Multi-Protocol Correlation: Data Record Analyses and Correlator Design

by

Ole-Martin Rodal & Erdem Özkan

Supervisors:

Frode Gorseth & Frank Li

This Masters Thesis is carried out as a part of the education at the University of Agder and is therefore approved as a part of this education.

University of Agder
Faculty of Engineering and Science
Department of ICT

Abstract

This thesis has two main goals. The first one is to design a user configurable multi-protocol correlator and implement a prototype of said design. The second goal is to identify and propose a method to match different data records from different protocols.

In essence, this thesis is about correlation of records which contain information about protocols or services that are generally used in telecommunication networks. In order to reach the two main goals of this thesis, we need to combine our knowledge from the programming world with our knowledge from the networking world. Correlation can be done on multiple levels; you can correlate protocol messages, and you can correlate whole calls or transactions which allows you to perform correlation across sections of a network. We approach this problem by gathering protocol signaling data, specifications on how the protocols work, and log files with examples. With this knowledge we were able to identify many of the problem areas related to correlation of the main protocols to be used in this thesis. We designed a configurable correlator that could be configured to overcome the problem areas related to correlation provided enough data was given. The prototype correlator was tested both on correctness and performance. Then, in order to validate the correctness and preciseness of our developed prototype correlator, we compare the correlation results obtained from our tool with the results obtained using Utel Systemś STINGA NGN Monitor. The comparison shows that the correlation results from our prototype correlator are satisfactory.

Preface

This thesis is a document on the problem, method and result of the master thesis course IKT 590 from the master program at the Faculty of Engineering and Science at University of Agder in Grimstad.
This thesis, *Multi-Protocol Correlation: Data Record Analyses and Correlator Design*, was initially proposed by Utel Systems, and tweaked into a master thesis by Ole-Martin Rodal and Erdem Özkan with the approval of Frank Li and Frode Gorseth. Utel Systems is a telecommunication company based in Grimstad that specialized in telecommunication testing and monitoring. They create protocol analyzers and probes to monitor telecommunication traffic, as well as protocol simulators.
This thesis is done by two students working as a group. The external supervisor from Utel Systemś Managing Director Frode Gorseth, the internal project supervisor from University in Agder is Professor Frank Li.

Ole-Martin Rodal
Erdem Özkan
Grimstad 25.05.2011

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Since the creation of the Internet we have seen a steady increase in the number of protocols and the amount of communication traffic, and this is in addition to the traditional telephone networks. What we get from this is millions of devices connected together in small and large networks, which in turn will communicate with each other. These networks have not just covered the earth like a spider web, but also introduced various types of languages which they use to communicate with each other. We call those languages *protocols*. The need for tools to monitor, analyze and detect chokepoints and anomalies across networks is growing as different protocols and technologies begin to communicate with each other in an increasing fashion. In this thesis the focus will be on how to achieve this; predominantly for telecommunication protocols, but the base ideas in this thesis will also be valid for other protocols and technologies.

There are two main parts of this thesis, modeling and implementation, and correlation theory. The first part can be divided into two major sections, the modeling part and the implementation part. The modeling part is about modeling how it is possible to gather data from the different protocols and put the relevant signaling data together. The process of finding and putting relevant data together is a process we call *correlation*, and the program produced from this model is called a *correlator*. The implementation part is about the implementation of a prototype

version of the model. This prototype correlator will be a simplified version of the whole mode, but the core component, user configurable multi-protocol correlation is implemented. Its goal is to prove or disprove our ideas on how to correlate. The second part is the theory on how to correctly correlate a small subset of protocols defined in this thesis problem statement.

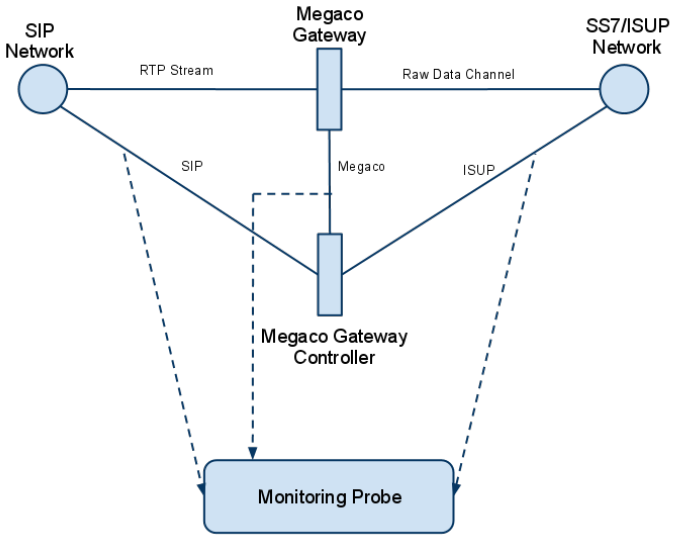


Figure 1.1: An illustration of how a system may look like.

In the scenario illustrated in Figure 1.1 we have an interaction between three main signaling protocols. Session Initiation Protocol (SIP) and ISUP all play a very important role in setting up a call. The signaling information from the different protocols is gathered by a monitoring probe. All captured data is then analyzed and manipulated based on some pre-selected user variables.

1.1 History

For the traditional telecommunication network such as SS7 there are a few protocols such as ISUP, MAP, CAP and Intelligent Network Application Part (INAP) which define most of the services that an SS7 network provides.

Recently, we have witnessed the growth of Internet Protocol (IP) based telephony. In this IP based telephony the primary protocols which define most the services available is H.323 and SIP with all the extensions to the original specification. In addition to this development of pure IP based telephony the classic implementation of the SS7, whose protocols are traditionally carried over MTP 1, 2 and 3, we have seen an updated version of SS7. The new version called SS7 over Signaling Transport (SIGTRAN) works in the same way as the old one, but the MTP transport protocol have been switched out with IP.

Presently we are seeing that the two networks, IP and SS7 are converging with services such as calling lane lines from computer programs such as Skype. This converging of the networks has caused some challenges for the operators of the telecom networks such as billing and call tracing. In order to address those challenges, the need to figure out how to easily correlate calls and services carried over the different protocols has become more important and that is largely what this thesis is all about.

1.2 Background

There are a myriad of different protocols in various types and layers in telecommunication. The protocols range from the physical layer to the application layer, and they follow the standard Open Systems Interconnection (OSI) model. Although the project theoretically aims to design a system that is independent of protocols, the sheer size of such a project is too large and complex to be completed in this thesis. As such this project will have its boundaries reduced to a few different protocols across a few architectures.

Correlation in telecommunication is a very broad field with a lot of the groundwork already done. However, the complexity of the different protocols working together, and the speed at which the different telecommunication networks and the Internet are converging is often causing the established ways of correlating to fail. This usually happens because they are to focused on one specific case. There are two major types of correlation:

1. Message correlation, which is about finding and grouping messages belonging to the same calls and transactions.
2. xDR correlation, which is one step up from message correlation. xDRs are the result of message correlation.
3. End-to-end correlation, which is correlation of signaling data from multiple links of a network and even mulitple protocols. It is the most complex type of correltion. Some scenarios are just not possible to correlate, and in other cases the information is incomplete, and a critical message may be missing.

In telecommunication; business, service and network operators often store protocol messages captured by monitoring probes in a protocol message repository. Protocol messages from the repository are correlated into records (call/transaction records - in general xDRs). It is on these logs that this thesis will be performing the correlation on.

End-to-end is the process of doing correlation over multiple sections of a network. An example is a monitoring probe that gathers information on both sides of a switch, if this probe is all you know then end-to-end will be from one side of the switch to the other.

1.3 Thesis definition

Utel Systems is the problem owner and the definition was written by them. This thesis will focus mostly on the following telecommunication protocols: ISUP [10],

SIP [13], Media Gateway Control Protocol (Megaco) [11] and CAP [16]. They are all application layer protocols, and they are all deeply involved in setting up and terminating calls, or working as a bridge in between the different protocols in today's telecommunication network. It is these four protocols that this thesis will focus on when trying to correlate signaling data.

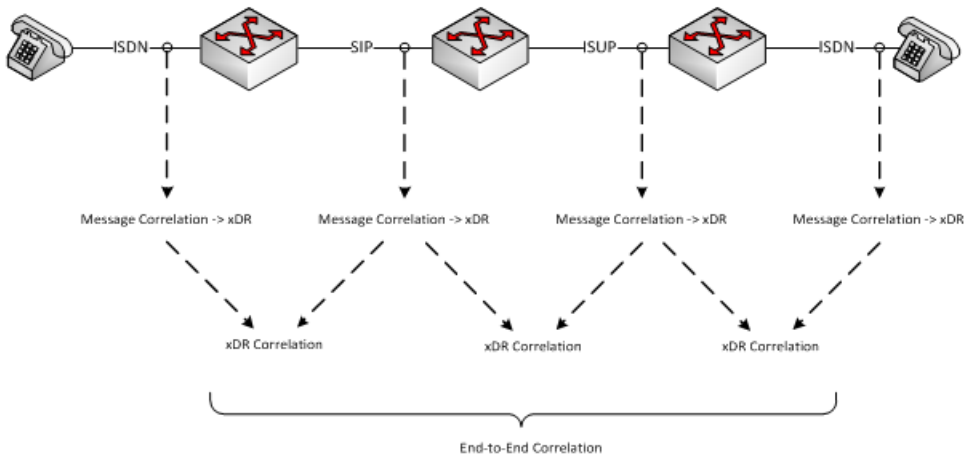


Figure 1.2: Utel Systems illustration of an example system

The image in Figure 1.2 how Utel Systems imagines the overall system for data gathering. Our work in this thesis will be limited to the two lower halves of the Figure, xDR correlation and end-to-end correlation. In both of those cases the actual correlation process is performed on xDRs.

In summary; this thesis consists of the following tasks:

- Describe a general model for end-to-end correlation of xDRs. The design should be as generic as possible, but focusing on a limited number of protocols; ISUP, SIP, Megaco and CAP.
- Propose a method to match data records from these protocols by investigating the similarities between different networks and protocols.
- Identify and discuss challenges related to:

  - Call Detail Record (CDR) correlation involving intelligent network services.
  - General and configurable correlator vs. hardcoded correlator.
  - The need for timers within the correlator model.

- Implement a general and user configurable xDR correlator based on the proposed method.
- Validate and evaluate the implemented correlator by using end-to-end correlation of a specific case, preferably using ISUP-CAP and/or ISUP-SIP and/or SIP-Megaco.

This thesis has two main goals. The first one is to design 'a user configurable multi-protocol correlator and implement a prototype correlator based on the design. The second one is to identify and propose a method to match different data records from the different protocols i.e. propose a way to correlate between the different protocols. Before we can start the modeling process, all possible interactions between the key protocols we will focus on needs to be found. To model a correlator, we must identify firstly all detect all possible interactions between protocols. A concrete example would be how the Media Gateway (MG) that separates SIP and ISUP operates, and by figuring that out it is possible to identify which parameters that needs to correspond with each other.

1.4 Literature Review

Correlation in general is a field that has been studied extensively. The roots can be found in statistical mathematics, but we will for the most part focus on reading specifications and figuring out what kind of parameters that is important in correlation. This project is divided into two major branches in which we can find literature. The first one; correlation is a very math heavy subject. The second one; is all the network and protocol specifications.

1.4.1 Related Work

Statistical Approach

We have found some work which is related to what this project is about. One solution for correlation of messages or xDRs which has already been done is called *Message Correlation and Business Protocol Discovery in Service Interaction Logs* [4]. This solution is interesting in the sense that it assumes that the parameters required for correlation are missing, and as such they have used a graph theory approach. The main issue with this approach is uncertainty. Since it is based on probabilistic reasoning, it cannot guarantee correct correlation, but it can approximate it. Another issue is focused on message correlation not end-to-end. It should however be possible to adapt it for end-to-end.

Deterministic Approach

Message Correlation and Web Service Protocol Mining from Inaccurate Logs [14] which tries to solve correlation by using deterministic computing. This solution is too far away from the telecommunication world to be of anything other than superficial use.

A completely different solution that takes our attention is called *Multi-protocol call trace on General Packet Radio Service (GPRS) Gb-Gr* [3]. It introduces a method of performing a multi-protocol call trace on GPRS Gb-Gr interfaces of a Global System for Mobile Communications (GSM) network. Transaction Detail Records (TDRs) are the major data source for the correlation algorithms used in this approach to look for and cover specific scenarios. This approach also focuses on the slow decoding process on huge amounts of Protocol Data Units (PDUs) and tries to bring a solution that needs less time. It desires a fast and easy way to perform procedure trace on GPRS data without involving a lot of bandwidth while allowing correlation of GPRS and GSM results. However, this solution belongs to its proprietary owners and is limited to GPRS, so it is not that interesting for us in this project.

Protocol Specific Approach

MEGACO correlation method [15] is one example of correlation, but this one is too specific for the Megaco protocol. These are a few examples of what other people have done in this research area. This solution is a little too focused on Megaco, to be of use to us when we are supposed to design and implement a prototype of a generic and configurable correlator. It might however be useful to look at when trying to configure the correlator to work for Megaco.

Summary

The end-to-end correlation field varies from protocol to protocol, thus it is hard to refer to previous research that may be of significant value to us. The above approaches are all specialized either on protocols or on the method of solution. We need an approach that can be specialized based on the users wishes, but at the same time is generic enough to cover all protocols.

1.4.2 Technical Literature

Many papers and all the specifications related to the four protocols specified in the problem statement (ISUP, SIP, Megaco and CAP) have been studied in order to complete this thesis. Documents of importance to this thesis are the SIP specification RFC 3261 [13], as that one explains the basic case for how SIP is supposed to be deployed and how it should work. In addition to the main specification for SIP there are also other supplementary specifications on how SIP should work with protocols and special services for SIP; such as RFC 3858 which updates the RFC 3261 to require Advanced Encryption Standard (AES) for Secure/Multipurpose Internet Mail Extensions (S/MIME) [12]. For ISUP the main specification is ITU-T Q.763 [10], and like SIP there are also supplementary specifications which try to cover special cases. Megaco has H.248.1 [11] as the main technical specification. Finally, CAP has TS 29.078 [1] as the main technical specification and is by itself an extension to the MAP specification [2].

For non-technical specifications our main research references are *Signaling and Switching for Packet Telephony* [17, Ch.10], which has a very good and detailed explanation on how Megaco works. Additionally, we have *Signaling System No. 7 (SS7/C7) Protocol, Architecture and Services* [6] which has a thorough explanation on how ISUP works, both in IP network, (SS7 over SIGTRAN) and in the traditional circuit switched telecommunication network.

1.5 Problem Delimitation

Correlation is a huge field, so drawing the borders is crucial to have a good progress on such a vast project such as this one. There are various reasons for drawing boundaries on this project, the main reason for the limiting is time. Some of the other reasons are because of our knowledge and interest.

1.5.1 Approach Limitations

One of the other approaches considered early on in the creation of the thesis definition was to use a statistical approach to problem of xDR correlation, however due to the uncertainties of the results produced by this approach we decided not to use it.

1.5.2 Design Limitations

GUI design is not specifically covered in this thesis, and as such the current GUI implemented in the prototype is very basic. Additionally, the correlator implemented will only be a prototype because a full implementation will take to long.

1.5.3 Protocol Limitations

The four mainstream protocols SIP, ISUP, CAP and Megaco mentioned in section 1.3 will be the protocols focused on. Other protocols are not specifically focused

on, however thought has been given to make the prototype correlator as generic as possible.

1.5.4 Correlation Limitations

Another excluded field is Message correlation, briefly explained in section 2.2.3 will not covered in this thesis as the subject of focus is xDR correlation.

1.6 Problem Solution

In short the problems that need to be solved in this thesis are which protocol parameters that are relevant for xDR and end-to-end correlation. The other major part of this modeling a generic and configurable xDR correlator and implement a prototype of this.

Most of the existing solutions available were of little use because either too specific in their area of application or too generic and uncertain. For this thesis the solution we used was, for the most part, already outlined by the problem description. As such we focused our attention on correlating the selected protocols ISUP, SIP, Megaco and CAP. The correlator is designed in such a way that the engine, which contains the actual correlation algorithm, is separated from the user interface. In addition, the design of linked lists allows the correlator to operate on real-time data or by reading data from a file. Our implementation of the prototype correlator differs a little from the overall design. The reason for the deviation is because we simplified some of the problems due to time constraints, but the overall design idea is carried over from design to the implementation.

The last part of our solution is about how to perform correct correlation between the four protocols ISUP, SIP, Megaco and CAP for the most common scenarios and some of the more complex scenarios. Unfortunately, because of the incredible complexity and amount of possible scenarios, we had to limit our solution to a subset of the infinitely large set.

1.7 Report outline

Chapter 2, **Background**, provides information on what xDRs are before moving on to explaining more specific cases of xDRs such as CDRs. After that sub-chapter we move over to other important terms, such as explaining what correlation in general is. Once correlation is explained the chapter continues on to a more detailed explanation on what message correlation and xDR correlation is. After that a brief explanation of programming architecture and the OSI model is presented before moving on to an in-depth explanation of the similarities between the four major protocols in this thesis; ISUP, SIP, Megaco and CAP.

Chapter 3, **Proposed Solution**, provides information on our solution in various sections that each covers a different topic/aspect. This chapter starts with introducing the Scenario Classification that provides how the correlation is handled for different correlation scenarios. That chapter is followed by Correlation Theory which introduces the solution in a theoretical way. After that Correlator Design is introduced. This covers the high level model and design information about the protocol correlation. The next chapter deals with the key details on our implementation of the prototype protocol correlator. Finally the last section of Chapter 3 focuses on the user interface of the correlator and introduces the interface components.

Chapter 4, **Validation and Testing**, starts with an introduction to validation and testing, and is followed by a section about STINGA NGN Monitorś role for testing. In the following section we mention the significant details about time performance testing. The chapter is finalized by introducing the topics on test process of the scenarios and the testing of implemented correlator prototype code.

Chapter 5, **Discussions**, focuses on discussions. In the first section we challenge our solution under major topics such as hardcoded vs. user configurable implementation, alternatives to sliding time window and the differences of xDRs and how they impact correlation. The last section in this chapter aims to convey different approaches to the same thesis definition; the major example of this is using graph theory.

Chapter 6, **Conclusions and further work**, will in brief terms explain our

solution and how well we completed the tasks given to us. After that section will try to predict the impact that our solution has for Utel Systems and us. Finally it will explain some of the further work that we did not implement because it was not part of this thesis.

Chapter 2

Background

In this chapter the underlying technologies and some of the most interesting ideas around correlation and programming architecture will be presented. It is broken down into the concept of what correlation is, and the type of data which is to be correlated. Finally, different types of programming architecture that can be used to program a correlator will be presented.

2.1 xDR

Different Call/Transaction Detail Records all have a specific purpose and content. xDR is a generic type of detail record where the x letter is to be replaced with another letter depending on the type of detail record. The data structure and format of xDRs are very loosely defined and configurable. A more specific detail record is CDR which is the most common one in this thesis; it usually refers to protocols like Integrated Services Digital Network (ISDN), ISUP, SIP and H.323 - i.e. call related protocols. Another type of detail record is TDR, and this usually refers to protocols like Transaction Capabilities Application Part (TCAP), MAP, CAP and Intelligent Network Application Part (INAP). A third, and for us not very interesting, type of xDR is IP Detail Record (IPDR) which usually refers to more general IP-based protocols like Domain Name System (DNS).

xDRs also come in another format called Enhanced xDR which is exactly the same, but they also include the raw signaling information inside the xDR. This signaling information is stored inside the xDR in plain-text as a set of hex bytes, and by decoding information inside the hex bytes the raw signaling information can also be presented.

2.1.1 CDR

CDRs contains information about one particular call.

```
CDR                             'UQoS cdr message'
MI              00240033        'UQoS CDR message indicator' Local, Primary, International UP, International NI
DATE            20091021 015915 'UQoS message DTG'
TRKGRP          BGD_MIR_MBAZAR  'Trunk Group'
DID             98637           'Destination IDentifier'
CSYM            BD              'Country symbol (short country name)'
CC              880             'Country code'
NPFX            88017           'Number prefix (unique prefix)'
TENR            [illegible]     'Terminating Number (digits after country code)'
CDIROTE                         'Call direction (Incoming/Outbound/Local+TErminating/TRansit or STP)'
OPCO            100             'Originating operator (carrier)'
DPCO            3051            'Destination operator (carrier)'
64KBIT          0               '1=64Kbit 0=not 64kbit'
OPC             12A5            'OPC'
DPC             2473            'DPC'
SYS             00              'System'
TS              0C              'Timeslot'
BNR             [illegible]     'B number'
ANR             [illegible]     'A number'
FATE            C046            'Fate'
CAUSE           16              'Cause Value'
Call clearing
HT              8               'Holdtime'
VUT             8               'Waittime, no answer'
ACT             0A              'Calling party category'
TREQ            00              'Transmission medium requirement'
BNRI            0410            'ISDN B number indicator (NatOfAdr, II, NrPlan)'
ANRI            0413            'ISDN A number indicator (NatOfAdr, II, NrPlan, Pres, Scre)'
VALID_ANR1                      'Wether the A-Number is valid or not'
NULL                            'UQoS null termination'
```

Figure 2.1: An ISUP CDR example

The most interesting parameters in Figure 2.1 are the Date, Called Party Number (CdPN), Calling Party Number (CPN), Cause and Hold Time (HT) parameters. These make the date in which the call was placed, who called who, how it was terminated and how long the call lasted. The Date parameter signifies the date in which the call was initiated and the Date format is yyyy-mm-dd hh-mm-ss. CdPN is the complete CdPN and CPN is the complete CPN. Cause is how the call was terminated; in the current example cause value is 16, which is normal clearing. Hold Time, this is how long the resources on the system were allocated (in seconds).

2.2 Correlation

Correlation is an important concept to understand in this thesis. There are many different types of correlation possible in this area of telecommunication. The easiest one to understand is message correlation, and after that comes xDR correlation. xDR correlation is difficult because it can be on both protocols and services. Finally, the highest level of correlation is end-to-end correlation.

2.2.1 Concept of Correlation

The concept of correlation is rooted in statistical mathematics. The basic concept is to use statistical probably to say whether there is a connection between two or more units. If the probability is above a certain threshold then we can say that they belong together can connect them together.

2.2.2 Data Topography

The topography of the gathered data can help in many cases of correlation, either by speeding up the correlation process or by previously uncorrelatable cases. This is due to the addition information that a correlator will know about the system. Unfortunately, in many cases topography is not known. Proposed solution in this thesis will assume that topography is not know, this is done for two reasons. The first reason is because the solution is to be constructed in such a way that it will be generic and user configurable, and that make it difficult to map this solution to any one network topography. Additionally making it configurable for network topography is such a large task that it is a master thesis in itself and not part of this thesis.

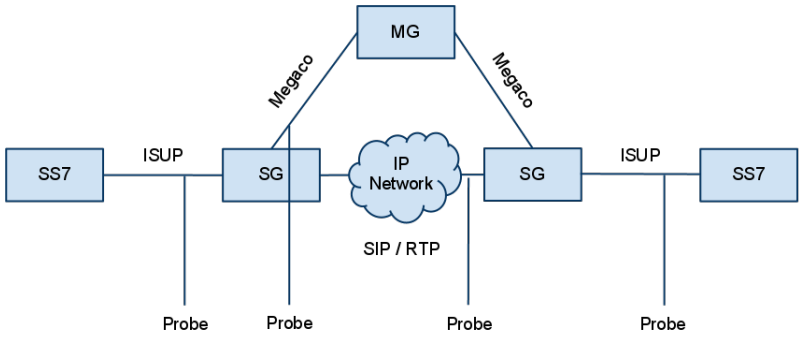


Figure 2.2: A simple telecommunication network.

In the case presented in Figure 2.2 we can see that knowing the topography is not essential for this example. However, by knowing the topography we can reduce and simplify the correlation process. Since we will know which way the traffic will pass and as such any ISUP xDR should create a corresponding Megaco xDR.

2.2.3 Message Correlation

Message correlation is the act of trying to correlate the different signals from taken from one monitoring probe. They are usually correlated based on calls, so that messages belonging to the same calls are correlated together. From this process it is possible to create xDRs. Message correlation is usually an easier form of correlation then xDR in that there is less ambiguity because correlation is done always using the same protocol and on data gathered from the same section of a network.

A message correlator collects the received messages captured by a monitoring probe as input and correlates them by a given category. There are various types of xDRs and the type of correlation performed decides the type of xDR that will be generated.

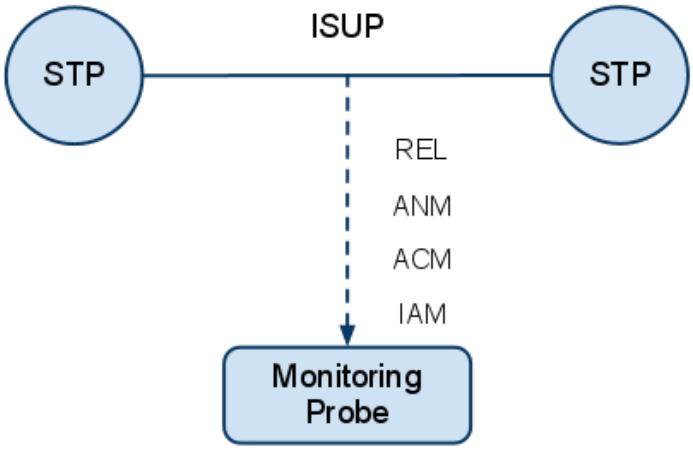


Figure 2.3: ISUP message correltion scenario.

Figure 2.3 shows a simple ISUP message correaltion scenario. From the figure it looks like we are bound to one protocol. However, since the network link can carry more then just the ISUP protocol it is important that signaling messages are filtered by protocol before message correlation is performed.

2.2.4 xDR Correlation

xDR correlation is similar to message correlation, but it is on a higher level of abstraction. Instead of correlating individual messages from one probe into one single unit, the idea here is to correlate the individual units from multiple probes into another single unit.

xDR correlation can easily be many magnitudes more complex than normal message correlation. The reason for it being more complex is because there can be more than one protocol to correlate together. Even with the same protocol there may be information missing in order to correlate xDRs properly. An example of

this is the monitoring of a call that is using special numbers e.g. number portability. Another example is correlating xDRs that contain different protocols.

1	DEV NAME MGW SIDE	Media Gateway name		PORT TYPE	ROUTE	TYPE	point code TSS owr	Point code AXE10 de	CIC's	SNT Port
2	/1	ip_		C7/IUP	PSTN		2-	2-	1	128
3	/2	ip_		C7/IUP	PSTN		2-	2-	2	128
4	/3	ip_		C7/IUP	PSTN		2-	2-	3	128
5	/4	ip_		C7/IUP	PSTN		2-	2-	4	128
6	/5	ip_		C7/IUP	PSTN		2-	2-	5	128
7	/6	ip_		C7/IUP	PSTN		2-	2-	6	128
8	/7	ip_		C7/IUP	PSTN		2-	2-	7	128
9	/8	ip_		C7/IUP	PSTN		2-	2-	8	128
10	/9	ip_		C7/IUP	PSTN		2-	2-	9	128
11	/10	ip_		C7/IUP	PSTN		2-	2-	10	128
12	/11	ip_		C7/IUP	PSTN		2-	2-	11	128
13	/12	ip_		C7/IUP	PSTN		2-	2-	12	128
14	/13	ip_		C7/IUP	PSTN		2-	2-	13	128
15	/14	ip_		C7/IUP	PSTN		2-	2-	14	128
16	/15	ip_		C7/IUP	PSTN		2-	2-	15	128
17				C7/IUP	PSTN	SL0	2-	2-		128
18	/17	ip_		C7/IUP	PSTN		2-	2-	17	128
19	/18	ip_		C7/IUP	PSTN		2-	2-	18	128
20	/19	ip_		C7/IUP	PSTN		2-	2-	19	128

Figure 2.4: ISUP - Megaco gateway mapping

As Figure 2.4 shows the information inside a gateway or a device serving a similar purpose might not always be exposed to the probes, and without this information it is impossible to accurately correlate the two xDRs together even though they belong to the same call. Without the information about the mapping between Circuit Identification Code (CIC), Originating Point Code (OPC), Destination Point Code (DPC), IP and Port Number which is not present in the signaling information it would be impossible to correlate this particular case. In this case this information is only known by the switch and the ones who configured the switch or network.

2.3 Programming Architecture

We made some high level architectural software design topics and had a look on popular types on software design approaches such as modular programming, component based programming, and service oriented architecture.

Modular programming is a programming approach applied on design of a system by putting related functions together inside a module such that each module

has its own interface and implementation. This approach might be applicable for big scaled and multi functional systems, but the system that is implemented for this project, is aimed to be compact and limited in functionality. As such modular programming is not practical to apply for the design of our protocol correlator.

Component Based Programming (CBP) is an application development model which can be defined as one step higher level than object oriented programming. While classes are built on objects in object oriented programming; component based/oriented programming is built on the use of interfaces. In CBP, instead of using objects, there are standard interfaces for configuration and execution. Each component has special interfaces that specify semantics and roles. The main advantage of component is reusable and replaceable [8, Ch. 17].

Service Oriented Architecture (SOA) is a relatively new multi-layered approach for software business whose components serve functions to services so it allows scalability and re-usability. This makes components more independent. In this type of architecture, services are supposed to be platform independent, and incompatibility has to be prevented to have success on interoperability. This approach fits for big scaled business software models [8, Ch. 16].

In this project, the traditional Object Oriented Programming (OOP) approach is used and many items are defined as objects; various reasons made us to make this decision. First of all, we already have enough knowledge on OOP. It also offers relatively fast implementation in a limited time. OOP makes it easy to write easy-to-understand code, so it is a bonus when more people work on the same code. Thus, if somebody wants to make extra coding on the protocol correlator it is easy to start after a fast adaptation on the object oriented code [5, Ch. 2].

2.3.1 Performance

When it comes to design a system, performance becomes one of the key fields which should be focused on. Performance is the term used when measuring how well a given system can handle various degrees of workload. This workload can be the process time of input data, scalability, reliability and resource usage. It is

practically not possible to create a perfect system at all points of performance. It is critical on decide on priorities, that is which aspects of performance are critical for the system.

Like all systems, a protocol correlator such as the one defined in this project has some prior performance criteria. First of all, such a correlator system needs good correlation performance especially if it is expected to work on real time data. In a large communication network it is possible to collect thousands of xDR every single second if there are enough probes and or traffic; thus, a correlator system needs to have a fast working algorithm to process and correlate input data to keep the system stable. A slow algorithm can cause various problems such as program termination, bottlenecks, instability and unreliable results. Another important performance criteria is memory usage. Working with a huge bunch of data needs a cleanup process. Otherwise, sooner or later, the system will encounter memory issues, regardless of how much memory the system has.

2.4 OSI Model

The OSI model is an International Organization for Standardization (ISO) standard and a 7 layer model whose propose is to provide a standard which all protocols should try and adhere to [9].

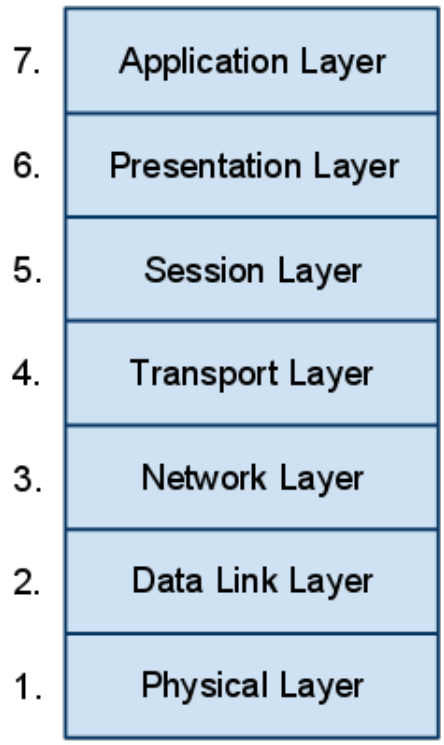


Figure 2.5: The OSI Model stack from seven to one in ascending order

The protocols specifically mentioned in Section 1.3 (SIP, Megaco and CAP) are all application layer protocols. However, it is interesting to look at the protocols they are carried on as well. ISUP and CAP for example can be carried over either Message Transfer Part 3 (MTP3) or IP; it just needs a transport layer protocol that works as a bridge between the network layer and the application layer. SIP can be transported using Transmission Control Protocol (TCP), User Datagram Protocol (UDP) or Stream Control Transmission Protocol (SCTP).

2.4.1 Application Layer

The application layer is the highest laye. CAP, Megaco and SIP are all application layer protocols. ISUP can be argued to belong here as well, however this protocol breaks the Application layer a little by having network related information such as CIC.

2.4.2 Transport Layer

Transport layer is the fourth level of the OSI model. On this layer we will find protocols such as TCP/UDP which are primarily used by SIP and Megaco in this thesis. Other protocols include SCTP which can be used by SIP, Megaco and other application layer protocols in the SS7 suite provided it is SS7 over SIGTRAN (including CAP and ISUP).

2.4.3 Network Layer

Network layer is the third level of the OSI model. The primary protocol in this layer is IP, which can be used to carry all the application layer protocols in this thesis, provided that the correct transport layer protocols and routing soft-switches are present. On the Public Switched Telephone Network (PSTN) side of the core network, MTP3 is the most common protocol to handle the network layer.

2.5 Protocols

The four major protocols that will be covered in this thesis are ISUP, SIP, Megaco and CAP. They are all pillars of their respective branch of any telecommunication network, and are all widely used in the world today both inside and outside the telecommunication world.

2.5.1 ISUP

ISUP is and application layer protocol designed by International Telecommunication Union (ITU) in 1984. It is part of the SS7 protocol suite. The protocol is used to set up calls in a PSTN network. The signaling part of ISUP is separated from the data stream. ISDN is the signaling protocol that exists at the end of the network where the end users are, and ISUP is the signaling protocol that is used inside the core telecommunication network proper. ISUP was designed to be carried over the old PSTN network, but today there are solutions that allow it to be carried over IP using another protocol called SCTP which works similar to TCP [6, Ch. 8].

ISUP has a few important parameters to note. OPC / DPC in addition to CIC are the values that are used to uniquely identify one call on one link of an ISUP network within a limited timeframe since CIC is reused [10].

There are two types of nodes that handle most of the network, Signaling Transfer Point (STP) and Service Switching Point (SSP). STP primarily handles routing, but also other things such as packet conversion and firewall functions. STP has their own point code, so messages captured on different sides of an STP will have different OPC - DPC. SSP is primarily processing the data traffic, such as voice or fax.

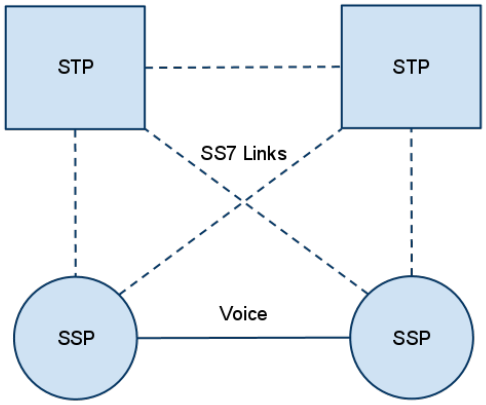


Figure 2.6: Interaction between SSP and STP in an SS7 network

There are a few important message types for ISUP, and they are the following [6, Ch. 8]:

- IAM (Initial Address Message)
  - This message is used to initiate the call; it sets up the timeslot and usually contains the number.
- SAM (Subsequent Address Message)
  - This message is used to send CdPN digits. This message is used if the IAM message does not contain a complete CdPN.
- ACM (Address Complete Message)
  - This is an answer message that is received once the whole number is received on the other end.
- ANM (Answer Message)

  - Once the phone is picked up at the other end this message is sent to the caller.

- REL (Release)
  - This message is used to end a call, as such timeslots will be released and the call ended

- RLC (Release Complete)
  - This message is used to acknowledge that timeslots have been released at the other end as well.

2.5.2 SIP

SIP, works much in the same way as ISUP. It is an application layer protocol that was initially designed by Internet Engineering Task Force (IETF) and accepted as a 3rd. Generation Partnership Project (3GPP) standard in the year 2000 to be used as a protocol to set up media transmission over IP. It has since been adopted by ITU as an ITU standard as well, and there are currently many extensions to the original SIP design which allow it perform more complex and much needed services which today exist for most telephone users.

SIP has only one important parameter to watch in order to uniquely identify one specific SIP call. Call-id is the parameter responsible for identifying a SIP call [13, Sec. 8.1].

There are a few important message types for SIP, and they are the following [13]:

- Invite
  - This message is used to initiate the session or call, it is the SIP equivalent of IAM.

- 180 Ringing

  - This message is used to signal that the phone is ringing on the other side.

- 200 OK
  - This is a generic response message; it is used to as a response to signal that the recipient is ready to receive data.

- Bye
  - This message is used to terminate a session.

2.5.3 Megaco (H.248)

Megaco is a signaling protocol that provides communication between Gateway and Gateway controller. It is the newest call control protocol which addresses the MG and Media Gateway Controller (MGC). Media Gateway converts the voice IP packets received from circuit switching networks. Media Gateway Controller provides good management of this traffic. In other words it works like a Call Agent or Softswitch. Megaco provides distant management of VoIP Gateway, Digital Subscriber Line Access Multiplexer (DSLAM) and Multi-Protocol Label Switching (MPLS) routers. It is similar to Media Gateway Control Protocol (MGCP), but it also unifies networks such as Asynchronous Transfer Mode (ATM).

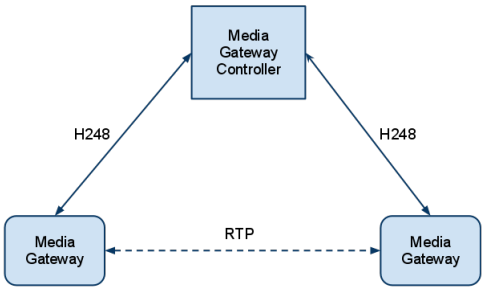
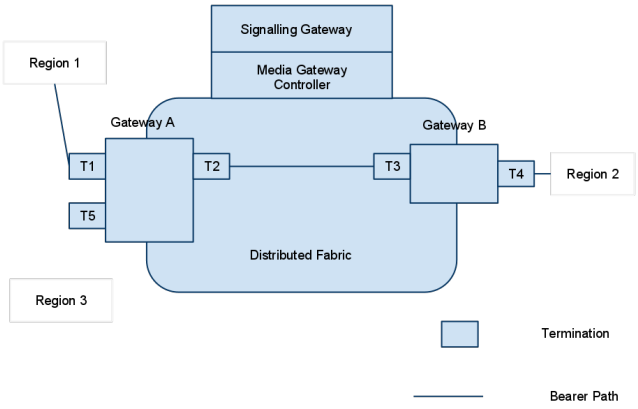
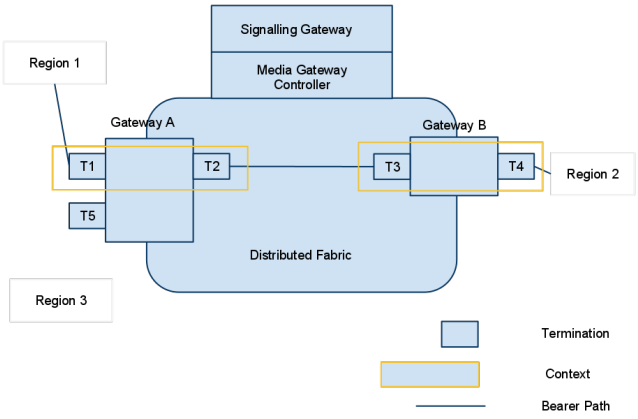


Figure 2.7: Basic representation of Megaco MGC communication

There are two essential components in Megaco: termination and context. Terminations are the places where media streams enter and/or exit media gateways. Contexts describe the bindings between terminations. A context is created when the first termination is added and is deleted when the last termination is subtracted [17, Ch.10].



(a) Megaco connection models : Termination



(b) Megaco connection models : Contexts

Figure 2.8: Two Megaco network images illustrating context and termination.

Megaco Commands: There are totally eight commands in Megaco. Notify can be called by just MG and ServiceChance can be issued by both MGC and MG. All the remaining commands can be called just by MGC, the Add, Modify, Subtract, and Notify commands are the workhorses; for many call flows, this set of four commands is sufficient [17, Ch.10].

Command	Description
Add:	Adds a termination to a context.
Modify:	Issued by MGC when it wants to modify the properties, events or signals of a termination.
Subtract:	Removes a termination from its current context.
Move:	Moves a termination to a different context.
AuditValue:	Returns the current state of termination. Called by MGC when it requires information about properties, events, signals or statistics about termination.
AuditCapabilities:	Called by MGC when it wants to check which termination properties are supported by an MG.
Notify:	Issued by the MG when it needs to inform the MGC that event has occurred.
ServiceChange	Can be issued by MG or MGC to take termination out of service or return termination to service.

Table 2.1: Megaco commands

Setting up the call

Lets assume that the user-1 calls user-2 with ISUP via an SS7 network. When Answer Message (ANM) message is received by signaling gateway, it sends a message to MGC about the call setup from the SS7 network.

MGC sends two Add commands. The First one tells MG-A to create a context. The second one tells MG-A to place an Real-time Transport Protocol (RTP) termination in that context and set its Mode as ReceiveOnly. MG-A replies and the Reply message contains the address and port number that it selects. Then MGC

contacts MG-B with two Add commands. First Add orders MG to create an ISDN line and an RTP termination and also the context to keep these terminations. The second Add is populated with the IP address and port number assigned by MG-A. This is the reason of sending Add in SendReceive mode for the RTP termination. MG-B replies with a Reply command and it includes the IP address and port number of the RTP termination that is created. Since then, MGC is ready to send a Modify command to MG-A, which changes the mode to SendReceive.

Tearing down the call

When either of the parties ends the conversation, the related MG (here MG-A) receives two Subtract commands to remove both terminations from the related context. The second Subtract command removes the last termination in the context, therefore also the context is removed with the last subtract. MGC also sends the same Subtract commands for MG-B.

2.5.4 CAP

Its purpose is to provide services to mobile phone subscribers. CAP is also an extension to MAP; the services provided by MAP are the basic mobile phone services such as Short Message Service (SMS) while CAP provides services such as special numbers. The CAP protocol is defined using the Abstract Syntax Notation No.1 (ASN.1) language, which gives it a rigid but flexible structure. This application layer protocol is used in the Intelligent Network architecture. It provides good support for roaming services for subscribers outside the home Public Line Mobile Network (PLMN). It also allows home networks to monitor and control calls made by the subscriber [1].

Figure 2.9 illustrates a scenario of call forwarding using CAP. In this example someone using a land-line phone is calling someone using a mobile device. The CAP call forwading service overwrites the normal GSM routing [16, p. 60].

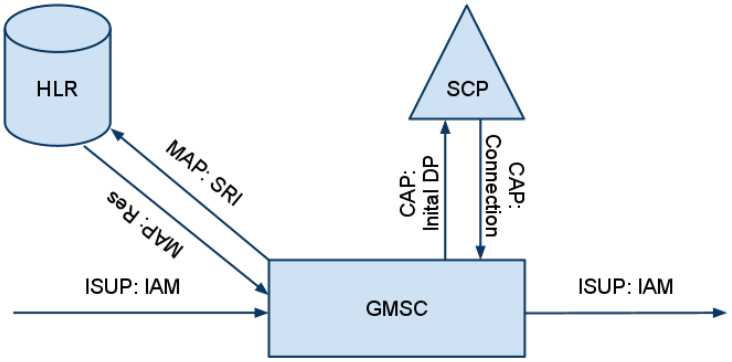


Figure 2.9: Basic network interaction between ISUP, MAP and CAP. [16, p. 60]

CAP has many different messages, and the most prominent are listed in Table 2.2 [16, Ch. 3.5].

Messages	Description
Initial DP:	Used to request information about the called party.
Continue:	Generic message used to continue the call handling, different meaning depending on what message it is in response too
Connect:	Used as a response to Initial DP. It contains information important to connect to the called party
Release Call:	Used to release an ongoing call or one in the process of being established.

Table 2.2: A table of CAP Messages

2.5.5 Similarities

SIP and ISUP are protocols that largely have overlapping roles; ISUP however does incorporate parameters and ideas from the lower OSI layers as well. CAP, MAP and INAP serve a similar role for mobile services to what ISUP does for

the land-line SS7 network, with CAP handling many special cases such as special numbers and pre-paid billing. Megaco is in a unique position because it serves as a bridge between protocols such as SIP and ISUP, and similarities in this situation are CIC, OPC and DPC in ISUP have a correlation with Terminating ID in Megaco, but unfortunately there is no way to correlate directly without an operator held mapping table.

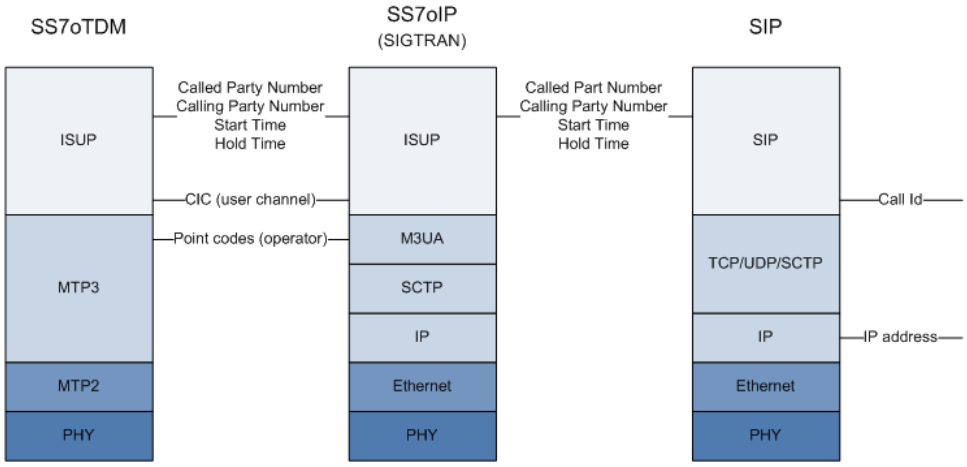


Figure 2.10: Similarities between SIP and ISUP, they have the most similarities since both protocols have overlapping roles.

Figure 2.10 illustrates the protocol stack for SIP and ISUP. ISUP is shown in the figure over both TDM and IP. It is interesting to note that most of the important parameters for calls are shared by both SIP and ISUP regardless of whether of which network layer protocol is used.

Chapter 3

Proposed Solution

This chapter covers our proposed solutions for the two main parts of this thesis. Section 3.1 contains how we classify the different scenarios. Section 3.3 and 3.4 contains how to correlate the scenarios. The second part is section 3.5 and section 3.6 which contains our solution for the design of the correlator and our implementation of it.

3.1 Scenario Classification

Performing correlation correctly for all possible scenarios is close to impossible due to the many different implementations of any given protocol, for example American National Standards Institute (ANSI) ISUP and ITU ISUP. In addition, there are different network structuring which can cause additional complexity. In an attempt to reduce the amount of ever growing complexity we have classified the different scenarios. The first scenario is the most basic case of correlation within one specific protocol. This thesis will for the most part focus on the basic scenarios, with a few exceptions mentioned in sub-section 3.1.2.

3.1.1 Basic Scenario

The basic scenarios are the easiest to correlate scenarios for each protocol. They can be extended into more complex scenarios, but most of the complex problem scenarios are not covered in this thesis as the thesis defintion only requires a few to be covered. Some of the more complicated problem scenarios are listed in subsection 3.1.2.

Protocol	Protocol
ISUP	ISUP
SIP	SIP
Megaco	Megaco
CAP	CAP
ISUP	SIP
ISUP	CAP
SIP	Megaco
ISUP	Megaco

Table 3.1: A table of the basic correlation scenarios

3.1.2 Problem Scenarios

The problem areas in xDR correlation is too many to be completely covered in this chapter. A small subset of all the complex scenarios are presented in Table 3.2 and will be covered in subsection 3.4.

Number Portability
Special Numbers
National-International Numbers
Error Case
Correlation on more than two protocols

Table 3.2: A table of the problematic correlation scenarios

3.2 Protocol Correlation

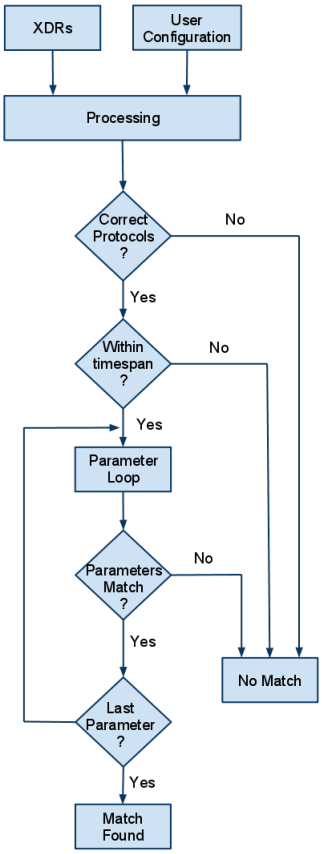


Figure 3.1: Generic Correaltion Flowchart

For all the correlation scenarios presented in section 3.3 we have a generic flowchart in Figure 3.1 that illustrates how the correlation process is to be done in the most generic way possible. The correlator is sent two xDRs and the end-users configuration. The first test is to see if they are of the protocol type that the user has previously selected. The next test determines if the xDRs are within the time span, which is previously defined by the user. The final section is a loop where one parameter pair, previously defined by the user, is selected and compared until all the

pairs have been processed. If they are all a match, then the xDRs can be correlated together.

3.3 Correlation of Basic Scenarios

3.3.1 ISUP - ISUP Correlation

[1] This section is about our thoughts on how to correlate between multiple ISUP CDRs. We've found out that correlating between ISUP CDRs can range from very easy to very hard depending on the complexity of the scenarios. The problem scenarios mentioned in subsection 3.1.2 are some of the special cases that can make ISUP correlation problematic.

CDR		'UQoS cdr message'
MI	00240033	'UQoS CDR message indicator
DATE	20091021 015915	'UQoS message DTG'
TRKGRP	BGD_MIR_MBAZAR	'Trunk Group'
DID	98637	'Destination Identifier'
CSYM	BD	'Country symbol (short country name)'
CC	880	'Country code'
NPFX	88017	'Number prefix (unique prefix)'
TENR	[illegible]	'Terminating Number (digits after cc)
CDIROTE		'Call direction
(Incoming/Outbound/Local+Terminating/Transit or STP)'		
OPCO	100	'Originating operator (carrier)'
DPCO	3051	'Destination operator (carrier)'
64KBIT	0	'1=64Kbit 0=not 64kbit'
OPC	12A5	'OPC'
DPC	2473	'DPC'
SYS	00	'System'
TS	0C	'Timeslot'
BNR	[illegible]	'B number'
ANR	[illegible]	'A number'
FATE	C046	'Fate'
CAUSE	16	'Cause value'
HT	8	'Holdtime'
VUT	8	'Waittime, no answer'
ACT	0A	'Calling party category'
TREQ	00	'Transmission medium requirement'
BNRI	0410	'ISDN B number indicator (NatOfAdr, II,
NrPlan)'		
ANRI	0413	'ISDN A number indicator (NatOfAdr, II,
NrPlan, Pres, Scre)'		
VALID_ANR1		'Whether the A-Number is valid or not'
NULL		'UQoS null termination'

(a) ISUP CDR 1

CDR		'UQoS cdr message'
MI	00240033	'UQoS CDR message indicator
DATE	20091021 015915	'UQoS message DTG'
TRKGRP	BGD_MIR_MBAZAR	'Trunk Group'
DID	98637	'Destination Identifier'
CSYM	BD	'Country symbol (short country name)'
CC	880	'Country code'
NPFX	88017	'Number prefix (unique prefix)'
TENR	[illegible]	'Terminating Number (digits after cc)
CDIROTE		'Call direction
(Incoming/Outbound/Local+Terminating/Transit or STP)'		
OPCO	100	'Originating operator (carrier)'
DPCO	3051	'Destination operator (carrier)'
64KBIT	0	'1=64Kbit 0=not 64kbit'
OPC	2473	'OPC'
DPC	4287	'DPC'
SYS	00	'System'
TS	0A	'Timeslot'
BNR	[illegible]	'B number'
ANR	[illegible]	'A number'
FATE	C046	'Fate'
CAUSE	16	'Cause value'
HT	8	'Holdtime'
VUT	8	'Waittime, no answer'
ACT	0A	'Calling party category'
TREQ	00	'Transmission medium requirement'
BNRI	0410	'ISDN B number indicator (NatOfAdr, II,
NrPlan)'		
ANRI	0413	'ISDN A number indicator (NatOfAdr, II,
NrPlan, Pres, Scre)'		
VALID_ANR1		'Whether the A-Number is valid or not'
NULL		'UQoS null termination'

(b) ISUP CDR 2

Figure 3.2: Visual representation of two ISUP CDRs which will be correlated together.

[1] The ideas on ISUP - ISUP correlation brought forth in this subsection are based on discussions with engineers at Utel Systems, and as such there are no references.

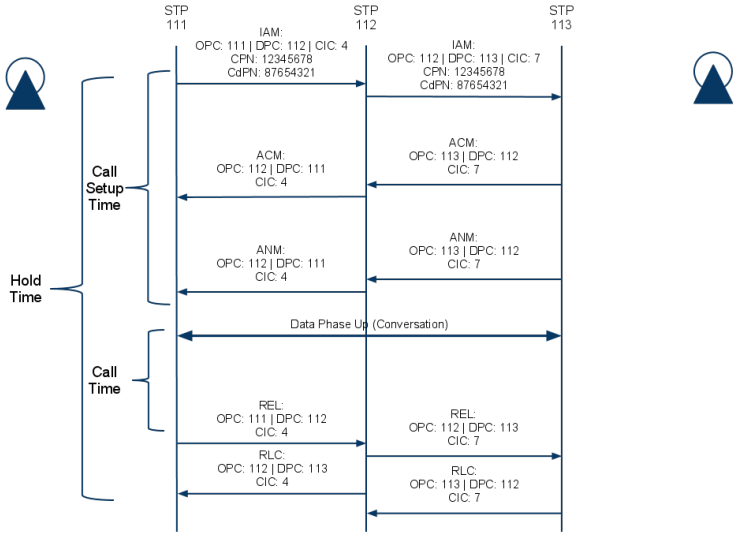


Figure 3.3: Basic ISUP call flow scenario.

From Figure 3.3 we can see that CdPNs, CPN, Date/Time, Call Time and Hold Time is correlate ISUP with ISUP correctly for the basic scenarios. The different parameters need to match, but some slack (from a few milliseconds to a few seconds) can be given for Date/Time, Call Time and Hold Time. The reason for this is because of processing time inside the switches and monitoring probe synchronization can cause the CDRs to have a slight difference in the timestamp.

There are pitfalls that make correlation between ISUP CDRs difficult if not impossible in some cases. One case where you monitor the traffic to and from the gateway in and out of a country, you will typically get CDRs with the international prefix on one side and without the international prefix on the other.

In that case it is important to also check the *National Indicator* parameter in ISUP, and if the value indicates international number then check the numbering plan to find out the country code. Depending on the country code the first one, two or three digits need to be removed before correlation can begin. Another case is the one previously mentioned where the use of special numbers is involved and in order to accurately perform correlation such CDRs it is imperative that additional

xDRs with the number switching is included.

3.3.2 SIP - SIP Correlation

[2] Correlating SIP - SIP CDRs are usually a straight forward business. In most cases we can limit the numbers by time, and just check to see if they have the same Call-Id parameter. However in order to make the correlation process more accurate we suggest additional parameters to be used in the correlation process. Correlating on other important parameters such as Hold Time, IP Address, Port Number, CdPN and CPN in addition to the most important one, Call-Id is what we suggest. These parameters must be used if SIP is going to be correlated with any other protocol, such as ISUP.

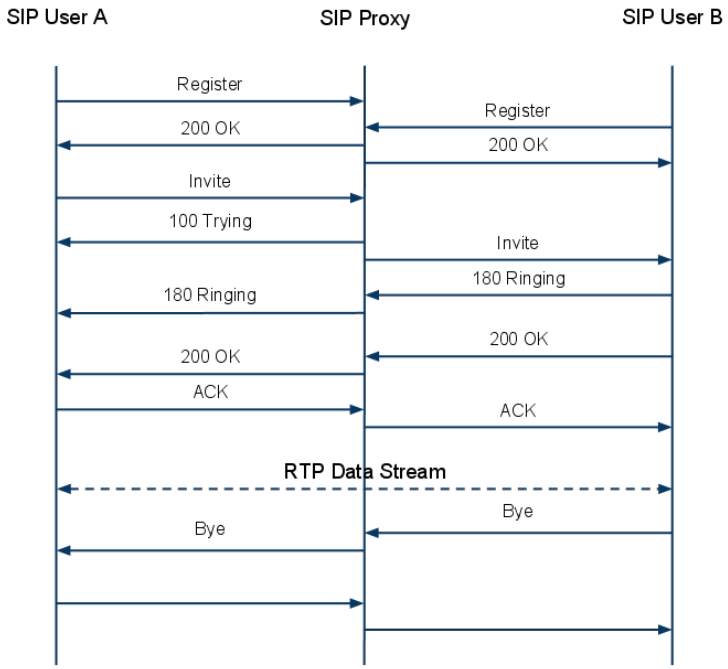


Figure 3.4: SIP - SIP call flow.

[2]The ideas on SIP - SIP correlation brought forth in this subsection are based on discussions with engineers at Utel Systems, and as such there are no references.

3.3.3 Megaco - Megaco Correlation

[3] In Megaco - Megaco correlation there are a few important parameters in the protocol that needs to be paired with each other in order to perform a correct correlation.

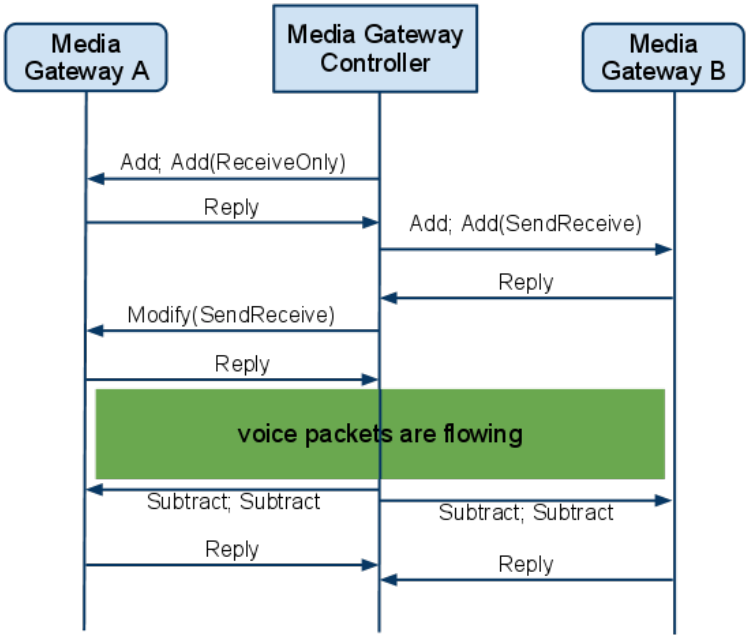


Figure 3.5: A simple example Megaco - Megaco call flow.

In a more detailed and realistic network, there might be multiple MGCs and even more MGs. In addition there are many other protocols, ISUP, and SIP that comes onto the picture as well, because Megaco is a bridge protocol.

3.3.4 CAP - CAP Correlation

CAP - CAP can be Correlated on a parameter called Global Call Reference (GCR) which is created at the Mobile Switching Service (MSC) where the call is initiated

[3]The ideas on Megaco - Megaco correlation brought forth in this subsection are based on discussions with engineers at Utel Systems, and as such there are no references.

and is carried over to other sections of the network if the call is routed there [16, Ch. 7.5]. So the way to correlate CAP - CAP is to just match the GCR number.

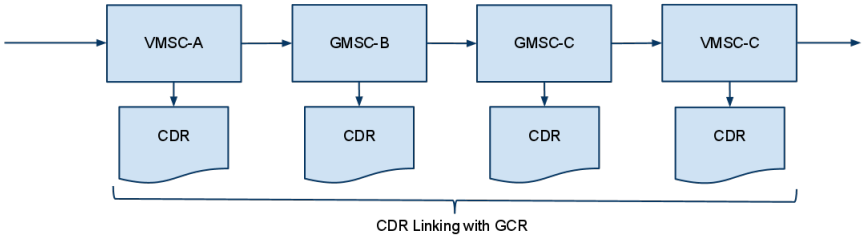


Figure 3.6: CAP - CAP Correlation with GCR [16, Ch. 7.5]

3.3.5 ISUP - SIP Correlation

[4] In order to correlate xDRs from the two protocols SIP and ISUP, some assumptions need to be created. The first assumption is that the SIP xDR contains the actual CdPN and not just an alias. The second assumption is that the monitoring is done on both sides of a gateway that separates the two protocols. Assuming those assumptions are correct we can proceed with the correlation. The parameter to correlate on is the timestamp (call time and hold time), if they are too far apart then it would be wrong to try and connect the xDRs together. CPN in ISUP should be connected to the From field in SIP, assuming that the CPN is present as it is an optional parameter in ISUP. The CdPN will have to be present on both sides, in ISUP as CdPN and in SIP as the To field. The parameter that indicates that this call is an international call should always be checked and international number prefixes needs to be compensated for [7, Ch 8.2.1.1].

The first scenario we will look at is a standard SIP - ISUP scenario where the international part of the number is removed on the ISUP part. This makes the CdPN slightly different than in the SIP part. The solution to this problem is to take the international indicator into account when doing the correlation. This does create some additional problems that can be solved a little differently. The first problem is

[4]The ideas on ISUP - SIP correlation brought forth in this subsection are based on discussions with engineers at Utel Systems and one particular reference.

the fact that we do not necessarily know which number is the international number; it can even be more than one number. So to solve this we need to either just compare and say that we allow some slack. This makes it a little uncertain and prone to failure, but statistically the chance is very low. The other solution would be to look up in a database and see what the number would be with and without a international number. This adds additional intelligence, and makes the correlation slow because it needs to check a database each and every time. With that problem solved, all we need to do is to compare the ISUP CdPN with the SIP CdPN and do the same for CPN if present. It will also be useful to make sure that the timestamps are not too different. They need to be within at least x seconds of each other, with x being a user specified parameter, a good default number is one. That should be all we need to correlate SIP and ISUP calls, the results of our test can be found in the result chapter.

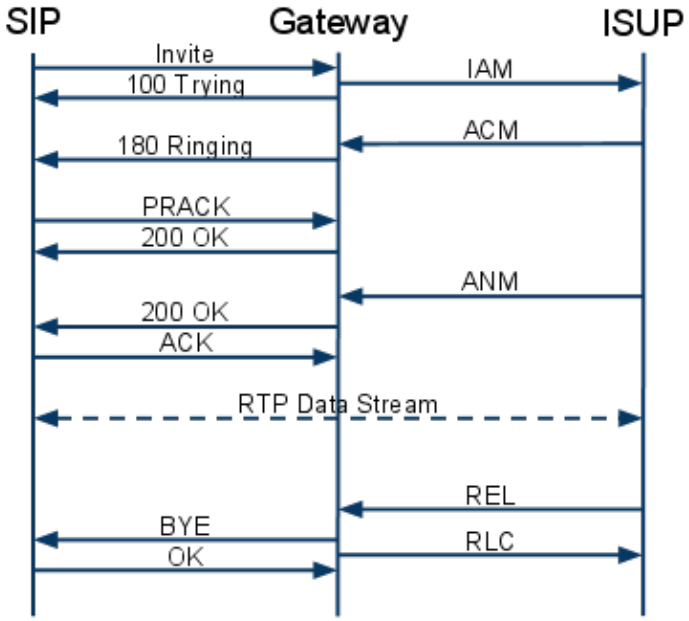


Figure 3.7: SIP - ISUP call flow. Note that Megaco or MGCP is left out in order to simplify it.

In this case we have a call coming from the SIP side and going to someone/-something on the SS7 side. The To field in the Invite message should be linked to the CdPN in the Initial Address Message (IAM). Other parameters inside the IAM that should be connected to the SIP invite are **International Number Prefix** Parameter. Address Complete Message (ACM) is connected to the Ringing message in SIP but since there are not any special parameters in either to correlate on it can be ignored. PRACK i.e. Provisional Acknowledge and the response can safely be ignored as well. ANM message is connected to the generic SIP OK message; there arent any specific parameters which are required to correlate on here. Once the acknowledgement is received the data phase is up, RTP on the SIP side and a-law or u-law raw data on the ISUP side. The ISUP side hangs up the phone first and the Release Message (REL) is connected to the BYE message on the SIP side. The final parameters needed for correlation are the Call Time/Conversation Time and the Hold Time; they should be roughly the same on both sides. Within 5% of each other is a good number. Slight time discrepancies are usually caused by processing delay in switches.

3.3.6 ISUP - CAP Correlation

[5] Correlating between ISUP and CAP is very similar to correlating between ISUP and ISUP. In both cases the correlation process is done on matching CPN to CPN, and CdPN to CdPN. Unfortunately, call time is not a parameter which we can use to increase the accuracy of the correlation process because it will not or even approximately the same in both xDRs.

[5]The ideas on ISUP - CAP correlation brought forth in this subsection are based on discussions with engineers at Utel Systems, and as such there are no references.

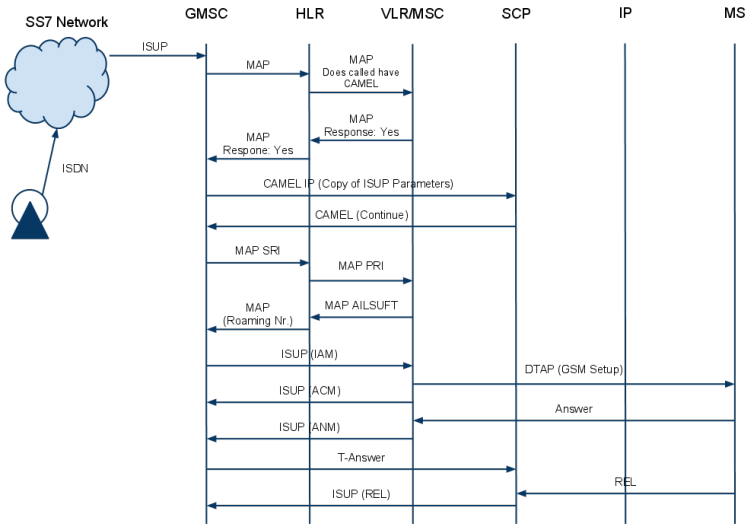


Figure 3.8: Call flow of setting up a call ISDN subscriber to a mobile subscriber where CAP is supported.

The scenario illustrated in Figure 3.8 is a basic call flow where an ISDN subscriber calls a mobile subscriber of CAP where other protocols are included.

3.3.7 SIP - Megaco Correlation

[6] Correlating between SIP and Megaco is in most cases possible. The most important parameters to look at when correlating between SIP and Megaco xDRs are SIP RTP stream IP address and port number with Megaco Remote Medium. If the parameters are the same then we can connect them together as long as the time difference between the xDRs is short.

[6]The ideas on SIP - Megaco correlation brought forth in this subsection are based on discussions with engineers at Utel Systems, and as such there are no references.

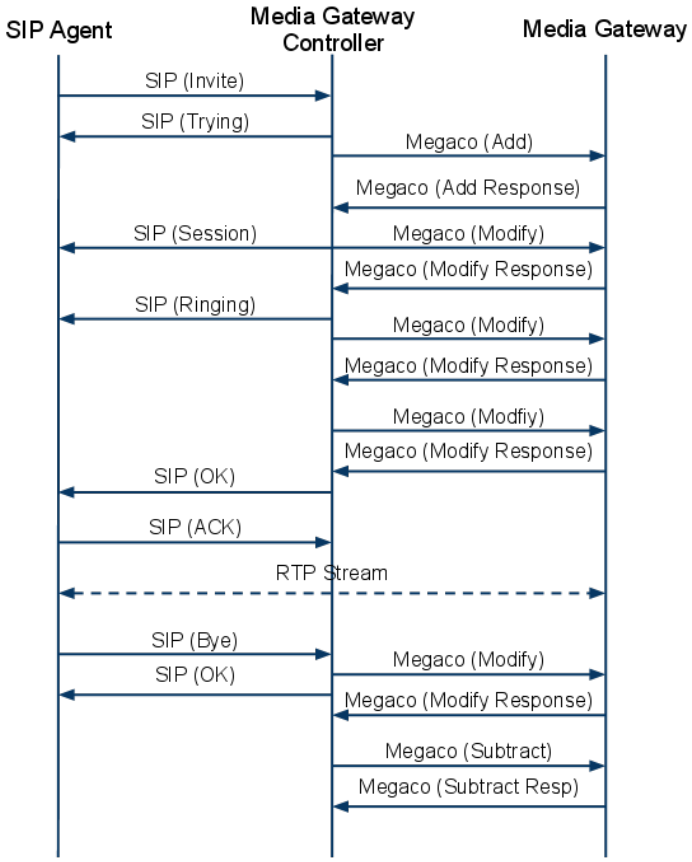


Figure 3.9: SIP - Megaco call flow.

In Figure 3.9, the left side messages are correlated together into a SIP CDR primarily using the call-id parameter inside the SIP messages, and the Megaco messages are correlated together primarily using the transaction id inside the Megaco messages. The two sides should then be correlated together based on the RTP IP address and port number inside the SIP invite message and the corresponding RTP IP address and port number inside the Megaco message.

3.3.8 ISUP - Megaco Correlation

[7] Correlating ISUP and Megaco together is generally a difficult affair as it is impossible to correctly correlate them based on the information present in the signaling information. In order to correlate them, the Transaction Id inside the Megaco protocol needs to match with some information that only the operator of the network has. The Transaction Id in Megaco is supposed to correspond to the CIC code in ISUP, but without knowing the topography it will is impossible unless there is only one E1/T1 interface. The topography of the network is something that the system operator must input into the correlator in order to achieve successful correlation.

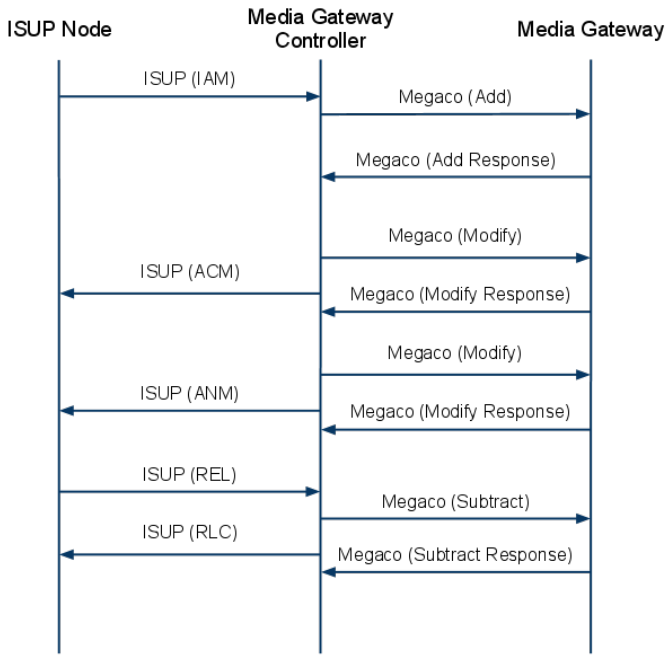


Figure 3.10: ISUP and Megaco call flow.

[7] The ideas on ISUP - Megaco correlation brought forth in this subsection are based on discussions with engineers at Utel Systems, and as such there are no references.

3.4 Correlation of Specific Problem Scenarios

3.4.1 Number Portability

Number portability is an intelligent network service that is particularly difficult because of the different ways to implement that type of functionality. According to Lee Dryburgh and Jeff Hewett [6, Ch. 8] there are four main ways to implement number portability in ISUP. Because of the different ways in which they function, its not possible to create on correlation method that can correctly correlate them all. There is a similarity between some of these methods, specifically the first one and the problem shown in subsubsection 3.4.2 special numbers.

In the first method it is necessary to monitor the transmission between the centrally administered database and the node sending the query. With the information in that transmission it should be possible to correctly correlate the different xDRs.

The second method is similar to the first method in how to correlate, with the exception that the database is switched out with a query to the network that owns the number the CdPN. In this method the network in which the call initiated first tries to route the CdPN number and will receive a error message back, at which point it will query the CdPNs network for the new routing number.

The third method is almost the same as the second, but instead of querying the CdPNs network for the new number, the new number is included in the error message reply.

The fourth and final method has the CdPNs network just reroute the call from their end. In this case additional correlation is impossible because it does not look like a case of number portability from the network in which the call was initiated.

Correlating method one, two and three requires a two-stage correlation process. First the xDR of the first link one needs to be matched with the xDR from which contains the CdPN conversion. Once process is done, the new xDR can be correlated with the glsacro09 from link two.

3.4.2 Special Numbers

Special numbers can cause some problems when trying to correlate xDRs together. A major part of correlation is trying to connect the xDRs based on the CdPN, however in the case where special numbers are in the picture this becomes a problem.

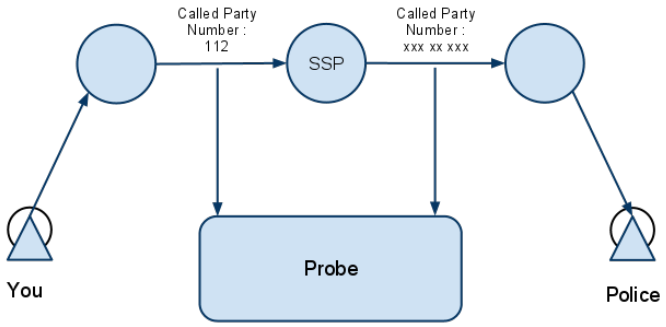


Figure 3.11: A scenario where someone calls a special number.

The scenario illustrated in Figure 3.11 is an example where a telephone subscriber calls a number that is part of the special number group, and in this case it is the police in Norway (112). On one section of an SSP the monitor probe captures this and records the number as 112 in a CDR. The problem arises when the SSP does a number analysis of the special number 112 and converts it the normal eight digit number of the nearest police station. In this example we have two CDRs that belong to each other and should be correlated since they are the same call, but they have two different CdPNs. [8]In this paticular case the only way to perform correct correlation is to know the number conversion. This information can be input into the correlator prior to correlation, or a third xDR containing the information about the number conversion needs to be included in the correlation.

[8]The ideas on correlation brought forth in this paragraph are based on discussions with engineers at Utel Systems. As such, there are no references.

3.4.3 International and National Numbers

International and national numbers can cause some problems when correlating CDR that belong to the same call. An example is where information from the probe is gathered on both sides for a international gateway. That way you will often end up with two CDRs, one with the International number pre-fix and one without.

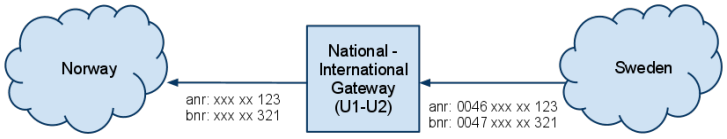


Figure 3.12: A scenario with and without national number prefix.

In the example illustrated in Figure 3.12, two CDRs will be created; they are essentially one call and should be correlated into one call. Because the numbers are not same on both sides, the correlating process is more difficult. Because some countries also use area number pre-fixes the correlation process can be even more difficult, but it should still be possible to perform a number match either based number comparison from the right to left, and if the numbers are close enough for the user specified breakpoint than we can correlated them. [9]The national indicator in ISUP is used to indicate whether the call is national or international, so the complex correlation case is only applicable if the CDRs to correlate have different values in the national indicator parameter.

3.4.4 Error Case

Errors can occur, one example of this is an ISUP system where all the timeslots are taken or an incomplete number; then the reply message will be in the form of an error message. At this point it will be impossible to correlate further because the

[9]The parameters to choose from and how to perform correlation brought forth in this paragraph are based on discussions with engineers at Utel Systems. As such, there are no references.

call ends here.

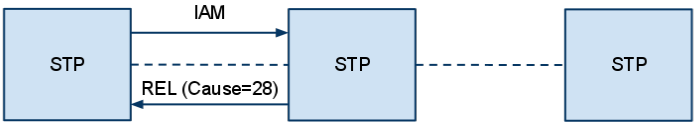


Figure 3.13: A scenario with an error

In the example illustrated in Figure 3.13 one CDR will be created, and by looking at the cause value for termination we can why the call was terminated. If the termination cause was 28 (invalid number format), then we can stop correlation of that xDR because we know that there will not be any more xDRs that will match the current case.

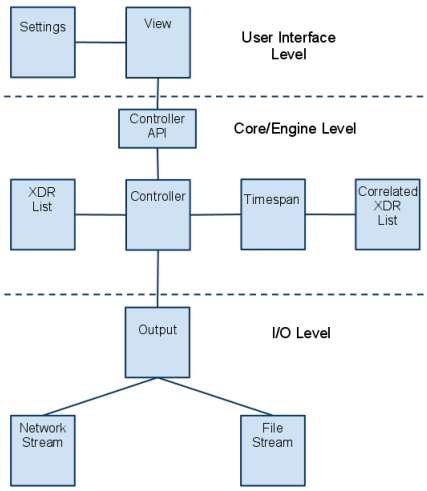
3.4.5 Correlation on more than two protocols

For correlation of more than two protocols, we thought of two possible solutions. The first was to allow correlation of more than two protocols at the same time, but unfortunately this causes the potential correlation problems to be very complex.

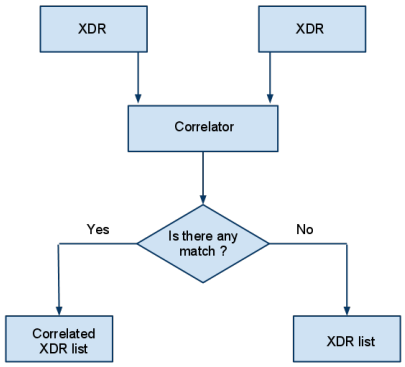
An example of this case is correlation of ISUP, SIP and Megaco. This correlation case requires information about the mapping of ISUP to Megaco to be manually added. It requires the Session Description Protocol (SDP) IP address and port numbers in SIP and Megaco to be correlated, and it requires the CPN and CdPN in SIP and ISUP to be correlated. The complex web of cross protocol parameter dependency is very difficult to program into a fast and generic correlator, as such we have opted for the use of hierarchical correlation because reduces the scenario complexity of correlation. Hierarchical correlation approach on the same example mentioned earlier in the text is done by just performing correlation on two protocols, and then correlate the result with the third protocol. The order of the protocols should not make any difference in the final result. This is further touched on in section 6.3 Further Work as our prototype correlator does not include this possibility.

3.5 Correlator Design

Our design of the correlator is a highlevel diagram illustrated in Figure 3.14(a). The core idea in this design is the seperation of GUI and the Input/Output (IO) section of the program from the core/engine of the program. The main focus will be on the engine design in subsection 3.5.1 as that is the part of the program in which the acutal correaltion process happens.



(a) Our ideas for the separation of the different components of the program.



(b) Highlevel Correlator Flow Chart

Figure 3.14: Highlevel representation of our correlator

The strength of this design is the ease in which it is possible to add and remove different components from the core/engine of the program. This will make it much easier to implement changes to reading and decoding the CDR files. You just need to import another DLL component. The user interface is interchangeable as well with this design. The engine of this program remains pretty much static and the design for this will be covered in subsection 3.5.1. The GUI and IO components are able to talk to the engine using a set of Application Program Interface (API) calls that will be explained in the subsection 3.5.1 Engine Design.

3.5.1 Engine Design

The idea of the engine of the program is to make it as separate from the other components and as generic as possible. It is for this reason that we have decided to distance the engine from the rest of the program using a set of API calls. This will allow the GUI and the IO components to be modified or redesigned without recompiling the whole program. Another bonus of this design is the ease in which it is possible to convert it to a different platform. If the core/engine is written in C, C++ or Java it will be easy to make it cross platform, as long as no operation system specific API calls is embedded into the engine. All that is needed is recompilation on the new platform. In the case of Java this isn't even needed, as it will just be complied to Java bytecode and interprited by the Java virtual machine on the system.

So far the set of API calls needed to operate the engine are as follows:

Listing 3.1: API

```
int createController();
int hookToDataStream(int controllerId, int portNumber);
int readFromFile(int controllerId, char * FilePath);
void setSlidingTimeWindowSize(int controllerId, int seconds);
int getSlidingTimeWindowSize(int controllerId);
void setSlackPercentage(int controllerId, float percent);
float getSlackPercentage(int controllerId);
void setProtocolsToCorrelate(int controllerId, int[] ProtocolSet);
void setProtocolParametersToCorrealte(int controllerId, list<string> leftSideProtocols, list<s
int doCorrelation(int controllerId);
int writeResultToDatabase(int controllerId);
int writeResultToFile(int controllerId char * folderPath, char * fileName);
```

The *createController* API call is the first function to be called. It will create a controller that will handle the whole correlation process. The return value will be the controllerId. The whole point of using this system is to be able to create and perform multiple concurrent correlations on the same or different xDR data sources.

The *hookToDataStream* API call is used if correlation is to be done in real-time.

The program must be told which port to listen to and which correlation controller that should be the one listening. TCP is the transport protocol of choice. It will return -1 in the event of an error or 1 if successful.

The *readFromFile* API call is used if correlation is to be done on xDRs present in a file. It should require an absolute Linux type file path and which controller that should read. This will make it easier to write the code and it will prevent errors like sometimes using absolute file path sometimes using relative, or a mismatch between Windows style file paths or Linux style file paths. This function will return -1 in the event of an error or 1 if successful.

The *setSlidingTimeWindowSize* API call is used to set the sliding time window size. The sliding time window is the parameter that decides how far back the algorithm should look when trying to find xDRs to correlate with. The two parameters it requires are the controllerId and the time window size, integer in seconds.

The *getSlidingTimeWindowSize* API call will return the currently set size of the time window in seconds. It requires that the id of the controller in question is be past to it.

The *setSlackPercentage* API call is used to set the amount of slack allowed when performing correlation on parameters such as time. Having the slack as a percentage rather than a fixed amount is the best choice because it is possible that long calls may have some discrepancy in the call time parameter. If this discrepancy is a few seconds for an hour long call, then that isn't more then:

$$(3sec * 100)/3600sec = 0,083\%$$

We can generally allow a little more slack than that, but if the call lasted only 30 seconds it would look totally different.

$$(3sec * 100)/30sec = 10\%$$

If we had only allowed for fixed numbers then the short conversation, which shouldn't have been correlated would have been correlated.

The *getSlackPercentage* API call will return the Slack Percentage; this value is a number between 0 and 1, where 1 is 100%.

The *setProtocolsToCorrelate* API call is used to fix the protocol that will be correlated on. The correlator will then ignore protocols that are from a different type then the ones specified. Each number represents a protocol,

$$-1 = nothing, 0 = ISUP, 1 = SIP, 2 = Megaco, 3 = CAP.$$

This can be expanded to include many other types of protocols.

The *setProtocolParametersToCorrealte* API call is used to fix the different protocol parameters together. This function takes two string lists as arguments and the controllerId, The parameter names present in the string lists will be compared to the parameters present in the xDRs that are to be compared.

The *doCorrelation* API call requires that the controllerId is pass to it and it returns -1 if correlation is not performed. This is the case if no parameters or protocols are set, there is not data or the controllerId doesn't exist. 1 is returned if correlation is performed successfully.

The *writeResultToDatabase* API call is used to write the correlated data to the database. It must be provided the controllerId and connection string which should contain the parameters necessary in order to connect to the database server. It will return -1 if connection is refused or 1 if successful.

The *writeResultToFile* API call is used to write the correlated data to a file. Filename, controllerId and the absolute file path are the required input parameters for this function.

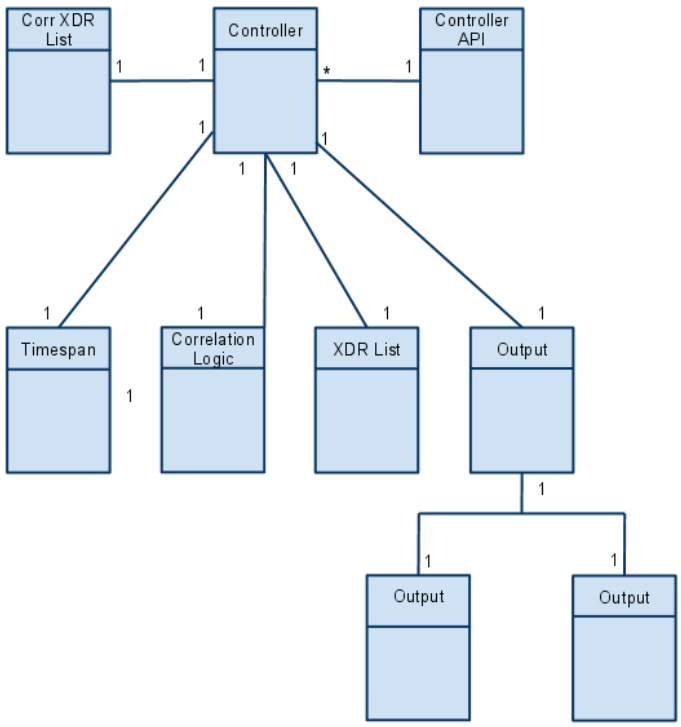


Figure 3.15: Our idea for the engine looks like. Loosely based on the UML 2.0 standard.

By using the design in Figure 3.15 the correlator will become very generic, it will allow the user to choose which parameters that he/she wants to correlate on. Additionally, it will allow the area in which the correlator will look for xDRs to correlate on to be manipulated. Another interesting effect produced by this design is the ability to be able to perform multiple concurrent correlations on the same or different xDR pools. The controllerAPIClass will act as the bridge between the user interface part of the program and the engine itself. Every time the createController function is called a new controller is created. This controller is the class that keeps track of all the unique parameters, xDRs, timers and output for one correlation process.

3.5.2 Sliding Time Window

The sliding time window is a very important part of the correlator. The sliding time window allows the algorithm to only look at a given time span when checking for xDRs to correlate. This is based on the users preferences. The major advantage of using the idea of a sliding time window is that it reduces the search space to a fraction of the whole search space. Another advantage of the sliding time window is that it allows the end of the timeline to continually be extended.

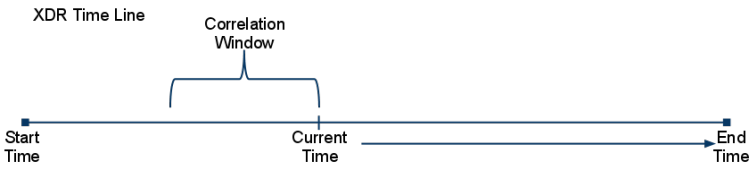


Figure 3.16: The sliding time window.

In Figure 3.16 we have a current time that keeps on increasing and a user configurable time window which extends back x number of seconds and is the time window in which the correlator will look for xDRs to correlate with the current one.

3.6 Correlator Implementation

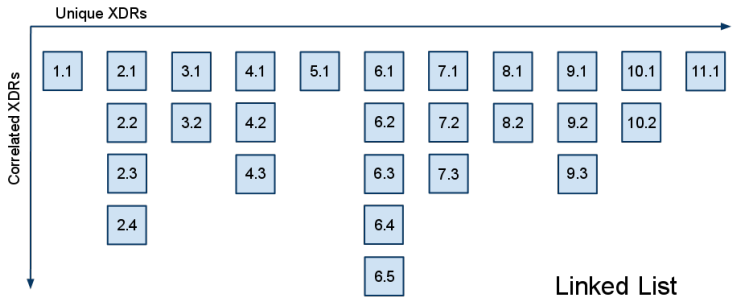


Figure 3.17: The Linked List design.

Figure 3.17 shows how the linked list may look like. Horizontally we can find the different xDRs, and vertically we can find the xDRs that belong to each other based on the parameter that the user specified. 2.X series are one correlated xDR, while 2.1 are one specific xDR in that instance.

Correlator program runs the correlation algorithm on a linked list and inserts the correlated xDRs in this list. Linked lists are flexible, fits for correlation logic and works fine for both real time correlation and offline correlation which uses a file logged from the network and stored.

Correlation algorithm works on a linked list of linked lists (2-dimensional linked list). First dimension is called a horizontal list and the second dimension is called a vertical list. Horizontal list is the main list that keeps the correlated xDRs as single items. We can also call it the correlation stream. Vertical list stores the correlated xDRs in the same list. Non correlated xDRs are stored individually in a vertical list (e.g 1.1, 3.1 and 8.1).

Listing 3.2: Linked List

```
LinkedList<LinkedList<KeyValuePair<DateTime, XDRClass>>> correlatedXDRs;
```

Here *LinkedList* is a two-way linked list which is possible to move back and forth through the list. *XDRClass* is the class that keeps all xDR related values

inside an xDR provided by the logs. All xDRs received by the engine are stored in this format in memory. Before each xDR is processed by the algorithm, all xDRs are encapsulated with *DateTime* as a key value in *KeyValuePair* type. This is the type for objects in the linked list.

Structure of the correlator code consists of various functions which are connected to each other and work together. Figure 3.18 shows the call relations of the functions which have a role for the correlation part of the system.

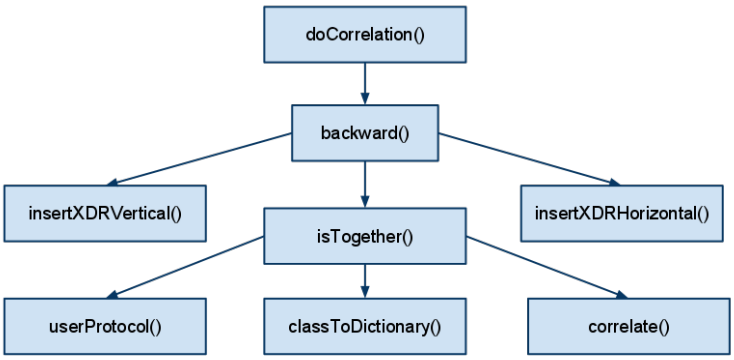


Figure 3.18: The prototype call hierarchy

doCorrelation()

This function will itterate through the linked list and send xDRs to compare with each other by calling *backward()*. The correlation process function is done when the thread exists this function.

backward()

Each xDR received by the system is sent to the algorithm by the *backward()* function. *backward()* function looks for a match in the linked list in a pre-defined number of seconds range backward direction. This function also receives the new xDR packet as input and place it to the appropriate place in the correlation stream

(linked list). It scans the xDRs in X seconds (defaults to 1) range and calls *isCorrelated()* function to check if there is a match between the new xDR with the existing xDRs. It scans each vertical list in the outer loop and scans each xDR in the vertical list in the inner loop. If it finds a match with any of the xDRs in that range, then it calls the *insertXDRVertical()*, and that function inserts the new xDR by using its timestamp. If it fails to correlate the new xDR with any of the xDRs in the current vertical list, then it moves back by one horizontally to scan the next vertical list. If backward fails correlate the new xDR with any of the xDRs in the x seconds range, then it calls *insertXDRHorizontal()* to create a new vertical list branch and insert the new CDR to the brand new vertical list. In that case new xDR becomes the first member of its own group.

insertXDRVertical()

This function is called when *backward()* finds a match in one of the existing correlated groups. That is because it needs the pointer for the vertical list which the new xDR will be inserted in and also the new xDR.

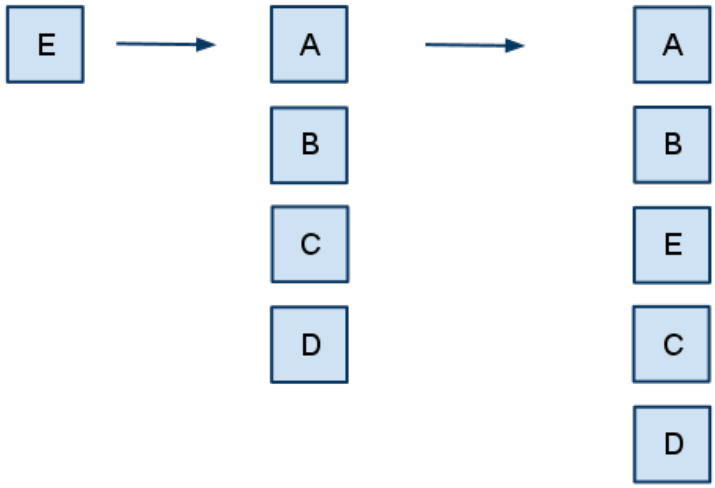


Figure 3.19: Implementation of the Linked List Vertical

Figure 3.19 illustrates a vertical insertion. Here A, B, C and D are correlated xDRs as a group and located under the same vertical list. Algorithm decides that the new xDR E needs to be correlated with that group and is inserted in this vertical list. The list is time ordered, so it is important where the xDR are inserted. During the insertion of a new xDR, the algorithm starts to compare the timestamps of the new xDR and the last xDR in the list. If the new xDRs timestamp is bigger than the other one, new xDR is inserted to the end of the list, otherwise algorithm checks the new xDRs timestamp with the previous one in the list and so on. If the new xDRs timestamp is smaller than all xDRs in the list, then the new xDR is inserted in the top of the list. Thus, the time order is ensured as the xDR that has the smallest time stamp is in the top of the list and the one which has the biggest timestamp is located in the bottom of the list and it is always updated for every new insertion. In the figure above, the new xDR E is inserted between B and C. So the timestamp of E is smaller than timestamp C and bigger than timestamp B.

insertXDRHorizontal()

The way the main algorithm works, all new xDRs needs to be inserted to the main list. *insertXDRHorizontal()* is called when a new xDR does not have any match with any of xDRs in the correlation range. Depending on the incoming xDRs, this xDR might be a xDR which could not be correlated, or the first xDR of a group in which the remaining xDRs has not been received or not been processed by the algorithm yet.

This function basically creates a new xDR branch/vertical list for the new algorithm and inserts the new xDR into it. It scans the horizontal list within the range and inserts the new list to the main list in where it belongs to in regards to its timestamp. There is no global time in real networks so there is no guarantee to receive xDRs sequentially based on time. Additionally, normal network delay to and from monitoring probes can also cause the ordering to be different. So it is important to make the insertion by the timestamp value and keep the list ordered by time.

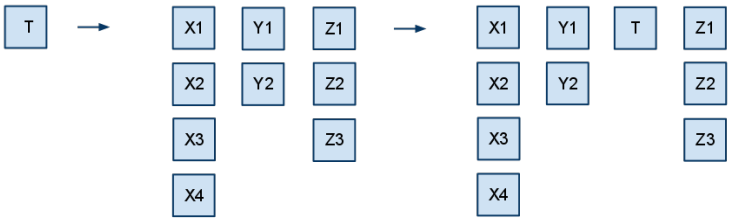


Figure 3.20: Implementation of the Linked List Horizontal

Figure 3.20 illustrates the horizontal insertion of a xDR. Here T is a new xDR which needs to be inserted to the correlated xDR list. As it is mentioned before, when the algorithm could not correlate a new item with any other items in the correlation range, it still needs to be inserted. The reason for this is because it can be the first xDR of a new group (based on the end-users whishes) received by the system. It is important that each new item (in this case, it is T) will be inserted in the correct position based on the timestamp in ascending order from the left. It can be seen from the figure above that new item T is inserted between group Y and group Z. The algorithm starts to check the timestamps from the right-end (timeline is from left to the right) of the list. In this scenario, it seems timestamp of T is smaller than Z3, then it checks the timestamp of Y2 and new item Ts timestamp is bigger than Y2 (the one which has the biggest timestamp for group Y). The algorithm uses the timestamp of the item in the bottom of each vertical list as the timestamp of the vertical list.

isTogether()

The *isTogether()* function takes two parameters and tries to find out if they are connected based on the preferences previously set by the end users, and it returns a true or false value based on whether it managed to correlate them together. The two input parameters are xDRs.

userProtocol()

userProtocol() function sets the protocol values which will be needed for the correlation by reading the protocol information of the xDRs. It also filters the xDRs which will not be correlated by returning false.

classToDictionary()

This function is responsible to set the values that will be needed for correlation gathered from the xDRs which will be attempted to correlate.

correlate()

This is exactly where the two xDRs are decided to be correlated or not. It works dependent on the user selection.

3.7 User Interface

The program is designed around the idea that the engine is separated from the user interface. Because of this design it is entirely possible to create either a console type of user interface or a graphical user interface. Figure 3.21 illustates how the prototype correlator looks like, and Table 3.3 explains the buttons available in the prototype correlator.

File	Allows the user to select file, and the program will attempt to open and decode the information inside the file. The type of file it takes is special file format that Utel System owns.
Options	Allows the user to change the size of the time-span and accuracy of the call time. In addition to this, the user can choose which parameters and protocols that the correlator should attempt to correlate on.
Start Correlation	Make the correlator attempt to correlate based on previously set parameters and it will return the result of its correlation.
Query	Allows the user to store data to a database, the possibilities are storing all the xDRs and storing the correlated xDRs on an MSSQL database. Correlated xDRs are stored as multiple records to avoid data loss.
Explorer	Allows the user to view the data stored in the database.

Table 3.3: Table of Prototype correlator buttons

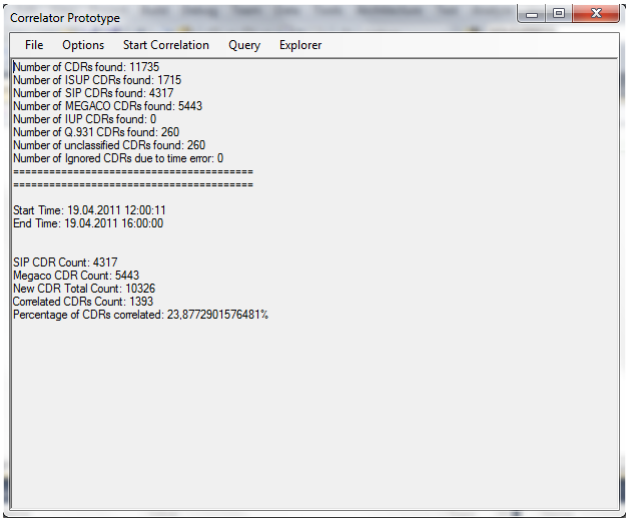


Figure 3.21: GUI of the prototype correlator

Chapter 4

Validation and Testing

Having two people involved in this project gives the opportunity for validation of coding tasks written by the members internally. Tasks which need to be implemented are created together by the group members. A group member implements a defined task, and when it is done, the other group member becomes responsible for validation of that task.

4.1 Usage of STINGA NGN Monitor

The STINGA NGN Monitor is a program provided to the members of this thesis by Utel Systems as a supplementary tool for testing and general idea gathering. The STINGA NGN Monitor is a program that Utel Systems uses to receive signaling data and present that data in a clear way. In addition, the NGN Monitor can be used to read and decode the Utel Systems proprietary xDR file format. The NGN Monitor is able to perform correlation on some cases, and SIP - Megaco is one such case.

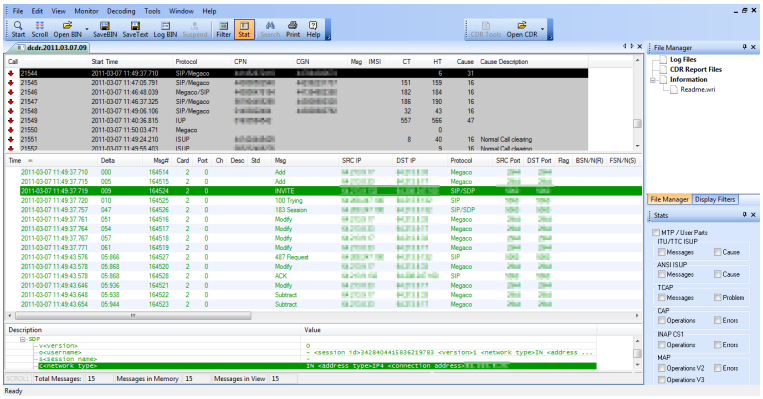


Figure 4.1: STINGA NGN Monitor.

In Figure 4.1 we can see that STINGA NGN Monitor has correlated a SIP and a Megaco CDRs together

4.2 Time Performance Testing

We started by running the same correlation test 10 times on the same dataset and gathered the result. Additionally, we tried the same thing on a different dataset. The correlation process is based on the parameters mentioned in subsection 3.3.7, which handles basic SIP - Megaco correlation.

Type of XDRs	Total xDRs in set	SIP xDRs	Megaco xDRs	% of set to compare	Processing Time
SIP/Megaco	11735	4317	5443	83,17%	38 ms
SIP/Megaco	11735	4317	5443	83,17%	49 ms
SIP/Megaco	11735	4317	5443	83,17%	51 ms
SIP/Megaco	11735	4317	5443	83,17%	63 ms
SIP/Megaco	11735	4317	5443	83,17%	64 ms
SIP/Megaco	11735	4317	5443	83,17%	52 ms
SIP/Megaco	11735	4317	5443	83,17%	52 ms
SIP/Megaco	11735	4317	5443	83,17%	62 ms
SIP/Megaco	11735	4317	5443	83,17%	65 ms
SIP/Megaco	11735	4317	5443	83,17%	35 ms
SIP/Megaco	32428	4446	5751	31,45%	68 ms
SIP/Megaco	32428	4446	5751	31,45%	64 ms
SIP/Megaco	32428	4446	5751	31,45%	78 ms
SIP/Megaco	32428	4446	5751	31,45%	68 ms
SIP/Megaco	32428	4446	5751	31,45%	58 ms
SIP/Megaco	32428	4446	5751	31,45%	67 ms
SIP/Megaco	32428	4446	5751	31,45%	71 ms
SIP/Megaco	32428	4446	5751	31,45%	73 ms
SIP/Megaco	32428	4446	5751	31,45%	68 ms
SIP/Megaco	32428	4446	5751	31,45%	73 ms

Table 4.1: A dataset of SIP/Megaco correlation time performance, performed on an Intel Core i5 2,26 Ghz processor running of a single thread using the Windows 7 x64 operating system.

Table 4.1 shows that the average correlation time for the first set is completed in 53 milliseconds on an Intel Core i5 2,26Ghz processor. The average correlation time on the second set of data which is twice as large was only 68 milliseconds on the same type of CPU. This data however does not show the whole picture because all the xDRs in question are not SIP and Megaco; in this dataset there is a mix of ISUP, IUP, SIP and Megaco. From this data we can say that the performance of the correlator code has a satisfying speed considering the size of the dataset.

The reason for this correlation speed is largely because the correlator algorithm performs correlation by working with just the data in the memory.

Type of CDR	ISUP CDRs	% of set to compare	Processing Time
ISUP/ISUP	82914	100%	1345 ms
ISUP/ISUP	82914	100%	1541 ms
ISUP/ISUP	82914	100%	1203 ms
ISUP/ISUP	82914	100%	1225 ms
ISUP/ISUP	82914	100%	1210 ms
ISUP/ISUP	82914	100%	1214 ms
ISUP/ISUP	82914	100%	1217 ms
ISUP/ISUP	82914	100%	1183 ms
ISUP/ISUP	82914	100%	1254 ms
ISUP/ISUP	82914	100%	1251 ms

Table 4.2: A dataset of ISUP Correlation Time performance on an Intel Core i5 2,26 Ghz processor running on a single thread using the Windows 7 x64 operating system.

We also tried a different correlation process on a much larger sample with a more complex correlation process. In the case illustrated in Table 4.2 the correlation time is drastically increased to 1264 milliseconds in average, but that is still within acceptable limits considering correlation time. We believe that there are two main reasons for the increased time. The first reason being that the correlation pool is much larger, 82914 CDRs to correlate compared to the 10197 and 9760 from the previous correlation pools in table 4.1. The second reason is the actual correlation process itself; when correlating ISUP with ISUP we use a parameter called hold time. Our correlator compares them by seeing if they are within the user specified percentage. This process is much more CPU intensive because it requires division of numbers. The type of correlation performed in Table 4.2 is based on the basic ISUP - ISUP correlation case mentioned in subsection 3.3.1.

The design of the algorithm is optimized for the correlation operations and the two dimensional linked list architecture used as the main data structure gives good

performance results for the correlation algorithm. On the other hand, there are some operations that use database such as writing all xDRs from the input file to the database, exploring xDRs, and writing correlated xDRs to database. Some of them are mostly for debugging purposes which users rarely need to use. Those operations obviously cannot be as fast as memory operations.

4.3 Scenario and Prototype Testing

Since the problem descriptions called for us to specifically test one specific scenario, we choose SIP - Megaco correlation. This allowed us to validate the correlation results from our prototype using the STINGA NGN Monitor program provided by Utel Systems, since the NGN Monitor is capable of performing correlation on SIP and Megaco. The biggest problem we had with validation was the fact that we had problems getting the proper statistics from the NGN Monitor without counting the SIP Megaco correlation cases manually. Despite of this problem we were able to validate our results with that of the STINGA NGN Monitor provided by Utel Systems.

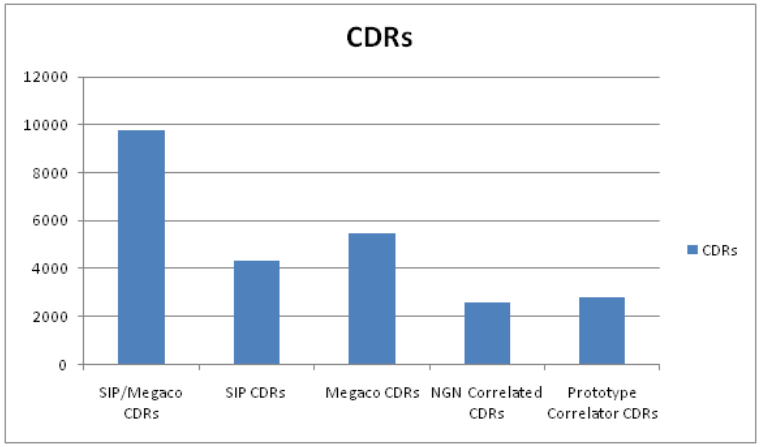


Figure 4.2: Correlation results comparisons chart.

From the data presented in Figure 4.2 we can see that there is a slight difference

in the number of CDRs correlated by our prototype correlator and the Utel Systems STINGA NGN Monitor. We believe that this number can be attributed to our correlation process. The one big difference in our correlation process is that the sliding time window timer is reset if a match is found, and this gives our correlator the possibility to find more CDRs outside the specified target zone. The reason for our decision to implement this feature is because we want the correlation process to be as correct as possible if multiple hops is present. In hindsight we probably should have implemented the ability to turn on and off that feature. Another huge factor for the slight discrepancy of the results between our correlator and Utel Systems STINGA NGN Monitor, is that we do not know exactly what type of method they use in their correlation process.

Given that the Correlation numbers are so close we believe that our correlator is performing as it should and the correlation process we completed correctly.

Chapter 5

Discussions

5.1 Our Solution

5.1.1 Hardcoded vs. Configurable

The solution we proposed in section 3.5 is a configurable protocol correlator, and the problem description limited us to designing a configurable protocol correlator. This approach is, we would argue, the best one. The reason for this is that it reduces the amount of code needed to be written and makes it easier and faster to create updates for the program. A configurable protocol correlator does not have to include a lot of code covering every conceivable case and that is why the code size for the main correlation algorithm will be reduced. With that said, all the parameters have to come on somewhere and that will typically be ourside the main algorithm. Aside from the benefits for the programmers there are also benefits for the end users and field in which the program will work. The program will no longer be limited to correlation of telecommunication protocols which is its primarily objective. The end users will be able to choose on which parameters they want to perform correlation, and this can be interpreted as something both good and bad at the same time. It shifts the burden of figuring out which parameters to correctly perform correlation on over to the end user, but in most cases this is a good

thing, because there are a regional differences in which communication protocols are used and how they are used. Configuring the correlator can also be performed by Utel System, and it is faster and cheaper than coding in new functionality in an existing product. An example of this is ITU-ISUP and ANSI-ISUP. The final case we will argue as to why a configurable solution is better is that there are cases where correlation is impossible because of missing information. As such, allowing the end-user to input the missing information for a much more accurate correlation.

Using a hardcoded approach does have some benefits as it allows the initial programming to go faster. It is also easier to create a prototype, but continued development will be difficult.

5.1.2 Alternatives to Sliding Time Window

In the design and prototype parts of this thesis we decided to go for the approach called Sliding Time Window. The reason for this is because it greatly narrows down the search space which makes it possible to perform correlation in real-time and on large amount of xDRs. Additionally, most xDRs are chronologically close to each other, this means that similar xDRs are within a small timeframe. In our implementation of the Sliding Time Window were only looking backwards x number of the seconds, where x is a value chosen by the end user. Another possibility is to look both backwards and forwards x number of seconds; however the reason we chose not to do this was because it created potential problems when performing real-time correlation.

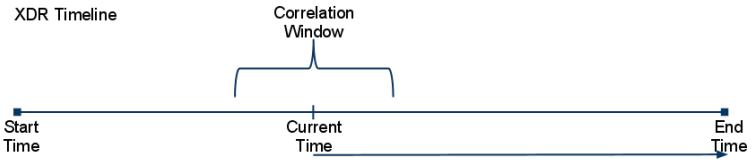


Figure 5.1: Sliding time window looking forwards and backwards.

In Figure 5.1 we can see that issues can arise if the future time part of the

correlation window surpasses the end time.

There are different ways to search through the xDR list in order to find xDRs that can be correlated. The easiest and arguably worst way to do this is to do an exhaustive search; this will cause the currently selected xDRs to be compared to every xDR in the whole list. The two major reasons for why we didn't use this option were that it is very slow; we can see that without even testing it.

5.1.3 xDR Differences

xDR is a generic term for a configurable format and this has presented us with some challenges on correlation. Due to the generic nature of xDRs which creates the possibility of many different types of xDRs that needs to be processed differently we have tried to be as generic in our approach as possible. Unfortunately we cannot be certain that we have thought of all possible scenarios and this is a potential problem area in the future should the correlator be extended to include more types of xDRs.

The main differences that we have found are what type of protocol or service that an xDR is for. For protocols there should be few problems when extending the correlator to incorporate the new xDR types. The main issue is when the xDRs contain services and not protocols, and this can cause unforeseen problems that we have not been able to consider when we designed the protocol correlator. The reason for this is that it was not specifically specified in the thesis description, see section 1.3.

Our ideas on a service based xDR are things like SMS xDRs which contain information from both CAP and MAP protocols; the actual text, CPN, CdPN, billing information and stuff like that. The way we can achieve correlation on that example is through the use of hierarchical correlation which we touch on how to do in section 6.3 further work.

5.2 Other Apporaches

Our approach was for the most part already set from the start by the problem description and our discussions with Utel Systems. However, in addition to this we have looked at other possible approaches to solve the problems in this thesis. Some other ways to approach the problem in this thesis and the advantages and disadvantages of them will be presented below.

5.2.1 Graph Theory

The first promising alternative approach that we have looked at entrails the use of graph theory to correlate the xDRs rather than use our current solution, in which the user chooses the parameters to correlate on. Using graph theory for the correlation means creating an instance for the network topology based on a graph structure(links and nodes) and optimizing it for the protocols which needs to be supported and handling all correlation steps by processing the graph items nodes and links.

To use graph theory the correlator should already know the topography of the network which your probes are located, or the correlator can create the topology itself by using the xDRs. In graph theory approach on xDR correlation, correlator probably needs probes everywhere in the network. Otherwise having accurate results from graph algorithms might be hard. It may be possible to use graph theory in the future of the xDRs themselves contain information about the network topography. However since they currently do not and not knowing the topography will almost nullify any benefits of using graph theory due to low accuracy it is currently not feasible.

Chapter 6

Conclusions and further work

6.1 Concluding Remarks

In this project the main goal was to create a generic protocol correlator model and implementing a prototype that is capable of handling the xDR data. The specific protocols focused on are ISUP, SIP, Megaco and CAP. Additionally the multi-protocol correlator should also allow the end-user to select which protocols and which parameters inside the protocols to correlate on. Looking at the work outlined for us in the problem statement we can say with confidence that the overall goal of this thesis has been achieved. There have been problem areas and problem statements which we have had to make slight adjustments to. An example of that is in the early planning stages the idea was to include major statistical analysis. Looking through the program specified in 1.3 this is what we have done.

1. A configurable correlator model has been designed.
2. A bridge has been implemented. It connects the xDR reader which reads the Utel format xDRs to the Protocol correlator which is written for this project.
3. Correlator algorithm has been designed and implemented in C#.
4. User interface has been implemented.

5. Correlator performance testing is acceptable.

6. Correlator correlation testing has been compared with Utel Systems STINGA NGN Monitor and the result is acceptable.

6.2 Main Contribution

Our proposed solution on how to correlate between the different protocols ISUP, SIP, Megaco and CAP have been discussed with engineers from Utel Systems; and some of them have been verified by using the prototype protocol correlator and the STINGA NGN Monitor from Utel System. SIP - Megaco Correlation is one such combination. The multi-protocol correlator design and source code can be of use to Utel Systems.

6.3 Further Work

Correlation algorithm is open for changes and/or improvements for different purposes and we have thought of a few improvements. The first major place for improvement is multi-threading, as that will allow the correlation process to both go smoother and faster. It will also take advantage of todays multi-core processors. The second place for improvement to the algorithm is including the functionality for merging and splitting groups on the linked list, as that will allow the correlator to perform additional correlation on the result of the first correlation process. This hierarchical correlation process is advantageous because it allows us to break up complex correlation processes into smaller and simpler tasks.

6.3.1 Multi-Threading

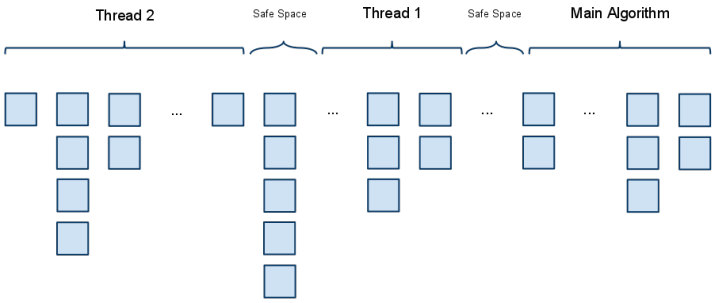


Figure 6.1: Multiple Theads working on the same dataset

Figure 6.1 illustrates a workaround about how to implement extra criteria for correlation on the current architecture. It is seen that the main algorithm is running in the end of the list and it keeps updating the list continuously. Thread 1 is running behind the main algorithm and is followed by Thread 2. This allows the working range off the threads and the main correlation process stay away from overlapping. It is possible to do that by using timestamp information. Also leaving safe space (a range defined with certain amounts of seconds) will guarantee avoiding any overlaps.

Main correlator algorithm focuses mostly on mandatory and multipurpose correlation criteria.

Implementing new correlation rules and criteria or doing temporary changes on the main algorithm might be an unwelcome decision in some cases; especially if the additions or changes are temporarily or rarely used, and doing the new implementation separately is usually better for development approach. Implementing a new criteria, correlation support or extensions which serves extraordinary rules for a new protocol might need different design approaches.

Main correlation algorithm leaves the correlated xDRs as groups in the link lists. Assume that there is another thread implemented and activated behind the

main algorithm which re-processes the xDR groups in the list. Also assume this thread is able to combine and split the groups in the list depending on its decision. There is one combining and one splitting example given below.

Working with double link lists has some advantages. It is flexible, fast and quite suited for correlation logic.

6.3.2 Combining Groups

The basic idea of combining groups is that the main algorithm can change the structure of an existing linked list based on the some new parameters and possibly new data as well. Figure 6.2 shows the idea of combining groups in a pre existing linked list created in a previous correlation process.

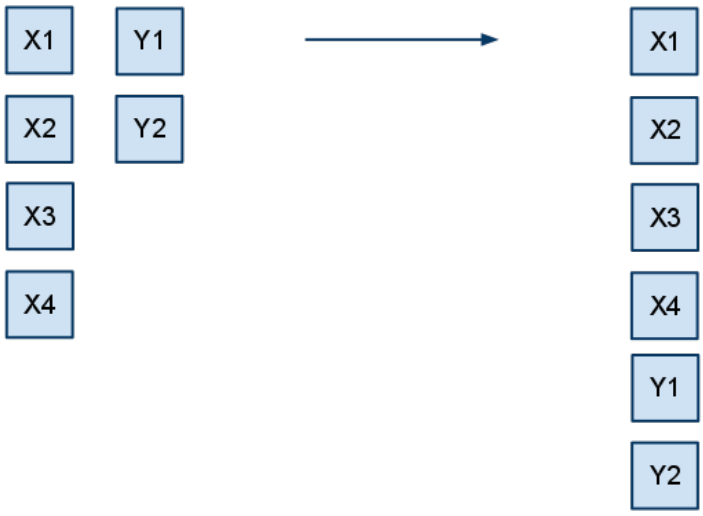


Figure 6.2: Combining Groups with the linked list.

6.3.3 Splitting Groups

Having the correlation algorithm combining groups on a pre-existing linked list gives the correlator program (as previously mentioned in section 6.3) many advantages. However, it is only useful if splitting of groups is also implemented. Figure 6.3 shows the idea of splitting groups in a pre-existing linked list created in a previous correlation process.

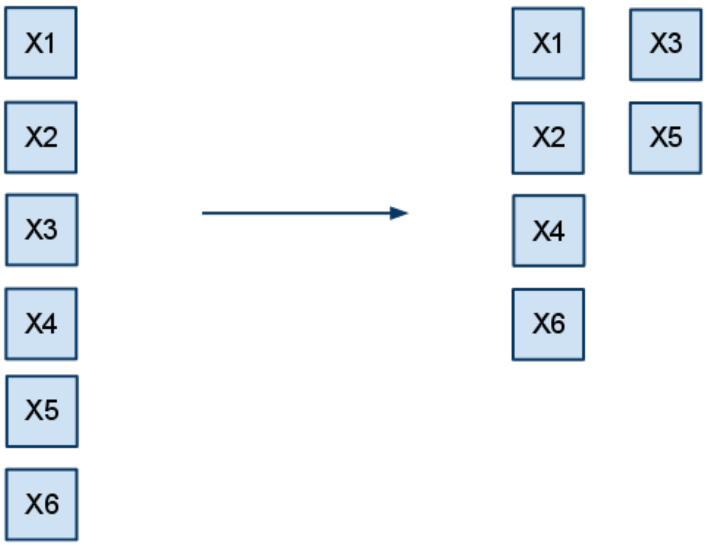


Figure 6.3: Splitting a group with the linked list.

Bibliography

[1] 3GPP. Customized Applications for Mobile network Enhanced Logic (CAMEL) Phase X; CAMEL Application Part (CAP) specification, 2010. http://www.3gpp.org/ftp/Specs/html-info/29078.htm.

[2] 3GPP. Mobile Application Part (MAP) specification, 2011. http://www.3gpp.org/ftp/Specs/html-info/29002.htm.

[3] A. Biasutto A. Bovo and A. Nicchio. Multi-protocol call trace on GPRS Gb-Gr. http://www.patentstorm.us/patents/6915110/description.html, July 2005.

[4] H. Kheddouci B. Serrour, D. P. Gasparotto and B. Benatallah. Message Correlation and Business Protocol Discovery in Service Interaction Logs. In *Proceedings of the 20th international conference on Advanced Information Systems Engineering*, CAiSE '08, pages 405–419, Berlin, Heidelberg, 2008. Springer-Verlag.

[5] D. Kiong D. Poo and S. Ashok. *Object-Oriented Programming and Java*. Springer, 2008.

[6] L. Dryburgh and J. Hewett. *Signaling System No. 7 (SS7/C7) Protocol, Architecture and Services*. Cisco Press, 2004.

[7] et al. G. Camarillo. Integrated Services Digital Network (ISDN) User Part (ISUP) to Session Initiation Protocol (SIP) Mapping.

[8] H. Gomaa. *Software Modeling and Design*. Cambridge, 2011.

[9] ISO. Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model, 1994. http://standards.iso.org/ittf/PubliclyAvailableStandards/s020269_ISO_IEC_7498-1_1994(E).zip.

[10] ITU-T. Signalling System No. 7 - ISDN user part format and codes, 1999. http://www.itu.int/rec/T-REC-Q.763/en.

[11] ITU-T. ITU-T, H.248.1, Gateway Control Protocol, 2005. http://www.itu.int/rec/T-REC-H.248.1-200509-I/en.

[12] et al. J. Peterson. S/MIME Advanced Encryption Standard (AES), Requirement for the Session Initiation Protocol (SIP), 2004. http://tools.ietf.org/html/rfc3853.

[13] et al. J. Rosenberg. SIP: Session Initation Protocol, 2002. http://tools.ietf.org/html/rfc3261.

[14] F. Daniel M-S. Hacid F. Casati K. Musaraj, T. Yoshida and B. Benatallah. Message Correlation and Web Service Protocol Mining from Inaccurate Logs. In *Proceedings of the 2010 IEEE International Conference on Web Services*, ICWS '10, pages 259–266, Washington, DC, USA, 2010. IEEE Computer Society.

[15] M. Mangri and M. Nafornita. MEGACO correlation method. In *Proceedings of the 14th WSEAS international conference on Communications*, ICCOM'10, pages 252–258, Stevens Point, Wisconsin, USA, 2010. World Scientific and Engineering Academy and Society (WSEAS).

[16] R. Noldus. *CAMEL: Intelligent Networks for the GSM, GPRS and UMTS Network*. Wiley, 2006.

[17] M. Stafford. *Signaling And Switching For Packet Telephony (Artech House Telecommunications Library)*. Artech House, 2004.

Acronyms

3GPP 3rd. Generation Partnership Project. 33

ACM Address Complete Message. 49

AES Advanced Encryption Standard. 16

ANM Answer Message. 36, 49

ANSI American National Standards Institute. 40, 77

API Application Program Interface. 57–60

ASN.1 Abstract Syntax Notation No.1. 37

ATM Asynchronous Transfer Mode. 34

CAP CAMEL Application Part. 5, 8, 11, 13, 14, 16–19, 21, 29, 30, 37–39, 46, 47, 49, 78, 80, 81

CBP Component Based Programming. 27

CdPN Called Party Number. 22, 32, 44, 45, 47–49, 53, 54, 56, 78

CDR Call Detail Record. 14, 19, 21, 22, 43–45, 51, 54–57, 65, 71

CIC Circuit Identification Code. 26, 30, 31, 39, 52

CPN Calling Party Number. 22, 44, 45, 47–49, 56, 78

DNS Domain Name System. 21

DPC Destination Point Code. 26, 31, 39

DSLAM Digital Subscriber Line Access Multiplexer. 34

GCR Global Call Reference. 6, 46, 47

GPRS General Packet Radio Service. 15

GSM Global System for Mobile Communications. 15

GUI Graphical User Interface. 6, 17, 57, 58, 69

HT Hold Time. 22

IAM Initial Address Message. 49

IETF Internet Engineering Task Force. 33

INAP Intelligent Network Application Part. 11, 38

INAP Intelligent Network Application Part. 21

IO Input/Output. 57, 58

IP Internet Protocol. 11, 17, 26, 29–31, 33, 34, 37, 45, 50, 51, 56

IPDR IP Detail Record. 21

ISDN Integrated Services Digital Network. 21, 31, 37

ISO International Organization for Standardization. 28

ISUP ISDN User Part. 5, 10–14, 16–19, 21, 24–26, 29–33, 36, 38–40, 43–49, 52, 53, 55, 56, 72, 73, 77, 80, 81

ITU International Telecommunication Union. 31, 33, 40, 77

MAP Mobile Application Part. 5, 11, 16, 21, 37, 38, 78

Megaco Media Gateway Control Protocol. 13, 14, 16–19, 24, 29, 30, 34–36, 39, 46, 50–52, 56, 70–72, 74, 80, 81

MG Media Gateway. 14, 34, 36, 37, 46

MGC Media Gateway Controller. 34, 36, 37, 46

MGCP Media Gateway Control Protocol. 34

MPLS Multi-Protocol Label Switching. 34

MSC Mobile Switching Service. 46

MTP3 Message Transfer Part 3. 29, 30

OOP Object Oriented Programming. 27

OPC Originating Point Code. 26, 31, 39

OSI Open Systems Interconnection. 11, 19, 28, 30, 38

PDU Protocol Data Unit. 15

PLMN Public Line Mobile Network. 37

PSTN Public Switched Telephone Network. 30, 31

REL Release Message. 49

RTP Real-time Transport Protocol. 36, 37, 49–51

S/MIME Secure/Multipurpose Internet Mail Extensions. 16

SCTP Stream Control Transmission Protocol. 29–31

SDP Session Description Protocol. 56

SIGTRAN Signaling Transport. 11, 17, 30

SIP Session Initiation Protocol. 10, 13, 14, 16–19, 21, 29, 30, 33, 38, 39, 45–51, 56, 70–72, 74, 80, 81

SMS Short Message Service. 37, 78

SOA Service Oriented Architecture. 27

SS7 Signaling System no.7. 5, 11, 17, 30–32, 36, 39, 49

SSP Service Switching Point. 31, 54

STP Signaling Transfer Point. 31

TCAP Transaction Capabilities Application Part. 21

TCP Transmission Control Protocol. 29–31, 59

TDR Transaction Detail Record. 15, 21

UDP User Datagram Protocol. 29, 30

xDR x Detail Record. 2–4, 12–15, 17–19, 21–26, 28, 41–43, 45, 47, 49, 50, 53, 54, 56, 58–70, 72, 74, 77–80, 82, 83

Appendix A

User Manual

This is a short introduction into using the prototype correlator. This correlator is only a quick prototype, as such we do guarantee that it will run correctly or smoothly on any system.

The following is a preliminary checklist of what is required in order to run the program.

1. Make sure that you have *Correlator_GUI.exe*.
2. Make sure that *decoder_dll.dll* is in the same directory as *Correlator_GUI.exe*.
3. Make sure that the .NET framework is installed.

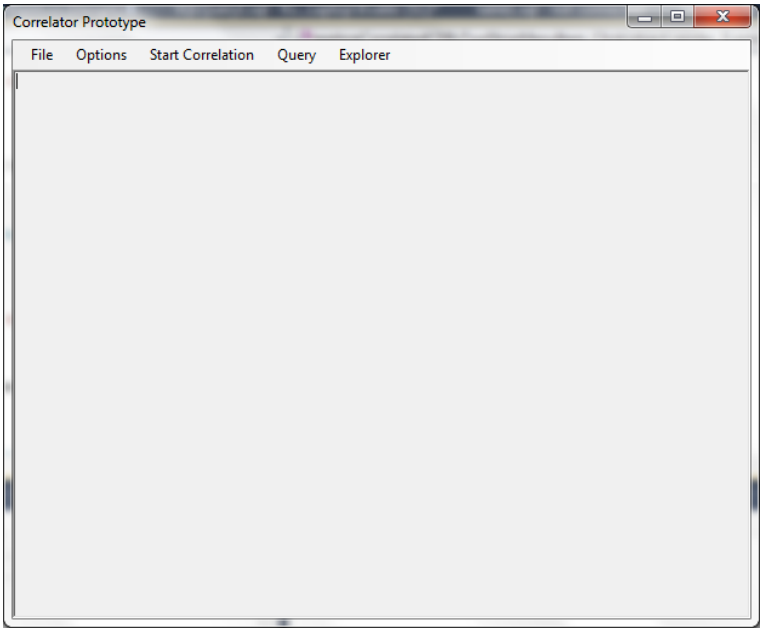


Figure A.1: The Correlator GUI at the startup of the program.

Once the program is the started; a simple GUI will greet you, and Figure A.1 illustrates that. The first thing you need to do is click on *File* and choose a xDR file containing xDRs.

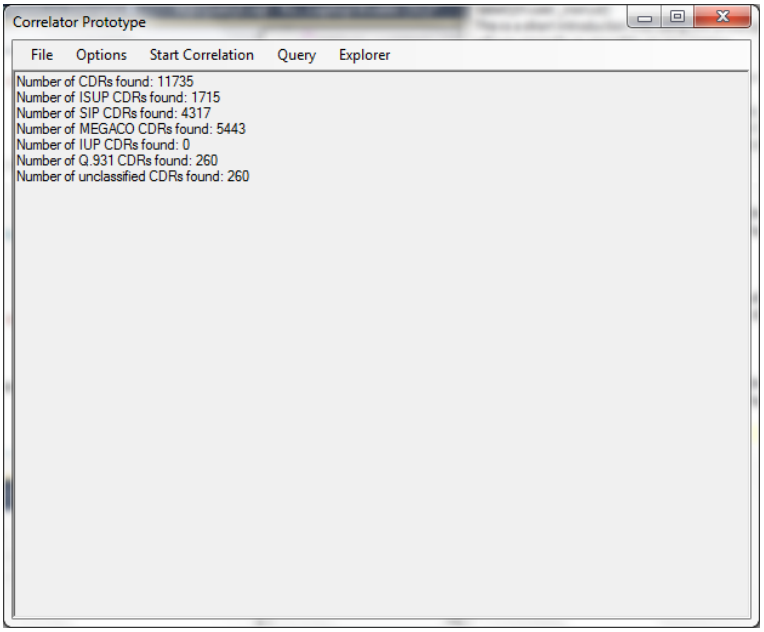


Figure A.2: The Correlator GUI after a file has been loaded into the system.

Figure A.2 illustrates how the program GUI looks like once a file has been loaded into the system. The next task is to choose the parameters which the correlator will preform its correlation on. In order to do that you will first need to press *Options* and then *Select Parameters*; you will now be greeted by a new dialog.

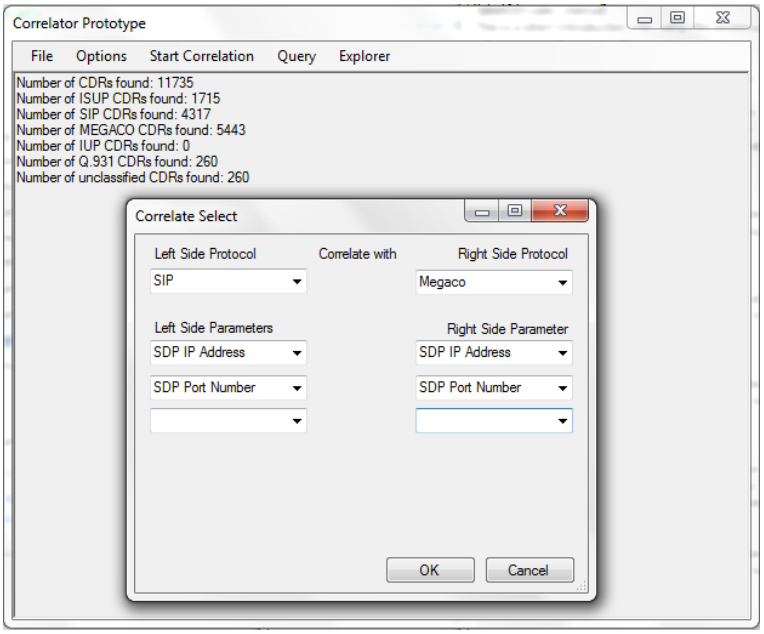


Figure A.3: The Parameter Select dialog with the correlator GUI in the background.

Figure A.3 illustates an example of how the parameter select dialog may look like. In This example SIP and Megaco are the protocols that the correlator will attempt to correlate on, and SDP IP and Port number are the parameters. Click *OK* to save the preferences. At this point you can either click *Start Correlation* to perform the correaltion process or click on *Options* and then on *Misc Options*.

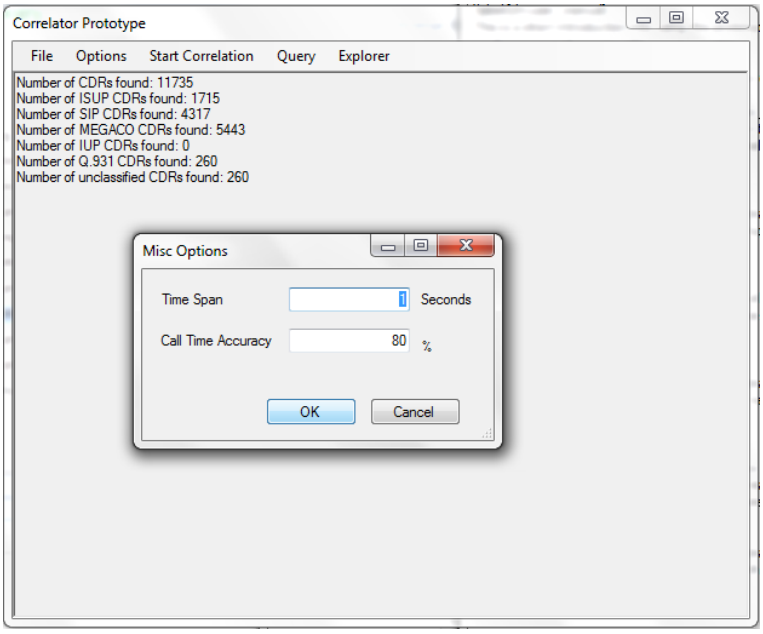


Figure A.4: The Misc Options dialog with the correlator GUI in the background.

Figure A.4 illustrates the Misc Options dialog. This dialog contains two correlator parameters that the user can manipulate. *Time Span* allows you to change the size of the time window in which the correlator will look for similar xDRs. *Call Time Accuracy* allows you to change the accuracy required in order for the correlator to mach Time based parameters.

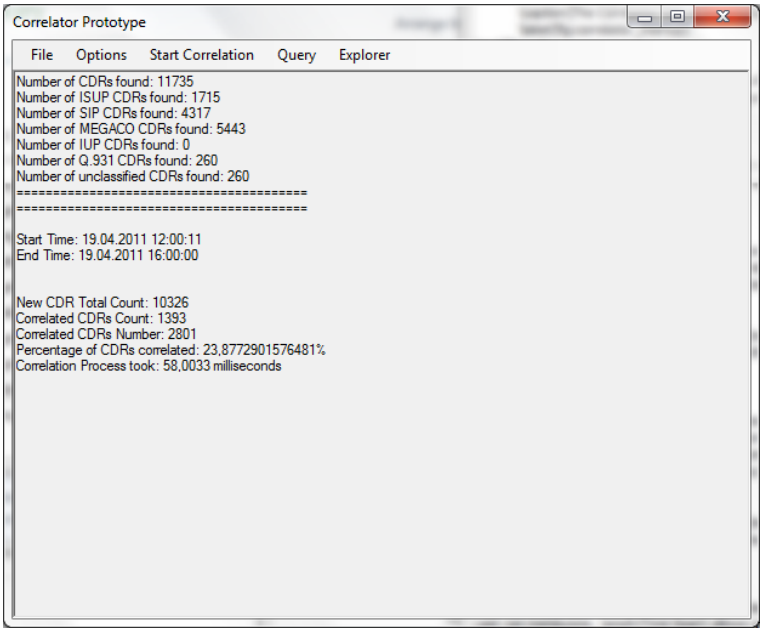


Figure A.5: The Correlator GUI after a correaltion is performed.

Figure A.5 illustrates how the correlator GUI looks like after a correaltion is performed. At this point the only thing left is to same the xDRs and the correlated xDRs to a database. In oder to do that you need to click on *Query* and then on *log file to db* and finally on *write correlated CDRs to database*. This action has not been tested to work on other computers due to the limited time available. *Explorer* allows you to view the data present in the database.