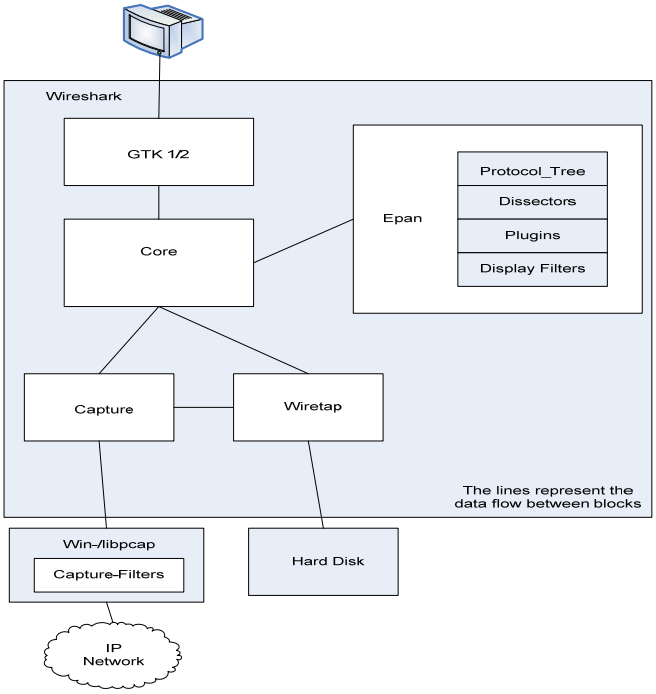


Figure 4-3: Wireshark function blocks [LSW04]

The functionality of Wireshark and how it works will be focused in this document. Figure 4-3 depicts the function block diagrams of this application. This figure illustrates how the Wireshark will capture the packets and specifies the packet protocol type and other relevant information of frames. The lines between blocks in this figure illustrate the information flow among the various blocks. This application works below the Linux kernel sub-systems. Clearly, it can detect the frames sooner than the kernel itself and it knows if the network interface receives a packet. The Core block is the major block that holds other application parts together. Epan is the packet analyzer and displays filter engine. The Wiretap block is responsible for saving and exporting the captured file. The Capture block is the capture engine of this application.

The Capture block will receive the packets from the NIC and save them in to the hard disk. The packets can be read in their natural format in this application. The important point is the capturing time. The capturing time of the packet will be established in microsecond precision in the capture file. The NIC will be active a flag when it wants to pass the packet to the Wireshark application. The Wireshark can dissect the packets while the packets are loading from a file.

Packet Generator and Transceiver

The main assignment in the test cases, which will be declared further in this document, is the simulation of a normal transaction between two end-stations. This transaction can be

executed in specified conditions between two hosts, which play the role of the computer systems in end-stations. The communication will be started, when the computer generates a specified packet with a defined size and transmit them to the network. The packets must be generated frequently without any perceptible interval, because these packets want to simulate the voice data, which will be packetize in VoIP applications. These assignments will be executed in a Perl script file in an end-station system. At another side the supplementary application that is run on the end-station will receive and detect the transmitted packets.

The script files that are used to ease the test process for the measurements in this thesis are Perl socket program files. These files are two specified client server interactions programs, which one of them is run on client machine and the other one is run at server system, which simulates the backbone network in this environment. In this project the client machine is the computer that comprises the link aggregation and named *volare-s0*. The *volare-s0* consists of the interface between the operator and the redundant Ethernet network. The management computer that explained before is the server machine in this network. Across this computer, the commands are transmitted to the network. Client receives them, executes them and answers the server requests, thus sends its requests to the server. Both programs codes are given in the appendix B with additional detailed explanations.

The script program in client will send a UDP packet consecutively to the server machine. The server has an open door to receive the packets, which has been sent to it. In this case only one system sends the message to it but in the reality it has to support more clients. These packets must simulate the VoIP packets. In consideration to the VoIP properties that this protocol does not provide a specified checksum or an algorithm to check the packet corrections and other safety appointments in these codes the UDP packets form have been chosen. In Perl script language it is possible to generate the both TCP and UDP packets. By using the TCP some checksums must be applied by the sender and receiver, but by the use of UDP these check points are relinquished. Therefore, if a UDP packet is generated, a packet with simpler frame structure will be transferred. This protocol is faster and also less complex than the TCP packets. UDP adds only 8 to 12 octets to the packet header. In contrast, TCP will add 24 octets to the packets as header and it requires checksums. Furthermore, the positive and correct reception of the packets has to be acknowledged in TCP packet transmission. Therefore, the UDP packet will be generate here, because it has the similar properties. [Com00, p.198]

The main assignment of these two script files is to create a socket between the client and server and transmit the generated packets from the client to the server. The port address and the packet format must be given in this command. To write such a file the following steps must be achieved for a server program [Bar02, p.99]. The socket must be created and connected to a port over its address. Then the socket will listen to the port and receive the frames, which are sent from the client.

Moreover, the same process must be achieved for the client program. The socket will be created and the generated packets must be sent to the destination. For the client machine the following steps must be achieved [Bar02, p.100]:

1. Create a socket
2. Connect the socket to the server

The socket that will be created here is an object-oriented interface. This type of socket simplifies the socket programming in the network assignment. The module *IO::Socket* in the Perl core provides this socket. In this class the parameters, such as the peer address, the peer port, the transport service protocol and also the local port, can be configured [Bar02, p. 153].

Another module that has been used in the Perl script programs is the *Time::HiRes*. This module implements an interface for the high-resolution time and timers [WSA02]. The client and the server programs must register the time of transmission and the reception of the packet by the Perl programs. The packet will be generated at the client machine and will be transmitted to the destination (server machine). If the generated packet has been transmitted successfully into the network the client program will record the transmission time. In server side exactly the same sequences will be done after the successfully reception of the packet. Therefore, this module in the Perl core can perform the accurate time measurements.

The generated packets and also the received packets at both sides of communication will be saved in the related files in the client and the server machines. The *FileHandle* module provides the permission to open a file and read and write from and in it. The read and write privileges must be activated to this file. Due to the breaking the program before the finishing, the *autoflush()* function will end the file after the program endpoint. Otherwise, the program will be finished by the normal function *close()*.

4.5 Test Environment

The test environment, also known as the network architecture is shown again in Figure 4-4. This figure depicts the entire links and devices in the test network. All links and devices are named to ease the test description. As explained in the previously the entire network is configured to use the link aggregation. The LAGs are defined in this environment for all network components. As shown in figure 4-4, two VLANs are defined on the router to connect two subnets to each other. In client machine has been implemented two VLANs to connect this computer system to the specified subnet and also to the ICT subnet to control its assignments remotely. The hatched line shown in this figure, illustrate the management connections between the management PC and each network device. The Layer-2 switches are stacked and operate as a single unit.

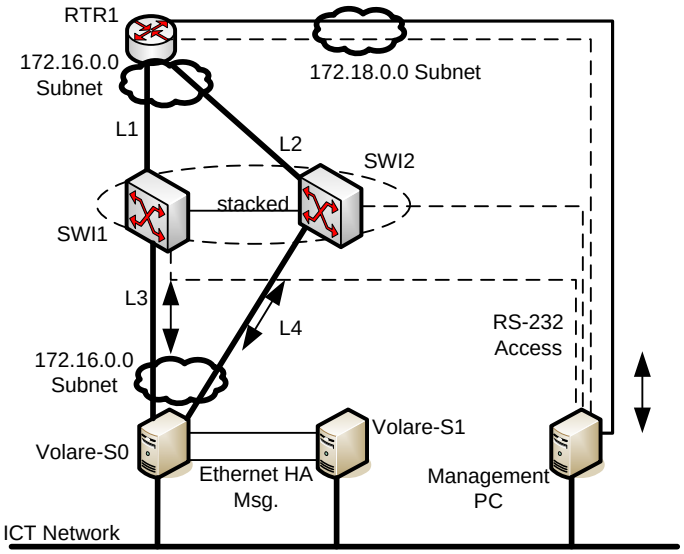


Figure 4-4: Test network environment

The switches and routers have been configured as described in Appendix A. After configuring the switches and router, two similar NICs were installed in the client computer *volare-s0*. These two network interfaces have been aggregated as a single virtual port called *bond0* in this process. The Intel e1000 NICs cards were used here. Finally the configuration was checked, in which the packet from client could be sent successfully to the server thus *volare-PC*. The preparation for the test requires the tools that were described in last section. The Perl client and server programs are ready to test. The Wireshark application are also opened at both client and server side to capture the specified interfaces to each side.

It is important in this thesis to measure the exact time in which the packet will be generated at the sender and the exact time that the packet is detected by the receiver, because it is possible to measure the time duration between these two assignments and also to measure the convergence time. Therefore, the global time and the hardware time of the sender and receiver must be synchronized before starting the tests. On that account the *volare-s0* and the *volare-pc*, which are connected to the ICT network apart, must be synchronized with the one same external reference. In this case the “TU-Wien” servers are used for the synchronization. These servers will be separately synchronized everyday with various global time references automatically. Therefore the time of these servers can be assumed as a precise reference time, although these times can have up to 5 milliseconds offset with the reference time.

The time synchronization is performed here through the Network Time Protocol (NTP). The Linux computers can provide this protocol over *ntpdate* command, which synchronizes the client system with given IP address that is here the Vienna University of Technology’s sever IP address. The only problem here is the precision of the NTP protocol that is 1 millisecond for synchronization. However, in some cases these tests require microsecond’s precision for an accurate measurement of the transfer time. For providing the synchronization in the

microsecond area, an external hardware or an external oscillator is required. One way to ignore the explained problem is to enlarge the packets to increase the transfer times in milliseconds. The NTP precision can be provided sometimes in microseconds.

In the tests that will be executed in this thesis, UDP packets will be generated and sent to the partner system. As explained before the Perl client program is responsible for generating and preparing the packet to transfer. In this program a UDP packet will be defined. This packet consists of specified random characters as data payload. This field will be checked at each side after transmission to explore the correction of transmission. The random characters can be any ASCII characters that are defined for the Perl programming core.

4.6 Test Case Description

In this section the entire test process will be described. The description of the test cases is relevant for the entire document. Through precisely definition of the test cases, the correctness of the test analysis will be approached in the further works. The test cases here show how the system behavior either in convergence scenario or for normal transaction is.

4.6.1 Packet Transmission

The first test case that will be explored in this thesis is the time estimation and measurements, which is performed between the sender and the receiver in the network by transmitting the packets. The sender, which provides a packet, will send the packet to the destination and also to the backbone network at a measurable time. The receiver will detect the packet at a time as well. It is important for this thesis to find out how the packet will be generated and transmitted to the network and how a computer system receives and detects this packet to confirm the transmission correction.

This test case will be named test case 1 in this document. Before beginning this experiment, the entire systems and devices, which have been involved in the transmission assignment, will be explored. The stations, modules and all buffers and memory locations that a packet visits them will be named and investigated to specify their influences on the convergence time.

In the first test case the client machine (*volare-s0*) will develop a random data payload and put it into the UDP packet. This machine transmits the generated packet to the networks and the packet will be forwarded from the devices in the network to the destination (backbone network). The specifications of computer systems and also the network devices such as switches and routers will be discovered and the time that a packet spends in each modules and stations will be estimated. After the precise measurements, it is possible to verify the time estimation in this experiment. Therefore, the explored modules and network devices help to analyze the convergence time much more precisely.

The test procedure is executed as following steps:

1. Run the Wireshark application in both client and server to capture all UDP packets, which are sent and received the end stations.
2. Run the Perl client and server script programs to generate and send the packets across a socket consecutively and receive the same packet at destination across a socket in server side.
3. Break all applications running after transmit about 10000 packets. (client and server Perl programs and Wireshark capturing)
4. Save the packets captured by Wireshark run on end-stations and the packets generated and detected by Perl scripts in external files.

After these steps the saved files can be analyzed and the recorded times can be measured. The measured time shows the transmission duration. The bit rate during the transmission and also the number of the sent and received frames can be explored. Through analyzing this information, it is possible to estimate the transmission time from the sender to receiver with respect and knowledge about all stations and modules in computer systems and network devices, which have been involved in these situations. This test case and the related analysis have been declared in Chapter 5.

For these normal operation tests and time measurements, different states will be explored. The first form of measurements is performed across a third computer device, which is connected to the client and the server via Secure Shell Host (SSH) and Putty. In other words the third computer system connected to both client and server over a third subnet and controls them remotely. This job has an advantage. The client and server are not involved into any additional appointments, such as displaying the results on an output device like a monitor, which will burden the computer system and influence the test operation performance. The second form is the normal operation of the test cases directly from the related client and server machines. In this case these machines are involved to display the contents of transmitted and received packets on the monitor.

The Putty is a free *telnet/SSH* client program, which can be used in Windows and UNIX systems. Due to use the graphical interface of Wireshark in that third PC, the X11 server must be active. Therefore, in the Putty the X11 options will be enabled. The X11 server will be active through the *Cygwin*, although the aim of this project is not to describe these tools.

Moreover, it is important to add some information in consideration to the test result, thus the times, which the application tools will accommodate and how the times are produced that the Perl program will emit. The Wireshark returns the time in the time of the day format. It means the time will be given as the date of the captured hours, minutes, seconds and microsecond of the time. When both systems are synchronized and the all analysis and time observations are in milliseconds and microseconds, then it is assumed that the two time outputs are compatible and they can be used for analysis in this thesis.

4.6.2 Link and Interface Failures

The link failure is one of the major test cases of the proposed solution that is analyzed in this project. The LACP must be explored, how this mechanism reacts to the link failures and also node failures. The link failure occurs between the two nodes. In a LAG, there are multiple links, which they are participated to forward the frames. If a link fails, the other links of the LAG assume the load that is transmitted across the failed link. Therefore, a link failure can be recovered in a link aggregation.

The second test case that will be investigated to analyze the failover behavior of link aggregation mechanism in an Ethernet network is the exploration of aggregated link failure during the transmission. Another test case is the failure of the NICs and also ports, either in computer systems or in network components. In all these cases, the connection between two nodes will be broken and the other links, interfaces and ports must be assuming the communication between these two devices.

In this experiment the packet generation and transmission into the test environment is similar to the test case number 1. In other words, the client machine will generate a UDP packet and send it to the server machine thus in backbone network. During the test one of the aggregated links, which is named in Figure 4-4, will be disconnected. After analyzing the stored files and transmitted and detected times the failover behavior and the failover time of this system with this mechanism can be explored. This test case will be named as the test case number 2 in this document.

After starting the server program the machine wait until the first packet can be detected and received by the NIC. In other word the program starts the listening to the created socket. Then the *client.pl* is run immediately. The sent frames and the send time are shown after this event on the console. However, these messages are shown very quickly and the information will be saved in the file that the Perl program created to record the packets and the transmission time. The Wireshark at both sides captures the entire received and sent frames. The link can be unplugged before the client sends many frames. However, the operator has to notice the approximated time of the link failure, because the link failure produced by the operator itself. Finally the capturing of interface will be stopped and the frames are imported to the main page of the Wireshark software.

The convergence time of a link failure within a link aggregation is the major exploration in this work. The general question that must be answered here is speed of a system to detect a link failure. The test cases declare how the process will be executed. The precisely measurement is on focus. The Wireshark application and two explained Perl script programs aid to this precise measurement. Furthermore, the complete network was configured as expected in consideration to project requirements. The configuration of the devices is explained completely in appendix A.

The first problem is how the exact fail time will be appointed and another one is the measurement accuracy. As explained before in introduction, the convergence time for this network must be less than 1 second. Therefore, measurement tools must be enough precisely (in milliseconds or microseconds) and the client and server machines must be synchronized. Clearly, for precisely measurements of the convergence time, the hardware time of each sender and receiver has to be synchronized with a third time server. The time offset between these two computer systems must be calculated in the test results.

After ensuring about correctness of the configuration and connections, it is required to send a message from the client to the server. By capturing the packets within the links over Wireshark, it is possible to examine the convergence time of a link failure. The client sends a message to the server and server receives the message. If a link fails, the designated LACP at the side of failed link decides to forward the packets through another active link. All these analysis and test results will be discussed in Chapter 5.

4.6.3 Stacked-Switch Failure

To verify the functionality of LACP on the devices in the Ethernet network, the behavior of this protocol will be investigated on stacked switches. The test of the device for this protocol is achieved, if the designated device unplugged. In other words the device will be removed from the network. Therefore, all related and connected links of this device will be removed. As shown in Figure 4-4, if one of the two stacked-switches fails, one of the two ways that aggregated in the LAG will be disconnected. Moreover, the redundancy between router and computer has been broken.

In such events the LACP must reconfigure the setting and distributes the frames through the active links. When the Wireshark captures the frames, this software observes all latencies and also other frames, which are sent during the transmission to the interface and the all information, can be saved and imported. Therefore, the LACP can be analyzed, explored and tested in this subject.

The test process is the similar to the test process of the link failure. Only instead of the link failure one of the stacked switches will be unplugged. Another way to discover the stacking mechanism in the switches is the reload of the stacked switches during the operation. In this case the reload command will be given in the command shells of the related switch and then this component will be reloaded. These two test cases will be investigated in this document. Finally the results are exported from the Wireshark and Perl programs. The same server-client script files are executed on the backbone network and the end-stations. This test case will be named as the test case number 3.

It is clear that the behavior of the system if one of the stacked switches will be unplugged or reloaded. These two cases will be done separately in this thesis. As explained in Chapter 3.6

the two stacked-switches have been divided into a master and backup units. Therefore, the behavior of each role of these devices must be covered in this test case.

4.6.4 The Behavior of Network Interface Cards

The Linux kernel will be connected to the hardware over the device driver. The device driver is responsible for the entire hardware duties. Therefore, in the Network Interface Card (NIC) all receive and transmit assignments are controlled over the networking device driver. As the results different network device drivers can cause different system and also network interface behaviors.

The aim of this diploma thesis is the analyzing the whole effective issues on the system, when the link aggregation mechanism has been utilized to explore the convergence time of the network. LACP will control all related fault treatments. Therefore, it is important to investigate, how different NICs with various device drivers will react to the link aggregation and LACP either in the normal communication or by a fault case. In this sub-chapter will be explored the both operating case of two different NICs and the results can be compared and analyzed. These results are convergence time difference, bit rate during the transmission and also the stability of the NICs during the communication.

The forth test case in this document try to verify the influence the networking device driver in each computer system with various operating systems. The device driver manage the device at failure cases, although the other tools such as bonding driver and LACP are involved in this work as well. The behavior the various network devices and their drivers can be explored here. How the system reacts to a failure and how quickly the two aggregated NICs will respond to the system requests will be investigated in this test case.

For this project three NICs will use in a computer system. The duty of one of them is not important here. The other two cards will be aggregated in a LAG to perform a single logical Ethernet interface. Here will be utilizing two exact the same NICs to investigate better the differences of various NICs. The Intel PRO E1000 and Linksys EG1032 have been used in this project. The Intel PRO network cards have been used for all previous tests. Therefore, if the same test cases are done for the second model, the obtained results will be compared and the differences between these two NICs can be observed.

The various NICs will be investigated in the test environment. The same test process as the previous test cases will be done here for this case. Therefore, the Perl script programs and also the other tools will be utilized to measure the error detection and error recovery time. However, the system, test environment and also the configuration and parameters remain as the same, but the results of various situations can be differently for various NICs. This test case will be named as the test case number 4.

After installation and test of the device drivers the functionality of these cards will be tests through some easy ARP applications. In the next step, the bonding driver will be configured

for the two cards, which will be aggregated. At the bonding driver the interfaces will be selected and aggregated to a LAG. Finally, the interface information of these two cards will be observed and checked, if these two are configured correct. Now the LAG at end-station has been configured and the other devices found the LAG as a member in the network environment and also as the partner of another LAG at the direct connected network device.

The various tests and test cases are defined in this thesis to discover the ability of the LACP and link aggregation. These four test cases are executed to explore the properties of the LACP. The tests will be done multiple times to achieve an appropriated analysis and statistics. In the next chapter all these cases will be done. The expected events and system behaviors will be investigated before the tests. Finally after multiple tests and experiments for each case, the results and the analysis of all achieved effects and issues will be given and the capability of this protocol can be assessed.

4.7 Analysis the Packet Transmission and Reception

Before finishing this Chapter, it is necessary to prepare the work and collect the necessary knowledge for tests. Therefore, the required analysis and information must be assembled to lead this thesis to obtain correct and precise results. In this sub-chapter all modules and stations that a packet will visit must be investigated, while this packet is sent from client to the server. As explained before and with reference to the Figure 4-4, in this thesis, two Linux computer systems, two Layer-2 switches and also one router are used.

4.7.1 Packet Transfer Line within Linux Computer Systems

The client and server are two normal computer systems that run Debian Linux UBUNTU version 7.4.5 as operating system. It is necessary to find in which stations and/or modules during the transmission a packet is delayed the most. However, as explained before, such information can give the clear information to discover the influences of these modules on convergence time. Therefore, here in this document, it will be analyzed first which modules and station a packet will visit. The operating systems are responsible for system behavior against assignments. The only part that can be differently between various systems is the NICs and its device driver.

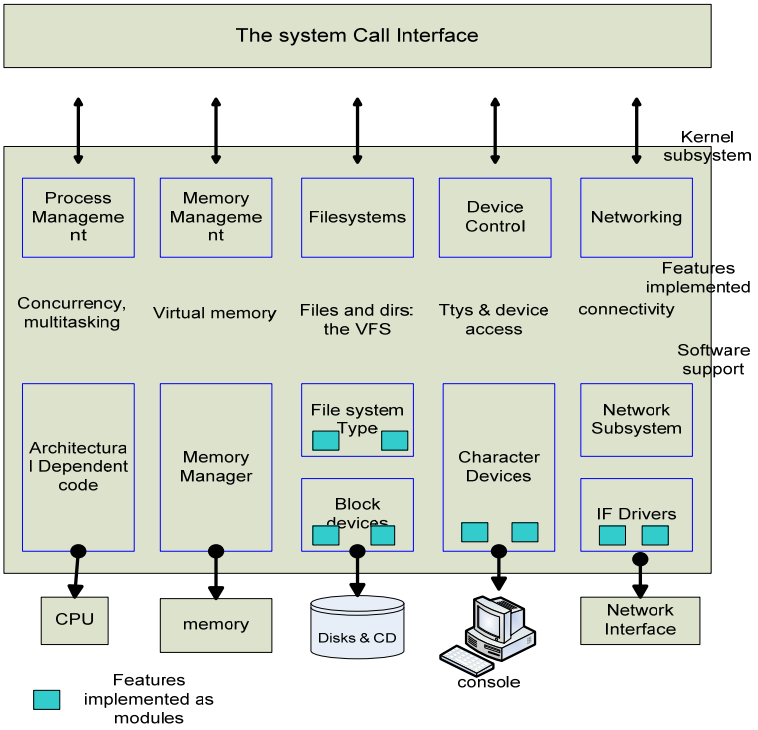


Figure 4-5: Linux Kernel [CRK05, p. 5]

Due to special attributes of the Linux operating system, this thesis will focus on the Linux properties in networking appointments. Figure 5-2 shows Linux kernel block diagram and where the hardware meets the operating system. In this document all modules, blocks and software will be declared, which will affect a packet at operating system during the transmission and reception. It is required to know how the packet will be received and processed by an operating system. Therefore, the networking kernel subsystem, networking software and modules (see Figure 5-2) will be more concentrated in this document.

The NIC is a physical device in a computer to provide the connection to the network and it is responsible for sending and receiving data over the computer network and converting the data to the readable machine codes. Then the device has to form the data in to the exact frame that it has been sent to. The sending and receiving of a frame are two different jobs for the NIC and each of them has its own progress.

The device drivers in Linux are black boxes, which hide the details of the system in other words how the hardware works. The device drivers are made of piece of programming that are separated from the kernel and plugged the device to the system at runtime. The role of device drivers is providing mechanisms. [CRK05, p. 2]

The part of the kernel and also the device drivers that will be discussed in this document are the networking and network interface device drivers. The networking must be managed by the operating systems. The incoming packets are asynchronous, as they have to be identified

collected and delivered across programs, applications and also network interfaces. Therefore, the kernel is the only responsible part for the routing and addressing issues of the incoming and outgoing packets. [CRK05, p. 497]

Each protocol in the networking is like a language. In the Linux kernel each protocol is provided through a module of code, which they provide different services to the socket layer. [Cox96]

The network transactions are performed through the network interfaces. These interfaces can be hardware or software unit. The network driver knows nothing about connections, but it handles the packets. In the Linux operating system the network interfaces are handled differently from other hardware devices. They are not mapped to the file system, such as */dev/ttyx*/. They are named such as *ethx* and the name does not have a place in the file system. In other words the communication between the network device driver and the kernel is completely different.

The network drivers have to support some tasks, such as setting the address and modifying the transmission parameters and maintaining the traffic and error statistics [CRK05, p. 500]. Thus the network API supports these tasks. The network driver put only the hardware header to the packet and the other higher-level headers of the packet will be attached to the packet on networking subsystems and the appropriate applications.

The NICs that will be used in this thesis are two different marks. The first one is the Intel PRO/1000s with the device driver e1000 registered for the kernel. The Gigabit controller 82541PI from Intel has been integrated on this card. The second type is the Linksys EG1032 with networking chipset RTL8169S from RealTek. Both devices are Gigabit Ethernet cards and they support all 3-operation choices 10 Mbit/s, 100 Mbit/s and 1000 Mbit/s. The network drivers request resources from kernel at running time and the driver insert a data structure for each detected interface into a global list of network devices. The *struct net_device* will be defined in *linux/netdevice.h*. At definition time, some other parameters are given in the named file and some resources are given to this device [Cox96].

The transmission and reception of the packets by devices in Linux based computer system is always similar and the skeleton of this structure of network device drivers is given in the kernel of the Linux under the path *drivers/net/skeleton.c*.

The *sk_buff* is the only buffer used by networking. This buffer provides the general buffering and the flow control facilities. Sets of functions in sk_buff library control the attached memory and also modify the list of *sk_buffs* [Cox96]. This buffer is like a stack and methods, such as put and push grow the buffer and also remove *sk_buff* from the list. The tail room is the part, which is allocated when the buffer obtains by the allocation function and the headroom is the part of the buffer that is reserved for further structures that will be added in the memory area as a *sk_buff*.

Packet Transmission

For transmission of a packet, at first the Perl application in the client program, which explained before, will generate a UDP packet. This means that the specified numbers of bits will be randomly generated as data payload. Then this data packet will be given to the socket. The socket is a software block, which is implemented to prepare the data for transmission, whereas the packet is the UDP packet, the appropriated fields such as UDP header will be attached to the packet. After this point the application notices that the kernel will send a packet.

The kernel runs the *hard_start_transmit* method to put the data packet to the sending queue. The kernel put the data in the named queue in the form of "socket structure buffer" *(struct sk_buff)*. This structure is defined in the *linux/skbuff.h*. A socket buffer is just a packet. The *skb* is the pointer to *sk_buff*. The socket buffer passed to *hard_start_xmit*. The interface does not modify the packet. Only some transmission-level headers must be attached to the packet. The transmission function performs some simple checksums on the packet and transmits the data through the related hardware function. The *hard_start_xmit* returns the 0 if it is success. [CRK05, p. 516]

There are three additional methods to guarantee the correction of the transmission. At first the *hard_start_xmit* method will be protected by concurrent call with *xmit_lock*, which is defined in the *net_device* structure. The second important issue is the limited hardware interface memory for the outgoing packets. If this memory is full, the driver will make the kernel aware to stop the transmission with this method *netif_stop_queue* until the hardware is ready to accept new data. The *netif_wake_queue* method will perform the restart of accepting the packets by the hardware after running the *netif_stop_queue*.

The last notice in this area is the timeout of the sending of a packet. The device driver is not responsible for detecting timeout problem. It is only set the timeout value in the *net_driver* structure. If the current system time exceeds the driver's *trans_time*, the networking layer will call the *tx_timeout* method. The method is responsible for clearing the problem up.

Packet Reception

The reception of packets in Linux is more complex than transmission, because the reception of a packet requires more tasks. Therefore, the reception of packets takes more time. It means from the time that a packet is received by the hardware interface until it is ready to use by the upper layer, which takes more time than an application wants to send a packet until this packet is completely sent. There are two methods to receive a packet: interrupt driven and polled. Most drivers in Linux such as device drivers that will be used for this thesis implement the interrupt driven method. [Cox96]

The hardware receives the packet and sends it to the memory. When the packet is successfully received and it is already located in the computer memory, the receive function will be called from the interrupt handler. The receive function receives the pointer to the

packet and also the length of the packet. At the next step this function only sends the packet information to the upper networking layer. The packet will be allocated to the socket buffer through the *dev_aloc_skb* method. This method needs the packet size (length). Some parameters are implemented in the receive function to provide some additional information for the networking block at kernel. For example, the *protocol* and *dev* show the packet protocols and also the interface device, which is received by the frame. Some checksum tasks will be executed to check the receive correction such as hardware. Finally the driver will update the statistics counters such as *rx_packets*, *rx_bytes*, *tx_packets* and *tx_bytes* [CRK05, p. 522]. In Figure 4-6, it is depicted which queues the packet will be met, until it is ready to use for the applications. At first the packet will be queued in the device, then in the socket buffer and finally in the memory to use for the upper layer applications.

The CPU will call the interrupt handler that will handle the received packet, when either a packet has arrived or a transmission of a packet is complete. There is a status register on the physical device, which can distinguish between done transmissions and arrival new packet states. Some functions and methods are called from the interrupt handler to send the data to protocols and manage buffers such as flow control.

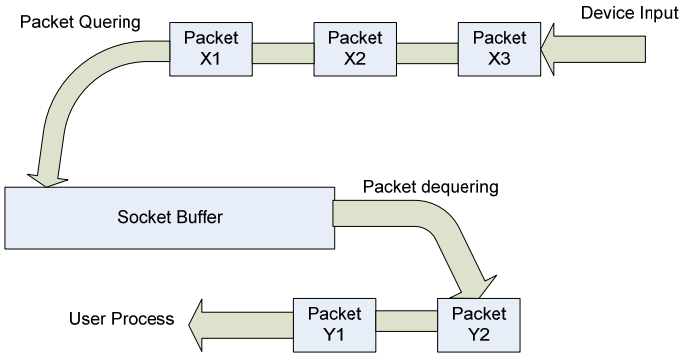


Figure 4-6: Network device data flow

Another state that is interesting for this work is the carrier state that shows whether the hardware is up and ready to use. For example if there is no wire connected to the Ethernet adapter this carrier state will be vanished and the link will go down. The device driver has the ability to modify this state. [CRK05, p.522]

The interfaces that are used in this thesis support the Media Independent Interface (MII), which is an IEEE 802.3 standard, because the NICs utilized in this work must operate with Bonding Driver (see Chapter 4-4) and this driver operates only with interfaces, which supports the MII. This standard declares how the Ethernet transceivers connect with the network controller. Therefore, the Linux consists of the *mii.h*. The device drivers of the utilized interfaces fill the *mii_if_info* structure with information about the physical ID of transceivers. Many additional functions can be run if the device driver supports the MII. Most

of these functions are used by *ethtool* and *mii-tool* at the Linux kernel networking subsystems.

The network device must perform some parameters and information about network protocols. It helps if the system is sensible to the various protocols. These parameters can be flags and variables. The largest data payload size that can be accepted by protocols and the minimum size of packets for each protocol can be named as such parameters. These parameters are necessary for the net-tools like *ifconfig* program. The list of these protocols and suitable parameters are given in RFC 1700 [Rey94]. Any updates in this list can be defined for the Linux kernel as well.

At the end of this sub-section, it is interesting to know how long it takes a packet to be generated until it is sent successfully and how long it takes a packet to arrive until it is detected exactly by an application. These times in the Linux operating system can not be exactly measured but this project attempts to estimate the receive time and also the transmit time roughly. In further sub-sections the other modules and stations that will be affected by the packets in the way from sender to receiver will be discussed. As explained in the Figure 4-4 the data packet is sent from the client system to the server and the packet will be passed through two stacked switches and one router.

4.7.2 Packet Transfer Line within Switches and Router

This sub-chapter has so far tried to explore the packet developing, transmission and also the reception of the packet at client and server machines. However, the only problem is the accuracy and correction of this time estimation. As mentioned before the sufficient result needs the sufficient analysis. Therefore, it will be attempted to find a solution for the precise measurement.

Additional features of Perl script programming provide the ability of precise measurements the time of packet generation. At the server side the time of detecting the packet can be measured in the exact same way as mentioned in the sender. If the time is calculable during this way, then the time division in any modules and stations can be specified. This method is extendable for all other tests in similar Ethernet networks.

As shown in Figure 4-4 two stacked switches and a router are located between the client and the server. These Ethernet network devices are all used to make the network extendable for integration to other clients. Now the question is how long it takes for a network device to receive the packet, forward it to the other ports and to send it out to another device. The switching and forwarding latency in the switches and routers can be estimated by investigating their technical specification datasheets.

At first it is necessary to find out how a switch and/or a router work. There are three methods of switching and forwarding the packets by these network devices [Cis07];

1. *Cut-Through*: The switch reads the first six bytes of the packet as soon as the packet is detected. These six bytes consist of MAC address. The destination will be specified and the packet is sent to the destination.
2. *Store-and-Forward*: The switch saves the entire packet in the temporary memory area (buffer). The switch checks the CRC of the packets and other possible problems. In addition to this the switch looks up the MAC address and send the packet to the destination.
3. *Fragment-Free*: This method is very similar to the cut-through method. The only difference is that the switch saves the first 64 bytes because the most collisions and errors occur within the first 64 bytes of a packet. This method is used seldom in the common switches.

The switches and the router that are used in this project (*Dell PowerConnect 3424 and Netgear GSM7312*) use the "*store-and-forward*" method for the switching of packets. The both devices have 128 Mbyte RAM for the temporary save of the packets.

There are also three physical designs for the Ethernet switches [Cis07]. Shared-memory, Matrix and Bus-Architecture can be named for these three designs. Devices here in this project use the first design (shared-memory), whereas all incoming packets in all ports are saved in the shared device's buffer. For the matrix form the incoming and outgoing addresses are perform a matrix to find the relationship between the incoming packet and the outgoing destination.

The network devices that are used here have the following specifications that are listed in the Table 4-1 and 4-2. These numbers are taken from the device datasheets, which is delivered with the device.

Dell PowerConnect 3424				
Switching capacity [Gbit/s]	**Forwarding rate [Mpps]**	**RAM [Mbyte]**	**Queuing method**	**Data transfer rate [Mbit/s]**
12,8	9,5	128	Weighted Round-Rubin	100

Table 4-1: Technical information of Dell switches

NetGear GSM7213			
Switching latency	**RAM [Mbyte]**	**Bandwidth [Gbit/s]**	**Data transfer rate [Gbit/s]**
Less than 80 μs for 64-byte frames	128	24	Up to 1

Table 4-2: Technical information of Netgear router

If any of these two devices are explored separately, it is obvious that the Layer-2 switches are clear faster than the Layer-3 switches. The more processing appointments of the router can be named as the result. The router must calculate and look up the MAC and IP address of the packets separately. However the Layer-2 switches only look up the MAC address in their address table.

The ports and cables that are used in and for these devices are 100 Mbit/s ports and links. Therefore the receiving and sending of the packets by each port can be calculated easily. The time that a 122-byte packet will be transferred across a port is approximately 10 µs. It means each port that is involved in the transmission of the packet needs 10 µs for any transactions within the forwarding. These actions are receiving the packet into the device and sending packets out of device. However, the 100 Mbit/s ports cannot perform really this communication speed. This assertion has been obtained from many statistics and studies in this field, because of existing header and footer in a frame. Such ports can provide maximum 80 percent of their abilities.

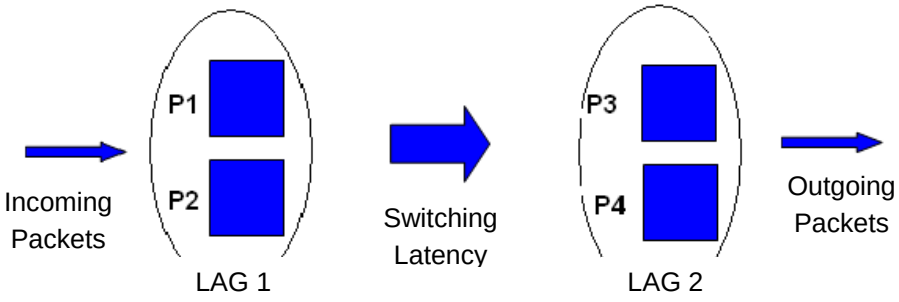


Figure 4-7: Packet procedure within a network device

Another subject that is noticed here is the usage of LACP in both devices in these tests. Two ports that are aggregated to a LAG receive the packets that are collected by the device. As explained in Chapter 3-5 the link aggregation is the optional sub-layer in the link-layer of the OSI reference model. Therefore, the packet collections in one side and the packet distribution at the other side are added to the time estimation of the packets. These processes depend on the processing speed of the device. Hence the switching capacity of the switch and switching latency of the router can be an accurate reference to calculate these processes.

In conclusion, the Ethernet network devices, which are shown in the Figure 4-7, are responsible for forwarding the packets. This figure depicts that packet delivery can be divided into three steps. The first step is the reception of packets into the device from the sender. The square parts show the aggregated ports in this component. The packet reception can be performed within about 10 µs or more. Then the packet must be switched to the outgoing port in the second step. This process can be executed in a few microseconds in switches, but in the router it can take up to 80 µs. The packets that are generated in the test that will be executed

in this thesis are 122 and more. Therefore the packet switching latency can be calculated approximately 160 μs. At the end of the second step the packet is ready to transmit out of the device. Finally the third step is the sending of the packet. This process takes about 10 μs for a 122-byte frame. The ports transmission and reception latency can provide maximum 80 percent of the port transfer speed. Table 4-3 summarizes all information and parameters explained above. The second and the fourth columns depict the time delay during the incoming and outgoing of the frames in a network device. Due to the transfer speed of the Fast Ethernet ports, all delays are similar in this case. The deviation of this 15 μs from the calculated 10 μs is only the worst-case transfer speed performance in a Fast-Ethernet port. The second column shows the forwarding latency in the network device.

	Delay at incoming process [μs]	**Delay at forwarding process [μs]**	**Delay at outgoing process [μs]**
Switch	15	70	15
Router	15	160	15

Table 4-3: Transmission delay in network devices

The precise calculation of the time will be given at description of each test in further chapter. By description of each test the test procedure and each steps of test will be depicted and declared.

Chapter 5 Test Results and Analysis

Up to now the basic definitions and the theoretical parts of this work have been discussed. Now in this chapter the entire tests and results are given. After each test, the arguments are given that can be assigned to declare the results. Finally the analysis will be given, which helps to better realize the ability of LACP and also the link aggregation (IEEE 802.3ad).

5.1 Analysis of the Transfer Time

Estimation of the Transfer Time

The transfer latency between client and server during normal operation is the first test executed to investigate the time estimation for the client server transaction. In the normal operation the *volare-s0* (client machine) generates a packet, put it in outgoing queue and finally sends it to the server (*volare-pc*) across the Ethernet NIC. Through the way of the packet to destination, it meets the switches and routers. These devices forward the packet directly to the destination. All connections from *volare-s0* up to the router are performed with link aggregation to approach higher availability.

The server receives the packet in its Ethernet network interface card. The server passes it to the buffer and finally through analyzing the received frame, it detects the packet protocols and content and passes it to memory device and the higher-level applications.

At the client side, there are two NICs, which are aggregated with the help of the bonding driver (see Chapter 4-4). The bonding driver works as a device driver for the *bond0* interface, which is a virtual interface and shows only the LAG that consists of two NICs. However, this driver is located higher than the NIC's driver. In other words, the network card driver works and controls the device activities, but the bonding driver controls and manages these two cards and the received and transmitted data from them. As explained test environment remains the network shown in Figure 4-4.

The transfer time estimation analysis is given in Table 5-1. As described in Chapter 4-7 the packet visits various buffers, modules and station on the way to the destination until it is

detected and ready for use for the application layer in the server machine. The data payload generation is performed through the application during the Perl program. This process can be provided in a few microseconds, but it depends on the processor and also on the hardware abilities of the computer system. After generation of the packet, the Perl application requests a socket from kernel and prepares the packet for transmission. The packet will be sent to the transmission buffer and the kernel prepares the environments and related parts to transmit the data. The protocol's header and also the related physical and IP addresses will be added to the packet within the Kernel subsystems. At this time the Perl program registers the time of successful packet preparation in the time register. However, the packet transmission from the interface cannot be confirmed for the script program. The processes, which a frame will be transmitted across the interface card has been declared in Chapter 4-7-1. These appointments can take about 10 to 20 μs for a 1000-bit packet.

The second point that a packets must wait for the hardware transmit it is the network interface card. The 1 Gbit/s interface cards used here cannot really perform this transmission speed, as the cables and the other partner devices in the network are all fast Ethernet devices. However, these cards have the ability to operate in Fast-Ethernet speed too. A fast Ethernet transfer time is about one second for one million bytes. However, this transfer time is gained theoretically. The ability of the hardware in real transfer is about 20 percent less than the basis. Clearly, the transformation time for a 122-byte packet is the theoretical 122 μs, but in the real operation the transfer time can be approximately 150 μs, thus 20 percent more. The PCI Bus inefficiencies can easily increase this time (the network interface cards work on PCI Bus on the computer systems).

After transmitting the packet from the Linux computer system, the packets will visit the network devices through the destination machine. The latency of each device can be calculated from the device datasheets. The calculated times in Table 5-1 has been obtained from the analyses of switching latency in 4-7-2. The interesting point is the reception time from the server machine. The integrated Fast-Ethernet card in this machine will receive the packets within 150 μs as well. The reception latency in this network interface can take the same as the transfer time, which is explained above for transmission. After arrival of the packets in the interface card, the buffering of the packet and finally the detection of the packet format and protocol can take more time than the transmitting the packets. However, the reception of the packets takes more time as transmission as described the reason in 4-7. These times depend on the computer system abilities and can be estimated from 10 μs up to 20 μs more than the transmission latency.

Transfer latency for a 122-byte packet	Time estimation [μs]
Packet generation	10 to 20
Average packet transfer time	150
Transfer latency across the Ethernet cables (7 meters)	40
Switch transfer latency	100
Router transfer latency	200
Frame reception at destination	150
Packet detection within the Linux kernel	30 to 40

Table 5-1: Analysis of the transfer time

Table 5-1 lists the transfer time in each node. The sum of the time latency here is about 700 μs for a 122-byte packet transfer. The correction of the analyses here can be explored over the executed tests further in this sub-chapter.

Measurements of the Transfer Time

At first test the packets will be generated and sent to the destination (*volare-pc*) and the received packet will be detected and analyzed at the server side. The packet size is 122-byte. It consists of random characters, which is filled in the data payload by the client Perl program. In this test case, 9233 packets have been generated and sent to the server. These packets have been transmitted in a specified time. In test case 1 it has been specified to transmit about 10000 packets for each test. The Perl script program is not so accurate to stop the transmission after 10000 packets. Therefore, the operator will be stopping the program approximately after about 10000 packets. All of the generated packets are sent completely to the destination and all of them arrive successfully to the receiver.

Table 5-2 consists of the first results, which are taken from the first test analysis. As shown in this table, four values are saved and given. The Wireshark gives back the time of capturing data at the interface. It means the time that Wireshark gives back, is the time that the packet is completely sent or received by the interface. Therefore, there are two columns of values. One is the sent time that is captured by the Wireshark at sender and another one is the received time, which is the capture time of the packet through Wireshark at receiver. Other times are the send and receive times that were saved by the script programs in an external file. Therefore, there are two other columns to show the send and receive times, given by the Perl client and server programs.

	Transfer latency observed by Wireshark [μs]	Transfer latency observed by Perl [μs]	Difference between observed time at Wireshark and Perl in sender [μs]	Difference between observed time at Wireshark and Perl in receiver [μs]
Average value	515,93	614,258	11,2172	29,549
Max values	837	15.073	12,9	1.323,7
Min values	512	427	-1.575	22

Table 5-2: Measurements of the transfer time

Table 5-2 consists of difference between the send and receive times that are taken from Wireshark captures. The minimum and maximum values are given bellow the average count. As explained above, these average counts are achieved from 9233 packets, which each packet has a transmission and arrival time. According to the results and synchronization, the arrival time is after the send time.

The second column comprises the average of difference between the send time of the socket in the client machine, which runs the client script Perl program and the receive time of the socket at the server machine. The server runs the server Perl script. The time that a packet is successfully generated and put into socket is observed as send time in the *volare-s0* and the time that a frame is completely detected and received by the socket in *volare-pc* are observed as arrival time. The difference between these two times can be assumed as a different time from generation and detection by the Perl application. These observation times are saved in external files immediately after generation and detection processes through the Perl programs. Table 5-2 includes the average of these differences and shows the maximum and minimum of the counts amount as well.

The interesting case in the second column is the big difference between the maximum value and average value. In consideration to the analysis of the client and the server behaviors, it is obvious that at the beginning of the transaction between the client and the server a little instability can be observed. It means that the time, in which a packet generated and available in the socket and the time of reception by the socket in the server side is longer than expected. In other words in comparison to the times of capturing the frames by the Wireshark, this interval time is longer. At the other side the Perl application run on the server requires longer time as expected to detect the received packet and establish the arrived frame. However, after about 60 to 100 packets it becomes stable and approaches the average interval time. This case can be seen in the other similar tests.

The third column in Table 5-2 shows the time difference between the time that is recorded by Perl application and the time established by the Wireshark at client side during the

transmission. The average value depicts that the Wireshark will be informed sooner than Perl application that a packet was completely sent. However, at the beginning of transmission the Perl script program is sooner. This has the exact same reasons as case that was described in the last paragraph. Therefore, it can be seen that the minimum value of this time interval is a negative number.

The forth column is the same as the third column but at the server side. According to the explanation of that column, this column shows the average value of the difference time between the detection of a packet at Perl application and the reception registered by the Wireshark. The only difference between these two columns is the difference between the server and client duties. Moreover, the packet receiving is completely different than the transmit packet. The difference is explained in 4.71. As a result, the capturing time at Wireshark is sooner than the establish time at the socket of the Perl script program. As shown in Table 5-2 after capturing the frame by Wireshark, it takes about 22 μs until the socket detects the arrived frame.

The interesting point about the third and forth column is the processing time within a Linux computer system. In consideration to the Wireshark description in Chapter 4, it is clear that the Wireshark is the first application that detects the received packet. As explained in the sub-Chapter 4-7-1, the receive times is longer than the transmit time because the reception is more complex than the transmission. The results that can be seen here confirm this theory. The average interval time at server side is approximately 22 μs, although this interval is about 10 μs on the client machine. In conclusion, if the time estimation between the packet generation and the packet detection will be measured, the analysis shows how the time can be divided.

In conclusion to the first test case, it is necessary to allude to the test results. This test case was analyzed more than one time, although the tests results remained the same. The similar average interval times, minimum and maximum values were obtained. The only interesting point in some cases was the difference of numbers of the generated packets, sent packets and the received packets. In some cases a few packets were generated and recorded in the Perl client program but they were never transmitted to the destination. The reason for this problem is described quite a little in sub-chapter 4-7. When an application wants to transmit data, it will send its data to the network device driver. However, if the network interface is not already useful or the transmit buffer is full, the network driver asks the kernel not to deliver any more packets until it is ready to transmit new data. In other words, the Kernel stops the packets to forward to the device buffer. As the mentioned reason, sometimes a few packets at the beginning of the transmission will be generated but not transferred across the network interface.

	Transfer latency observed by Wireshark [µs]	Transfer latency observed by Perl [µs]	Difference between observed time at Wireshark and Perl in sender [µs]	Difference between observed time at Wireshark and Perl in receiver [µs]
Average value	9 238,482	24 182,11	13,5065	16,592938
Max values	9 342	156 768	13	14,754
Min values	9 234	9 250	12	23

Table 5-3: Measurement of the transfer latency

For the second point of test case, which is defined to analyze the time estimation between the source and the destination, is the executing normal transaction between two end-stations and observing the packet procedure and flow within the pre-specified test network environment. In this case the results will be displayed on the monitor of the *volare-pc*. It means the third remote computer will be ignored, which is used in the last described test. In this case the test analysis shows if there are any influences when the computer system is involved in other assignments, such as displaying the results on monitor and some other memory managements.

Table 5-3 shows the same values, account and description as the last table. The only difference is the content of the table. In the mentioned state, the same test procedure was executed. During the reception the packets the *volare-pc* displays the status of the communication on the monitor. In other words it shows which packet number is sent through client and which packet number will be received. Furthermore, the content of the detected frame is displayed on the monitor.

In this test case a 122-byte packet has been sent inclusive the header and footer bytes, which are attached to the data payload. The transaction will be executed unlimited times. After approximately 10000 sent packet the transmission will be broken. The transmitted and arrived packets that were captured with Wireshark are saved in a readable file.

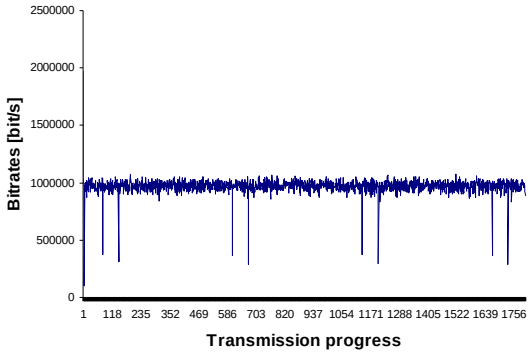
After analyzing the recorded files and packets in the mentioned files, it is clear that all packets, which are generated through the script file, have been transmitted successfully and that also, all these packets have been received completely at the destination. The only interesting point is that the receiver could not save all received packets. Although in this test 8092 packets have been established as generated and transmitted frames, the server could only display 5721 packets. However, as explained above, all transmitted packets have been receive in receiver and the Wireshark run on server machine has been detected all packets. Due to high bit rate and also the many processing assignments in the server machine, this computer could not display all the frames on the monitor.

The progress duration of frames during the way from network device driver and the Perl application can cause this trouble. After receiving the packet in the NIC, the device driver will pass the frame to the Linux kernel to be used by other applications and kernel blocks. In this step all packets has been detected. Therefore the Wireshark has been captured all transmitted packets. After detection of the packet, the packet will be saved in the memory and if an application such as console wants to display a data, it will request to take the pointer to the saved data and pass it to the display driver. For all these progresses the data must be buffered in various places. The Wireshark has captured all packets, whereas Wireshark has the opportunity to get the packet earlier than other application in the kernel and it has an access to the device buffer. However, the socket in the Perl program is located in upper levels at the kernel view.

Another interesting situation in this test case in comparison to the last explained test is the average time from the sender to the receiver. The average value in the second column show the packet's transmission from client to the server takes 9230 µs, although this value was 550 µs in the last case. Except of the instabilities of the reception and the display of the contents of the packet on an output device, the time synchronization can cause this huge difference between the two same test cases in the same test environment. The acceptable time offset for the client and server machines are about 5 ms and unfortunately within this experiment the synchronization offset between the client and server machines with the reference timeserver was near the most acceptable offset.

At the client machine, it takes about 12 µs between the packet generation and the successful packet transmission. In comparison to Table 5-2 it has not changed. However, the server Perl program saves the important situation. After checking all saved packet in this file and comparing them to the captures and transmitted frames recorded in other files, only the conformed lines are analyzed and the value in the third and fourth columns are resulted to this. It is clear that some instability is obvious in these results but as the given arguments above, server Perl script with a big delay saves some packets in the file. After monitoring the results, some packets are detected by the socket exact within the similar time as Wireshark with only 23 µs. These results are the same as shown in the Table 5-2. Therefore, the instability and weakness of the *volare-pc* machine and also the high-speed transaction between this machine and the *volare-s0* causes these troubles in this test case.

In conclusion to the comparison of these two tests, it can be argued that if the displaying of the packet is abandoned, also the size and speed of packet transmission is reduced. The mentioned instability will be abolished and the duration time between sender and receiver decreased. To confirm this argument, the test has been executed another time without displaying any received data on the monitor and the result was exactly as expected. All generated data could transmit successfully and the server Perl program received and saved all packets in the appropriate file. In other words, no packet was lost and also there is no instability observed on the system side. In this test 3100 packets were sent and the socket receives all of them, thus the Perl programs detected all transmitted and received packets.



Average bitrates: 970,161 kbit/s

Max bitrates: 1968 kbit/s

Min bitrates: 272 kbit/s

Figure 5-1: Bitrates progress

For the last pre-analyzing the behavior of the link aggregation, the bitrates progress of the normal transmission will be investigated. To explore this progress, a normal communication between the client and server will be executed. A 228-byte packet will be transmitted to the destination. The server in a few sub-seconds will receive each two following packets. The received bits and the time of the reception have a relationship with each other to provide the bit rate. After calculating the bitrates for each packet, it is possible to draw the related chart and show the bitrates progress in this test. Figure 5-1 depicts the affair explained above. The average bitrates is 970 kbit/s for 1802 transmitted frames. The maximum and minimum values are given as well.

5.2 Convergence Time in Response to the Link and Node Failures

In this sub-chapter, the main analysis and measurements are given, which are the major arguments to assess the LACP, if the suggested high availability solution is sufficient for usage in the safety critical voice communication systems. Therefore, in this section the issues such as error detection, fault treatment and error recovery, which influence on the convergence time, will be discussed and explored additionally.

5.2.1 Analysis the Effective Parameters on Convergence Time

Error Detection

As explained before, the LACP were utilized in the pre-specified communication environment, to increase the availability of Ethernet network and thus reduce the failover

time in this environment. In the safety critical voice communication systems, the high availability has another acceptation. In such systems the high availability, which has special meaning in the definition of availability, must be utilized to protect the network from unexpected events and failures. Therefore, to gain appropriate statistics and also obtain higher availability more than normal modes, it is necessary to set a technology, which helps to pretend and maybe to converge the failures faster than normal operation.

As the result, the LACP will be assumed to achieve the failover requirements of the network. However, it is not clear, if the results and measurements guide the user to utilize this mechanism in the future. In further parts of this thesis some special tests and researches will be accomplished to approach the aim of this project.

In this section, Figure 4-4 will be referred for tests and analyses, as the test network environment. All specifications and settings remain unchanged. The client computer system, two stacked-switches and also the routers utilize the LACP. The server machine simulates the provider or a backbone network.

As explained in Chapter 3-5, by the use of link aggregation and LACP, it is possible to increase the availability of the system. The link aggregation helps to make a multiple point-to-point connection. Therefore, if any of these links fail, the system can recover it faster than normal. Other aggregated links in the LAG can assume the load on this connection. By the use of LACP all aggregated links are involved in forwarding the data. In other words, if a device sends the packet to its partner through the LAG, all aggregated links in that LAG share the forwarding load.

Moreover, in Chapter 3 some protocols and technologies are mentioned, which are working as primary and secondary devices. Therefore, if the primary link fails the secondary replaces it. However, such mechanisms can be effective, but the error detection and error recovery can take much more time to reduce the number of lost packet as expected. Moreover, such protocols with the big amount of packet lost during a failure cannot be used for safety critical applications.

In Chapter 3.5 LACP and IEEE 802.3ad has been discussed. Therefore, this section will be referred to this chapter, when some special information will be given for the analysis about LACP reactions. To analyze the behavior of this protocol to a link failure, at first the error detection will be explored. How a device, which is utilizing IEEE 802.3ad standard, will detect that a link fails.

In the network environment, the two stacked-switches and also the router utilize the link aggregation. The router has a LAG consisting of two links, which connect the router to the stacked switches. Two stacked switches have also two LAGs to connect to the *volare-s0* in one hand and the router in another hand. The *volare-s0* (client machine) is using a LAG to connect to the switch. All these LAGs are controlled with the LACP, which works parallel with the related LAG in all mentioned devices.

As explained in the definition and description of the LACP and also in Clause 43 [IEEE8023ad], the LACP is the only responsible unit for controlling and checking the link status and also the packet ordering and packet lose in this mechanism. The LACP checks the link status frequently by a specified interval. This interval is set during the configuring LACP on the related device (see Appendix A). In switches and routers this interval can be set between two predefined values. The timeout can be set as short or long terms. If the timeout is set as long term, which is predefined as default by network devices, the LACP checks the port status and link availability each 30 s. However, this interval is longer than expected. If the error detection takes within 30 s, then the high availability of the network cannot be assessed. However, there is another term that can be applied for the LACP timeout. If the timeout is set as the short term, then the link status will be checked every 1 s. Therefore, the network device will find out about the error in approximately 1 s at the worst case.

A counter provides this timeout that will be run at the time of setting the LACP. By each timer timeout, the LACP sends a LACPDU (LACP Packet Data Unit) to the aggregated ports in the LAG. The aggregated ports will answer this request and give some parameters, such as link status and the partner status back. If after this transaction, the LACP learns a link status down, then it will inform the system to forward the packets only from active ports. This will help the system to decrease the packet loss as much as possible.

Another error detection possibility is the physical layer at each network device. If a connection is cut, the involved port will inform the system that it has no connections any more. Therefore the system controller, which is integrated in any network devices, will alert all blocks and algorithms, which used this port for executing their operations, not to use this port. This appointment will be performed through a status change report.

The error detection in the client machine has been defined differently from the network devices such as router and switch. The computer has a definitely stronger processor. Therefore, it can perform the error detection with the better time resolution. As mentioned in the Chapter 4-7 the device driver in an operating system is responsible for control the operations of the hardware. Moreover, the client machine in this example is a Linux computer with three NICs. Two of these three NICs are aggregated and perform a LAG. The bonding driver (see Chapter 4-4) in this machine works as the device driver for the link aggregation group. In other words, it is responsible for handling the failure occurrence in this interface (*bond0*, LAG).

The bonding driver in Linux operating system can be configured for some integrated network interface cards. As explained in Chapter 4-4 the bonding driver has more than one operating modes. One of them is the LACP mode, which is used in this thesis as the operating mode. In this mode the specified NICs will be aggregated in a LAG but in contrast to the protocol specifications only one of them is active in due time. Moreover, the bonding tool is specified for more operating modes and it can work with only one active interface. In this operating mode the LACPDU will be generated and transmitted by the bonding driver to the aggregated interfaces and the status and information of the partner of this link aggregation will be

reported to the Linux kernel. Therefore, if the active link fails, the bonding tool will reconfigure the routing and status tables and activate the other aggregated interface. As the result, if the inactive link fails the network finds any failure and remains in the working status.

By definition the LAG in the bonding driver, some parameters will be given. These parameters are the name of group, the operating modes (here LACP) and finally the MII (Media Independent Interface). The MII is an IEEE 802.3 standard describing how Ethernet transceivers can interface with the network controller [CRK05, p. 540]. The MII in this driver operates like the *mii-tool* of the Linux itself. The *mii-tool* in Linux checks the Ethernet interface status during the operation and save some statistics and also link status. If an interface status changes into the down and it stops working, the *mii-tool* will inform the operating system about this event and register the event occurrence in a log file.

As the result, the parameter, which will be set for the MII, shows the time interval in sub-seconds between every two-link status checks of the aggregated link in the bonding interface. In other words this term shows how often the bonding driver will check the status of the aggregated interfaces. This parameter is configurable and the count can be set in what the system need. A common interval setting is 100 ms, which provide a frequently status check for every 100 ms. It is absolutely adequate for the safety critical applications. However, it can be set to 50 ms as well. The lower rates can influence the ability of the operating system. [Dav00]

To summarize this topic, in the test network environment that is used for this thesis, there are three error detection sections. If a link fails between the client machine and the two-stacked switches, then the both switch and Linux computer system can detect the failure. However, the Linux bonding driver has a shorter time interval to check the link status, but the error detection depends on the time of occurring the failure. Therefore, the error detection can be guaranteed in less than 100 ms or in another case in 50 ms (MII parameter). The switch can detect the failure as well. If the failure occurs near the LACP timeout and this time distance is less than the MII timeout, then the error will be detected by the switch.

The connections between switches and router can fail as well. These network devices can detect the error. There is here exactly the same situation such as connection fail between switches and client machine. Each device that has the nearest interval timeout can detect the connection failures. Therefore, the error detection between these two devices can be performed less than maximum one second.

Error Recovery

As explained above by describing the bonding driver, there are two methods to monitor the port status, the ARP and MII. In this case the MII will be used. At the definition of bonding driver, except the 802.3ad-operating mode, the *carrier_use* must be defined. The parameter 0 is specified to it. It means the MII and ETHTOOL in the Linux will assume the link status

monitoring. The *"miimon = 100"* has been given to specifying the link status monitoring time interval. All these parameters will help to learn the link status. Due to the fact, at the beginning of the bonding driver all aggregated interfaces are assumed as up.

The bonding driver selects one of two aggregated links as the major interface and will pass entire packets only over this link. The selection method will be specified by the HASH policy. If the active link goes down and the driver learns its inactivity at the monitoring timeout, then it reconfigures the Hash policy and passes the packets over another link in the LAG. The both slaves in the 802.3ad-operating mode are active but the packets are only transmitted over one of them. Therefore, the recovery of the mentioned failure cannot take much more time. After specification the ink status down from MII of the Linux kernel, Hash policy will be run to select the second link as the transmission port. However, in this case that declared in this thesis, there are only two active interface aggregated in the bonding tools. Therefore, immediately after detecting the link failure the other link will be replaced to this failed link and the transmission goes on.

It can be assumed that the link failure will occur within the *miimon* parameter of bonding driver, thus during the 100 ms monitoring interval. If this event occurs at the first of this time period it can be taken more than 100 ms. However, if this event occurs near the end of monitoring period, then the recovery of this error cannot take more than a few sub-seconds. Clearly, the link failure occurs at 99^{th} millisecond of the monitoring interval. Then the system can be recovered in less than 10 ms.

Fault Treatment

Up to now, the entire error detection and error recovery has been explained. The network environment, which is described in Chapter 4 are the points of concern in this thesis. The aim of this work is to reduce the convergence time. Not only were the connection status at the network focused but also the behavior of the device in case of failure and also the failover time by the use of the link aggregation.

The safety critical voice communication systems must support the five nine percent (99,999%) availability. The convergence time of such systems must be kept as low as possible. Moreover, the system, which utilizes the link aggregation, has some restriction to provide this mechanism. As the result, the exploring of the fault treatment of this mechanism is an essential point for the aims of this work.

The LACP is responsible for all disturbances in the system, which comprises of LAG. The link aggregation control unit consists of all information about a LAG and its partner group. It must update its information about the ports and also the frame distributor and collectors frequently. As explained in Chapter 3-5 the link will pass the received frame to the link aggregation layer, which is the same for all aggregated links and the control unit can check all progresses that are done for this frame until it leaves this layer (see Figure 3-3). Moreover, all these concerns will be executed for the frames, which are given to this layer to send to the

destination from related port. Then the LAC is obviously the most important unit in the link aggregation for the fault treatment. The LACP is the protocol, which has been defined to operate in LAC.

The appointments such as load sharing, frame reordering and also the connection establishment between the MAC sub-layer and upper layers are in responsibility of the link aggregation control protocol. Therefore, this unit will investigate the status of the network connections across passing a LACPDU. The transmitting interval of this data unit has been explained in sub-chapter 5-1.

If the LACP finds out that one of the aggregated links is down it will remove the related control parser/multiplexer from the link aggregation sub-layer and then the aggregator will remove the information of this unit from its aggregator parsers and multiplexers. After that the frame distributors and collectors know that this aggregator is not available any more. As the result, no frames will be distributed and collected to and from this aggregator and so the failure will be recovered.

In addition to the failover behavior of the link aggregation, if a link is added to the LAG a related control parser and multiplexer unit will be created to connect this physical port over MAC sub-layer to the link aggregation sub-layer. The created control parser/multiplexer will be connected to the aggregator. If this link was located to the aggregator and the aggregator knows its information then the LACP will only update its status for the aggregator for the use of frame distributor and collector. However, if this link is new members of the LAG, then the link aggregation control unit will create the aggregator parser/multiplexer and attaches it to the frame collector and multiplexer.

In conclusion, the fault treatment of the system using LACP can be referred to all above described concerns. In consideration to the test results, it can be approved that the link aggregation can perform the adequate convergence time in case of link failures and the recovery time is not more than assumed value.

5.2.2 Link Failure

In consideration to the mentioned proposition, the confirmation of the failover behavior of the link aggregation will be evidenced in the following tests. In these cases the link failure for all devices will be tested to declare the error recovery of the network environment. Figure 4-4 and the test case 2 will be referred for these tests. There are two important aggregated links shown in figure 4-4. They were named as the L3 and L4. The following tests will assess the behavior of the network if one of them is disconnected. As explained in bonding driver, one of these links is inactive and it is standby to assume the burden of the network if the active link fails. The following test shows that the L3 connects the inactive interface to the network components and the L4 connects the active interface.

The first case will be executed for the Link number 3. Clearly the connection between the *volare-s0* and the master unit of stacked switches will be disconnected to observe the event influences at the network and packet transmission. Table 5-4 shows all the important information about this test.

After starting the programs and transmission the L3 (see Figure 4-4) was disconnected and the after a few seconds the transmission was stopped. The entire transaction in this case took about 4 seconds. The client Perl program has registered 1802 sent packets. The packet size in this test is 228 bytes. The Wireshark application has captured 1802 packets as well.

At server side the Perl script program has detected and registered the entire 1802 transmitted packets. It means there are no lost packets during the test. This shows the inactivity of the interface, which has been connected to the stacked-switches with the L3. The evidence of this theory is the next test, when the L4 failure will be disconnected and the packet transmission correction will be investigated.

Table 5-4 shows that the 1802 228-byte packets have been sent and the exact the same number of packets and bits has been received at receive. The average bitrates during the 3,407227 s transmission runtime at sender and receiver are not the same. Since the duration time at receiver is a little longer, nearly here in this experiment the reception duration is 3,417781 s. The link number 3 was not an active link. Another similar test, which has been executed after this test, has confirmed the inactivity of link 3 in the network. At the second test, 1964 packet with same size as the last test were transmitted and all these packets has been received and registered successfully at the receiver. The difference between average bitrates at sender and the average bitrates at receiver is the duration time at sender and receiver. This experiment has been repeated multiple time and the similar results have been obtained two of them are shown and given Table 5-4.

Test duration [s]	All transmitted bits	All received bits	Average bitrates at sender [kbit/s]	Average bitrates at receiver [kbit/s]
3,407227	3 286 848	3 286 848	964,6695	962,5296
3,780037	3 582 336	3 582 336	947,6986	947,7182

Table 5-4: The inactive interface failed, the system detected any failures

Due to the test results that were obtained in last two tests and because of the no error detection at both sides of transmission, the error recovery is out of the question here in this case.

To assess the failover time of the network by the use of link aggregation, the failure in link number 4 (see Figure 4-4) will be explored. Due to the results of the L3 failure, the active status of this connection is approval. The similar test, as the last two tests will be done here. The transmission will be started, after a while the link 4 will be disconnected and then after a couple seconds the transmission will be stopped. In this case the packets with the same size are transmitted over the network and the *volare-pc* will confirm and register the packet reception. This test has been repeated multiple times as well. Table 5-5 shows three experiments to obtain the failover time. The first column shows the average time, which the system has been spent to transmit a packet for each test. The convergence time of the system in each experiment has been given in the last column.

	Average time of the packet transmission [ms]	Count of transmitted packets	Count of received packets	Count of lost packets	Convergence Time of the system [ms]
1	1,91	2378	2330	48	93,837
2	1,907	2082	2029	53	104,812
3	1,946	3442	3399	43	85,385

Table 5-5: Multiple experiments to obtain convergence time of the system link failure

The transmission duration is approximately 4 s for the first try. The client Perl script program has generated 2378 228-byte packets. The Wireshark application on sender has captured exact the same generated packets. It means all generated packets could be sent successfully at sender side. However, the receiver only receives the 2330 packets. As the result, 48 packets have been lost because of the link failure.

	Average time of the packet transmission [ms]	Count of lost packets	Convergence Time of the system [ms]
Max Value	1,902	55	104,812
Min Value	1,96	39	76,455
Average Value	1,9293	47,6	93,03

Table 5-6: Convergence time of the system link failure (average value)

These 2378 packets are transmitted across the aggregation group within the 4,540056 s. Due to the link failure these packets are sent but the systems did not know that the link is not more up. After detecting the link down through the MII status report as explained above, the system selected another active link for transmission. For this experiment, the 2378 packets

were sent. At sender this interval time takes in average 1,91 ms. However, at some points it can be observed that the interval is 7 ms.

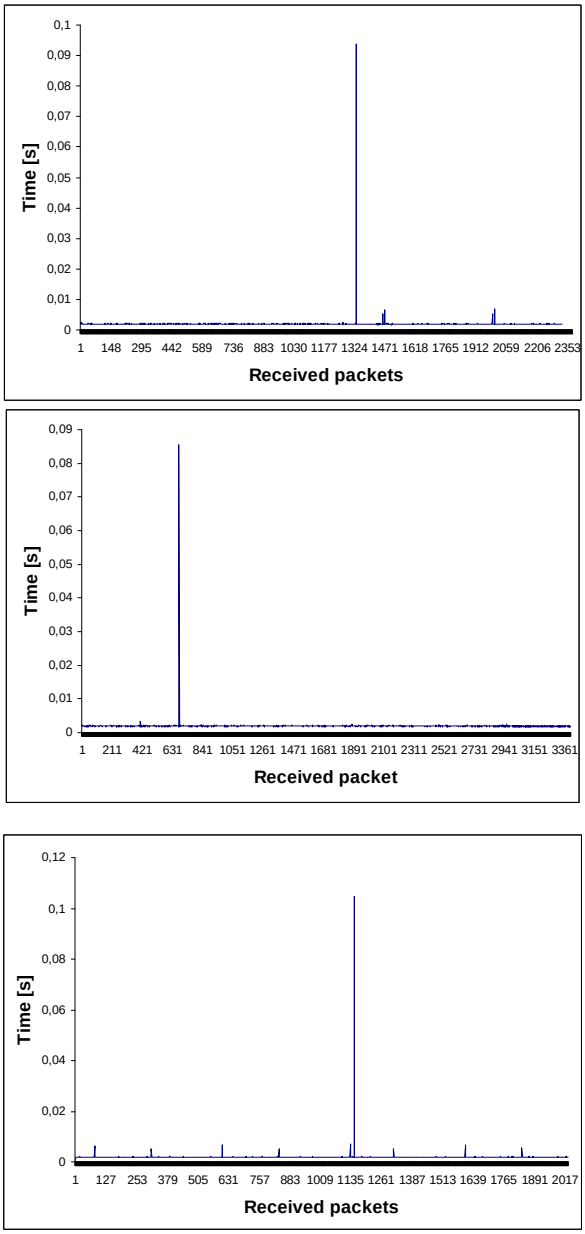


Figure 5-2: Convergence time in case of link failure

The communication time is 4,540008 s. 48 packets have not been transmitted to the destination. Therefore, a few times in between has been missed and no packets were received. In consideration to the Figure 5-2 the sender could not transmit these 48 packets for approximately 90 ms. The content of the generated and received packets, which are saved by

the Perl script files at client and server machines show that in this 94 ms 48 packets has been lost. Furthermore, Table 5-6 shows the average values of multiple times that the test has been executed and all the results. As seen the maximum convergence time in this case is not more than 105 ms, which is absolutely acceptable for the project specification. During the tests maximal 55 packets will be lost and the bitrates is not less than 900 kbit/s and not more than 950 kbit/s.

This subject can be confirmed by the average interval times at sender and receiver. The average time at sender as mentioned above is 1,91 ms and if the system is 93,8 ms down, then 49 packets will be lost and the 48 packets lost in this case can evidence for this event.

Another interesting case is the convergence time at *volare-s0* in occurrence of the failure for the main transport connection. This recovery time of the system is less than 93,8 ms, because the failover time is 93,8 ms. It can confirm the theory about recovery time for the bonding driver that explained above. The difference between two packets 1334 and 1335 at the receiver shows the convergence time of link aggregation. The 1334th generated packet is the same as the 1334th received packets and 1383rd packets at sender is the same as 1335th packet in receiver. It means all the 49 packets are lost in the one distance.

The mentioned analysis can be declared as the view of bitrates for this transmission. Figure 5-3 depicts the bit rate values during this test. As shown in this figure the bitrates will be raised at the beginning of the transaction between sender and receiver. After a couple packets the bit rate became stable and remains in a determinate area. However, as depicts at the failure time the bit rate becomes zero for a while until the connection is established again. After that the bit rate came back to the determined area. The average bit rate in this test case is about 940 kbit/s.

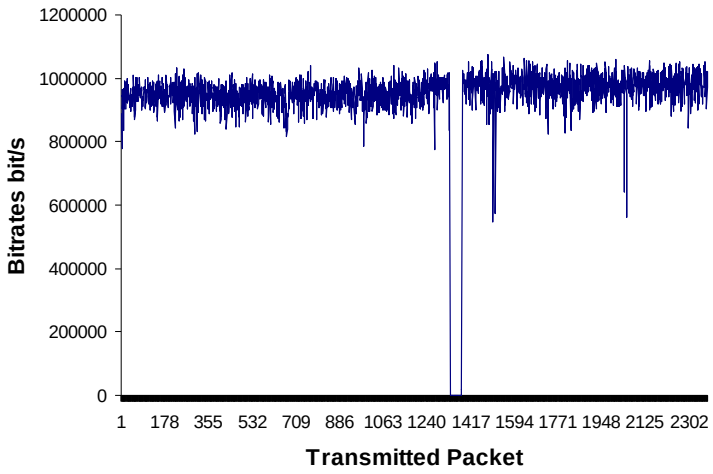


Figure 5-3: Bitrates chart during the transmission

To assess the test result correction in the last declared test case another same test has been executed to confirm the error recovery experiment by the use of link aggregation. In this test 2082 packets are generated and sent to the destination. The L4 was disconnected and after a while the test was stopped. The receiver has detected 2029 packets, 53 packets were lost during the link failure.

As expected the sender has sent in average each 1,91 ms to the receiver exact the same as the last case. 53 packets were lost and this results about 101 ms failover time. After concluding the results, the system shows that the system required 104,8 ms to recover the failure. This value proves the theory of convergence time that can be a little more than the 100 ms. The average bit rate is here 940,268 kbit/s.

The last test case that will be investigated in this sub-chapter is the check and test of adding a member to the LAG. When a link is disconnected then one of active members of a bonded interface group will be removed from the list of interface table. However, if the disconnected network card is attached and enable to the group again, then the information of this card will be added to the list of aggregated interfaces.

In this test process, the packet transmission will be executed from the client machine to the server machine. After a few seconds the active link will be disconnected and after a while it will be connected again. This process will take more than previous tests. The test results can be presented after analyzing the failover behavior of this network environment through the exploring the files, which are saved and recorded by the Wireshark application and the Perl script programs.

For the first experiment, the test process has been executed and 3442 packets were generated at the sender. The receiver has detected only 3399 packets. The transmission duration was approximately 6,6 s. The packet size here is 228 bytes again. The Perl script program has generated the packets every 1,94 ms exact the same rate as the previous experiments.

Furthermore, the second test was done for the same process and the same results have been obtained. The interesting point for this test case is the adding the disconnected link to the system again after a few seconds. After analyzing the results it is clear that when a link is active and it is involved to the packet transmission it will be not replaced with the new connected link. However, Hash policy and algorithm had selected the L4 for the transmission in bonding tool, but after disconnecting the L3 (the remained NIC) has assumed the transmission. The L4 has been connected to the system again but no changes were assumed in the bonding active NIC.

In conclusion to all above information, the behavior of the bonding driver can be explored as the following. The both NIC in this system are members of the LAG at the sender (*volare-s0*), one of them will be selected as the active NIC for transmitting the data and the other one remains as a hot standby redundant interface. Due to the disconnecting the active link this redundant link will be replaced with the failed NIC as the new active link and the status table of the interfaces will update after any changes. However, the important point as explained

before is when the failed link connects to the system again. After this event no interrupt will occur during the transmission. The status table will change the status of failed interface, but the bonding driver will not replace the new interface with the active NIC. In other words, during the successful transmission and if a NIC works in the bonding driver, no changes will be accepted by this driver.

Moreover, the Hash policy will not be run if at least one NIC is enable and working. In case of using more than two NICs, if the active NIC failed, then the Hash algorithm will be run to select the next active interface. However, if a link connected to the system and the related interface status changes to the up status again the Hash algorithm will not be run again until the active interface fails and the system must select the substitute for this interface. Figure 5-4 proves this theory. It shows the bit rate variances for the explained experiment in the last test case. At message 678 the L4 was disconnected. Then the bit rate became zero for 85 ms until another interface was substituted to the failed interface. Then the bit rate would be raised up again and remained, although the connection has been established after a few seconds.

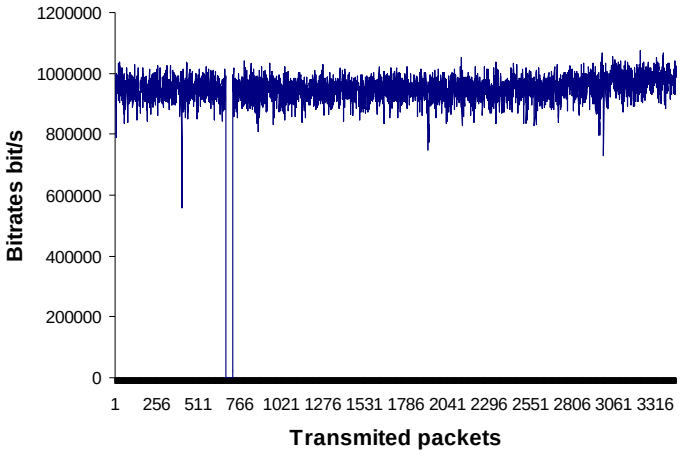


Figure 5-4: Transmission bitrates, when a link fails and recovered again

5.3 Failover Behavior of Stacked Switches

In this project has been used of two technologies to reduce the convergence time in an Ethernet network. The first one is the link aggregation, which the link failures have been explored in previous section. However, another failure that can occur during the communication between the client machines and the provider network is the device failure. The device-level redundancy is as important as the network connection redundancy. Whereas the link failure must be recovered within sub-seconds, the network developer must cover the device failure as well.

As explained in the introduction, it is appropriate to make each critical node redundant, but in some cases this requirement cannot be performed. Clearly, in an Ethernet network many protocols and technologies were provided for node redundancy. Some protocols and technologies, which are investigated in this work, have been described in Chapter 3. Nevertheless, the aim of this work is not to explore all of them, but some has been utilized in this network environment and investigating the behavior of them can be useful for this thesis.

One of the mentioned technologies in Chapter 3 is stacking. Due to the use of link aggregation in the test environment, the stacking mechanism must be utilized to enable the ability of using multiple links. Therefore, the stacking has been used to support the multiple connections between the Layer-2 switches and the client machines. The stacking is a mechanism, which can be used to improve the availability in switches. The description of stacking was given in Chapter 3-6. The two utilized switches in the test environment (see Figure 4-4) use the stacking mechanism. The test case number 3 has been used in this section. One of them is the master unit and another one is the backup unit. The operation basis of these two switches can be mentioned as following.

In this test case, the failover behavior of these stacking units will be investigated. The master unit and backup unit will be reloaded or power disconnected to observe the failure handling of the entire system. Such as previous test cases the client and server Perl script programs will be executed and the transmitted and arrival packets are saved in files as well as captured frames by the Wireshark.

For this experiment, the communication has been started and the master switch power disconnected after a while. Immediately after the disconnection of master unit the backup switch has been reloaded, because it must take the master unit configuration for new start. After the reconfiguration of the backup switch it became the role of master unit, the master LED is lighted, the ports of this device changed to status up again and the communication has been continued. However, the important point is the approximately long time for all these mentioned processes. Due to this reconfiguration, the communication between client and server machines has been broken and the generated UDP packets have not been transferred to the server, since the stacked switches are not up for a couple of seconds.

	Convergence time [s]
Min Value	10,218376
Max Value	14,012241
Average Value	12,341254
Switch reload average time	59,0065

Table 5-7: Convergence time, if master switch fails

Moreover, if the disconnected switch reconnected and can be back to the work, the status of the master and backup unit does not change any more and then it cannot assume the master role. Therefore, if the switch is connected again no changes can be provided. Nevertheless, the packet transmission appointment will not be broken for the second time.

In this test the 122-byte packets are generated and sent into the network and the test case has repeated multiple times (5 times) and the average values will be given. Table 5-7 shows the convergence time obtained from this test case. Clearly, in one experiment the client has generated 19233 packets in 15,163246 s but the receiver has received only 6510 messages. In other words, during the failure, which the error recovery has been taken 10,218376 s, 12723 packets have been lost.

In another test case, which declared in test case number 3, the master switch will be reloaded. In this case the master switch will be down after the enter the reload command, during the reload process the back unit will be assume as the master unit and it will be reloaded as well. Furthermore, before the backup unit has been accepted the assignment, the master unit will be up again. This process will take near one minute and within this progress the transmission is broken. Therefore, so many packets will be lost. The convergence time in this case has been given in Table 5-7.

In the files, which sender and receiver and also the information have saved results from captured frames. It is obvious that this experiment shows the instability of the stacking unit in case of failure. As the result, if the master switch fails the recovery time for this type of failure is more than 10 s. Therefore, the stacking mechanism cannot perform adequate convergence time to be used in the safety critical communication systems. The average value in this test case confirms the instability.

In addition, the stacking has been defined differently in various switches. The stacking has not been standardized and each manufacturer has defined this mechanism and algorithm in the way that is more appropriate for its designed device. Clearly, in the Dell switches utilized here in this project, when a master stacked unit fails, the backup switch will de restarted to become the stack unit. However, during this job these devices send each other some messages to change their role and configurations. To confirm this theory, one port has been defined to mirror the ports specified for stacking connections. However, the applications such as Wireshark and *tcpdump*, which has been implemented to show all frame types, cannot read the stacking messages. However, if the structures of the messages have been specified for the users, it is possible to define the structure for the capturing tools and it can show the messages. The stacking messages for other devices from other manufactures have their own structures. Furthermore, the other switch marks may behave differently to the master failure problems.

In conclusion to this issue, the stacking is not an appropriate mechanism to make the switches redundant, but only in this networking area. This mechanism has not a sufficient rates and value to fulfill the requirements. In fact of changing the implementation of the stacking in

switches to an appropriate algorithm to decrease the error recovery time, it is possible to perform this mechanism for a highly available Ethernet network. The framework of the network device has to be changed and implemented to cover the stacking mechanism.

5.4 Various Network Interface Cards

The importance of the device driver in Linux computer systems has been discussed in Chapter 4-7. In the last sub-chapter, two different general tests have been executed to explore the system failover behavior. The first test has been measured the time estimation between sender and receiver in the specified network environment and the time division between each two stations were estimated. In the second option, the fault treatment of the LACP has been explored, thus the error detection and error recovery has been measured. Therefore, the same examinations will be executed to cover the failover behavior of another NIC with another networking device driver. Test case number 4 will be referred here for this analysis.

The bonding driver, which will be used to aggregate the two NICs, is located in upper sub-layer than the networking device drivers. Furthermore, the bonding driver is a kernel-networking module, which performs these two network interface cards as a single virtual interface for the Linux kernel sub-systems. The user can develop the Linux networking device driver manually and it is not important to use driver, which is delivered with the card itself. Clearly, in this way it is possible to configure the device more compatible to the environment and also usage requirements.

The first differences between various NICs can be observed during the hardware installation. In the installation of hardware and related device driver in the Linux kernel, the appropriated file will be loaded in the Kernel and the system learns this driver as the related device driver. If two similar devices are utilized in a system a single driver will be installed and it works for the both devices. The Linux system shares higher compatibility with Intel cards than Linksys cards. After installing the Linksys devices, some extra processes must be executed to get the device operating. Moreover, the Intel cards have more stability in the system, although the Linksys cards cannot work with some modules in the Linux Kernel sub-systems.

Table 5-8 shows the values, which are obtained from test of Linksys NICs and the comparison the results with the similar situations and test case, which has obtained values from the test of normal client-server communication with the Intel NICs, thus the test case number 1. Tow experiments have been executed and the results compared with the execution of test case number 1 for Intel cards.

	Exemplar experiment			Average values from 5 experiments		
	Average value	**Maximum value**	**Minimum value**	**Average value**	**Maximum value**	**Minimum value**
Transmission duration for each packet (Linksys) [ms]	3,668791	3,782	3,661	2,994507	3,13	2,9575
Transmission duration for each packet (Intel) [ms]	1,98388	2,19	1,979	2,302206	2,4185	1,7445

Table 5-8: Comparison two NICs in normal communication

The time shown in Table 5-8 has been gained from the send and receives capture times. Five experiments have been executed to provide the table. The first part of this table shows an exemplar experiment that has been chosen from the various experiments to depict the differences between these two NICs in each try. The second part shows the average values for the entire test case in comparison to the Intel NICs. If this table is compared with the Table 5-3 and 5-2, the variation between these two NICs can be observed clearly. The values in this table have been obtained from four different tests. For each test 4 to 5 second communication has been done and 1500 up to 2400 228-byte packets have been transmitted across the network.

As seen in the Table 5-8, the transmission duration for both Intel and Linksys cards are the same, but the offset time after synchronization can cause the little difference. The difference in Intel cards between the maximum and minimum values is bigger than Linksys cards. It means that the operating system and the Linksys device driver work much coherent. However, the bottom line is a big difference or variety between these two cards cannot be observed. The ability of both cards in a link aggregation group is the same. They have the same bitrates and capabilities. Figure 5-5 and 5-6 depict the little differences between these two cards. As shown in these figures only at the beginning phase of the transmission the Linksys cards require much more time to reach to the stable rate. However, the average rate for both cards can be assumed similarly. The Linksys NIC have also a little more fluctuations than the Intel cards, but as the result, these two models, which have been compared here, do not differ within the normal operation in this network.

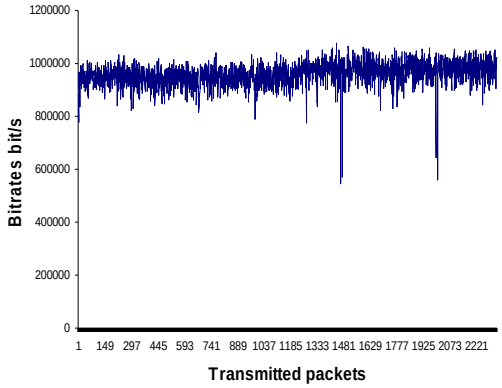


Figure 5-5: Bitrates, Intel cards

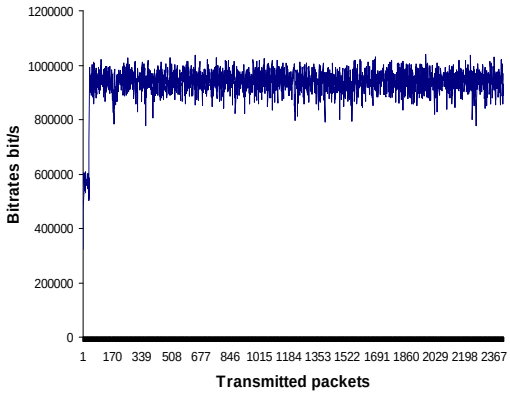


Figure 5-6: Bitrates, Linksys cards

For the next step, the mentioned transaction will be established between the client and server, but in this case the error recovery and error detection time will be explored. In this case during the transmission the link of active NIC between two bonded interfaces will be disconnected. The test results will be investigated and compared with entire test results obtained by experiments on Intel cards.

After executing the tests on the Linksys cards, it can be obtained that these cards operate exact alike with the Intel cards, although this behavior has been expected because the algorithms tolls and the modules on this network remain as the same for the both NICs. The two aggregated NICs here cannot operate parallel for transmitting the packets. As explained in this chapter, the one of the aggregated NICs will be selected as the active interface. Moreover, the Hash policy will be making this interface as the primary networking device. After installing the Linksys devices the interface, which is connected to the master unit (stacked switches) through the link number 3 (see Figure 4-4) will assume the communication appointments in the client machine. Furthermore, the bonding driver in the Linux operating system identifies this interface as the primary interface.

As expected, the link number 4 failure will not disturb the client server transaction. The L3 failure will cause the breaking the packet transmission and some packets will be lost until the backup interface is up. The error detection and the error recovery processes in the system remain as the same, but only the switching time between these two aggregated interfaces can distinguish the behavior of the Linksys and the Intel cards.

Test experiments	Transmission time [s]	Generated packets	Received packets	Lost packets	Convergence Time [ms]	Average bitrates kbit/s
1	4,343197	2248	2212	36	71,722	928,5
2	5,324597	2807	2803	4	9,453	960
3	5,442171	2635	2611	24	48,266	925,8

Table 5-9: Convergence time by the use of Linksys cards

Table 5-9 shows the test results on the Linksys cards. These cards have the same behavior and switching rate as the Intel cards. The only difference here in these experiments is the lower bitrates by the Linksys cards. Furthermore, the transmit capability of these cards are a little less than the bitrates in Intel cards. The difference between each to transmitted packet, thus the time interval that the NIC is ready again to transmit the next packet is bigger than the Intel cards. For Linksys cards it takes about 1,97 ms to transmit a 228-byte packet but the mentioned time in the Intel cards is about 1,91 ms. Whereas the process of generating the packets and transmitting them in all tests is the same and also the single difference between these two tests is the variety of the network interface cards, then the lower bitrates can be assumed as the lower capabilities and compatibility of these cards with the Linux operating system and also to other integrated hardware. Moreover, in Intel cards the number of the transmitted bits is clearly more than the Linksys cards, although the number of lost packets in the tests of the Linksys cards is less than the previous experiments. As shown in the Table 5-9 the generated, received and lost packets for each experiment has been given. The results can be compared with the contents of the Table 5-6 and 5-5 and also Figures 5-2 and 5-3.

As the result and through the comparison of the obtained values with the explained issues in Chapter 5-3, it is clearly resulted that the behavior of the Linksys cards with the PCI Bus of the computer system causes a little differences to the Intel cards and the transaction of these cards with the Linux kernel is slower. However, in the bottom of line this difference is much little to relinquish in the performance of this card. Furthermore, the influences of the higher bitrates cause no visible effect in the LAG and the network with the use of link aggregation.

To summarize the comparison of the two various NICs in the network shown in Figure 4-4, it is obvious that different network interfaces operate similarly in the system and also the communication appointments, which they have to perform them. Due to the results, the only inconvenience point for the Linksys cards was the complexity of operating with the bonding driver and also the link aggregation. However, this problem can be solved with the help of other tools and applications and this term can be named for the evaluation of this NIC.

5.5 Evaluating the High-Availability of Network Environment

Up to now the convergence time of the test environment in case of link and node failure has been investigated. In consideration to the results shown in Table 5-6 and 5-7, the convergence times of the system, if a link fails or if the stacked-switch fails, have been obtained. Due to the datasheets of the utilized network components, it is possible to earn the operation time of these components until the first failure (MTBF). Therefore, the availability of the system can be calculated here in this work. The following formula can estimate the availability of a system, if the downtime is available.

$$Availability = \left(1 - \frac{Downtime}{Downtime + Operation\ Time} \right) * 100$$

In consideration to the Table 5-5, the average convergence time of the system if the link or the NIC fails is 93 ms. The MTBF of the NICs and Ethernet links are about 1 103 000 hours. If the system wants to work 24 hours a day and 7 day a week, then the downtime per week must be less than 6,05 s and the system will be highly available. The convergence time of the system in this case shows if a links or a NIC fails the system will never leave the high availability area. However if the master unit between two stacked-switch fails the convergence time 12 s can ruin the high availability of the system in the week. The system with 25,9 s downtime in a month is also high available and through one system node failure per month the high availability can be guaranteed, although this massive downtime will ruin the safety critical VoIP communication system and it is not appropriate for this area. The MTBF of the Switch is about 250 000 hours and the MTBF of the router is 166 000 hours. Therefore, the system with this operation time can support the network high availability. They can provide six nine availability for their long operation times.

Chapter 6 Conclusion

This diploma thesis focused on the high availability in Ethernet networks for safety critical communication systems. VoIP applications will be run on a specified network environment. As explained in introduction, the VoIP does not dispose of a special safety mechanism to check the correctness of the transferred packets. In consideration to provide adequate service quality, the convergence time of the environment must be performed as low as possible. This work has been attempted to exhibit a solution to decrease the failover time in data-link layer and also increase the availability of an Ethernet network for a safety critical voice communication systems.

In order to ensure the availability of the VoIP services, the network has been designed in a redundant way. In addition, various protocols and mechanisms have been investigated here to improve the availability. The suggested link aggregation mechanism in this thesis has been offered the opportunity to support the link and interface redundancy for the specified network. This mechanism has been configured in the environment to increase the availability in each point-to-point connection.

The link aggregation mechanism has been precisely investigated in this work. The failover behavior of different devices and the entire system, which are utilized this technology, has been explored. The LACP controls the entire assignments of the aggregating process. Therefore, diverse test cases have been defined to lead precise analysis. The LACP has been used in various scenarios and situation to assess the reduction of the failover time by the use of this mechanism. The fault treatment of this mechanism has been probed and the results from defined test cases have verified the analysis.

The first essential point to analyze the failover behavior of the network at the end-system was the specification of the effective modules within the packet transmission. Due to the routing, forwarding, transmitting and receiving assignments, the packet will be delayed in diverse modules at different stations. In order to this, each delay must be estimated and possibly measured to calculate the comprehensive transmission time from sender to receiver. This work eases the analysis of the failover behavior and specifies the consequences of each module on the convergence time. The results show that the Linux computer systems and the device driver of the NICs have massive effects on the packet transmission time. However, the

link aggregation improves the availability of the network and it does not delay the packet transmission more than a couple of microseconds.

Moreover, the convergence time of the system regarding to the error detection and error recovery by the use of link aggregation has been probed. The possible failures in different scenarios can be named as link and the device failure within the transmission between end-systems. The error detection for devices, which is utilized the link aggregation, can be performed within sub-seconds. The error recovery process consists of the system reconfiguring and activation the standby redundant interface.

Previously, the importance of the networking device driver and its influences on the convergence time has been revealed. Therefore, various NICs have been explored and tested in this work. The results show that different NICs do not provide a cognizable operating and also convergence time difference in the systems, although in the test executions, the switching time between two similar interface cards can be performed faster than another exemplar NIC. The only difference that can be observed in this test case is the incompatibility of various NICs with the bonding driver and the link aggregation mechanism. This incompatibility causes a little instability during the packet transmission by the system. However, the error detection and error recovery can be performed in sub-seconds again by the use of diverse NICs.

To conclude the tests results in this document, it can be said that LACP can provide a convergence time approximately of 100 ms. However, this failover performance can be afforded if the link, the aggregated port, and or the active NIC in the Linux computer system fail. The achieved results are given in Chapter 5 to confirm the assertion. Theoretically, such convergence times do not have inappropriate effects on voice communication over Ethernet networks.

In consideration to the results for exploring the stacking mechanism at switches, some problems have been observed, which can menace the appropriate availability in the specified network environment. The failover behavior of two stacked switches causes difficulties in this network. Due to the experiments, if the master unit fails, the stacking mechanism cannot provide the convergence time better than 10 s. However, this convergence time does not guarantee the availability of the network, which will be used in safety critical voice communication systems. In worst-cases, if the master stacked unit is reloaded, the convergence time of the system is performed approximately of 1 minute. The stacking mechanism is a company-defined mechanism and its performance may differ from manufacture to manufacture. However, this thesis does not focus on investigating of different stacking mechanisms. This subject can be probed in further works to reduce the convergence time in these failure scenarios.

To summarize this work, it can be undertaken that the link aggregation mechanism and LACP are able to perform the convergence time in sub-seconds in the specified scenarios. However, each node must support the LACP requirements to allow the use of this

mechanism. Moreover, the stacking mechanism must be attended and focused for these network environments in further works to provide the better convergence time.

Abbreviations

ARP	Address Resolution Protocol
ATC	Air Traffic Control
BPDU	Bridge Protocol Data Unit
BGP	Border Gateway Protocol
COTS	Commercial Off-the-Shell
EIGRP	Enhanced Interior Gateway Routing Protocol
GLBP	Gateway Load Balancing Protocol
HA	High Availability
HSRP	Hot Standby Router Protocol
ICMP	Internet Control Message Protocol
IGP	Interior Gateway Protocol
IGRP	Interior Gateway Routing Protocol
IS-IS	Intermediate System to Intermediate System
ISP	Internet Service Provider
LACP	Link Aggregation Control Protocol
LAG	Link Aggregation Group
LAC	Link Aggregation Control
LACPDU	Link Aggregation Control Protocol Data Unit
MPLS	Multi Protocol Label Switching
MSTP	Multiple spanning Tree Protocol
NIC	Network Interface Card
NTP	Network Time Protocol

OCP	Operation Communication Process
OMP	Operation Main Process
OSPF	Open Shortest Path First
PSTN	Published Switched Telephone Networks
RIP	Router Information Protocol
RMS	Resource Manageable system
RPR	Resilient Packet Ring
RSTP	Rapid Spanning Tree Protocol
RTP	Real-Time Transfer Protocol
SCTP	Stream Control Transmission Protocol
SDH	Synchronous Digital Hierarchy
SONET	Synchronous Optical Network
STP	Spanning Tree Protocol
SIP	Session Initiation Protocol
TCP	Transmission control Protocol
UDP	User Datagram Protocol
VCS	Voice Communication System
VLAN	Virtual Local Area Network
VoIP	Voice over Internet Protocol
VRRP	Virtual Router Redundancy Protocol

References

[AGH01] Tim Anderson, Todd Grabbe, Julian Hammersley, Ira Horden, Ken Hosac, Stuart Levy, Vince Liggett, David McKinley, David Radecki, Brian Ramsey, Alan Stone, "*Providing Open Architecture High Availability Solutions*", Published by the HA Forum, Revision 1.0, 2001 *February*

[AJY00] C. Allaettinoglu, V. Jacobson, H. Yu, "*Towards Milli-Second IGP Convergence*", Packet Design Inc., NANOG 20, 22 – 24 October Washington D. C., 2000 *November*

[Bar02] Paul Barry, "*Programming the Network with Perl*", Published by John Wiley & Sons Limited, Institute of Technology, Carlow, Ireland, ISBN of 0471486701, 2002 *February*

[BFG07] Lukasz Budzisz, Ramon Ferrús, Karl-Johan Grinnemo, Anna Brunstrom, Ferran Casadevall, "*An analytical estimation of the failover time in SCTP multihoming scenarios*", IEEE WCNC 2007, Hong Kong, March 11-15, 2007 [CPR05] Chris Clegg, Jacky Pojzet, Melvyn Rees, "*ATS Ground Voice Network Implementation and Planning Guidelines*", European Organization for the safety of Air Navigation, EUROCONTROL Headquarter, General Public, Edition 1.0, 2005 *22nd February*

[CLML98] B. Cole, T. Li, P. Morton, D. Li, "*Cisco hot standby router protocol*", Network Working Group, Request for Comments: 2281, 1998 *March*

[Com00] Douglas E. Comer, "*Internetworking with TCP/IP, Principles, Protocols and Architecture*", Volume 1, fourth Edition, Published by Prentice Hall, ISBN 013183806, 2000

[Cox96] Alan Cox, "*Network Buffers and memory Management*" Linux Journal, issue 30 1996 *29th September*

[CR05] Mathew Caesar from UC Berkeley, Jennifer Rexford from Princeton University, "*BGP routing policies in ISP networks*", Publication in Nov - Dec 2005, Network IEEE Volume 19, issue 6, Pages (5 – 11), 2005 *November*

[CRK05] Jonathan Corbet, Allessandro Rubini, Greg Kroah-Hartman, "*Linux Device Drivers, where the kernel meet the hardware*", Published by O'Reilly, ISBN 0596005903, 2005

[CRP03] Chris Clegg, Mel Rees, Wolfgang Philipp, European Organization for the safety of Air Navigation, "*Voice communication system Procurement guidelines*", Edition 1.0, General Public, EATMP Infocentre Reference: 03052701, 2003 22nd *February*

[CV06] Anna Cavalla, Dario Vieira, "*Working Around BGP: An Improvement of BGP Session Maintenance*", 06. International conference on Networking and Services, 2006 ICNS apos; Page 41, France, 2006

[Dav00] Thomas Davis, "*Linux Bonding Driver*", release in Linux Kernel Documentation, Corrections by: Willy Tarreau, Constantine Gavrilov, Chad N. Tinde, Janice Girouard, Jay Vosburgh, 2000 *March*

[Del05] "Dell™ PowerConnect™ 34XX Systems, User's Guide", Dell Inc. 2005

[DYGN04] Fredrik Davik, Mete Yilmaz, Stein Gjessing and Necdet Uzun, "*IEEE 802.17 resilience Packet Ring Tutorial*", IEEE Communication Magazine Volume 42, Issue 3, Mar 2004, Pages (112 – 118), 2004 *March*

[ENI08] European Network and Information Security Agency (ENISA), "*Business and IT Continuity: Overview and Implementation Principles*", Conducted by the technical department of ENISA Risk management in cooperation with: Janet Beattie et al. - Glen Abbot Ltd., 2008 *February*

[Fin98] Norman Finn, "*Port Aggregation Protocol*", Cisco system Inc. 1998 *May*

[FM03] Werner Fischer, Christoph Mitasch, "*Linux High Availability mit Heartbeat und DRBD*", Slides, IBM Linux Congress, FH Hagenberg, Germany, 2003 26th *June*

[Goo02] Bur Goode, "*Voice over Internet Protocol*", In: Proceeding of the IEEE, Manuscript Volume 90, Issue 9, Sep 2002 Page(s): 1495 – 1517, 2002 *March*

[HA00] Johann Hörl, Bernhard K. Aichernig, "*Focus Requirements Engineering: Validating Voice Communication Requirements Using Lightweight Formal Methods*", IEEE Software, volume. 17, no. 3, pages (21-27), May/Jun, VOEST-Alpine, TU Graz 2000

[Hed88] C. Hedrick, "*Routing Information Protocol*", Network Working Group, Request for Comments: 1058, Rutgers university, 1988 *June*

[IEEE8023] IEEE 802.3 LAN/MAN CSMA/CD Access Method, "IEEE Standard for Information technology, Telecommunication and information exchange between systems, local and metropolitan area networks and specific requirements" Part3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications, IEEE Computer Society, IEEE 802.3-2005, IEEE 3 Park Avenue NY, USA 2005 9th *December*

[IEEE8023ad] IEEE 802.3 group, "Amendment to Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications- Aggregation of Multiple Link Segments" IEEE-SA Standards Board, IEEE Std. 802.3ad-2000, 2000 *March*

[IEEE80217] IEEE Draft Standard 802.17 for Resilient Packet Rings (RPR), Version 0.3, and Version 3.3, An Overview by Resilient Packet Rings Alliance, 2002 *June* and 2004 *April*

[Kop02] Hermann Kopetz, "*Real Time Systems, Design principles for distributed embedded applications*", Second Edition, ISBN: 0792398947, Vienna University of Technology, 2002

[KS03] Hartmut Kell, Markus Schaub, "*Product Analysis: HiPER Ring vs. RSTP*", ComConsult Technology Information GmbH, Edition 1.0, 2003 *November*

[KWW04] S. Knight, D. Weaver, D. Whipple, R. Hinden, D. Mitzel, P. Hunt, P. Higginson, M. Shand, A. Lindem, "*Virtual Router Redundancy Protocol*", Network Working Group, Request for Comments: 2338, 2004 *April*

[Moy98] J. Moy, Ascend Communications Inc., "Open shortest path first", Network Working Group, Request for Comments: 2328, 1998 *April*

[MS04] Bruce Miller, Elliott Stewart, "*Multi-Protocol switch label: Conformance and Performance Testing*", Company Ixia Inc., White Papers 2004

[MYM02] Kurrenai Murakami, Su-Hun Yun, Osamu Matsuda, Motoo Nishihara, Direceu Cavendish, "*New Transport Services for Next-Generation SONET/SDH Systems*", IEEE Communication Magazine, Volume: 40, Issue: 5, On page(s): 80-87, 2002 *August*

[OY02] L. Ong, J. Yoakum, "*An Introduction to the Stream Control Transmission Protocol*", Network Working Group, Request for Comments: 3286, 2002 *May*

[Pos80] J. Postel, "*User Datagram Protocol*", Request for Comments: 768, 1980 *28th August*

[Red04] Kumar Reddy, "*Building MPLS-Based Broadband Access VPNs*", ISBN: 1587051362, Published by Cisco Press, 2004

[RL95] Y. Rekhter, T. Li, "*Border Gateway Protocol*", Network Working Group, Request for Comments: 1771, Cisco Systems and T.J. Watson Research Center, IBM Corp., 1995 *March*

[RP94] J. Reynolds, J. Postel, "*Assigned Numbers*", Network Working Group, Request for Comments: 1700, 1994 *October*

[SB03] Markus Schaub, Petra Borowka, ComConsult Technology Information GmbH, "*Technologie Report: Designvarianten lokaler Netzwerk*", Pages (191 – 200), 2003 *March*

[Sch03] Wolfgang Schulte, "*Spanning Tree*", Funkschau, 15-16/2003, On Pages (55 – 58), 2003

[Sol02] Michael Soltau, "UNIX/LINUX Hochfuegbarkeit", ISBN-10: 3826607759, ISBN-13: 978-3826607752, First Edition, 2002 *September*

[Ste07] R. Stewart, "*Stream Control transmission Protocol*", Network Working Group, Request for Comments: 4960, 2007 *September*

[SV97] A.K. Somani, N.H. Vaidya, "*Understanding Fault tolerance and Reliability*", Computer, Volume 30, Issue 4, Apr 1997 Page(s): 45 – 50, University of Washington, 1997 *April*

[SXS00] R. Stewart, Q. Xie, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, V. Paxson, "*Stream Control transmission Protocol*", Network Working Group, Request for Comments: 2960, 2000 *October*

[WWD02] Gerard J. Waldron, Rachel Welch, James Dempsey, "*Voice-over-IP: The future of communication*", Covington & Burling, Washington, D.C., Center for Democracy and Technology, 2002 *April*

[WGXX06] Jigang Wang, Gouchang Gu, Shi-Bo Xie, Li-Feng Xu, "*An Fast Transparent Failover Scheme for Service Availability*", Proceedings of the First International Multi-Symposiums on Computer and Computational Sciences - Volume 1 (IMSCCS'06) - China, Pages: 654 – 661, Published by IEEE Computer Society, ISBN:0-7695-2581-4-01, 2006

[Wil06] Chris K. Williams, "Tuning dynamically routed internet protocol networks to achieve controlled and predictable failover during link instability", Military Communications Conference, MILCOM 2006, 23-25 Oct, On page(s): 1-6, Washington, D.C., 2006

[Yu05] James T. YU Ph.D. "*A New Approach to Developing High-Availability Server*", White Papers, School of computer science and information systems DePaul University, 2005 *April*

Internet References

[Cis97] Cisco Group, Whitepapers, "Understanding Spanning Tree Protocol" [online], and Available at: http://www.cisco.com/univercd/cc/td/doc/product/rtrmgmt/sw_ntman/cwsimain/cwsi2/cwsiug2/vlan2/stpapp.htm, 1997, [called on 23.03.2008]

[Cis02] Cisco Group, Whitepapers, "Gateway load balancing protocol" [online], Available at: http://www.cisco.com/en/US/docs/ios/12_2t/12_2t15/feature/guide/ft_glbp.html, 2002 [called 23.03.2008]

[Cis05] Cisco group, White Paper, "*IS-IS Network Types and Frame Relay Interfaces*" [Online], Available at: http://www.cisco.com/en/US/tech/tk365/technologies_tech_note09186a008009445a.shtml, ID: 13794, [Called on 23.05.2008] 2005 10th *August*

[Cis07] Cisco group, White paper, "*How LAN Switches Work*" [online], and Available at: http://www.cisco.com/en/US/tech/tk389/tk689/technologies_tech_note09186a00800a7af3.shtml, Document ID: 10607, 2007, [called on 25.02.2008]

[Els04] Kaled M.F. Elsayed, "*Overview of the Evolving IEEE 802.17 Resilient Packet Rings Standard*" [online], Department of Electronics and Communications Engineering Faculty of Engineering, Cairo University, Giza, Egypt 12613, 2004, Available at: www.cs.rice.edu/~eugeneng/teaching/f08/comp529/papers/rpr.pdf [called on 23.03.2008]

[Iee90] IEEE Standards Association, "IEEE Standard Glossary of Software Engineering Terminology" [online], IEEE Std 610.12-1990, and Available at: http://standards.ieee.org/reading/ieee/std_public/description/se/610.12_1990_desc.html, 1990 *September*, [called on 20.02.2008]

[LSW04] Ulf Lamping, Richard Sharpe, Ed Warnicke, "*Wireshark User's Guide: 24602*", Available at [online]: http://www.wireshark.org/docs/wsug_html/, 2004, [called on 26.07.2008]

[Var00] Enrique Vargas, "*High Availability Fundamentals*", Sun BluePrints™ Online, and Available at: www.sun.com/blueprints/1100/HAFund.pdf, 2000 *November*, [Called on 12.10.2008]

[WSA02] D. Wegscheid, R. Schertler, G. Aas and J. Hietaniemi, "*Perl documentation Time::HiRes*" [online], Available at: http://perldoc.perl.org/Time/HiRes.html, 1996 – 2002, [Called on 25.08.2008]

Appendix A Network Architecture and Device Configurations

A.1 Network Architecture

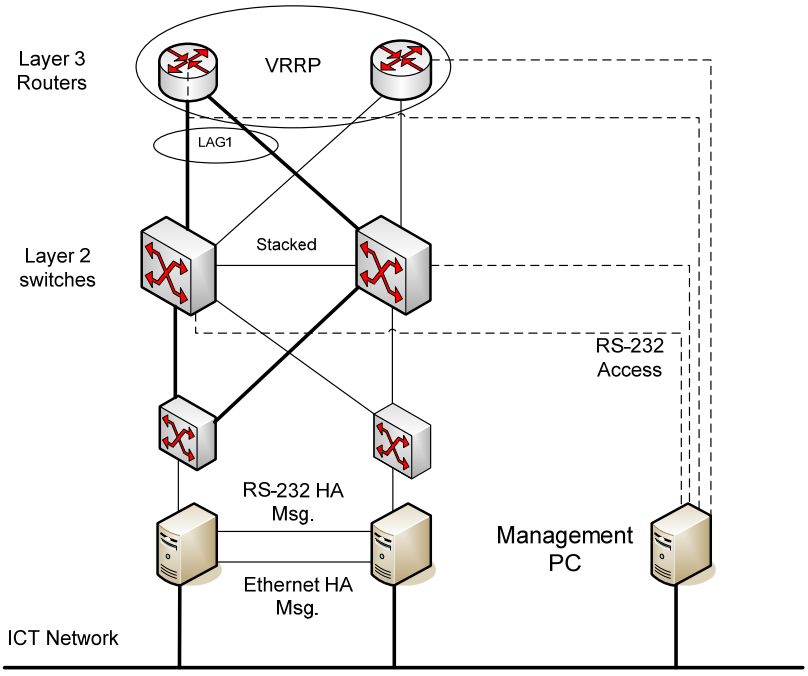


Figure A-1: Network architecture

The devices are used in this environment:

- NetGear GSM7312
- Dell PowerConnect 3424
- Dell

- Server with Linux HA
- Management PC, Linux Ubuntu

The NetGear GSM7312 routers are selected to use in this environment. This device supports all standards and protocols required in this project. The main protocols such as RSTP, Link Aggregation, LACP and routing protocols such as VRRP are supported by the NetGear GSM7312. A management PC manages these routers. The management PC is responsible for configuration and setting the routers up. The router is consists of 12 10/100 Ethernet ports, over them the routers are connected to the both networks; Wide Area Network and Local Area Network, and also a console port, which connect the router to the management PC. This connection is produced with an RS232 interface connection. And a set of LEDs shows the links activity and existence in the appropriate router.

After NetGear routers, the Layer 2 switches will be described. The Dell Powerconnect 3424 switches are used in this project. This model consists of 24 10/100 BASE Ethernet ports, two 1000 BASE Ethernet ports and also two stacking ports. It comprises a consol port like the routers. These switches are connected to Management PC too. Management PC can configure the switches. The Dell Powerconnect switches support all the required protocols and standards, such as LACP and RSTP. One of the requirements of this project is the ability of Layer 2 switches to operate as a single unit. Therefore, it is necessary to use the stacking feature that explained in section 3-6. On front panel of this device there are eight LEDs, which show the status of stacking in related device. Also these LEDs show the role of device among stacked devices. It can be the Master device or a Backup device, which can be second, third or further backup device. This switch model, Dell Powerconnect, supports six stacked devices.

Other parts of network shown in the following figure are the two non-manageable switches. These two devices allow the two PCs that allocated bellow them to connect multiple devices through a single Network Interface Card (NIC). A link connects the PC to the non-manageable switch. The packets sent from PC can be forwarded to maximal 4 different devices. As described before these switches are non-manageable. It means, a management PC cannot configure them. However they can recognize and forward special standard packets but they cannot distinguish between them. Also they cannot control the flow and recognize data rate, although the other devices, which explained above can do these assignments.

The two PCs, which they have three connections;

- Direct connection to each other over NIC
- Direct connection to institute intern network
- Connection to the WAN through the designed network

The Linux-HA runs on these two PC, which simulate a redundant processing device that wants to communicate with outside world across the designed redundant network. It means

these two PCs have to play the role of a redundant device wants to run the VoIP applications, communicate to WAN and used in Air Traffic Control. The Processes run on these PCs must be highly available. Therefore, one of them is appointed as the master PC and the other one is Backup or slave. The master slave connection is achieved over the Linux-HA features, which operate with heartbeat applications to obtain the high availability.

A.2 Configuring the Bonding Driver on the Linux Computer system

There are more than two ways to configure the bonding driver; Use of network initialization utilizations and without any tools with a script. There are several utilizations such as *sysconfig* and *initscript* but we prefer to write a script and configure the bonding devices manually.

In this method, the bonding module parameters are placed in */etc/modules.conf* or */etc/modprobe.conf*, then the *modprobe* commands are added to *init script.* There is an example shows how two network interfaces could be add in a bonding driver. This file is a script file.

```
modprobe bonding mode=802.3ad lacp_rate=1 miimon=100
modprobe e100
ifconfig bond0 172.16.1.1 netmask 255.255.255.0 up
ifenslave bond0 eth0
ifenslave bond0 eth1
```

This method does not support the automatically initiation and remove bonding devices. Therefore, the initiation and remove the bonding should be done manually through the extra scripts. For removing the bonding devices the following commands can be given. After these commands the bonding will be shut down.

```
# ifconfig bond0 down
 # rmmod bonding
 # rmmod e100
```

This method has the ability to define multiple bonding drivers with multiple operation modes. By using *max_bonds* it is possible to define multiple bonds with one single operation mode.

After determination network devices and bonding devices, it is possible to see the configuration and check it. In subdirectory */proc/net/bonding* is given all information about bonding, this file is a read only file and it cannot be changed.

```
Ethernet Channel Bonding Driver: v3.1.1 (September 26,
2006)
      Bonding Mode: IEEE 802.3ad Dynamic link aggregation
Transmit Hash Policy: layer2 (0)
MII Status: down
```

```
       Up Delay (ms): 0
       Down Delay (ms): 0
       802.3ad info
       LACP rate: fast
       Active Aggregator Info:
               Aggregator ID: 1
               Number of ports: 1
               Actor Key: 9
               Partner Key: 1
               Partner Mac Address: 00:11:22:33:44:55
       Slave Interface: eth2
       MII Status: up
       Link Failure Count: 0
       Permanent HW addr: 00:1b:11:19:b0:42
       Aggregator ID: 1
       Slave Interface: eth3
       MII Status: up
       Link Failure Count: 0
       Permanent HW addr: 00:50:ba:fa:c8:c1
              Aggregator ID: 2
```

And giving the commands *ifconfig* can check the network configuration.

```
root@volare-s0:/home/volare# ifconfig

bond0  Protokoll:Ethernet  Hardware Adresse 00:11:22:33:44:55
       inet Adresse:172.16.1.100 Bcast:172.16.255.255
Maske:255.255.0.0

eth0   Protokoll:Ethernet  Hardware Adresse 00:01:80:63:FD:CF
       inet Adresse:128.131.81.57 Bcast:128.131.81.255
Maske:255.255.254.0
       inet6 Adresse: fe80::201:80ff:fe63:fdcf/64
Gültigkeitsbereich: Verbindung

eth2  Protokoll:Ethernet  Hardware Adresse 00:11:22:33:44:55
      inet6 Adresse: fe80::211:22ff:fe33:4455/64 Gültigkeitsbereich:
Verbindung
```

A.3 Configuring the Link Aggregation Control Protocol on Switches and Router

In consideration to the project targets, and after configuring the link aggregation at the client sides, it is required to configure the Link Aggregation for router and switches. Other parameters would be set for this network. These settings configure the routing, separating the ports of both router and switch to different area for different configuration and finally the IP assignment for each device I the network.

A.3.1 Setting Up the Router

The router can be configured with respect to its Command Line Reference and the Hardware installation guide, which the both documents are delivered with the device. These two documents have adequate guidelines to configuring the device, but in this document the entire process will be explained. Configuring the router can be divided in following sequences:

1. Configuring two VLANs
2. Allocating an IP address to each VLANs in consider to the Network area
3. Configuring a Link Aggregation Group and allocating two port to this LAG

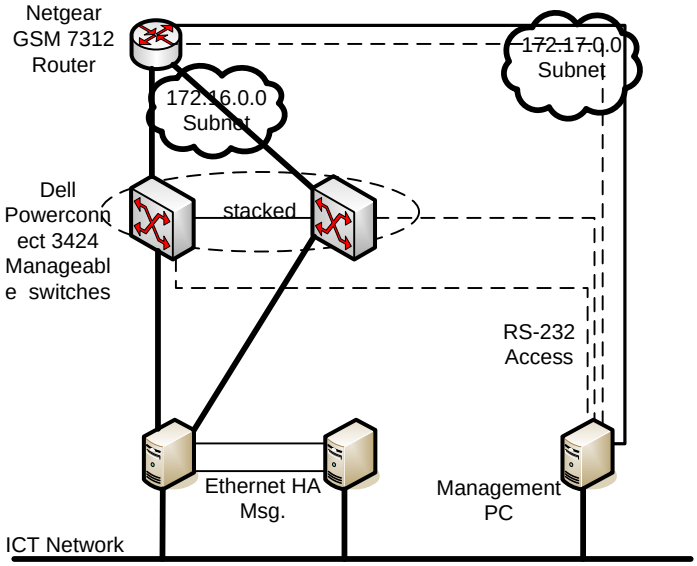


Figure A-2: Subnets in Ethernet network

At beginning it is necessary to describe the Ethernet subnets that we should work with them. The outside world of network environment is the 172.17.0 subnet. This subnet consists of the communication and connections between management PC, which simulates the outside world and the router as the entrance gate to the system that we work on it. The system designed to use here and the environment are located in 172.16.0.0 subnet. Each device in this subnet has an IP address, which is begun with 172.16.x.x. The router forwards all message from outside world (management PC) to the system inside and vice versa. This is the router duty and only the routing setting should be configured.

At first, it is necessary to define two VLANs for router, because only one IP address should be allocated to a VLAN. Virtual Local Area Network (VLAN) is a virtual local network in a physical device. The technical information is assembled in IEEE 802.1Q standard.

Each device in a local network can only communicate to other device and stations in the same local network. In other word, the packets can be sent and received only among the devices

located in the same subnet. The sending and receiving the packets to and from other subnets is only possible, if one or multiple routers connected these subnets anyhow to each other. Therefore, two VLANs are defined in the router and specify the forwarding packets among the two named subnets.

To configuring a VLAN these commands are given in a console of management PC, which can manage the router through a RS232 connection. Default VLAN 1 exists by any router. To manage the routing, two more VLAN is required. Therefore, it is necessary to connect the router management environment. This job is possible through *minicom* tool in Linux. This tool allows us to connect each device that connected to the management PC. This command performs this connection.

```
siadatkhoo@volare-pc:~$ minicom ttyUSB4
```

After login in the device through username *Admin* without any password, the router configuration can be changed to the appropriate configuration. For configuring two VLAN, these commands are given.

```
(RTR1)
User:admin
Password:
(RTR1) >enable
Password:
(RTR1) #vlan database     #gives    the    privilege    to
configure VLAN
(RTR1) (Vlan)# vlan 10    #create VLAN 10
(RTR1) (Vlan)# vlan 20    #create VLAN 20
```

After creating two VLANs, 4 ports are participated in each created VLAN through these commands. However, first the interface configuration should be active.

```
(RTR1) (Vlan)#exit
(RTR1) #configure
(RTR1) (Config)#interface range 0/1-0/4
(RTR1) (conf-if-range-0/1-0/4)# vlan participation include
10
```

The participation of ports 5 to 8 to VLAN number 20 is done exactly the same with appropriate parameters. After that, it is possible to allocate an IP address to each VLAN. This job is performed through these commands:

```
(RTR1) (Vlan)#vlan routing 10
(RTR1) (Vlan)#exit
(RTR1) #configure
(RTR1) (Config)#interface vlan 10
(RTR1) (Interface-vlan 10)#ip address 172.16.1.3 255.255.255.0
(RTR1) (Interface-vlan 10)#exit
(RTR1) (Config)#interface vlan 20
(RTR1) (Interface-vlan 20)#ip address 172.17.1.3 255.255.255.0
```

Finally the VLANs are configured. And now the link aggregation will be configured in this device. The link aggregation configuration is created and performed through these commands:

```
(RTR1) #configure
(RTR1) (Config)#port-channel 1
(RTR1) (Config)#interface range 0/1-0/2
(RTR1) (Interface-range 0/1-0/2)#addport 1/1
(RTR1) (Interface-range 0/1-0/2)#port lacpmode
(RTR1) (Interface-range 0/1-0/2)#exit
(RTR1) (Config)#exit
(RTR1) #show port-channel 1

            Port-                  Link
Log.       Channel            Adm. Trap  STP           Mbr      Port    Port
Intf        Name        Link  Mode Mode  Mode   Type   Ports    Speed   Active
------ --------------- ------ ---- ---- ------ ------- ------ --------- ------
lag 1    lag_11        Up      En.  En.  En.    Dynamic 0/1    Auto      True
                                                         0/2    Auto      True
```

Now the Link aggregation is configured and the LACP is active for this Link Aggregation Group. Therefore, this LAG can be used in the system test and the above table confirms the correction of process. In “appendix A” the complete router configuration are shown.

A.3.2 Setting Up the Switches

In this section, the switch configuration process will be declared. This process is a little different from configuration the router, although some parts of configuration are similar.

As explained in the first section of this chapter, the Dell switches (Powerconnect 3424) are used in the network environment. There are product guideline and command reference, which can be used to configure the switch. These documents are delivered with device.

In this project are used two same switches that should be redundant. As mentioned in chapter 3, to make this part of the network redundant, we performed stacking for these two switches. The stacking helps us to achieve higher availability, although this option has its own problems and trouble in both installing and configuring processes.

The configuration process of these two switches is executed in following phases;

1. Configuring the Stacking
2. Allocating an IP address to the stacked switches
3. Configuring two LAG and allocating each one two ports
4. Enabling the LACP on the stacked group

At the beginning of the process, the stacking should be performed on these two switches. The stacking process is not a standard and it depends on manufactures. This means that the software, which is run on the device, provides the stacking features. Each manufacture produces its own device with own software and drivers. Therefore, the stacking is different from company to company.

To configuring these two switches, the same tool in Linux is used. Also the same management PC operates the functions, communicates to switches in consideration with executing the command lines and configuring the network device. We use *"minicom"* to connect the switches. The first switch is the device with name ttyUSB2 and another one is ttyUSB3. Also this command allows us to enter the device configuration environment;

```
siadatkhoo@volare-pc:~$ minicom ttyUSB2
siadatkhoo@volare-pc:~$ minicom ttyUSB3
```

Configuring the stacking is performed manually. It is required to restart the device. Restarting the devices is executed through the following command;

```
SWI1#reload
```

However, before the restarting and configuring the stacking it is better to save the running configuration in the switch, in the reference guideline of Dell switches explained completely of the entire saving process.

While the device has been restarted and reloaded, we have to press the stacking button on the front panel of the switch. There are 7 LEDs to show the role of switch in the stacking program. 6 of 7 LEDs are numbered, because 6 switches can be stacked (in this model). We must press the button several times, until the LED with appropriated number blinks. After that the pressing must be stopped. For the first switch (switch number 1), the master switch, the LED number 1 blinks. Then the seventh LED, the master LED, blinks.

Next, exact the same process such as process that explained above is executed. In other words, the second switch is reloaded and the stacking button will be pressed until the LED number 2 blinks. Then the switch backup role will be appointed. After completing the process

```
       SWI1# show stack

Unit    MAC Address     Software    Master       G3         G4      Status
----  -----------------  ----------  --------  ---------  -------- -------
1    00:19:b9:ad:0c:6a  2.0.0.21   Enabled       2          2       master
2    00:19:b9:ac:fd:1f  2.0.0.21   Enabled       1          1       backup
Topology is Ring

Unit Unit Id After Reset
---- -------------------
 1            1
 2            2
```

the switch 2 can not be configured anymore through the management PC. The ports of this switch are shown as the ports of master switch, thus switch 1. In other words, these two switches are viewed as a single unit. By giving the following command the process result will be shown;

At next step, the Link Aggregation and also the Link Aggregation Control Protocol must be active for the single stacked unit. As shown in Figure 4.5 this unit must be connected to the computing unit and at another side to the router. For each side the switch need an appropriated Link Aggregation Group, Therefore, two LAGs must be configured for the stacked switches. In this project each Link Aggregation Group comprises two links. However, it can be improved up to eight links. The following commands lead to configure the Lag for the first side.

```
SWI1#configure
SWI1(config)# interface range Ethernet 1/e1,2/e1
SWI1(config-if)# channel-group 1 mode auto
SWI1(config-if)# exit
```

By aggregating the two selected links, which one of them belongs to the Switch 1 and another one belongs to Switch 2, the command “channel-group” is given. The first parameter is the number of LAG in the Switch. The second parameter is the mode of aggregation. The modes are divided to two categories. The mode “auto” means that the LACP is active for this LAG. The mode “on” allows the port to join a LAG without an LACP operation. The Link Aggregation Group is here named as the “port-channel”. Finally, after configuring the channel the interface (ports) configuration environment can be leaved. Through the following command the configuration of the LAG can be changed.

```
SWI1(config)#interface port-channel 1
```

The same process is executed to perform the second Link Aggregation Group.

```
SWI1#configure
SWI1(config-if)# interface range Ethernet 1/e2,2/e2
SWI1(config-if)# channel-group 2 mode auto
SWI1(config-if)# exit
```

In addition to configuration above, it is necessary to set the LACP active for the entire stacked unit. Therefore, the following commands are given to change the system priority and the LACP timeout, which are explained in sub-section 4.2.2. The required parameters are set in this section. As declared in sub-section 4.2.2 the time out must be set in short, because the system needs to monitor the link status more quickly. Moreover the system priority is not very important here, because the system used in this project is not so extended that the priority among LAGs is required. The default system priority for each unit is 1. This means the device has the highest priority between all units in the same level of system.

```
SWI1(config)#lacp system-priority 1
SWI1(config)#lacp timeout short
SWI1(config)#exit
```

Finally the configuration two LAGs in the stacked switches are completed and the results can be seen by give the following command. If the LACP configuring result is requested, the partner of each channel is shown in the result.

```
SWI1#show interface range port-channel 1,2

Channel  Ports
-------  -----
Ch1      Active: 1/e1, 2/e1
Ch2      Active: 1/e2, 2/e2
SWI1# show lacp port-channel all
Port-Channel ch1
       Port Type 100 Ethernet
       Attached Lag id:
       Actor
              System Priority:1
              MAC Address:    00:19:b9:ad:0c:6a
              Admin Key:      1000
              Oper Key:       1000

       Partner
              System Priority:1
              MAC Address:    00:18:4D:D3:96:1D
              Oper Key:       1000
Port-Channel ch2
       Port Type 100 Ethernet
       Attached Lag id:
       Actor
              System Priority:1
              MAC Address:    00:19:b9:ad:0c:6a
              Admin Key:      1000
              Oper Key:       1000
       Partner
              System Priority:1
              MAC Address:    00:11:22:33:44:55
              Oper Key:       1000
```

Now only the IP address must be set for the stacked switches. The allocated IP address must be in the 172.16.x.x subnet. These commands allocate the appropriate IP address to thus unit.

```
SWI1#configure

SWI1(config)#interface vlan 1

SWI1(config-if)#ip address 172.16.1.2 255.255.255.0
```

Finally the configuration of switches was completed and now this unit can be used in the network and the tests can be executed in the network.

A.4 Setting Up the Routing Table

The devices were configured to use. It is possible to communicate between the management PC and the computer stations in the network as sown in figure 4.5. However all configuration were executed and all devices are ready to use, the communication between two named machines cannot be observed. The only problem that fails is the routing setting in the entire network.

As mentioned in sub-section 4.1 two different subnets are connected to each other through the router. And the router has the ability to forward the messages from one to another. Therefore, it is not necessary to execute any additional configuration in router.

The stacked switches are connected only to the 172.16.x.x subnet. Therefore, they require no routing setting. The only nodes, which need to configure the routing, are the two end stations. These two computers run the Linux operating system and the Linux need to know how and across which subnets and devices the packets will be sent. By only one command this assignment is done and the Ethernet network is operable.

```
root@volare-s0:/home/volare#ip route add 172.17.1.0/24 via 172.16.1.100
netmask 255.255.255.0
```

It means that the volare-s0, which consists of link aggregation and is connected to the switches directly, will communicate with subnet 172.17.0.0 through the bond0 or through its Link Aggregation Group. The same command will be given to management PC and the route will be set. This machines see the subnet 172.16.x.x via the router, which it has the IP address 172.17.1.3.

```
root@volare:/home/siadatkhoo#ip route add –net 172.16.1.0/24 netmask
255.255.255.0 gw 172.17.1.3
```

The result of these two commands is shown in the following table.

Volare-s0			
Destination	Router	Netmask	Interface
172.17.1.0	172.16.1.3	255.255.255.0	Bond0
128.131.80.0	172.16.1.3	255.255.254.0	Eth0
172.16.0.0	172.16.1.3	255.255.255.0	Bond0
Volare-pc			
172.16.1.0	172.17.1.3	255.255.255.0	Eth2
172.18.1.0	172.17.1.3	255.255.255.0	Eth1
172.17.1.0	172.17.1.3	255.255.255.0	Eth2

Table A-1: Routing configuration at volare-s0 and Volare-pc

A.5 Router Configuration

After configuring the router and switches, they have the following configuration in their startup configuration file.

```
Current Configuration:
System Description "GSM7312 L3 Managed Gigabit Switch"
System Description 6.2.0.14
set prompt "RTR1"
network mgmt_vlan 10
vlan database
vlan  10
vlan  20
vlan routing 10
vlan routing 20
exit
configure
logging buffered
ip routing
lineconfig
exit
spanning-tree configuration name 00-18-4D-D3-96-1D
spanning-tree forward-time 4
spanning-tree max-age 6
spanning-tree hello-time 1
port-channel lag_11
interface  0/1
addport 1/1
exit
interface  0/2
addport 1/1
exit
snmp-server community public@10
snmp-server community public@20
router ospf
exit
router rip
exit
ip vrrp
interface  0/1
vlan pvid 10
vlan participation include 10
exit
interface  0/2
vlan pvid 10
vlan participation include 10
exit
interface  0/3
no port lacpmode
vlan pvid 10
vlan participation include 10
exit
interface  0/4
no port lacpmode
vlan pvid 10
vlan participation include 10
exit
interface  0/5
no port lacpmode
no spanning-tree port mode
vlan pvid 20
vlan participation include 20
exit
interface  0/6
no port lacpmode
vlan pvid 20
vlan participation include 20
exit
```

```
interface  0/7
no port lacpmode
vlan pvid 20
vlan participation include 20
exit
interface  0/8
no port lacpmode
vlan pvid 20
vlan participation include 20
exit
interface  0/9
no port lacpmode
exit
interface  0/10
no port lacpmode
exit
interface  0/11
no port lacpmode
exit
interface  0/12
no port lacpmode
exit
interface lag 1
vlan pvid 10
exit
interface vlan 10
routing
ip address  172.16.1.4  255.255.255.0
ip vrrp 10
ip vrrp 10 ip 172.16.1.5
exit
interface vlan 20
routing
ip address  172.18.1.4  255.255.255.0
ip vrrp 20
ip vrrp 20 mode
ip vrrp 20 ip 172.18.1.5
exit
exit
```

A.6 Switch Configuration

```
no spanning-tree
interface range ethernet 1/e1,2/e1
channel-group 1 mode auto
exit
interface range ethernet 1/e2,2/e2
channel-group 2 mode auto
exit
interface vlan 1
ip address 172.16.1.2 255.255.255.0
exit
hostname SWI1
```

```
Default settings:
Service tag: F9S6SB1
SW version 2.0.0.21 (date  15-Jan-2007 time  15:48:02)
Fast Ethernet Ports
===========================
no shutdown
speed 100
duplex full
negotiation
flow-control off
mdix auto
no back-pressure
Gigabit Ethernet Ports
============================
no shutdown
speed 1000
duplex full
negotiation
flow-control off
mdix auto
no back-pressure
interface vlan 1
interface port-channel 1 - 8
spanning-tree
spanning-tree mode STP
qos basic
```

Appendix B Script Programs

B.1 Shell Script Implementation

Due to the facts given in this chapter, the bonding driver will be installed through a script file. For installing a bonding driver some appointed commands must be executed. For example 5 lines must be executed each time. Therefore, by defining a script file and implementing all these commands in a single file, it is possible to execute all the named commands by calling the single script file. After calling the script files all commands are executed in the appointed sequence. However, the operator sees noting except the result.

In this project will be used some script files to run some functions and execute some commands. The first script file is *Bonding_start*, which installs and parameterizes the bonding driver. The following codes are the contents of this file.

```
#!/bin/bash
       modprobe bonding use_carrier=0 mode=4 miimon=100 lacp_rate=1
       #       modprobe       bonding       mode=1       arp_interval=10
arp_ip_target=172.16.1.3
       ifconfig eth1 down
       ifconfig eth2 down
       ifconfig bond0 hw ether 00:11:22:33:44:55
       ifconfig bond0 172.16.1.100 up
       ifenslave bond0 eth1
       ifenslave bond0 eth2
       route add -net 172.17.0.0 netmask 255.255.0.0 gw 172.16.1.100
```

At first line the LACP will be active, and the LACP-Timeout will be parameterized as short. Two interfaces eth2 and eth3 are aggregated and the IP address 172.16.1.100 will be allocated to bond0. The result of this script file is shown in the sub-section 4.3.2.

For removing the bonding and detaching the interfaces from bond0, another script file will be called. The file removes the interfaces from aggregation group and reinstalls them into

normal configuration. The following commands in the *bonding_stop* complete the assignment.

```
#!/bin/bash
ifconfig bond0 down
modprobe -r bonding
```

B.2 Perl Script Implementation

Other script files that are used to ease the test process for the measurements in this thesis are Perl socket program files. These files named as *client.pl* and *server.pl*. These files are two specified client server interactions programs, which one of them is run on client machine and the other one is run at server system. In this project the client machine is the computer that comprises the link aggregation and named *volare-s0*. This machine works as a client in the network. This machine runs the VoIP applications that are used in the air traffic control system on the control tower of the airport. The *volare-s0* consists of the interface between the operator and the redundant Ethernet network.

The management PC that explained before is the server machine in this network. Across this computer, the commands are transmitted to the network. Client receives them, executes them and answers the server requests, thus sends its requests to the server. The *server.pl* is the script file runs on server machine (*volare-pc*).

The most important line is the first line that a socket created in it.

At next step with an easy “if” command the connection will be established and used. During the connection establishment, the client sent a message to the server.

The script file at the server is a little more complex. The socket will be created and the socket listens to the designated port. The connection is established as long as the client message is empty.

Client.pl

```
#!/usr/bin/perl -w
# Client Program
use Time::HiRes qw( usleep ualarm gettimeofday tv_interval clock_gettime clock_getres
clock);
use IO::Socket::INET;
use Data::Random qw(:all);
use FileHandle;

open MYINFILE, "> sentfile.txt";
MYINFILE->autoflush(1);
print MYINFILE "Writing to sentfile.txt\n";

$i = 0;
$len = 256;
print ">> Client Program <<";

# storing random string
my @random_chars, @chars_help1, @chars_help2;

print @random_chars;

# Create a new socket

$MySocket=new IO::Socket::INET-
>new(PeerPort=>1234,Proto=>'udp',PeerAddr=>('172.18.1.3','172.18.1.5'));

# Send messages

  while (1)
{
    @chars_help1 = rand_chars(set => 'all', min => 93, max => 93);
    @random_chars = @chars_help1;

  # for($j=1 ; $j <= 10 ; $j++)
  #  {
     @chars_help2 = rand_chars(set => 'all', min => 93, max => 93);
     push(@random_chars,@chars_help2);
  #  }
      $msg = join ("",@random_chars);
        chomp $msg;
    if($msg ne '')
    {
             # print "\nSending '",$msg,"'";

               if($MySocket->send($msg))
            {
             ($sec, $usec) = gettimeofday();
             print " ''''the message   ",$i,".....<done>","\n";
             # print $def_msg;
             $i++;
            }
    }
    else
    {
        # Send an empty message to server and exit
        $MySocket->send('');
        $i++;
        exit 1;
    }
   print MYINFILE "Sent Messages : ", $msg, " number: ",$i, " length: ", length($msg),
"  at time : ", $sec, "  second and ", $usec, "  microsecond", "\n";
}
close MYINFILE;
```

Server.pl

```
#! /usr/bin/perl -w

# Server Program
use Time::HiRes qw( usleep ualarm gettimeofday tv_interval clock_gettime clock_getres
clock );
use IO::Socket::INET;
use FileHandle;

open MYOUTFILE, "> nettest.txt";
MYOUTFILE->autoflush(1);
print MYOUTFILE "Writing to nettest.txt\n";

$i = 0;

print ">> Server Program <<\n";

$MySocket = new IO::Socket::INET->new(LocalPort=>1234, Proto=>'udp' );

while( 1 )
{
  if ($MySocket->recv($text, 256))
  {
    ($sec, $usec) = gettimeofday();
    print "\nReceive message  '", $i, "'\n";
    $i++;
  }
   else
   {
     print "\n No message is received  ";
     exit 1;
    }
     print MYOUTFILE  "receive message: ",$text, " number:", $i, "  At time: ", $sec,
" second and ", $usec, " microsecond \n";
 }

close MYOUTFILE;
```

Appendix C Link Aggregation Control Packet Data Unit Structure

LACPDUs are basic IEEE 802.3 frames; they shall not be tagged (See Clause 3). The LACPDU structure shall be as shown in Figure 43-7 and as further described in the following field definitions:

a) *Destination Address (DA).* The DA in LACPDUs is the Slow_Protocols_Multicast address. Its use and encoding are specified in Annex 43B.

b) *Source Address (SA).* The SA in LACPDUs carries the individual MAC address associated with the port through which the LACPDU is transmitted.

c) *Length/Type.* LACPDUs are always Type encoded, and carry the Slow_Protocols_Type field value. The use and encoding of this type is specified in Annex 43B.

d) *Subtype.* The Subtype field identifies the specific Slow Protocol being encapsulated. LACPDUs carry the Subtype value 0x01.

e) *Version Number.* This identifies the LACP version; implementations conformant to this version of the standard carry the value 0x01.

f) *TLV_type = Actor Information.* This field indicates the nature of the information carried in this TLVtuple. Actor information is identified by the value 0x01.

g) *Actor_Information_Length.* This field indicates the length (in octets) of this TLV-tuple, Actor information uses a length value of 20 (0x14).

h) *Actor_System_Priority.* The priority assigned to this System (by management or administration policy), encoded as an unsigned integer.

i) *Actor_System.* The Actor's System ID, encoded as a MAC address.

j) *Actor_Key.* The operational Key value assigned to the port by the Actor, encoded as an unsigned integer.

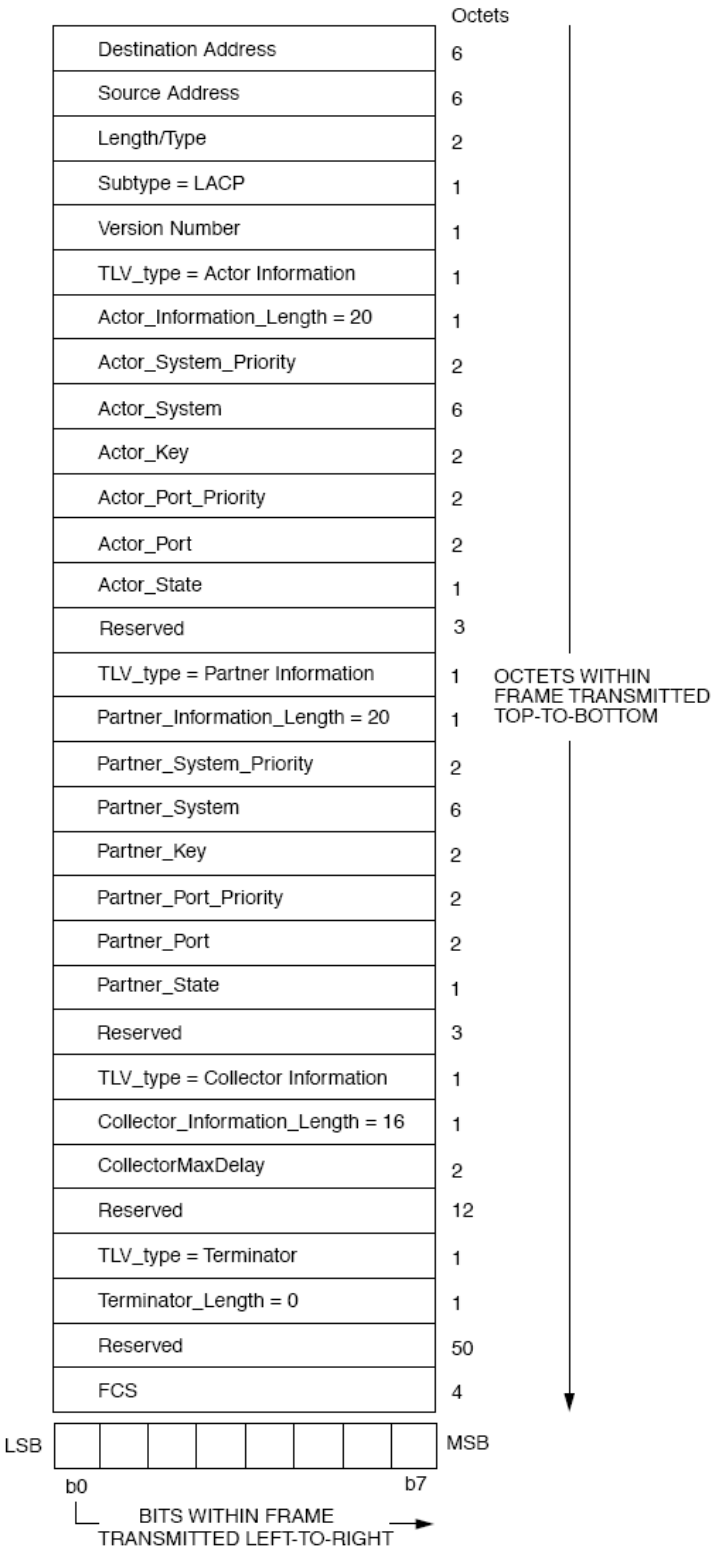


Figure C-1: LACPDU Structure [IEEE8023ad]

k) *Actor_Port_Priority*. The priority assigned to this port by the Actor (the System sending the PDU; assigned by management or administration policy), encoded as an unsigned integer.

l) *Actor_Port.* The port number assigned to the port by the Actor (the System sending the PDU), encoded as an unsigned integer.

m) *Actor_State.* The Actor's state variables for the port, encoded as individual bits within a single octet, as follows and as illustrated in Figure 43-8:

LACP_Activity is encoded in bit 0. This flag indicates the Activity control value with regard to this link. Active LACP is encoded as a 1; Passive LACP is encoded as a 0.

2) *LACP_Timeout* is encoded in bit 1. This flag indicates the Timeout control value with regard to this link. Short Timeout is encoded as a 1; Long Timeout is encoded as a 0.

3) *Aggregation* is encoded in bit 2. If TRUE (encoded as a 1), this flag indicates that the System considers this link to be *Aggregatable*; i.e., a potential candidate for aggregation. If FALSE

(encoded as a 0), the link is considered to be *Individual*; i.e., this link can be operated only as an individual link.

4) *Synchronization* is encoded in bit 3. If TRUE (encoded as a 1), the System considers this link to be *IN_SYNC*; i.e., it has been allocated to the correct Link Aggregation Group, the group has been associated with a compatible Aggregator, and the identity of the Link Aggregation Group is consistent with the System ID and operational Key information transmitted. If FALSE (encoded as a 0), then this link is currently *OUT_OF_SYNC;* i.e., it is not in the right Aggregation.

5) *Collecting* is encoded in bit 4. TRUE (encoded as a 1) means collection of incoming frames on this link is definitely enabled; i.e., collection is currently enabled and is not expected to be disabled in the absence of administrative changes or changes in received protocol information. Its value is otherwise FALSE (encoded as a 0);

6) *Distributing* is encoded in bit 5. FALSE (encoded as a 0) means distribution of outgoing frames on this link is definitely disabled; i.e., distribution is currently disabled and is not expected to be enabled in the absence of administrative changes or changes in received protocol information. Its value is otherwise TRUE (encoded as a 1);

7) *Defaulted* is encoded in bit 6. If TRUE (encoded as a 1), this flag indicates that the Actor's Receive machine is using defaulted operational Partner

information, administratively configured for the Partner. If FALSE (encoded as a 0), the operational Partner information in use has been received in a LACPDU.

8) *Expired* is encoded in bit 7. If TRUE (encoded as a 1), this flag indicates that the Actor's Receive machine is in the EXPIRED state; if FALSE (encoded as a 0), this flag indicates that the Actor's Receive machine is not in the EXPIRED state.

n) *Reserved.* These 3 octets are reserved for use in future extensions to the protocol. They shall be ignored on receipt and shall be transmitted as zeroes to claim compliance with Version 1 of this protocol.

o) *TLV_type = Partner Information.* This field indicates the nature of the information carried in this TLV-tuple. Partner information is identified by the integer value 0x02.

p) *Partner_Information_Length.* This field indicates the length (in octets) of this TLV-tuple, Partner information uses a length value of 20 (0x14).

q) *Partner_System_Priority.* The priority assigned to the Partner System (by management or administration policy), encoded as an unsigned integer.

r) *Partner_System.* The Partner's System ID, encoded as a MAC address.

s) *Partner_Key.* The operational Key value assigned to the port associated with this link by the Partner, encoded as an unsigned integer.

t) *Partner_Port_Priority.* The priority assigned to this port by the Partner (by management or administration policy), encoded as an unsigned integer.

u) *Partner_Port.* The port number associated with this link assigned to the port by the Partner, encoded as an unsigned integer.

v) *Partner_State.* The Actor's view of the Partner's state variables, depicted in Figure 43-8 and encoded as individual bits within a single octet, as defined for Actor_State.

w) *Reserved.* These 3 octets are reserved for use in future extensions to the protocol. They shall be ignored on receipt and shall be transmitted as zeroes to claim compliance with Version 1 of this protocol.

x) *TLV_type = Collector Information.* This field indicates the nature of the information carried in this TLV-tuple. Collector information is identified by the integer value 0x03.

y) *Collector_Information_Length.* This field indicates the length (in octets) of this TLV-tuple. Collector information uses a length value of 16 (0x10).

z) *CollectorMaxDelay.* This field contains the value of CollectorMaxDelay (43.2.3.1.1) of the station transmitting the LACPDU, encoded as an unsigned integer

number of tens of microseconds. The range of values for this parameter is 0 to 65 535 tens of microseconds (0.65535 seconds).

aa) *Reserved.* These 12 octets are reserved for use in future extensions to the protocol. They shall be ignored on receipt and shall be transmitted as zeroes to claim compliance with Version 1 of this protocol.

ab) *TLV_type = Terminator.* This field indicates the nature of the information carriein this TLV-tuple. Terminator (end of message) information is identified by the integer value 0x00.

ac) *Terminator_Length.* This field indicates the length (in octets) of this TLV-tuple. Terminator information uses a length value of 0 (0x00).

ad) *Reserved*. These 50 octets are reserved for use in future extensions to the protocol. They are ignored on receipt and are transmitted as zeroes to claim compliance with Version 1 of this protocol.

ae) *FCS.* This field is the Frame Check Sequence, typically generated by the underlying MAC.

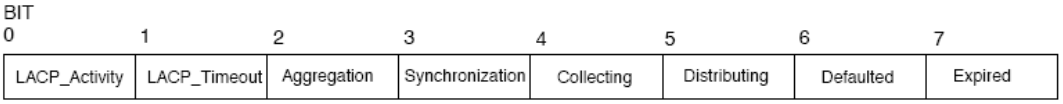


Figure C-2: Bit encoding of the Actor_State and Partner_State fields

The operation of this protocol is depends on the state machines, which execute a distinct function and control the protocol operation. The figure 4.3 shows these state machines and the interrelationship between them.

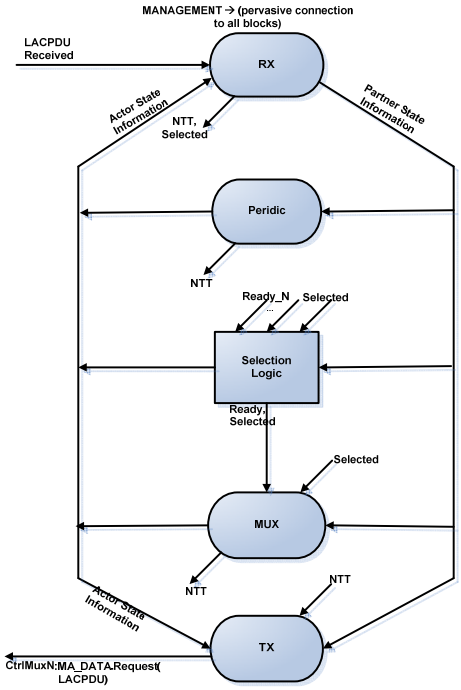


Figure C-3: Interrelationship among state machines [IEEE8023ad]

The state machines are explained in the following:

Receive machine: the assignment of this machine are receive the LACPDU from other partners, record the configuration information and also parameterize the LACP factors, such as Active LACP and LACP-Timeout. It can find, if the system need to transmit the LACPDU out of Transmission Period.

Periodic Transmission Machine: If LACPDU must exchange among the aggregation partners to maintain an aggregation, this machine execute this job.

Selection Logic: once an Aggregator should be selected to associate with a port, this machine will be called.

MUX Machine: Attaching ports to a selected Aggregator is the assignment of this machine, also the detaching a port from an Aggregator. Other responsibilities of this machine are the enabling and disabling the distributor and collector, by current protocol information.

Transmit Machine: this state machine is responsible for transmission LACPDU, in both periodic transmission and requirement of LACPDU by other state machines.

In this figure the NTT is the abbreviation of Need-To-Transmit, which is explained above. It means the other partners of an Actor need to upgrade its current information.