Requirements Analysis Report

CEng 491, Fall 2007

Auto-Identification / Classification of Common IP Protocols (ACCIPP)

Çağla ÇIĞ

Nazif İlker ERÇİN

Elvan GÜLEN

Can HOŞGÖR

Table of Contents

1. Introduction

In general terms, a packet sniffer (also known as a network analyzer or protocol analyzer) is a software or hardware that can monitor, state statistical information about and log traffic passing over a digital network or part of a network. As data is being transferred over the network in real time, the task of the sniffer is to capture ideally every packet and analyze its content in accordance with the appropriate RFC document or other specification. Most network analyzers allow port-specific tracking, i.e. they label protocol connections only by looking at port numbers. However, the need to overcome the limitations of traditional port-based protocol analysis arises since in today's networks an increasing ratio of the traffic (totaling roughly 5.6 million connections [1]) resist correct classification using different TCP/UDP port numbers. The reason for that increase is that today more users want to evade security monitoring and policy enforcement.

1.1 Problem Definition

Relying on well-known port numbers such as 80 for HTTP may not always be possible since applications may use arbitrary ports. The main reasons for that choice of usage are benign reasons and malicious intent. Benign reasons result from lack of user privileges, obfuscation, multiple versions; adversarial applications such as Skype bypassing firewalls. On the other hand, malicious intent results from the desire to evade from security monitoring like IRC bot-nets using ports other than the ones they are assigned to (666x/TCP). The necessity to distinguish these arises from the prevalence of the problem and has the consequence of the need for a new approach for dynamic analysis using auto sensing mechanism that performs port independent network analysis.

The auto-identification/classification of common IP protocols software to be developed for Siemens is a new system. It will be used as an application for capturing packets over the network and identifying most of the widely used IP protocols such as HTTP, NNTP, POP3,

IMAP, SMTP, WTP, SIP, FTP etc. The project is designed to run on both Windows and common flavors of UNIX thus it is platform independent. ACCIPP should be equipped with a user-friendly, intuitive, and easy-to-operate GUI that will provide quick and comfortable operation.

1.2 Project Goals

The project is aimed to satisfy the following goals:

- Identify the protocols such as SMTP, NNTP, POP3, IMAP etc.
- Capture some popular file formats like *avi*, *wmv*, *jpg* etc. from the detected protocols.
- Log instant messenger conversations.
- Give output in an appropriate format.
- Monitor and supervise network traffic for performance and security and bandwidth usage.
- Gather and report network statistics and help troubleshoot network problems.
- Generate and view reports in tables and charts on network usage.
- Filter suspect content such as spam, and denial of service attacks from network traffic.
- Spy on other network users and collect sensitive information.
- Debug client-server communications.
- Show relevant information like IP, protocol, host or server name etc.
- Determine when the identified protocol is no longer available in the flow through the identified port.
- High performance and low-latency (Real-Time) detection capability.
- Recognize incomplete protocol sessions.

1.3 User Classes and Characteristics

The expected user classes are as follows:

- Network administrators that need to enforce network policies.
- Network Software developers.
- Advanced users with expert knowledge and thorough understanding of network concepts.
- Students.
- Concerned parents.

The characteristics of these user classes can be generalized as:

- In need of a user friendly application to monitor and gather statistical and real-time information about network traffic
- In need of catching security breaches.
- Eager to optimize network performance.
- Curious about the working mechanism of network protocols.

1.4 Assumptions and Dependencies

Regarding the project design the following assumptions are made:

- Hardware and software requirements are properly met.
- ***Pcap*** files are correctly formatted and not corrupted.
- All packets except the encrypted ones, which are out of the scope of this project, are captured and stored in the ***pcap*** files.

The project dependencies are as follows:

- Properly configured and running X Server for *nix port.
- Qt is installed on the system that the software is working on.

- Availability of a properly working network sniffer for real-time analysis.

2. Project Process

Project process can be explained with the following subtopics:

2.1 Process Model

The project is designed to progress in discrete steps. Design and implementation phases are clearly separated. So the design phase will progress according to the linear sequential model. However, we are planning to release several prototypes therefore, the implementation phase including testing and debugging will progress with spiral model.

2.2 Team Organization

In this project, the structure of the team is decided to be Democratic Centralized (DC). The main reasons of this choice are that all team members have equal experience and same degree of engineering capability in managing and implementing a software project and there is no hierarchical order between team members. All communication between the team is horizontal. On the other hand, for efficiency reasons in certain aspects of the project flow, presence of a team leader is necessary. Therefore, one team member is designated to be the team leader throughout the project. The sole purpose of the leader is to make sure all scheduled deadlines are met, and guarantee successful cooperation within the team. The roles of team members and regulations for the team and project are determined at the beginning of the semester.

2.3 Project Constraints

Project constraints can be grouped like the following:

2.3.1 Time Constraints

Since senior project design is a two semester course, the project will have to be finished by the end of May 2007. All design, implementation and testing must strictly meet this deadline and complete in this 7 month period.

2.3.2 Language Constraints

For performance reasons, the language for the project is decided to be C++. Platform independency and code portability is an implementation constraint, thus all C++ code for this project will conform to ISO C++ standard. Development environment will be Visual C++ for Windows port, and a suitable GCC based environment for Unix/Linux ports. Qt library is planned to be used for the development of OS independent user interface system.

2.3.3 Data Constraints

A fair amount of primary storage space is required to hold various data structures used for analyzing data flow over the network. If the user chooses to save some data for later analysis, or the data cannot be processed in real time, the need for secondary storage space arises.

2.3.4 Performance Constraints

When used for real time analysis, ACCIPP will be exposed to high network traffic. Under these circumstances, the number of packets arriving per unit time will be quite large and average processing time given to a packet should be kept minimal. Since ACCIPP intends

to recognize a large number of protocols, the user should select only a subset of these in order to avoid starvation/packet drops. In a typical case where the number of packets per second is around 100 and presuming that the user might be running other applications, a maximum of 7-8 ms can be spent on each packet. Under such heavy load, ACCIPP should rely on predefined rule-based recognition engine rather than the relatively slow learning/training method.

2.3.5 User Interface Constraints

The interface must be kept simple and easy to use. Names of *menu*s and other *gui* elements will be easy to understand and straightforward. Accessibility features must be taken into consideration for handicapped users.

3. Research

Since none of the team members have prior experience in building network analysis applications, conducting market and literature research is compulsory. For this purpose, in this section the results of this research combined with the information gained from the meeting with Siemens will be presented.

3.1 Market Research on Existing Network Analysis Software

For market research, several network analysis software packages such as, EtherPeg, tcpxtract, Ether Detect and CAPSA were studied and compared. This market research was conducted in order to understand if currently available products satisfy customers' needs. The best way to evaluate a software package is to consult its users. Therefore by assigning the team members the role of potential users, advantages and disadvantages of these programs were investigated. Various usage scenarios were considered, and they were evaluated in terms of their weak and powerful sides according to these scenarios. In this

section, a brief summary about their distinguishing features and evaluation results are presented.

The common features used in evaluating the programs are explained below.

1. *Learning curve*: Time required for an average user to get used to the program is low.
2. *Performance*: The program can monitor and capture packets at wire speed.
3. *Reliability*: The program can operate consistently under heavy load.
4. *Supported Protocols*: The program is able to decode TCP/IP and many application protocols.
5. *Auto-sensing*: The program is able to identify the network protocol automatically.
6. *Flexible filters*: The program allows users to specify filters in terms of various parameters.
7. *Comprehensive output*: The program is able to generate reports in various detail levels.
8. *Database support*: The program is able to store information for later retrieval.

To demonstrate these programs' features clearly, a feature matrix is formed and each box is assigned a value. The resulting matrix is located in Appendix, part 8.2. From this feature-based comparison chart, it is concluded that there are mainly two categories of network analysis programs. The ones in the first category are able to recognize a large number of protocols, but since their protocol sensing mechanism is simply based on predefined port numbers, they are not adequate to extract valuable information from connections over non-standard port numbers. The ones in the second category, on the other hand, can detect and capture various file formats transmitted over an arbitrary protocol connection. However, since all these programs do is to “dump” recognized files, they are not perfectly suited for generic network analysis. This project distinguishes itself among others in that, it does not rely on port numbers for protocol detection and rather takes an AI based and probabilistic approach.

3.2 Literature Survey

In order to design and develop a network analysis application, a fair knowledge about network concepts and network programming methodology is required. A broad research on these topics were conducted. First of all, the team needed to become familiar with relevant terminology (such as "packet", "connection", "sniffer" etc.) and fundamental concepts such as network architecture and roles of the network elements, most importantly how packets are transmitted. This introductory knowledge was gained through reading articles on Wikipedia[1].

Next, a popular network sniffing application (Wireshark, formerly known as Ethereal) was installed. This application is chosen because it is a mature product and has the ability to capture network traffic as **pcap** files. Wireshark documentation was consulted whenever a difficulty was faced while using the program. Using Wireshark when analyzing packets, an overall idea about the inner structure of packets is gained, but it was not enough to give a deep insight about the protocols. In order to perform pattern recognition, complete specifications of the command-response relationship of the protocols were vital for the purpose of the project. Therefore, RFC documentations particular to the mentioned protocols were divided among team members. Consequently, the team had a thorough understanding of these protocols that will assist in pattern recognition.

The following is a list of RFCs that have been or are being planned to be studied in the first place.

- Network News Transfer Protocol (NNTP) [2]
- Internet Message Access Protocol (IMAP) [3]
- Simple Mail Transfer Protocol (SMTP) [4]
- Post Office Protocol Version 3 (POP3) [5]
- Hyper Text Transfer Protocol (HTTP) [6]
- Extensible Messaging and Presence Protocol (XMPP) [7]

- Session Initiation Protocol (SIP) [8]

Since this project is designed to be a stand-alone application, a module for real-time capturing of network packets in **pcap** format is needed to be developed. For this purpose, an open source library that could assist in detecting the network device, opening the device for sniffing, filtering traffic and capturing this information in **pcap** format is searched. As a consequence of this research, **libpcap** was found suitable for this purpose in both being an open-source and widely used library, and providing a packet filtering mechanism based on the BSD packet filter.

Having completed the research on fundamental necessities of a basic network monitoring and analysis application, the next step to be taken is to conduct research on the most distinguishing feature of the project namely pattern recognition based auto-sensing module.

Network protocols tend to have a well defined set of rules, and conversations between the client and the server computers must conform to these rules. Therefore, packets that belong to a protocol follow a common "pattern". If the pattern of the protocol is known, one can (at least probabilistically) decide, weather a packet can belong to the given protocol, by examining the character sequences the packet contains.

For the time being, it can be stated that, pattern recognition in the project can be done via two ways: Supervised Learning and Unsupervised Learning. Supervised Learning is implemented using machine learning techniques. Among these techniques "Support Vector Machines" is found to be most appropriate since it is used to detect and exploit complex patterns in data by clustering, classifying and ranking the data. Consequently it is decided to narrow down this part of the research to "Predictive analysis", "Support Vector Machines", and other pattern recognition techniques like "String Metrics" etc. Popular open-source tools which will make assistance with the execution of predictive analytics are WEKA, RapidMiner and SimMetrics. Regarding the major points mentioned in this section, the following publications and related informal material are planned to be studied further:

- A Tutorial on Support Vector Machines for Pattern Recognition [9]
- A Probabilistic Analysis of a Pattern Matching Problem [10]
- Video Lectures on Machine Learning, Pattern Recognition [11]
- Pattern Recognition on the Web [12]
- String Metrics for Information Integration [13]

3.3 Meeting with Siemens

Considering none of the team members have previously worked on such as large scale software project, lack of experience is a major risk for the project. In order to overcome this deficiency, to give the teams a starting point and a brief summary about the project constraints, a meeting with Sevgi Yaşar of Siemens Corporation was held on October 6, 2007.

Before attending the meeting none of the teams were fully aware of the problem description and accomplishment objectives of the project. In this meeting, various points about the project were made clear and requirements for the project were defined. Ms Yaşar was well prepared for the meeting, and communicated the subject clearly. She used visual aids such as diagrams and flow charts whenever verbal explanations became insufficient. The teams had the chance to ask questions for clarifications or state their ideas openly. Therefore, generally speaking, this meeting has been quite helpful for us. A brief summary of notes taken at this meeting can be found at Appendix, part 8.1.

4. Project Requirements

Understanding the needs of the project, the project requirements should be specified. During the determination of requirements analysis, the steps taken are as follows:

4.1 Functional Requirements

Below, the functional requirements for ACCIPP are explained briefly.

4.1.1 Capturing Packets

The input will be captured from a network device or taken as already existing pcap files.

4.1.2 Preprocessing

The captured packets may need reordering and/or defragmentation. The preprocessing mechanism handles these operations.

4.1.3 Filtering

The user of the system may not want to receive irrelevant data that s/he is not working on. Thus the filtering mechanism is employed to filter the packets which are of concern. Filters can be defined by several identities of connections such as IP addresses or protocol data.

4.1.4 Auto-Sensing

The system is expected to identify the packets without using port information. Auto-Sensing mechanism takes action in this identification process using some *Artificial Intelligence* algorithms.

4.1.5 Processing Identified Connections

The proper output for the analyzed protocol of the connection are sent to output mechanism.

4.1.6 Output Mechanism

The data that is received from the system, will be displayed as reports or user interface summaries. If asked, more detailed information about the connection can be given as output.

4.2 Non-Functional Requirements

In this section various non-functional requirements such as usability, portability, reliability and documentation will be mentioned.

4.2.1 Usability

The program has to be easily adaptable for novice users, and powerful enough for experienced users. User interface elements such as menu items and command buttons have to be as clear and self-explanatory as possible. They should provide tooltips where applicable. The resulting graphs should allow the user to obtain rapidly an overall grasp of the material presented.

4.2.2 Portability

The software package is designed to be a cross platform product, therefore it should not rely on machine and/or OS dependant functionality such as byte ordering and non-standardized system calls. Consequently the program will be able to compile on different computer systems without being altered.

4.2.3 Reliability

The software package is planned to be used in large and corporate networks, thus it is a critical requirement that the software functions consistently under such circumstances.

4.2.4 Documentation

User documentation includes *online help* and user manual for the product. A hardcopy of the *user's manual* will also be provided with the software package.

4.3 Software Requirements

In this section, the external software packages ACCIPP depends on will be presented.

4.3.1 Operating System

ACCIPP shall function on Windows versions starting from Windows 2000, and major Linux distributions like Debian, RedHat etc.

4.3.2 External Packages

ACCIPP requires the presence of an external libpcap compatible packet sniffer and an adequate network adapter in cases where real-time processing is deemed necessary.
In addition to that, Qt library must be installed in order to have user interface functionality.

4.4 Hardware Requirements

In this section, hardware requirements for the software project are presented.

4.4.1 Minimum Hardware

In order to have basic functionality, a system with 256 MB Memory, Pentium III class CPU, 10 MB Hard disk space is required.

4.4.2 Recommended Hardware

To be able to make full use of the auto-sensing facility and store statistical information in the database backend, a system with at least 1 GB Memory, 2.5 Ghz Pentium IV class or higher CPU, 4 GB Hard disk space is required.

5. System Analysis and Modeling

Modeling is a useful way to express the connections between the parts of the project. So it is functional to use models to make the project clear. The models that are formed to specify the project are mentioned below.

5.1 Structured Analysis – Functional Model

Visual explanations of ACCIPP are defined using several diagrams.

5.1.1 Level 0 of DFD (Context Diagram)

A very general view of the system is shown in Figure 1, Level 0 DFD.

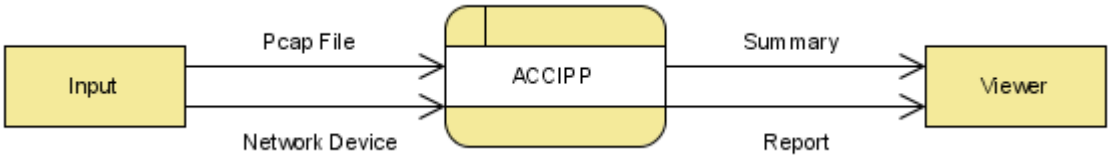

Figure 1: DFD Level0

5.1.2 Level 1 of DFD (Overview Diagram)

A more detailed version of the DFD above is as follows:

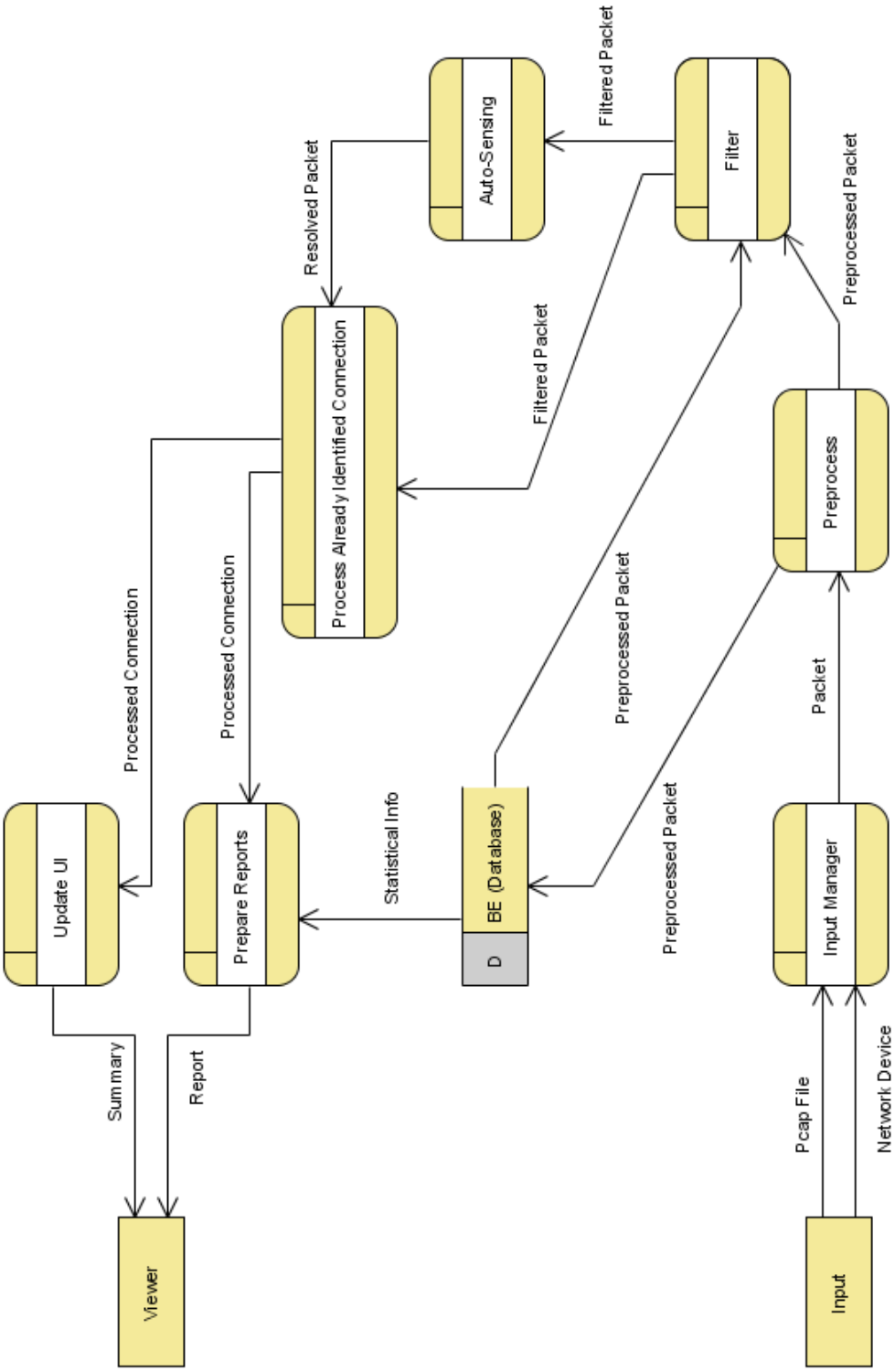

Figure 2: DFD Level1

5.1.3 Structure Chart for Overview Diagram

Another visual diagram is the structure chart, showed in Figure 3 below.

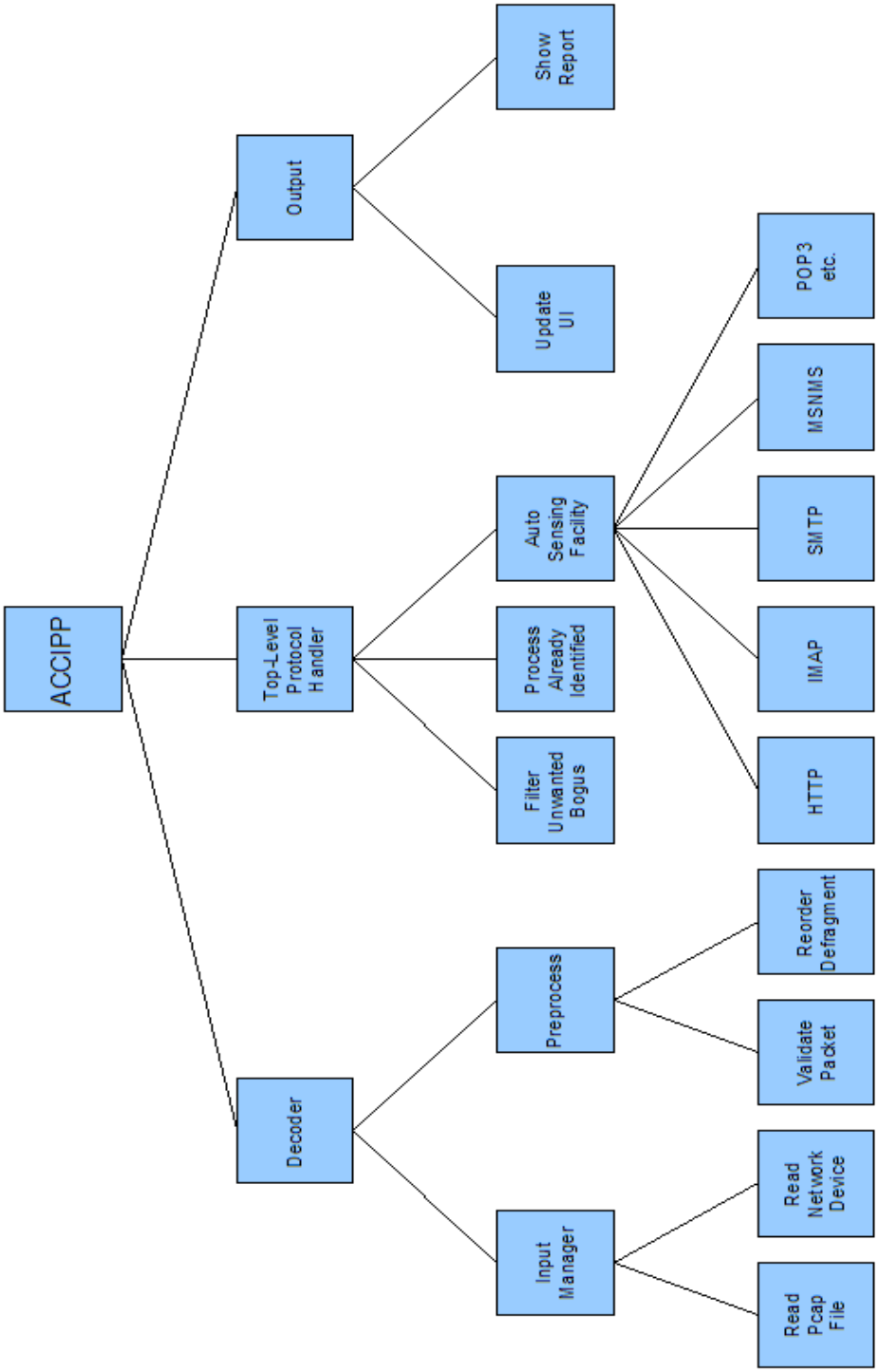

Figure 3: Structure Chart

5.2 Use Case Analysis

The Use Case Diagram for the project can be seen in the following figure :

Figure 4: Use Case Diagram

6. Project Schedule

Gantt chart is used to visualize the schedule of ACCIPP including only the first term of the project.

6.1 Gantt Chart

The Gantt chart of the project can be found in Appendix, part 8.3.

7. Risk Management Plan

As any software project can face with several undesirable circumstances, throughout its entire progress, ACCIPP would possibly come across with many of these. This problematic issues should be handled by an effective risk management plan. This plan should be considered as broad as possible so that the negative effects of risks that will show up during the future phases of the project can be minimized and the progress will maintain stable.

7.1 Project Risks

The following subtopics basically define the foreseen risks of the project.

7.1.1 Project Team (PT)

- *Health issues* affect team member availability and concentration on the project.
- *Team member unavailability* causes inefficient work on the project.
- *Lack of interest* may cause delays in project schedule.
- *Lack of responsibilities* affects the seriousness of the project work.
- *Lack of self-confidence* will introduce a simple project without any original ideas.

- *Lack of development experience* clearly creates conflicts with the project schedule. Because of this risk, wrong steps may be taken during the project.
- *Problems between group members* causes communication problems and consequently defects the integrity of the project.

7.1.2 Development Process (DP)

- *Flawed Design* may result in irreversable defections.
- *Difficulty of implementation* : Project may require complex algorithms.
- *Poor comments in code* : It is a disadvantage for readibility.
- *Module integrity problems* may influence the accuracy of the program.
- *Missing the deadline* will not be welcomed by the supervisors.
- *Quality of testing issues* : If testing phase is not comprehensive, then it affects the integrity in a negative manner and bugs may occur.

7.1.3 Company Related Risks (CR)

- *Problems with the supporting company* : Because of the lack of support and connections, the project may be affected negatively.
- *Changes of demands of supporting company* may cause waste of code, documentation and time.

7.1.4 Technical Issues (TI)

- *Lack of useful documentation* may delay the progress since the access to useful information takes more time.
- *Insufficiency of software tools* affects the development progress of the related parts of the program.

7.1.5 Project Arguments (PA)

- *Hardware constraints* : If hardware features do not meet the recommended system requirements, problems may occur.
- *Large project size* increases the complexity and it will be hard to manage.
- *Storage space limitations* may degrade the performance.

7.2 Risk Table

The risk table for ACCIPP can be found below.

RISKS	PROBABILITY	NATURE	IMPACT
Health issues	10%	PT	Negligible
Team member unavailability	30%	PT	Critical
Lack of interest	15%	PT	Marginal
Lack of responsibilities	20%	PT	Critical
Lack of self confidence	5%	PT	Marginal
Lack of development experience	35%	PT	Critical
Problems between group members	25%	PT	Marginal
Flawed design	5%	DP	Catastrophic
Difficulty of implementation	50%	DP	Catastrophic
Poor comments in code	25%	DP	Marginal
Module integrity problems	15%	DP	Catastrophic
Missing the deadline	10%	DP	Catastrophic
Quality of testing issues	30%	DP	Critical
Problems with the supporting company	20%	CR	Marginal
Changes of demands of supporting company	10%	CR	Critical
Lack of useful documentation	10%	TI	Marginal
Insufficiency of software tools	10%	TI	Marginal
Hardware constraints	5%	PA	Marginal
Large project size	20%	PA	Critical
Storage space limitations	20%	PA	Critical

8. Appendix

8.1 Notes from the Meeting with Sevgi Yaşar

From the meeting the ideas and suggestions of Ms. Yaşar can be summarized as follows:

- The following protocols are going to be recognized with this program: HTTP, SMTP, POP3, NNTP, IMAP, MSN, YAHOO, JABBER and SIP.
- Since the program is going to operate in real-time, C++ should be chosen as the development language.
- The testing phase of the program should first be conducted offline with the previously generated **pcap** files and the second part of the phase should be done in real-time.
- RFC documents found online can be observed for detailed structures of the **pcap** files of the particular protocols.
- Wireshark can be used for real-time monitoring of network traffic.

As a consequence of the meeting, the following model of the project was generated:

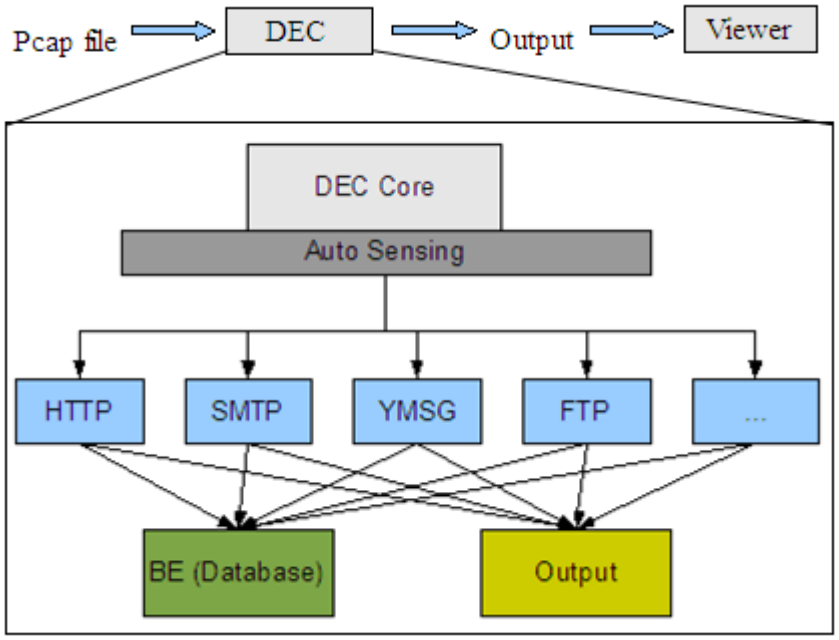

Figure 5: A simple model of the Project

8.2 Results of the Market Research

	Ease of Use	Performance	Reliability	Supported Protocols	Supported Formats	Auto-sensing	Flexible Filters	Comprehensive Output	Database Support
EtherPeg	Low	Real-time	Moderate	N/A	JPEG, GIF images	No	No	Exports output directly to files	None
tcpxtract	Low	Real-time	Moderate	N/A	Document, image, audio, video	No	No	Exports output directly to files	None
Ether Detect Packet Sniffer	Moderate	Real-time	High	All TCP and UDP	HTML, HTTP, XML	No	No	Syntax Highlighting for Supported Formats	None
Network Packet Analyzer CAPSA 6.5	High	Real-time	High, almost no packet drops	All leading protocols	N/A	No	Yes	Reports, statistics, graphs, Matrix view of peer conversations	None
LANWatch Traffic Monitoring	Moderate	Real-time	High	All leading protocols	N/A	No	Yes	Statistics, graphs, summary, detail	None
SoftPerfect Network Protocol Analyzer 2.5	Moderate	Real-time	Moderate	Mostly low-level protocols	N/A	No	Yes	Basic	None
Wireshark (Ethereal)	Low	Real-time	High	All leading protocols	N/A	No	Yes	Basic	None
Distinct Network Monitor	Excellent	Real-time	High	All leading protocols	N/A	No	Yes	Advanced - Statistics Building Module called ReportBuilder	Statics DB
Snoop	Very low	Real-time	Moderate	Most leading protocols	N/A	No	Yes	Binary Output File	None

8.3 Gantt Chart

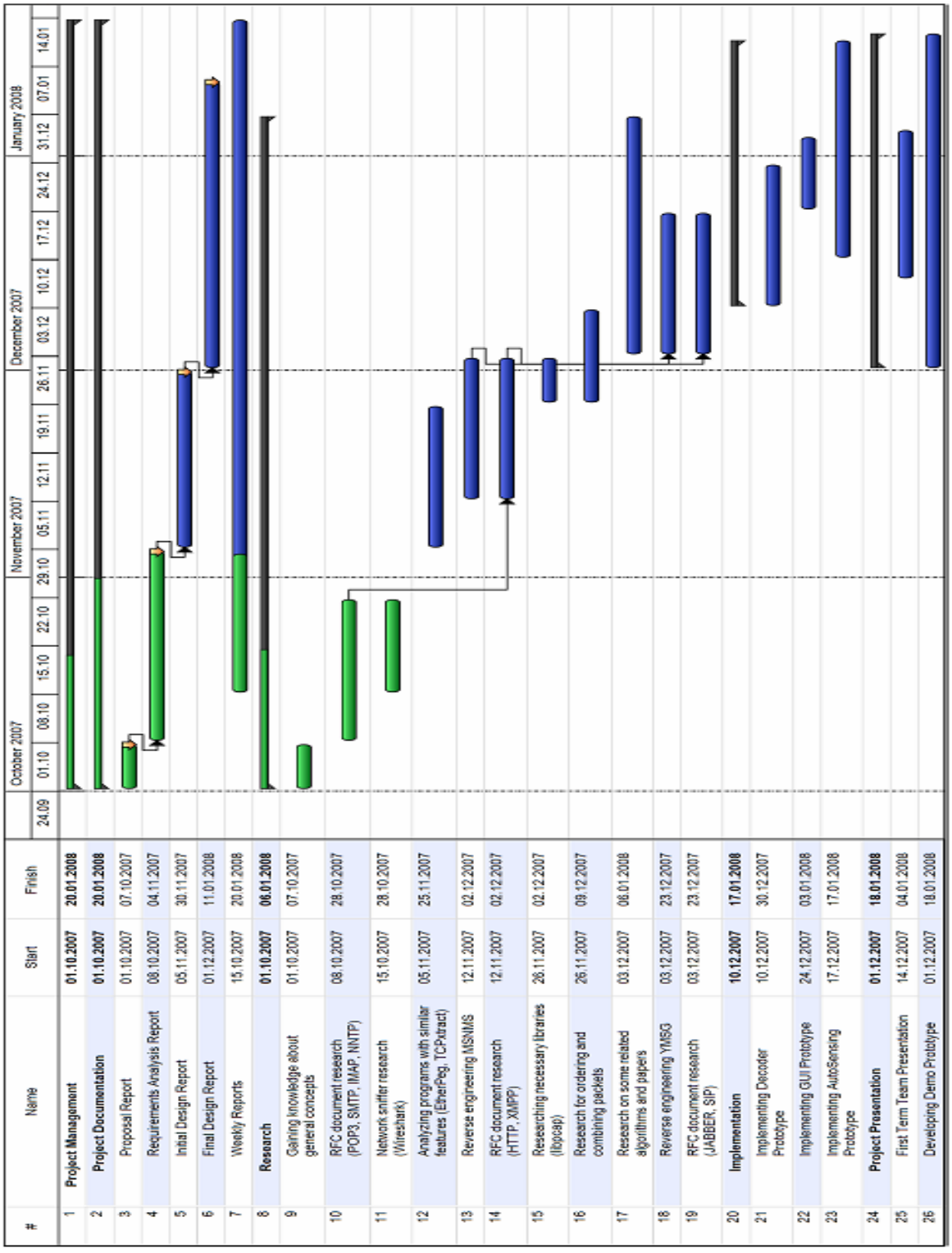

Figure 6: Gantt Chart for ACCIPP (1st Term only)

8.4 References

[1] Dreger H., Feldman A., et.al, "Dynamic Application-Layer Protocol Analysis for Network Intrusion Detection",

[2] The Internet Engineering Task Forms. http://www.ietf.org/rfc/rfc0977.txt.

[3] The Internet Engineering Task Forms. http://www.ietf.org/rfc/rfc3501.txt.

[4] The Internet Engineering Task Forms. http://www.ietf.org/rfc/rfc0821.txt.

[5] The Internet Engineering Task Forms. http://www.ietf.org/rfc/rfc1939.txt.

[6] The Internet Engineering Task Forms. http://www.ietf.org/rfc/rfc2616.txt.

[7] The Internet Engineering Task Forms. http://www.ietf.org/rfc/rfc3920.txt.

[8] The Internet Engineering Task Forms. http://www.ietf.org/rfc/rfc3261.txt.

[9] Burges C., "A Tutorial on Support Vector Machines for Pattern Recognition".

[10] Atallah M., et.al, "A Probabilistic Analysis of a Pattern Matching Problem", 1992.

[11] Video Lectures on Machine Learning, Pattern Recognition. http://videolectures.net/Top/Computer_Science/Machine_Learning/Pattern_Recognition.

[12] Pattern Recognition on the Web. http://cgm.cs.mcgill.ca/~godfried/teaching/pr-web.html.

[13] String Metrics for Information Integration. http://www.dcs.shef.ac.uk/~sam/stringmetrics.html.