SRI LANKA INSTITUTE OF INFORMATION TECHNOLOGY

Brute Force Grid

CTP Final Report 2009

CURTIN UNIVERSITY OF TECHNOLOGY

Declaration

We declare that the project has been successfully completed by ourselves and it is not a copy of another product or technology. All the aspects under this research project are successfully designed and developed by the group members of the project as a result of dedication and hard work.

Project Title : Brute Force Grid

Project ID : CTP / 2009 / CURTIN / 18

Group members :

- M. F. F Faraj
  (DCN / 07 / C3-0619 | Curtin ID : 14451524)
- W. A.H. Jayawilal
  (DCN / 07 / C4-0775 | Curtin ID : 14451825)
- W.A.R. Lasitha
  (DCN / 07 / C3-0634 | Curtin ID : 14451553)

Supervised by : MR. P.L. RUPASINGHE

Signature : _____ (29th October 2009)

Abstract

From the early Stone Age to the modern technical world, man has been seeking for better ways to get his work done more efficiently. Eventually he figured out by sharing his work among others it's easier and much faster and quicker for him to finish. So this dilemma of sharing work has improved and enhanced over the years by people and has spread to all the different aspects in our lives. When it comes to the field of Information Technology and Networking, it too has a great value for this concept since we all want is to get our things done quickly no matter what. 'Grid computing' and 'parallel processing' are some of the terms which are woven around in this concept. Giving a new definition to the world of parallel computing, people have been trying to run a process in parallel on a distributed network tied up with peer-to-peer concept, which is a powerful hybrid of well-known technologies. Though there are so many distributed network architectures available with grid-computing, yet there are no actual prototypes or implementations done regarding this concept. So our mission in undertaking this project was to come up with an actual implementation, a distributed system which is empowered by peer-to-peer concept and run user defined tasks on it, bringing a revolutionary era in the field of IT. Many rich features like, Address/node learning, process breakdown and information sharing make this a handy aggregation, delivering the user a refreshing feeling and making him to say nothing else than "oh! Now it's easy!"

We decided to run a brute force algorithm as the application on top of this framework since it needs a lot of computations, thus it allows us to illustrate the fruitfulness of our project. Themed BFG or the Brute Force Grid is such described implementation. The full details of this interesting project are discussed in detail in the pages to come.

Acknowledgements

- Mr. Lakmal Rupasingha
- Dr. Chandimal Jayawardena
- Dr. Malitha Wijesundara

Contents

I - Tables

II - Figures

III - Abbreviations

- IT – Information Technology
- BFG – Brute Force Grid
- CPU – Central Processing Unit
- P2P – Peer-to-peer
- PC – Personal Computer
- LAN – Local Area Network
- IP – Internet Protocol
- MD5 – Message-Digest Algorithm 5
- GUI – Graphical User Interface
- FT – File Transfer
- ASCII – American Standard Code for Information Interchange
- SQL – Structured Query Language
- WAN – Wide Area Network
- UDP – User Datagram Protocol
- TCP – Transmission Control Protocol
- SHA – Secure Hash Algorithmes
- DES – Data Encryptions Standard
- SSH – Secure Shell

1. Introduction

In today's world, internet plays one of the major roles in each of the individual life and the company's progress. According to the studies made, it has been proved that much concentration of the data traffic passing via the cables of this internet work is from the one major innovative idea of peer-to-peer file sharing. This is sharing the files over the overlay network which is presented in the internet in a distributed manner.

Our approach is to show that this major innovative idea of peer-to-peer is not limited to the file sharing but it can be also be used for much complex and interesting views of the current world. Such as process sharing, peer-to-peer multimedia streaming, etc…

Figure 1.1 – The mesh of Peer-to-Peer

For our experiment we took the concept of process sharing. Processes which need a very huge amount of CPU power. This can be divided into smaller modules and executed in different peers and recollected to combine and result the actual outcome.

Using this experiment we were planning to demonstrate that Peer-to-Peer concept can be successfully implemented in the field of process sharing using parallel processing the sub modules.

1.1 Background

The project Brute Force Grid is about analyzing the parallel process execution in more than one computer which was interconnected using the Peer-to-Peer network. The concept behind the project, which is using the P2P concept for parallel processing is under the main discussion but not considered as a much more important discussion of this time, where the internet is considered as a world revolutionary source of interconnection between personals.

There are plenty of research papers written about this issue. But no innovative result has been developed so far according to this specific concept.

- The research paper written by *Karl Fenech* and *Kevin Vella* describes the basic concept of parallel processing using the P2P network concept in their research paper *"Parallel Processing over a Peer-to-Peer Network. Constructing the Poor Man's Supercomputer."*

- The book written by *Fatos Xhafa* also speaks about the same concept we are going to deal around. The book is named as *"Parallel Programming, Models and Applications in Grid and P2P Systems "*

- *Algorithms and architectures for parallel processing* is a book and a theses written by *Hai Jin, Omer F. Rana, Yi Pan* about the parallel processing underlying concepts. In that study they have told about the P2P concept which can be used to develop a parallel process interface.

These are some of the much important work pieces that we went across when we are planning to develop a system which can be successfully used to show up the workability of the parallel processing for process sharing on top of P2P concept. There are some other website, forums and blog posts about this issue, which I have added to the reference section of the report.

Having all these considerations in mind we find the perfect application that can be used to demonstrate the concept. That is the brute force attack for hashes.

This is how www.webopedia.com defines hashes.

"Producing hash values for accessing data or for security. A hash value (or simply hash), also called a message digest, is a number generated from a string of text. The hash is substantially smaller than the text itself, and is generated by a formula in such a way that it is extremely unlikely that some other text will produce the same hash value.

Hashes play a role in security systems where they're used to ensure that transmitted messages have not been tampered with. The sender generates a hash of the message, encrypts it, and sends it with the message itself. The recipient then decrypts both the message and the hash, produces another hash from the received message, and compares the two hashes. If they're the same, there is a very high probability that the message was transmitted intact." [01]

We took this because, the brute force attack can be carried out portioned in multiple hosts easily and the specific process needs very huge amount of processing. This satisfies our basic need for the experiment to be carried out. We select the hash key braking concept using brute force attack to recover the key from the hash.

1.2 Motivation

The motivation for this project is built by the research papers that we read. It all contained the information to build a prototype, but still no high end product produced to make the concept of P2P works out the parallel processing. There are so many parallel processing frameworks, but neither of them were designed in a P2P distributed architecture over the public network, internet.

Other than this the research paper *"Parallel Processing over a Peer-to-Peer Network. Constructing the Poor Man's Supercomputer."* By *Karl Fenech* and *Kevin Vella.* This gave us big motivation to do the project in the field. The Poor man's Supercomputer theme is something which can make anyone try out the way to create a cheaper super computational model.

The problem of executing a process which needs a huge processing power such as a brute force attack cannot be carried out in one typical computer. This needs a special CPU powered super computers to make it done quickly and easily. But this cannot be achieved by a typical PC user who is in need of doing a complex task in a cheaper way. So we narrowed our scope to provide the users of our application the interface of a super computational powered architecture using a simple and easy model of P2P over the internet.

1.3 Objectives

Our project objective can be divided in to two major parts. One is the most straight away show off objective of efficient brute force attacks. And the other one is the basic objective of the project which is too show the P2P concept can be used to implement process sharing using parallel processing in an effective way.

- Efficient Brute force attack

  This can be achieved by providing an efficient and faster way for recovering the passwords from the hashes in the distributed network using the brute force application, is much better than the single host attack concept.

- Implement the concept of P2P for process sharing

  To show this we should create a complete working prototype that can be used to make the implementation of parallel process to share the processing power via the P2P network. The performance should be shown increased and provide a solution via a distributed system.

- Providing a mechanism to make the peers connect together in the initial state

  When a peer starts, it doesn't know the address of the other peers connected to the network. We should provide a solution for the initial connectivity when a new peer gets-up or taken down. Many peer to peer concepts face the problem of the initial setup of the network. We have made use of the currently available mechanism in torrents to address this problem. We shall discuss this more in below sections.

1.4 Contribution of the research

The project Brute Force Grid provides a new innovation towards the concept of parallel process sharing using the Peer to Peer concepts.

The research and the prototype designed can be used as a step toward the parallel processing using the P2P network. This project can be used to determine the efficiency of the process sharing over a single PC's performance over the brute force attack.

This also enables other researches to go ahead with the current research and provide better solutions. We have given the first step by building a framework which can be used to any P2P parallel processing applications. So this framework can be used to make the P2P concept work on the process sharing and parallel processing views.

Project prototype provides a low level of hashing recovery concept. This is the user interface that is used to enter all the details for a specific task. This actual prototype developed can be used to decode the hash script by the brute force attack and get the key. But this can be use to demonstrate the P2P concept working on the process sharing arena.

Overlay network definition

"An Overlay network is a term for network that run on top of existing infrastructure but provide certain additional functionality. Overlays are created for a number of different reasons. Some overlays are created to provide test beds for novel protocols and distributed systems that are hard to test on existing networks (for example, multicast and IPv6 experiments). Other overlay networks are used in production to provide features non-existent in the original protocols and infrastructure for instance, to provide additional security features to transactions done over mainstream TCP and IP protocols that were not designed with security in mind. Another example of features overlay networks can provide without drastic, hard-to-implement changes to the existing infrastructure is quality of service control." [02]

1.5 Organization of project

The project technical organization is based on the peer software; this simple application which is running on the host is the main component which makes the host a peer of the BFG P2P overlay network.

We design the framework from scratch, to make it utilized to the specific task of parallel processing as the current frameworks are dealing with file sharing concepts. The major parts available in the system are shown in the figure below.

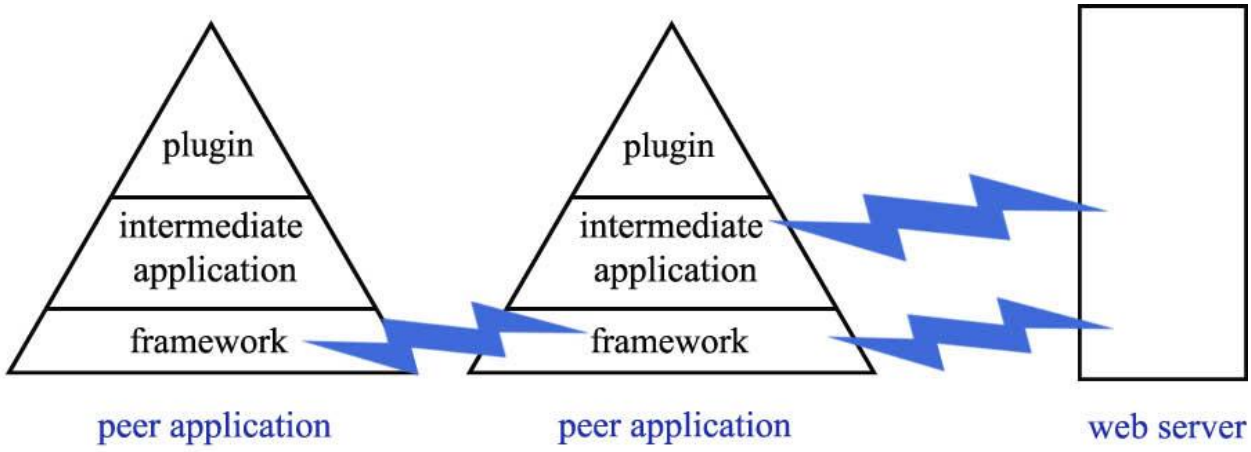

Figure 1.2 – Basic configuration between peers and web server

- The Peer application

  The application which runs on the host to make that host a peer of the overlay network
- Web server

  The server which is used to address the problem of initial network setup
- The application plug-in.

  Software used to recover the keys from the hash using the brute force mechanism.
- intermediate application

  The platform created to make the plug-in application run over the framework in a distributed manner.

These are important part or components that the brute force grid prototype is made of. We shall discuss about this topic thoroughly throughout the rest of the paper and not intended to speak much in this specific section of the report.

The project conceptual organization is done by going though some of the major phases which are in needed to complete the project successfully. This includes:

- Defining the experiment
- Developing the framework
- Utilizing the framework to be made as BFG specific application for brute force attacks
- Developing the plug-in, intermediate applications and the web server
- Concatenate the project components together and make it work together.
- Utilize the project to work efficiently
- Verify the P2P concept can be used to work in the process sharing
- Check the performance of the working Brute Force Grid with the single mode attack used to recover the key from the same hash.

2. Literature review

Since peer-to-peer concept, parallel and distributed computing have become interesting topics in the field of IT, people have done some researches, and many books and other materials have been written regarding this phenomena which has helped us in numerous ways in developing this project.

[3] The research paper “Parallel Processing over a Peer-to-Peer Network Constructing the Poor Man’s Supercomputer “ written by Karl Fenech and Kevin Vella describes about the assimilations of the technologies. They further talk about the p2p concept and the value of using it as a core technology with distributed computing. They say “Our vision is to tackle the aforementioned obstacles by building a P2P system capable of deploying user-defined tasks onto the network for distributed execution.”

[4] The book “Algorithms and the Architectures for parallel processing” by Hai Jin, Omer F. Rana, Yi Pan describes the parallel processing algorithms that are being used and it gives a major consideration on p2p concept and how it is important as a parallel computing algorithm.

[5] The book written by Fatos Xhafa also speaks about the same concept we are going to deal around. The book is named as “Parallel Programming, Models and Applications in Grid and P2P Systems “ which carries priceless information about parallel programming models and more about Grid computing and p2p concept.

[6] “The future of peer-to-peer computing” is another book written by A.W.Loo which talks about peer-to-peer grid computing architecture models. Further he talks about the pros and cons of each model and how they can be enhanced. Also he speaks about the future usage of this p2p concept and how it will become a prime technology when it comes to networking. Future challenges in grid computing and how p2p concept can be used to succeed them are also mentioned in here.

[7] “The evolution of the Grid” of D. De Roure, M.A. Baker, N.R. Jennings, and N.R. Shadbolt talks about the evolution of the grid computing technology, its journey from a concept to a technology and how widely it is used today.

[8] “Grid Computing: Making the Global Infrastructure a Reality” by F. Berman, G. Fox, and A.J.G. Hey describes a little bit of the history and how grid computing is today.

[9] In the book, “Defining the grid: a snapshot on the current view” by H. Stockinger clearly defines the grid concept and he further illustrates the current position of it and how it is being used as a main technology.

[10] “A Survey of Peer-to-Peer Content Distribution Technologies” by S. Androutselli Theotokis and D.Spinellis is a very interesting research article regarding p2p concept. In there they express their ideas about the p2p how it is important, its advantages and how it can be used to solve many networking issues and how it differs with other concepts. The results of their survey and the conclusion are also mentioned in there.

[11] “Distributed Computing: An Introduction” by L. Erlanger is a book which gives you an idea about the distributed computing.

[12] “Extracting Patterns and Relations from the World Wide Web,” by S. Brin, he considered the problem of extracting a relation to a particular data type which is scattered across thousands of independent information sources in many different formats. Furthermore, he presented a technique which exploits the duality between sets of patterns and relations to grow the target relation starting from a small sample.

[13] Piranha processing – utilizing your down time,” HPC wire (Electronic Newsletter) says that Java offers the basic infrastructure needed to integrate computers connected to the Internet into a seamless parallel computational resource: a flexible, easily-installed infrastructure for running coarse-grained parallel applications on numerous, anonymous machines. Ease of participation is seen as a key property for such a resource to realize the vision of a multiprocessing environment comprising thousands of computers. We present Javelin, a Java-based infrastructure for global computing. The system is based on Internet software technology that is essentially ubiquitous: Web technology. Its architecture and implementation require participants to have access only to a Java-enabled Web browser. The security constraints implied by this, the resulting architecture, and current implementation are presented. The Javelin architecture is intended to be a substrate on which various programming models may be implemented.

[14] “Computational intermediation and the evolution of computation as a commodity,” by A. Davies, indicate that The consumer who purchases computational power ultimately purchases a reduction in the time interval between the initiation and the completion of work. This paper looks at computation as a commodity and the nascent industry of computational intermediation, and proposes a model for the market for computational power as distinct from the market for computers. Some interesting results emerge. The model implies that the demand for computation is discontinuous and that there is a lower limit to the quantity of computation consumers will demand that is independent of the price of power. The

model identifies a range of computational powers that could be supplied by computational intermediaries but which will not be supplied by computer manufacturers, and suggests a model for pricing computation.

[15] H. Stockinger, from his book “Defining the grid: a snapshot on the current view,” indicates that The term “Grid” was introduced in early 1998 with the launch of the book “The Grid. Blueprint for a new computing infrastructure”. Since that time many technological changes have occurred in both hardware and software. One of the most important ones seems to be the wide acceptance of Web services. Although the basic Grid idea has not changed much in the last decade, many people have different ideas about what a Grid really is. In the following article we report on a survey where we invited many people in the field of Grid computing to give us their current understanding.

[16] S. Androutsellis-Theotokis and D. Spinellis, in their book “A Sur-vey of Peer-to-Peer Content Distribution Technolo-gies,” says that Content distribution networks (CDNs) are a type of distributed database using geographically dispersed servers to efficiently distribute large multimedia contents. Among various kinds of CDNs are those resembling peer-to-peer (P2P) networks in which all the servers are equivalent and autonomous, are easy to maintain and tolerant of faults. However, they differ from P2P networks in that the number of nodes joining and leaving the network is negligible. The main problems in CDNs are the placement of contents, and the location of content. Widely used CDNs either have inefficient flooding-like techniques for content location or restrict either content or index placement to use distributed hash tables for efficient content location. How ever, for the efficient distribution of contents, the contents must be optimally placed within the CDN, and no restrictions should be placed in the content or index placement algorithm.

[17] in Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities article by Dr. Amdahl introduce Amdahl's Law. Interestingly, it has no equations and only a single figure. For this issue of the SSCS News, Dr. Amdahl agreed to redraw the figure. In the available hard copy it was illegible.

[18] J.L. Gustafson in his book of Reevaluating Amdahl’s Law,” have "broken" the Amdahl's Law and to have justified massively parallel processing. An alternative formulation was proposed. This is often referred to as the Gustafson's Law [5] and has been widely refereed to as a "scaled speedup measure". In Gustafson's formulation, a new serial percentage is defined in reference to the overall processing time using P processors. Therefore it is dependent on P. This P dependent serial percentage is easier to obtain than that in Amdahl's formulation via computational experiments. But mathematically, Gustafson's formulation cannot be directly used to observe P's impact on speedup since it contains a P dependent variable.

[19] D.S. Milojičić, F. Douglis, Y. Paindaveine, R. Wheeler, and S. Zhou, showed that the Process migration is the act of transferring a process between two machines. It enables dynamic load distribution, fault resilience, eased system administration, and data access locality. Despite these goals and ongoing research efforts, migration has not achieved widespread use. With the increasing deployment of distributed systems in general, and distributed operating systems in particular, process migration is again receiving more attention in both research and product development. As high-performance facilities shift from supercomputers to networks of workstations, and with the ever-increasing role of the World Wide Web, they expect migration to play a more important role and eventually to be widely adopted. This survey reviews the field of process migration by summarizing the key concepts and giving an overview of the most important implementations. Design and implementation issues of process migration are analyzed in general, and then revisited for each of the case studies described: MOSIX, Sprite, Mach, and Load Sharing Facility. The benefits and drawbacks of process migration depend on the details of implementation and, therefore, this paper focuses on practical matters. This survey was help in understanding the potentials of process migration and why it has not caught on.

[20] On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing by I. Foster and A. Lamnitchi, have taken a step toward comparing and contrasting P2P and Grid computing. Basing their discussion whenever possible on the characteristics of deployed systems, rather than the unverified claims abundant in the literature, they review their target communities, resources, scale, applications, and technologies. On the basis of this review, he drew some initial conclusions concerning their interrelationship and future evolution. In brief, they argue that (1) both are concerned with the same general problem, namely, the organization of resource sharing within virtual communities; (2) both take the same general approach to solving this problem, namely the creation of overlay structures that coexist with, but need not correspond in structure to, underlying organizational structures; (3) each has made genuine technical advances, but each also has in current instantiations, crucial limitations, which we characterize (simplistically, but, we believe, usefully) as. Grid computing addresses infrastructure but not yet failure, whereas P2P addresses failure but not yet infrastructure; and (4) the complementary nature of the strengths and weaknesses of the two approaches suggests that the interests of the two communities are likely to grow closer over time.

[21] V. Donaldson, in his journal of “Program Speedup in a Heterogeneous Computing Network,” describe network and program models for heterogeneous networks, define notions of speedup and superliner speedup, and observe that speedup consists of both heterogeneous and parallel components. They also consider the case of linear tasks, give a lower bound for the speedup, and show that there is no theoretical upper limit on heterogeneous speedup. 1 Introduction Program speedup is a widely used measure of the performance of an algorithm on a multiprocessor or multicomputer. Although there are differing notions of the definition of speedup [7], speedup measurements are still almost universally quoted as proof of system efficiency.

[22] In “Technological Revolutions, Paradigm Shifts and Socio-Institutional Change,” C. Perez, puts forth a particular interpretation of the long wave phenomenon, which offers to provide criteria for guiding social creativity in times such as the present. Basically, the present period is defined as one of transition between two distinct technological styles -or techno-economic paradigms- and at the same time as the period of construction of a new mode of growth. Such construction would imply a process of deep, though gradual, change in ideas, behaviors, organizations and institutions, strongly related to the nature of the wave of technical change involved.

[23] L. Erlanger, “Distributed Computing: An Introduc-tion”, in his book states that the Current map generalization for massive spatial data is becoming more and more urgent. It is necessary to study the integration of new technologies, such as distributed computing, which is important to provide high-quality geographic information services. The research proposed an architecture of automated map generalization in distributed environments, and implements a prototype system to demonstrate the architecture. Finally an experiment to generalize contour data for entire China from scale 1:250,000 to 1:1,000,000 was performed with prototype system.

3. Design and implementation of Brute Force Grid

3.1 Major components of the project

P2P file sharing is a major network application which rules the traffic concentration of today's internet. Our research is to use the P2P network efficiently to build a process sharing via parallel processing on top of this existing P2P concept, to enable the single user get the huge processing power from the other peers connected to the network.

The project, Brute Force Grid is a result of demonstration prototype of our research via brute forces the hashes in a distributed manner.

- Peer application
    - This makes the host work as the peer of the brute force grid overlay Peer to Peer network.
- Web Server
    - This provides the service of initial network setup and addresses some other security issues for real time plug-in exchange.
- Application Plug-in
    - The actual application process which is used to do the parallel processing of. That is, this application is the one which is parallel processed.

The detail specification of the inner functions of these systems will be shown and discussed in the coming sections of this paper. Now let's move on to see how these components work together to form a complete Brute Force Grid system in the distributed network.

3.2 How the components work together

3.2.1 Peer Application and Web Server issues

Peer application is installed in the host and web server is installed in the public network within any peer or a separate host. The application plug-in is installed inside the peer application so that it is invoked using the peer application and it's under the control of it.

The Peer applications send the web server periodic probes to say that the specific peer is up and ready. So when the new peer is started. That is to run the peer application in one of the network host in the public network (or we can do this in the LAN environment too) the peer first send its existence to the web server. And this newly upped peer check its local IP list. This is the List which consists of the IP addresses of the remote peers that the local peer has previously connected to. These modules were discussed more specifically in the rest of the paper.

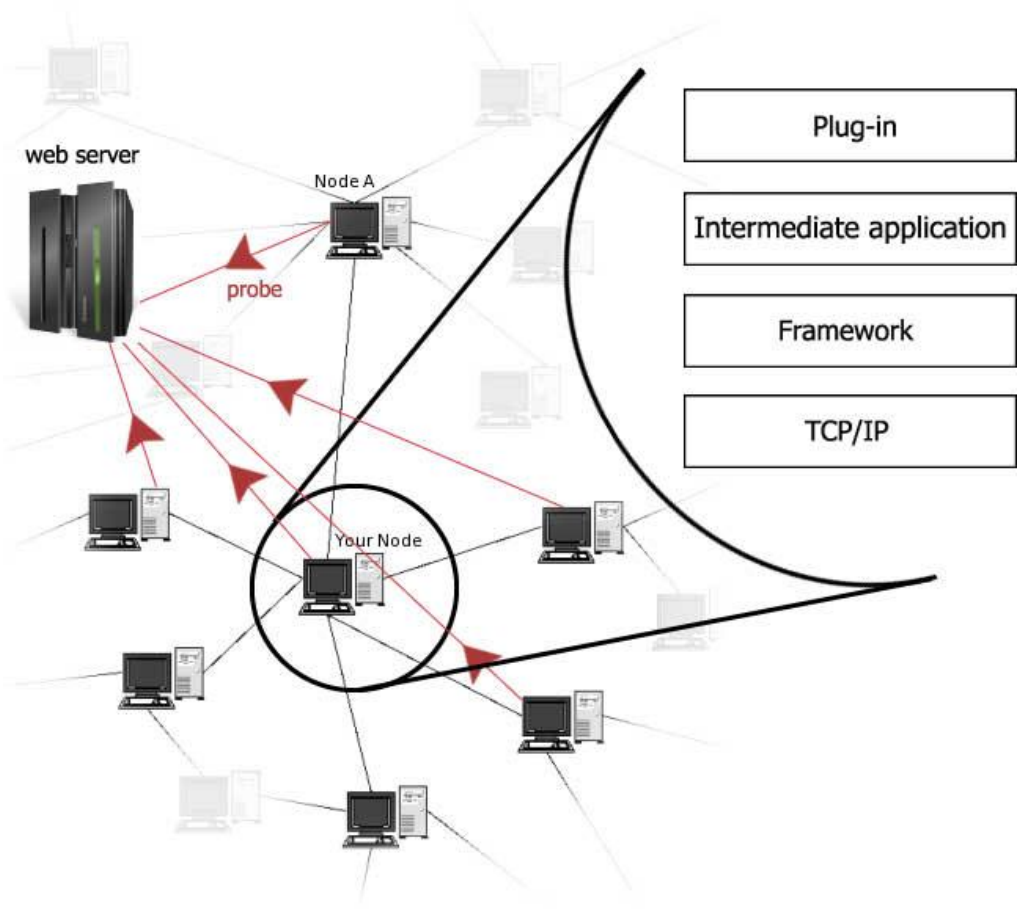

Figure 3.1 – overview of the component work structure

But in case of the peer application has started it's execution for the first time in that peer, there exists a problem. A problem of unknowing the IP addresses of the any of the peers in the overlay network. As if the new upped peer doesn't know any of the IP addresses of the existence network peer then it cannot be connected to the overlay network of the brute force grid. To address this problem, we have introduced the web server as the centralized body to enable the connection.

As each of the peers send the probes to the server telling that peer is in a connectable state; when a new peer is connected, it can get the other peers IP addresses from the web server and create connections with those peers and make the overlay network connection with them.

3.2.2 Problem of instability in the centralized systems

There is a problem of stability when the system is centralized. When the system gets centralized, and if the central body fails, the whole system has to be failed. So we were expecting to make the system as decentralized as much as we can. But to address the problem of initial setup of the network we need a central body. So there is no other way to remove this central web server from the system. This leads us to create additional methodologies to address the problems that might arise when the central web server fails. So without removing the web server from our system we gave solutions for the server fail situations, so that the system doesn't need to get crashed even if the server crashes. These solutions include:

- The backup copy of the last connected peers IP list so that when the next time the peer starts it doesn't need to rely only on the web server to fetch the IP list. Assuming that there would be even at least one peer that is being up since the local peer restarts.

- When a peer gets connected to another peer, each peer exchanges the other connectable peer IPs with each other. So that when a new connection is created by the new peer it request the remote peer, the IP list of the peers it has connected to. The remote peer responses the IP list so that the new peer can also make the connections with the peers which are being connected by the remote peer.

Below image describes the actual steps involved in the above mentioned scenario. Even though this doesn't provide a total solution for the non central web server, it does a very good job when the web server is unreachable. So that we can be sure that the system won't get crashed even though the central web server gets crashed.

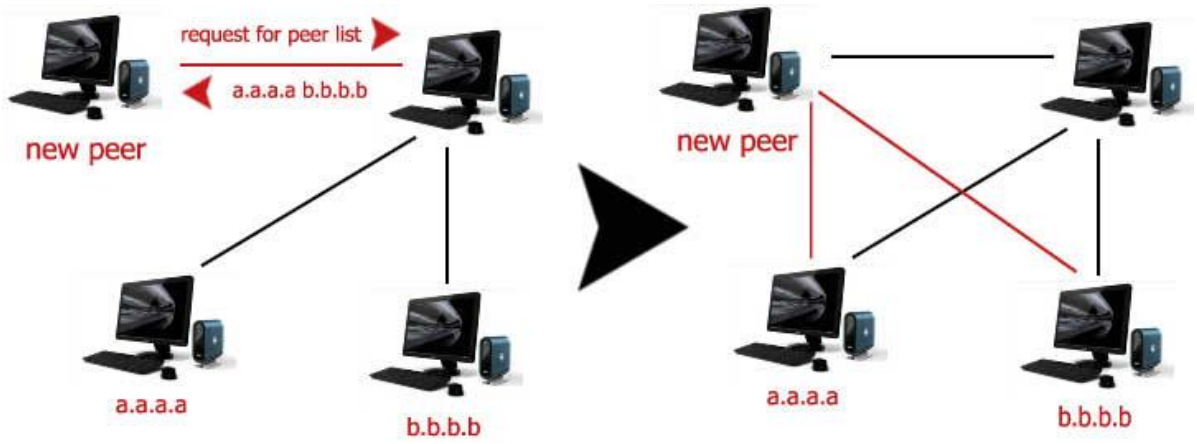

Figure 3.2 – Request peer-list from connected peers

3.2.3 Totally new peer turning up when the web server is down

Using these two mechanisms, we can ensure that even the server doesn't exist in the system, the system won't get stuck, and it can work as normal. The only remaining problem is the situation when the server crashes and there is a totally new peer which doesn't have any old connected IP list with it.

To make this problem solve, we have insert a manual way to tryout the peers. That is to enter the IP addresses manually and try to connect to it. If we know the IP address of even a single peer that is sufficient as when the local peer connects to that remote peer, it can request and gain the IP addresses that has been connected to that remote peer and initiate the connections to those peers.

3.2.4 Inter Peer Probing

The peer which has got turned on doesn't create the connections automatically. This is to utilize the resources available. There is no need of creating a connection when there is no job has to be done. But these peers were probed time to time to determine the number of connectable peers that is in ready on hand when a job is connected. This increase the performance when the peer wants to start a job, as it doesn't need to wait for the searching process for the peers.

3.2.5 Job Initiating by local peer

When a job gets initiated at a local peer, it fragments the jobs into pieces of sub jobs and keeps it in the local job list. While doing this, the peer connects to the connectable peers that were being probed and ensured upped. After connecting the peers, the local peer tries out the each segment to the peer connected. And when the peer accept the job it sets the sub job as assigned to that peer and wait for the response from that remote peer. The remote peers connected can be distributed the jobs simultaneously using threads and these threads waits till the result is received to the local peer from that remote peer.

There can be several responses after a sub job is assigned to a remote peer. According to the responses return from the remote peer, the local peer manipulates the local sub jobs. These are the responses which can be returned by the remote peer:

- Remote peer response the local peer telling that the job cannot be accepted. The local peer put the sub job to the unassigned state and assigns the sub job to another remote peer connected.

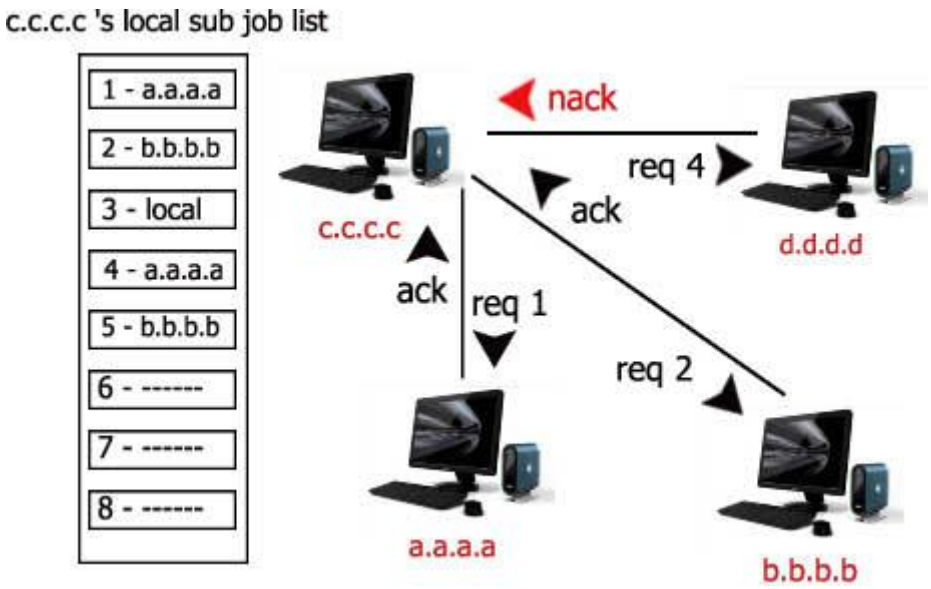

Figure 3.3 – Assigning sub jobs

> *"We assign two jobs per peer at a time. When one finishes another is sent to the remote peer and queued. This is to increase the performance by not keeping the remote peer to wait till next sub job is assigned to that peer. This is further discussed later. The local peer assigns a sub job to itself and processes it locally."*

- Remote peer response with the finish state, the local peer sets the local sub job to the finish sate and this job is not assigned to another peer again.

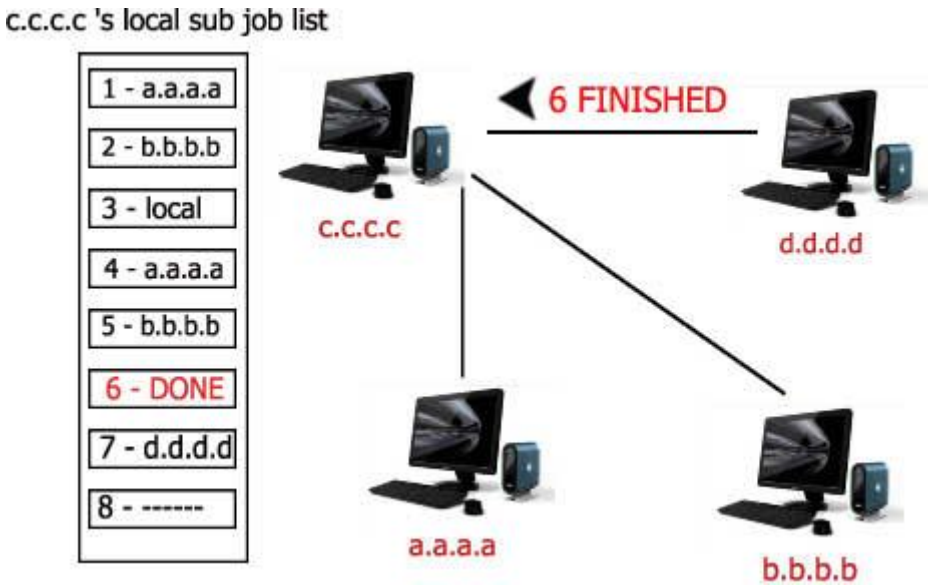

Figure 3.4 – Finish signal from the remote peer

- Remote host returns the response with a success and the recovered password key with it. At this time, the local peer informs all the other peers connected to stop the jobs provided by the local peer as the job already completed successfully. And the local peer removes all the connected peers and clears the local job list.

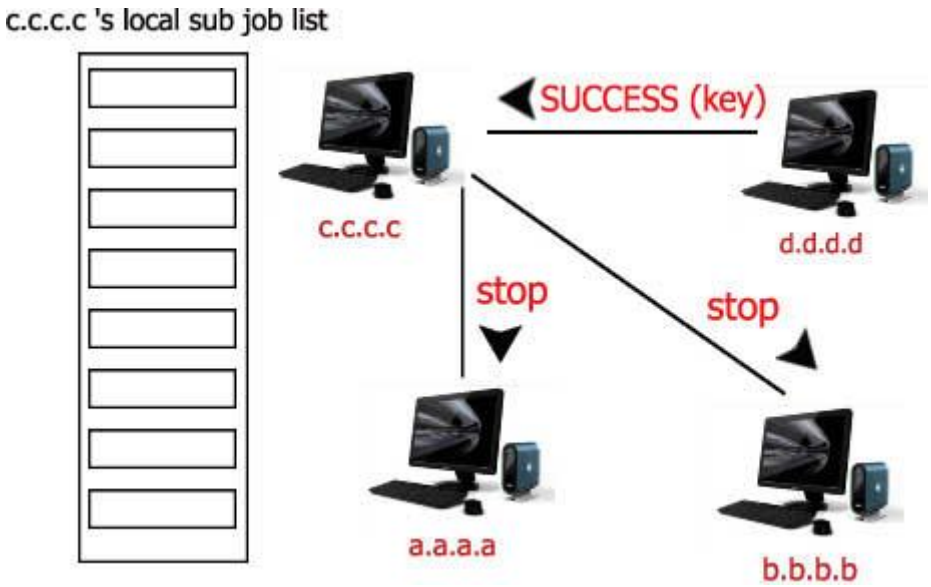

Figure 3.5 – Success signal from the remote peer

An initiated job can be paused or stopped at any time by the local peer. We have given the local peer the full ability to do this and inform other connected peers to stop the specific sub jobs from this local peer.

If the acknowledgment or negative acknowledgment is not received by the remote peer the local sub job is reset and considered as a new sub job and assigned to another peer.

The connectivity between the peers is available only if the job is distributed between them. If not the connection gets terminated by the peer.

3.2.6 Remote Job received scenario

Now let's look at the same scenario from the remote peer's perspective. A remote peer is now considered as the local peer. The local peer receives a connection establishment by the remote peer. Then the remote connection is accepted and the connected peer count is incremented by one. Then the remote peer sends the local peer a sub job to execute. The local peer checks the resources and determines whether to accept or decline the request by the remote peer.

This check can be done in various methods. As our project doesn't include this inside the scope, we have kept the interface of this to be developed further by any interested parties.

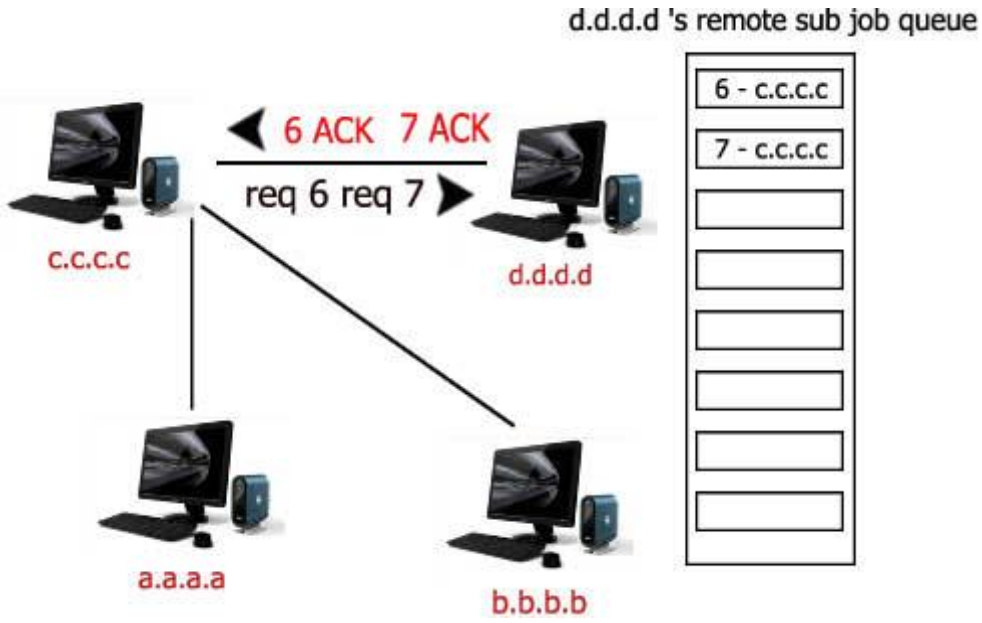

Figure 3.6 – Receives a new remote sub job

When the determination is confirmed the peer does two things. If accepted the peer acknowledge the remote peer or else is sends a negative acknowledgment saying that it cannot accept any sub jobs at this time.

The connected peer is always checked for the connectivity by sending probes to it. So when a probe sending fails, the remote peer is considered as the drop connected peer and the local resources are freed along with the remote job list's entries made by that peer.

So these create a situation where the started process acknowledged by the remote peer doesn't reply with a finish or a success. This sort of sub jobs were then reset as motioned before.

3.2.7 Automated plug-in transfer

The plug-ins was used to execute the sub jobs in a peer. So when a peer initiates the process of a specific job that specific peer should have that plug-in within hand. That is for example. If a peer wants to initiate a brute force attack for MD5 hashing algorithm then that local peer should have the MD5 plug-in to work out the sub jobs created by that job.

Apart from this all the peers that is going to execute the sub job should have that specific plug-in. To do this the remote peers should have a mechanism to get the plug-ins from a central location, which is something like a web server.

But as mentioned before such an activity might make the system to be centralized. So we planned to get that plug-in from the peer which initiates the job. But there exists a security issue. That is, the peer which initiates the process can make the plug-in like viruses and send them to other peer via this application.

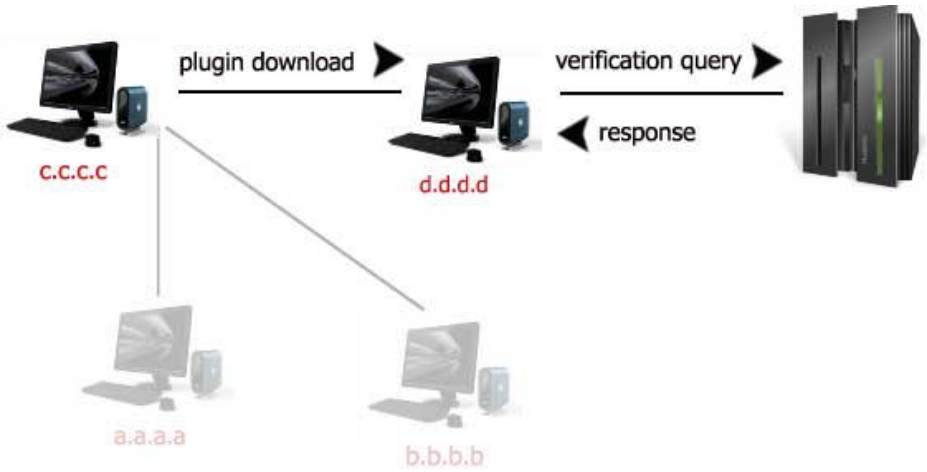

Figure 3.7 – Plug-in verification from the web server

So we include a security mechanism, which is all the plug-ins registered and certified by the central body, and the central body, that is the web server has the hash of each of this plug-in file. Before executing the downloaded plug-in from the peer, the peer which downloaded the plug-in check the server for the hash of that specific plug-in and confirm that the file is certified by the web server.

But again if the web server has crashed, the peer shows the user a warning message telling that the plug-in downloaded may have viruses and to use on our own risk. So at this point the user has the ability to decline the request along with deleting the remotely downloaded plug-in if he or she thinks that the plug-in may contain malicious codes inside.

3.3 Peer application

Peer application is the software piece which runs on the host that makes that specific host, the peer to the overlay Peer to Peer network of the brute force Grid.

Peer application is a set of modules working together to produce the protocol a working object. These modules are further divided into three categories.

- Data Collections
- Working Components
- GUI

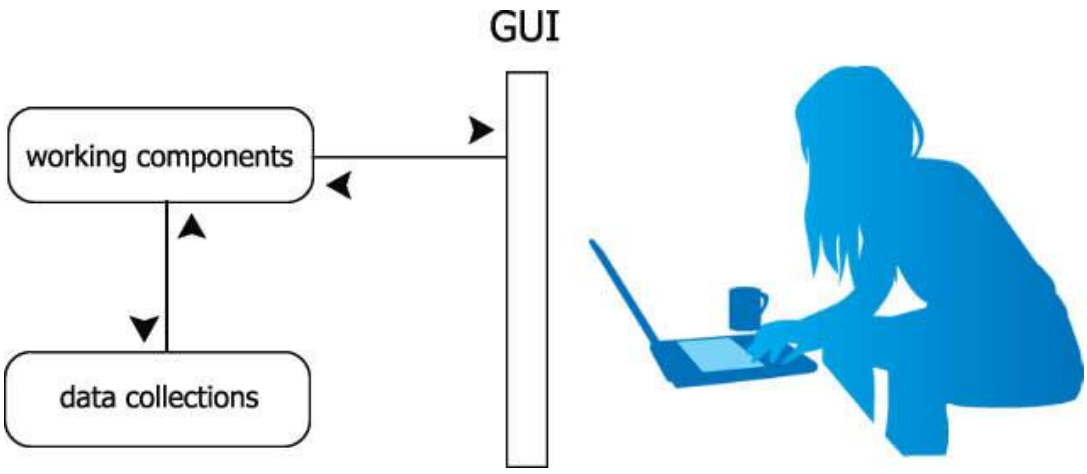

Figure 3.8 – the major components work together

3.3.1 Data Collections

Data collection is the location to store and manipulate the variables and other data structured need to execute the Peer Application. The data collection Module has several sub modules.

- **Available Decryptor List**

  This sub module consists of the plug-in list that is stored in the plug-in folder of that specific host. The check is done periodically after a specific time interval. This enables the detection of any new plug-ins installed on run time of the application.

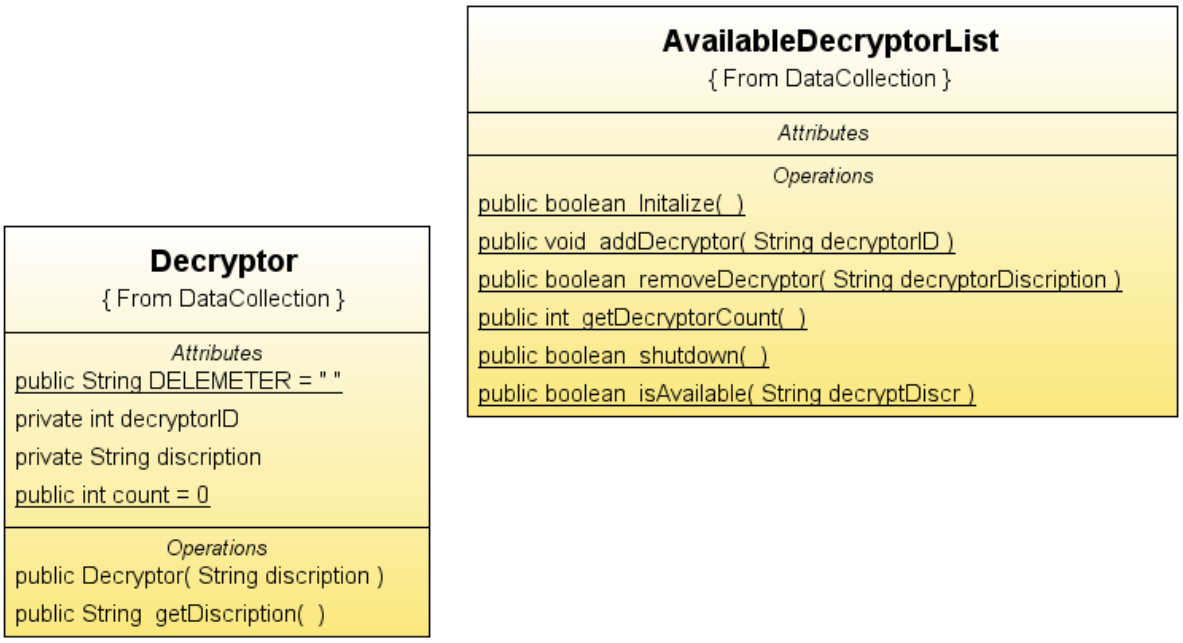

Figure 3.9 – Class Diagram of Available Decryptor List and Decryptor

- **Common Flags**

  Consists of the flags and other variables that is used to signal the inter thread or inter service modules on various occasions.

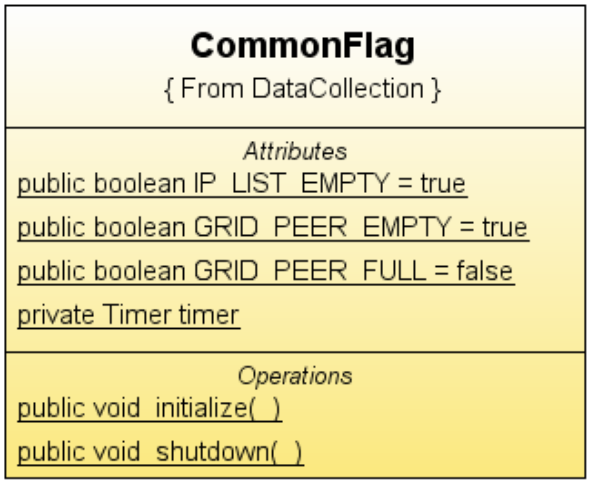

Figure 3.10 – Class Diagram of Common Flags

- **Common Register**

Common Register is the main sub module which contains all the configuration variables of the application. This includes the entries found in the settings GUI interface which is shown before. The variables are stored in the configuration file when the application closes, and restore from that file when the application restarts.

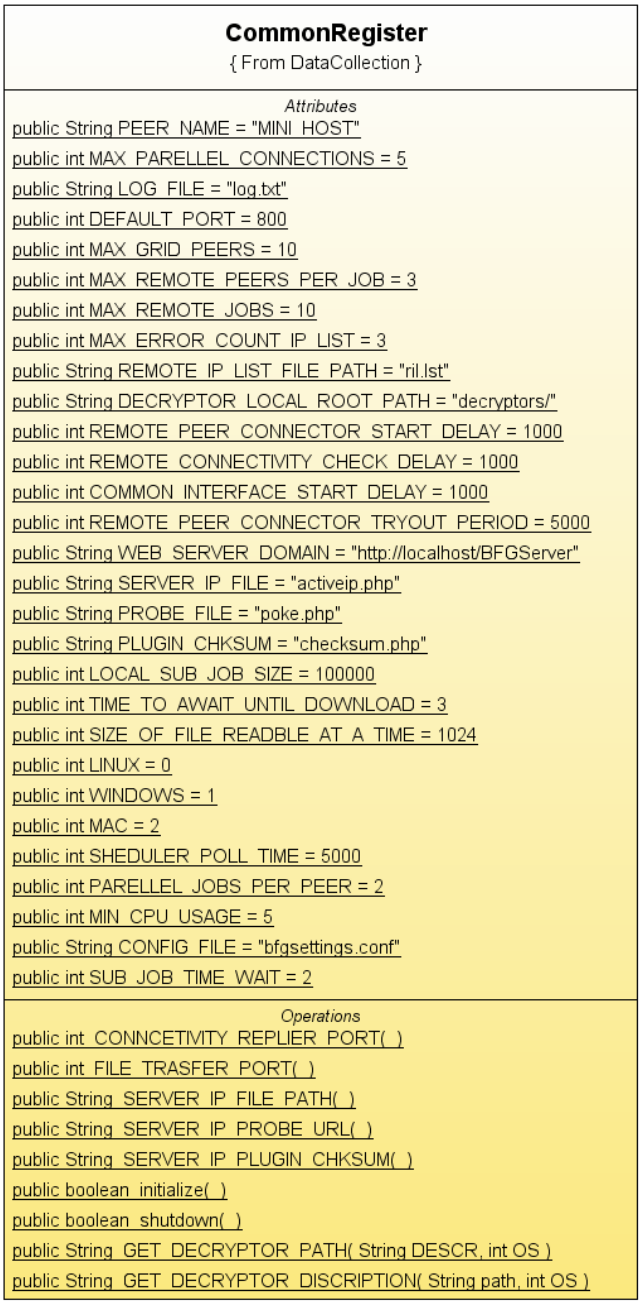

Figure 3.11 – Class Diagram of Common Register

- **Grid Peer List**

This contains the Peers information which is connected or connectable. Remote Peer Trier and the common Interface fills up this list. And when a peer goes down, the Grid Peer List module is updated.

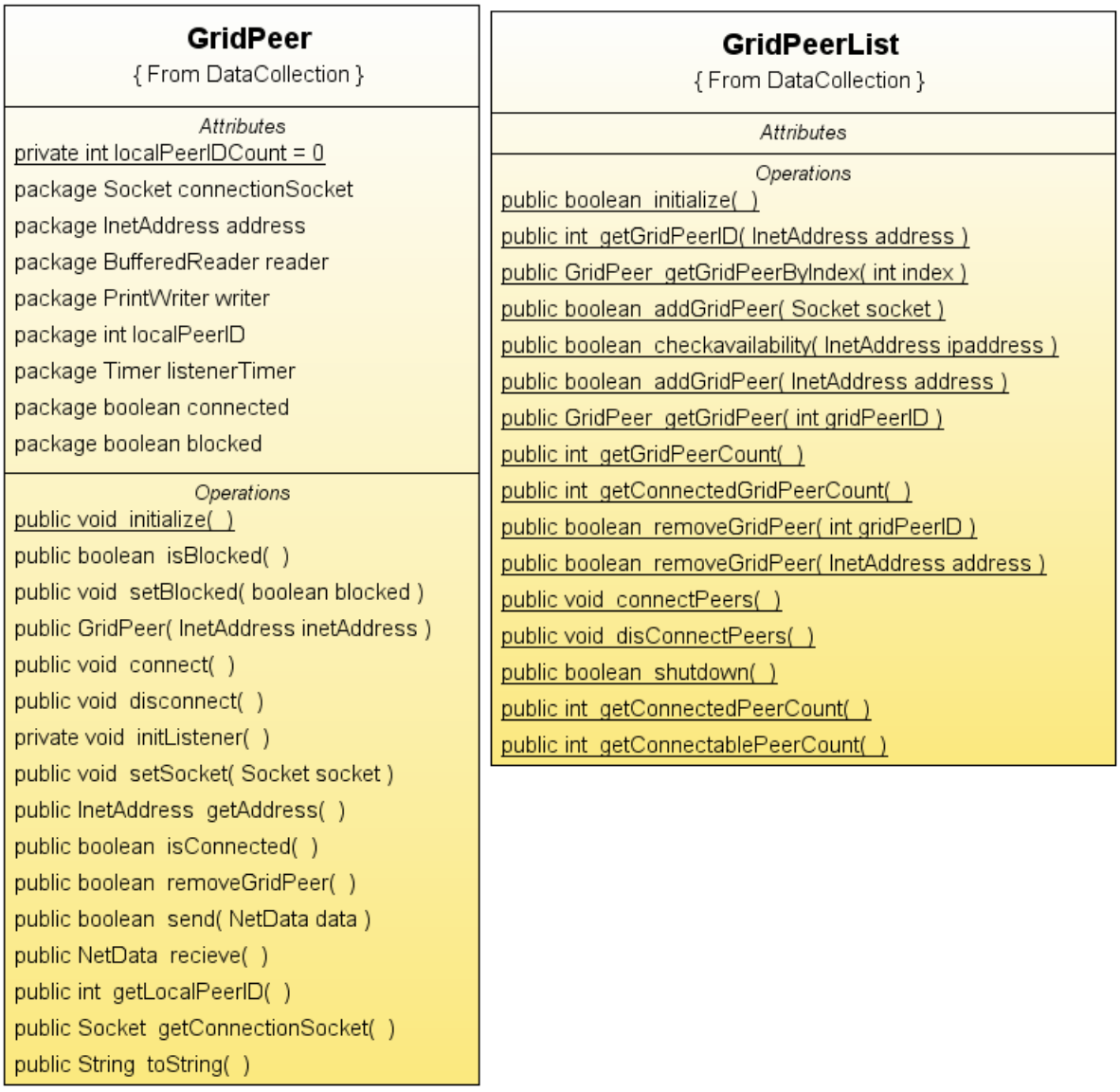

Figure 3.12 – Class Diagram of Grid Peer List and grid Peer.

- **Peer Info**

  This sub module is used to produce the system information to send to other peers, and to use for the system task such as platform detection. This component can be further developed according to the need and new features.

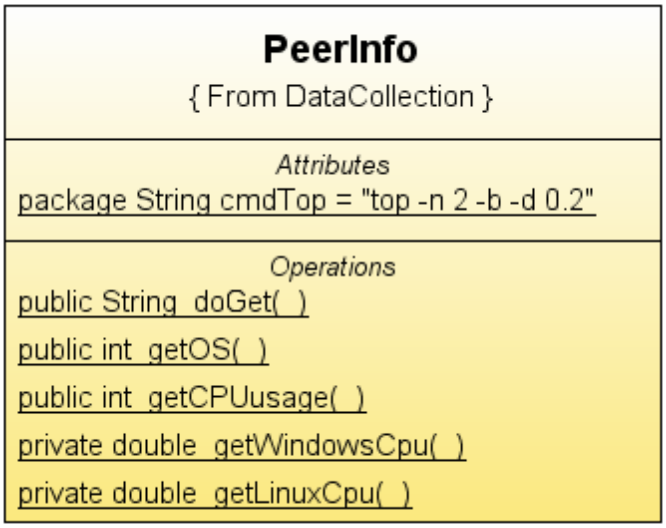

Figure 3.13 – Class Diagram of Peer Info

- **Local Job**

  This is the module used to handle the local job. When a new local job is inserted to this module, it automatically fragment the job into sub jobs using the information provided in the common register. This module also contains a state for each and every sub job created and for the whole job as one. These sates are useful when determining the job statues when the program is running.

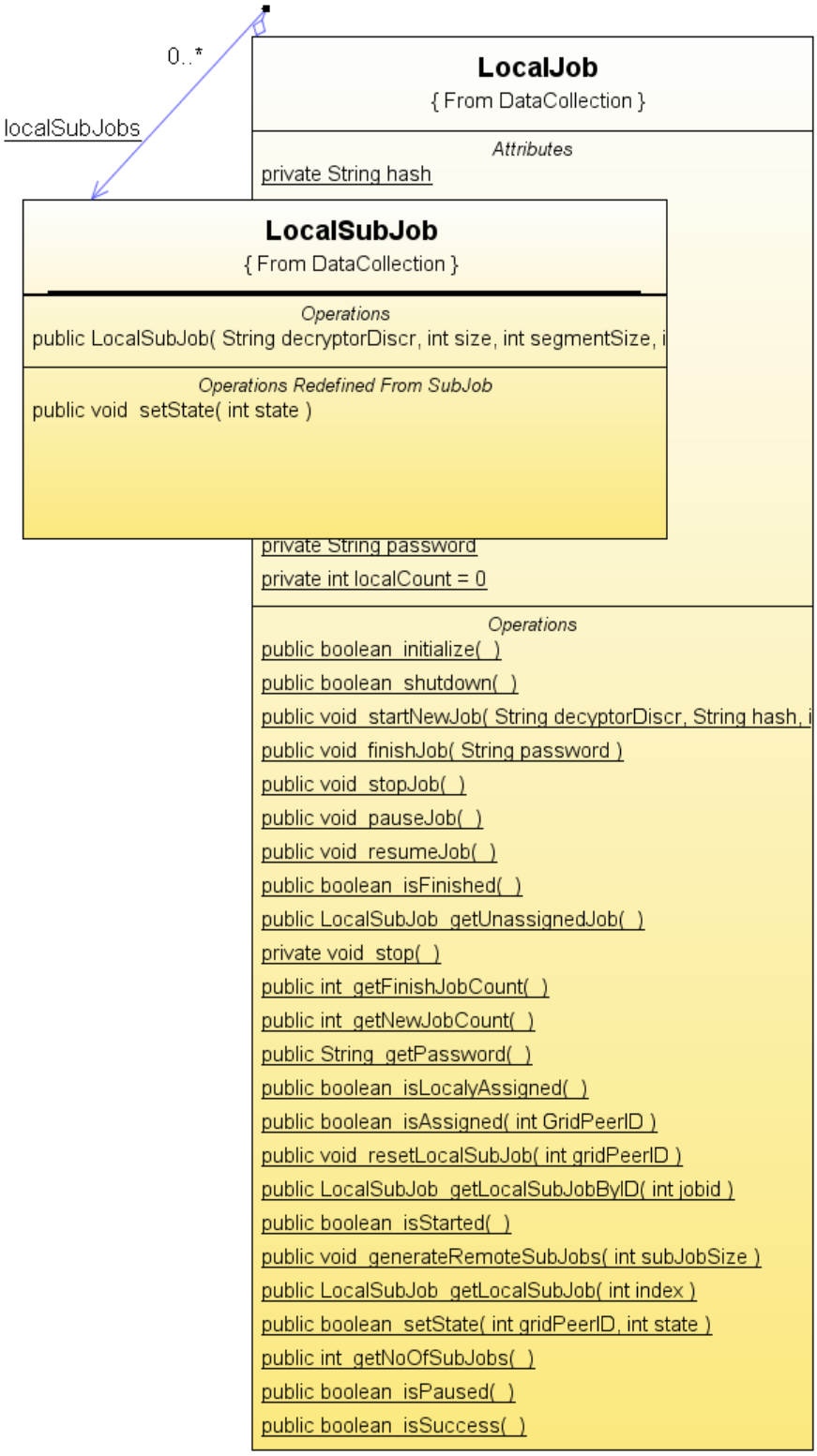

Figure 3.14 – Class Diagram of Local job and local sub job

- **Remote Job List**

This data structure module consists of the sub jobs received by the remote peers. When a remote peer is down, the sub jobs received by that specific peer is removed from this list. The list is decremented by the scheduler, when a sub job executed successfully.

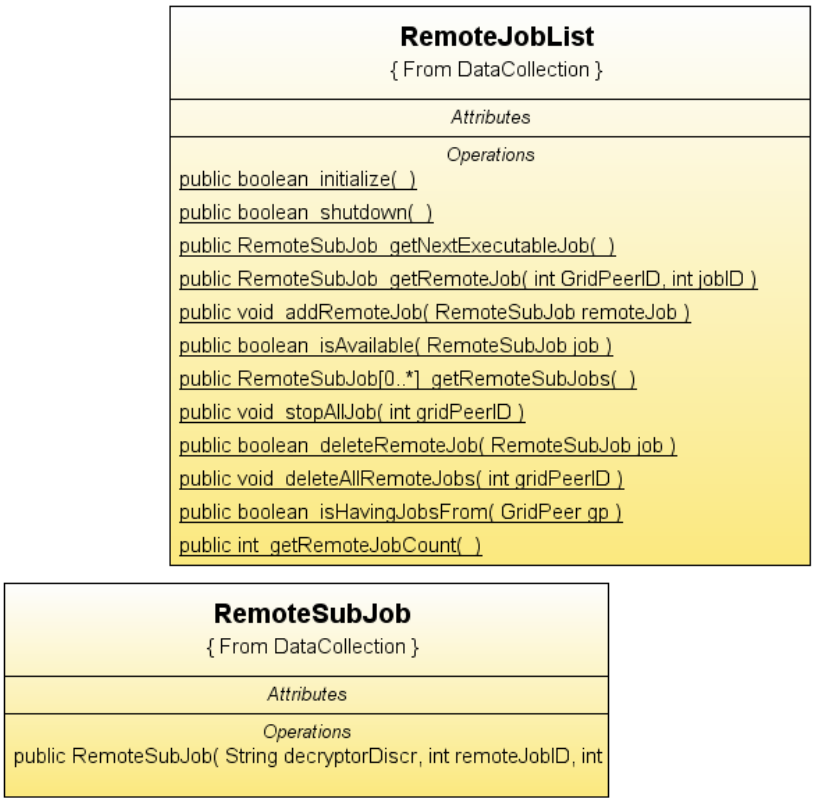

Figure 3.15 – Class Diagram of Remote Job List and Remote Sub Job

- **Net Data**

This data structure module is used to send the information from one peer to another. This module defines the way of signal and information transmission from one peer to another. We can create an object of this module; feed it with the information to transmit, and transmit to the peer. From where the peer can reconstruct the object and retrieve the information received by the peer.

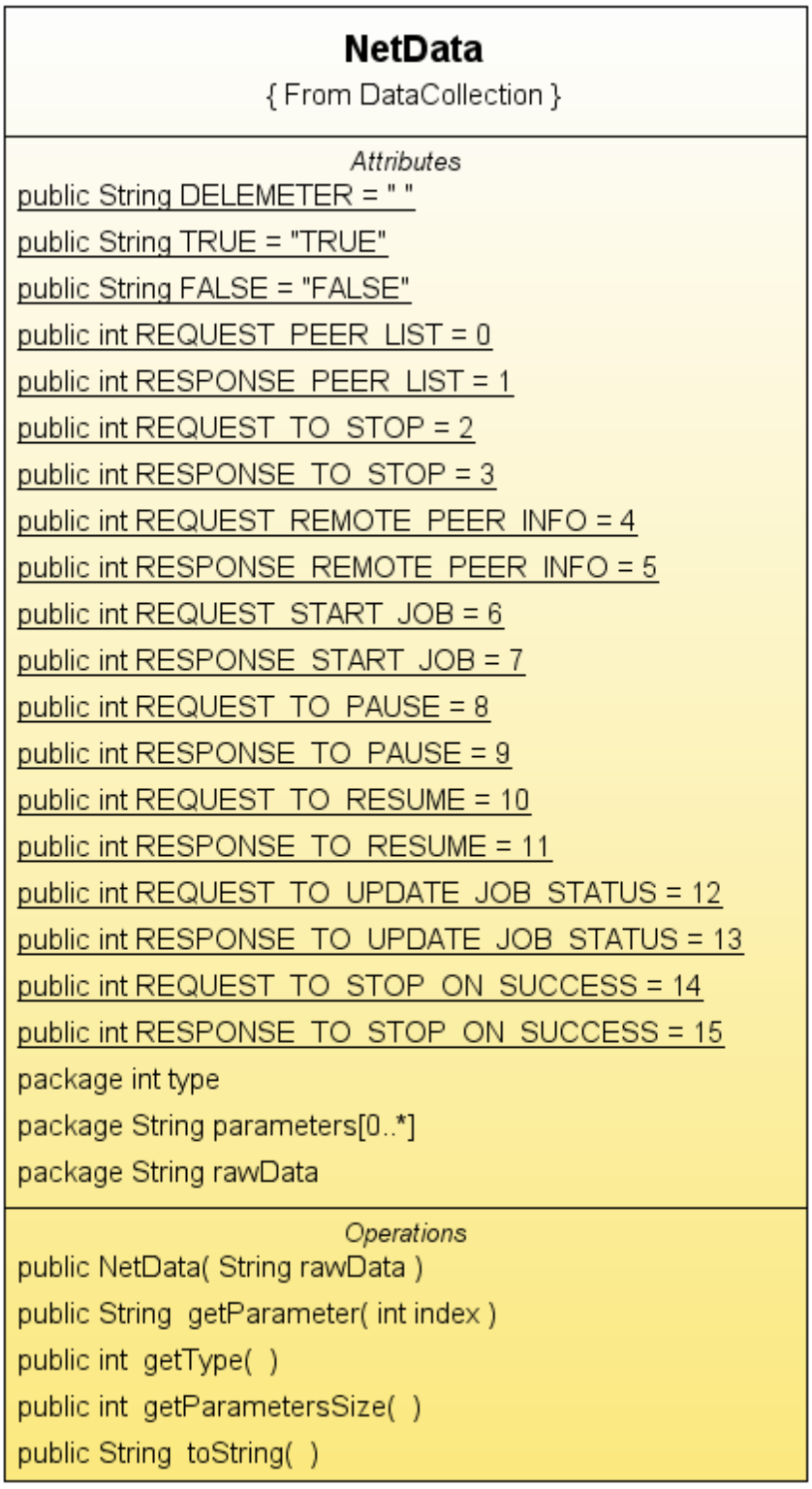

Figure 3.16 – Class Diagram of Net Data

- **Error and Messenger**

These two sub modules are used to notify the use about the events happening in the application. These are the modules used to update the console preview of the GUI.

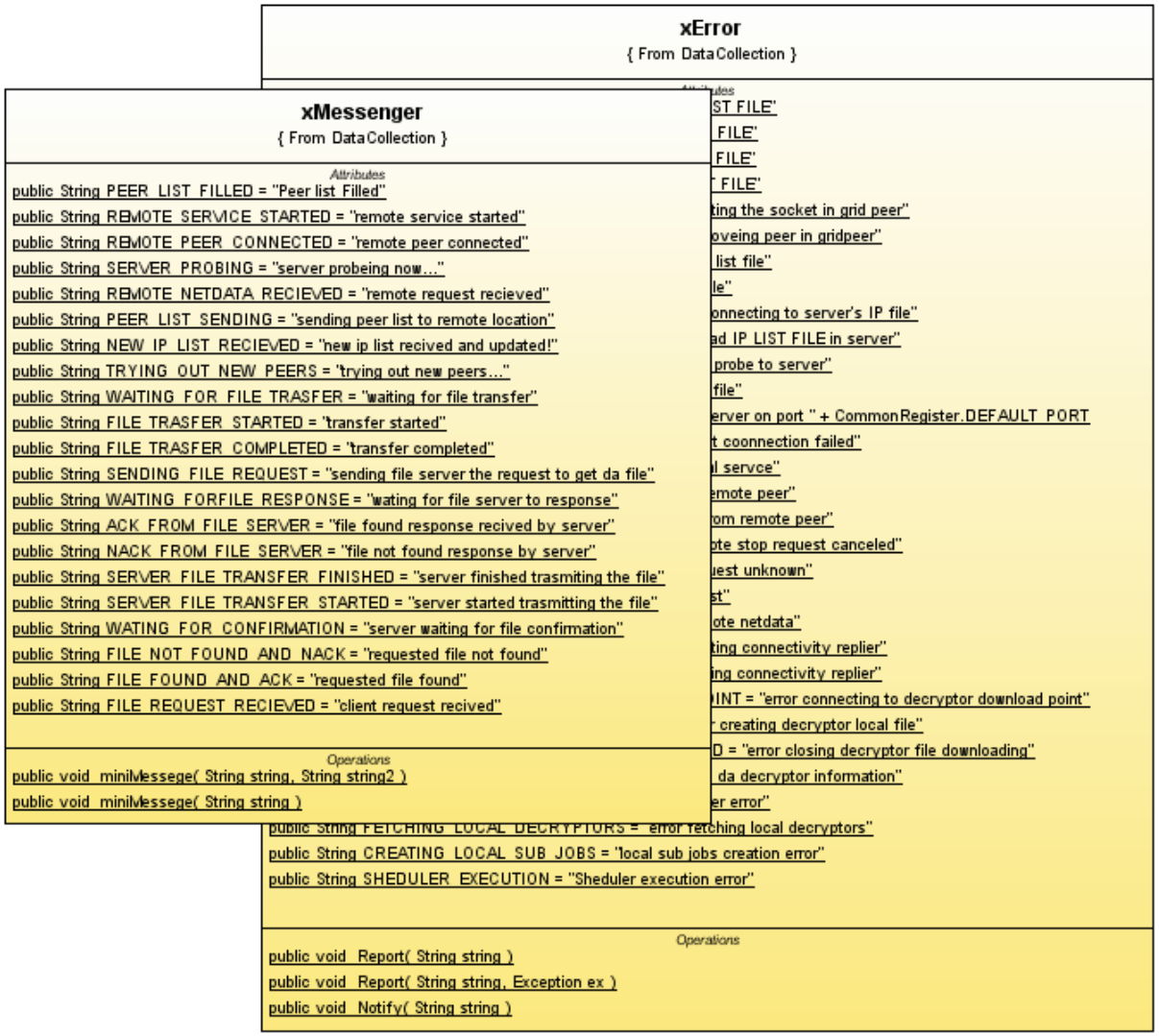

Figure 3.17 – Class Diagram of xMessenger and xError

3.3.2 Working components

This consists of the sub modules or services that executes while the program is running. These Modules use the data components to store and retrieve information.

- **Benchmark**

  This module is used to determine the statues of the peer. That is to accept the request from the remote peer or not. This can be based on many factors. We haven't implemented this part in this prototype. But we created free interface so that anyone can used to develop it later.

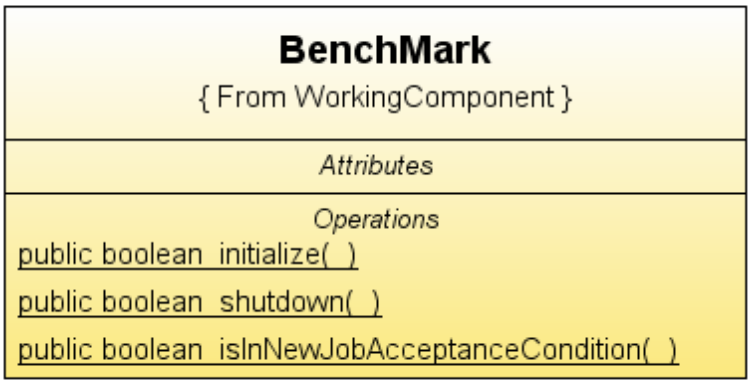

Figure 3.18 – Class Diagram of Benchmark

- **Common Interface**

  This is the listening server components which creates the remote connections and add to the grid peer list. This is the open interface to the other peers to establish a connection.

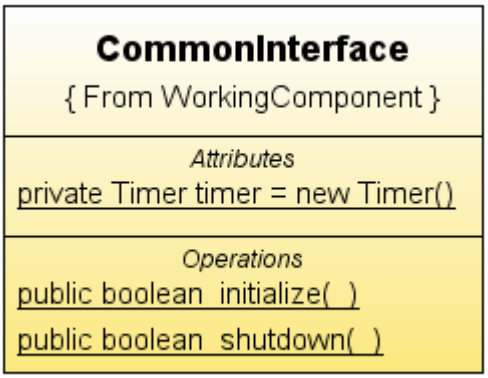

Figure 3.19 – Class Diagram of Common Interface

- **Connectivity Replier**

  When a peer is connected or set as connectable, the other peers send this peer periodic probes to check whether the peer is still in the connected or connectable state. This connectivity replier is the module which response to these probes.

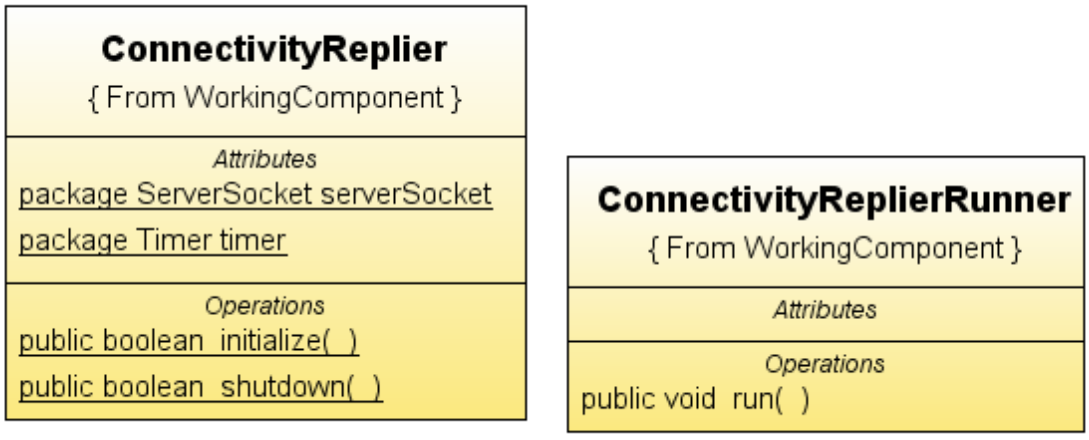

Figure 3.20 – Class Diagram of Connectivity replier

- **Decryptor Downloader**

  Decryptor downloader is used to download the plug-in from the job initiator peer to the local peer as if the plug-in is not available. This Module is responsible to check online on the web server about the plug-ins certification before continuing.

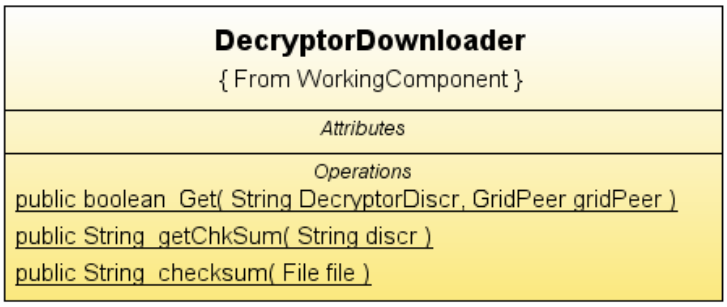

Figure 3.21 – Class Diagram of Decryptor Downloader

- **File Transfer Server**

  FT server is use to response to the Decryptor downloader. This module is used to transmit the plug-in file to other peers which has requested to download the file. FT server uses a separate port other than the default port.

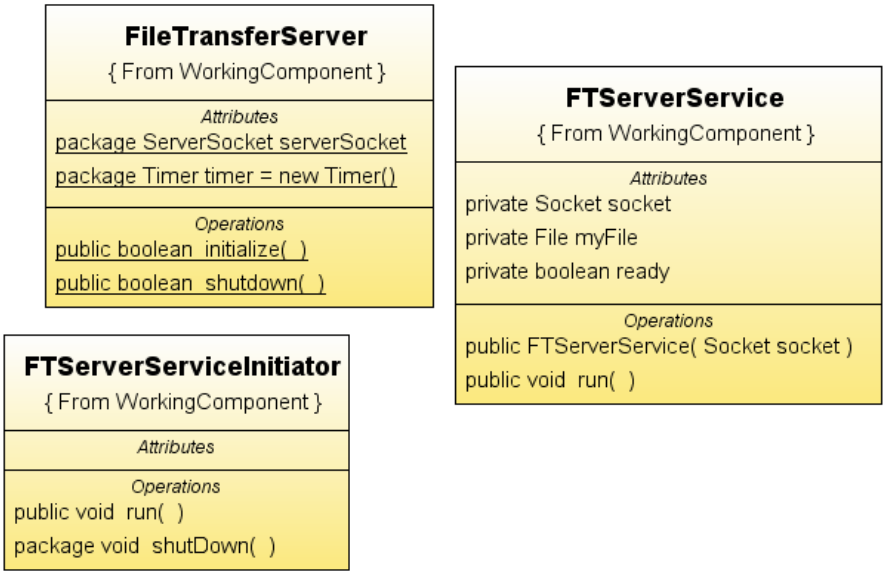

Figure 3.22 – Class Diagram of File Transfer Server

- **Initialize**

  This module is responsible to initialize each and every component available in the Brute Force Grid application.

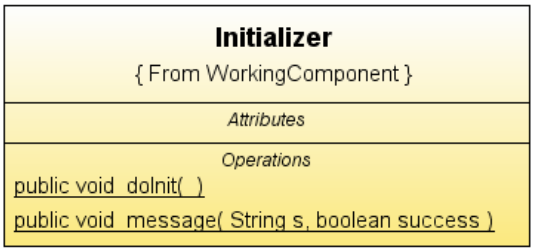

Figure 3.23– Class Diagram of Initializer

- **List Updater**

  List Updater is used to update the list available in the application according to the new peers connecting and peers disconnecting. This is the module responsible to remove the IP list and sub jobs as if the grid peer has got down.

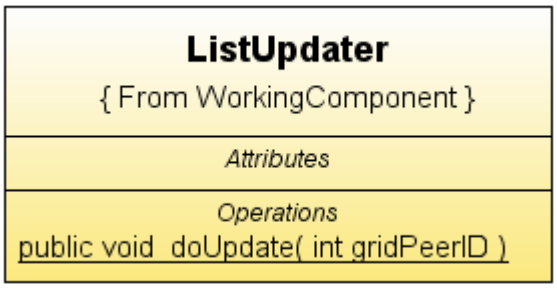

Figure 3.24 – Class Diagram of List Updater

- **Local Job Distributer**

Local Job Distributer is responsible to distribute the local sub jobs to the specific peer connected. This module gets activated when a net local job is put to the local job component. And when the size of the local sub jobs list is zero, this module deactivates.

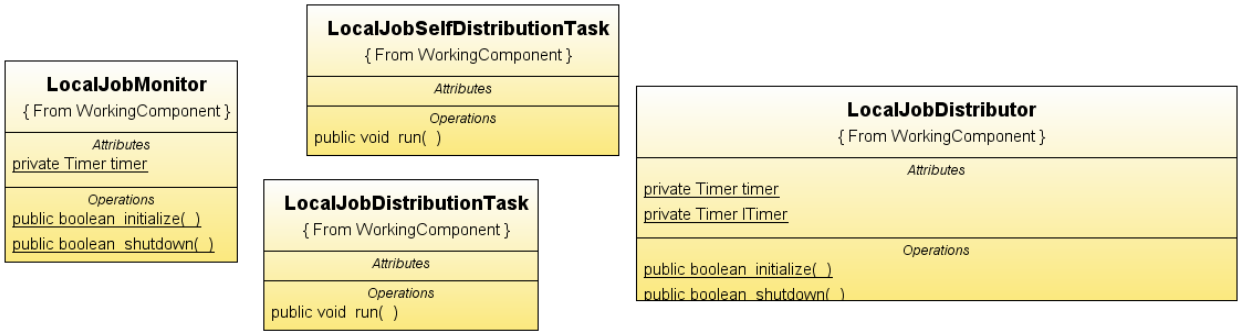

Figure 3.25 – Class Diagram of Local Job Distributer

- **Local Job Monitor**

This Module is responsible for monitor the local jobs. If a particular local job doesn't gets response from the remote peer or gets the failed response, this module is responsible to reset the sub job to the original state where the sub job can be assigned to another peer.

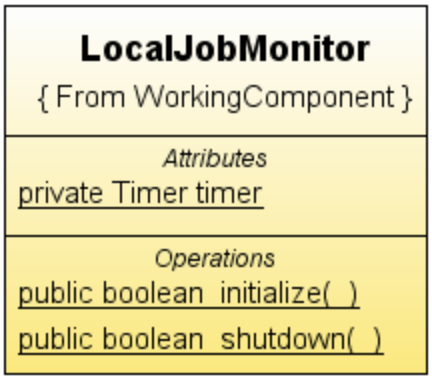

Figure 3.26 – Class Diagram of Local Job Monitor

- **Logger**

Logger is the module which logs the events and errors in the log file. This can be used to determine the progress when an error occurs. Or to trace down the process for any administrative needs.

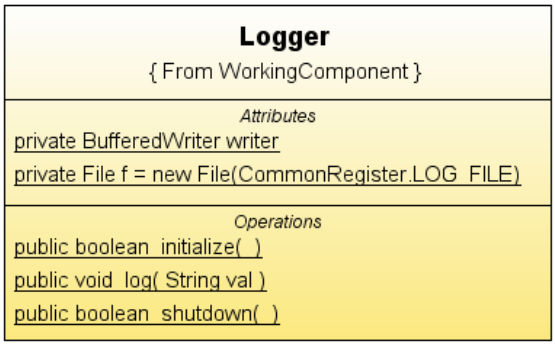

Figure 3.27 – Class Diagram of Logger

- **Net Data handler**

When a data is received by the remote peer to the local peer, this is the module that reconstructs the Net Data object and retrieves the information sent to the local peer. And this module is responsible to initiate any activities related to the request received.

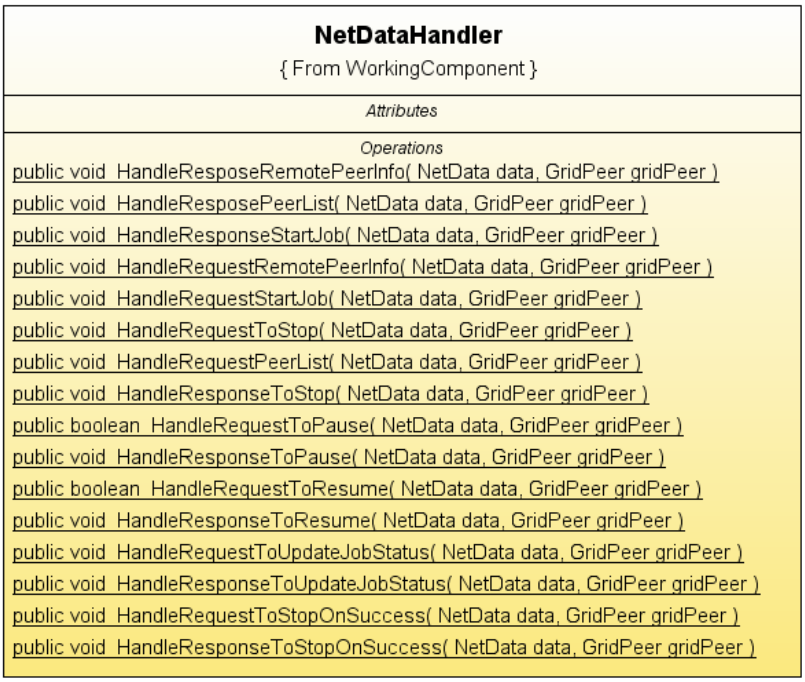

Figure 3.28 – Class Diagram of Net Data Handler

- **Peer Disconnect**

As said before, the connection is existed only if there are any transactions between the two peers. If not the Peer should be disconnected and kept in the connectable state. This should be done after checking the sub job lists as if there is no sub jobs exists before the peer is disconnected. This is done using the Peer Disconnect module.

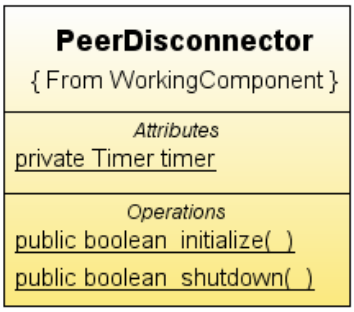

Figure 3.29 – Class Diagram of Peer Disconnect

- **Plug-in Interface**

This module is responsible for the invoking of the C++ plug-in using the JAVA program. That is the peer application. This module is the platform used by the peer application as well as the plug-in to interact with each other.

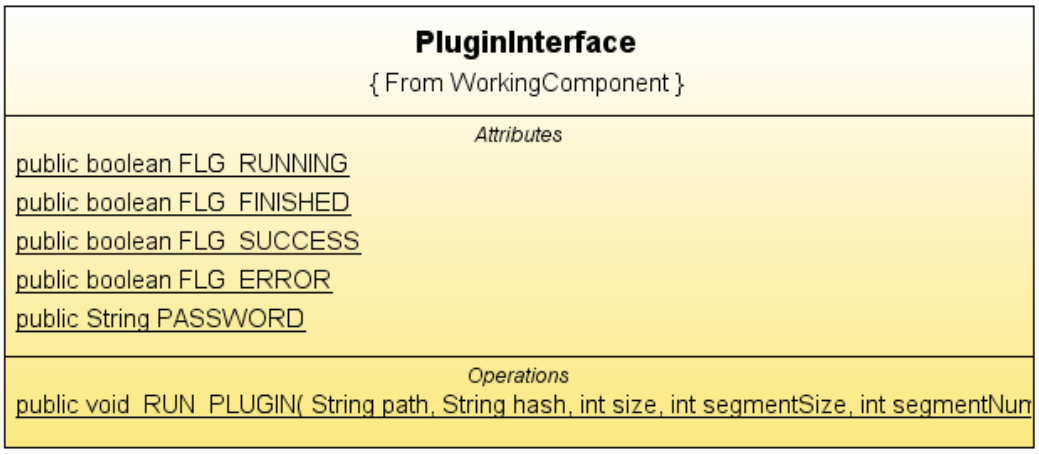

Figure 3.30 – Class Diagram of Plug-in Interface

- **RIL filler**

RIL filler is responsible to fetch the IP addresses connectable from the web server and store it in the Remote IP list data structure module as well as in the backup file.

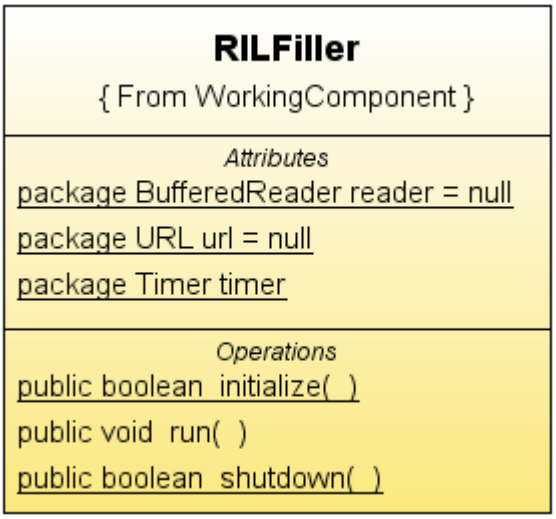

Figure 3.31 – Class Diagram of RIL filler

- **Remote Connectivity Monitor**

This module is used to check the connectivity between the peers. When a peer gets itself inside the grid peer list, this module automatically send probes to that peer and waits for the reply. If the reply did not received, the peer is considers to be down, and the List updater is called.

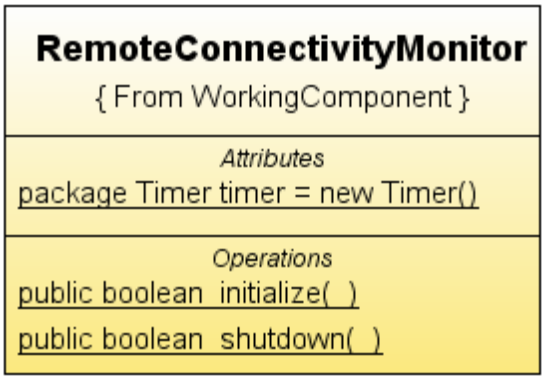

Figure 3.32 – Class Diagram of Remote Connectivity Monitor

- **Remote Listener**

  This sub module waits for the connected peers to send the signals and request to the local peer. And this module is responsible to call the net data handler when the request or some other data received.

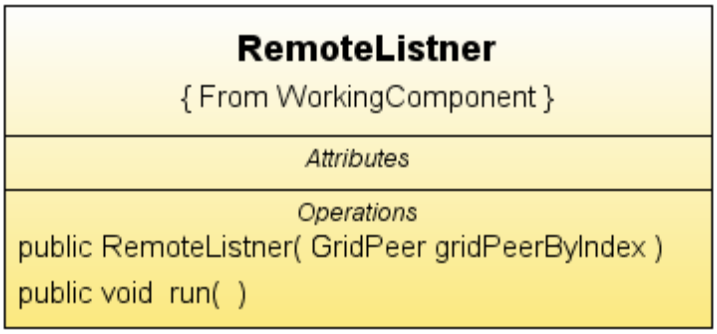

Figure 3.33 – Class Diagram of Remote Listener

- **Remote Peer Trier**

  Remote Peer Trier try-outs the IP address in the Remote IP List sub module and insert the entries to the Grid Peer List module. When trying out peers, the simultaneous try-out were carried out to increase the performance.

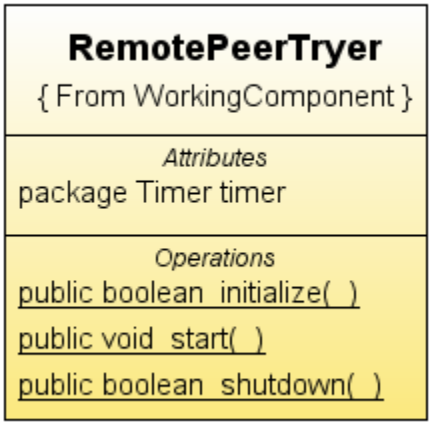

Figure 3.34 – Class Diagram of Remote Peer Trier

- **Remote Requestor**

  This module is used to send signals or request to the remote peer. This module is responsible to show off the whole content of the request send.

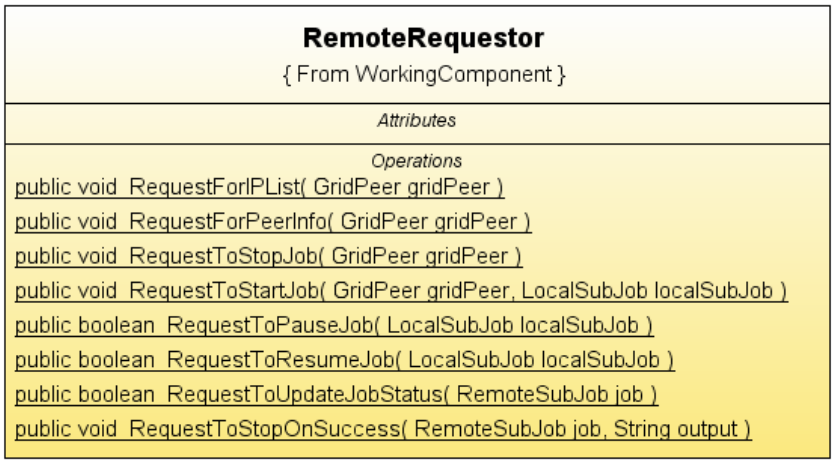

Figure 3.35 – Class Diagram of Remote Requestor

- **Remote Response**

  When a request is received from the remote peer, this is the module used to response to those requests. The response and the request definitions are set I the net data structure, this is used to determine the exact response received.

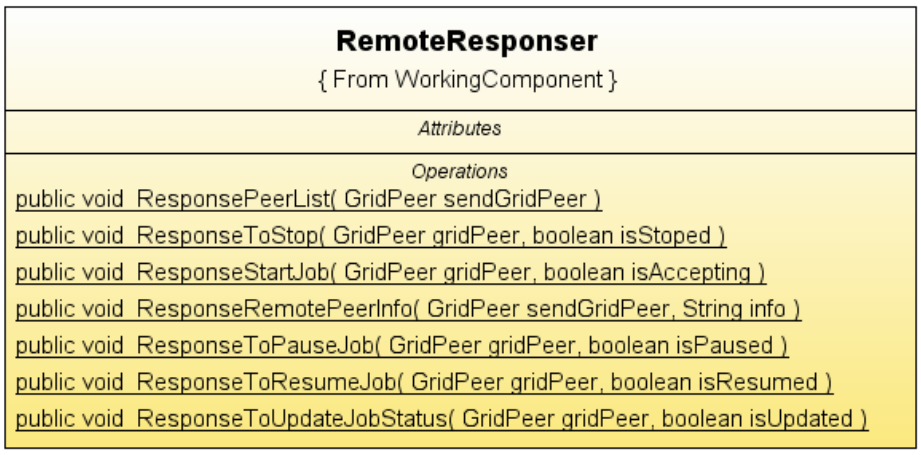

Figure 3.36 – Class Diagram of Remote Response

- **Server Prober**

The local Peer should probe to the web server periodically to inform that the peer is up and ready to be connected by other peers. This work is done by this module. This module connects to the web server and informs the existence.

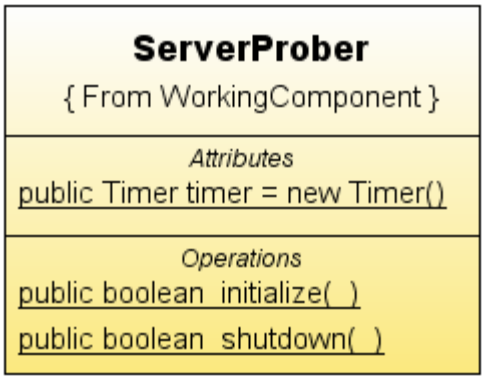

Figure 3.37 – Class Diagram of Server Prober

- **Scheduler**

Scheduler schedules the remote sub jobs one by one to the plug-in interface to execute for the result. When a sub job executed successfully it is removed from the list of the remote sub jobs. If there are no sub jobs to execute, the scheduler is put to sleep until the remote sub job is received.

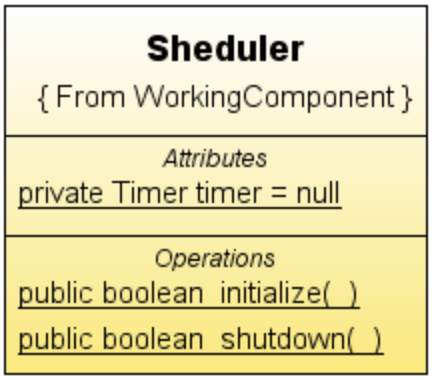

Figure 3.38 – Class Diagram of Scheduler

- **Shut downer**

Shut downer is used to shut down the services and modules initialized by the initialize process. This frees the resources and shut down all the working components and the data collections.

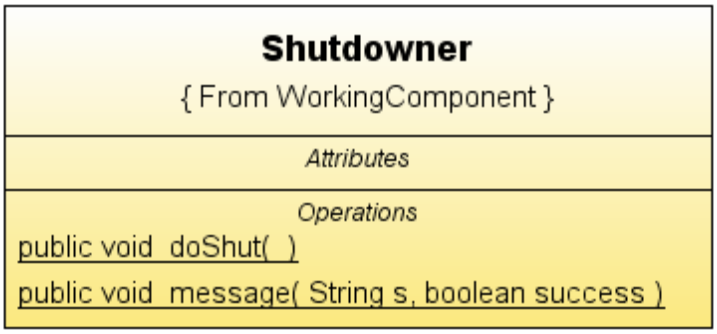

Figure 3.39 – Class Diagram of Shut Downer

3.3.3 GUI

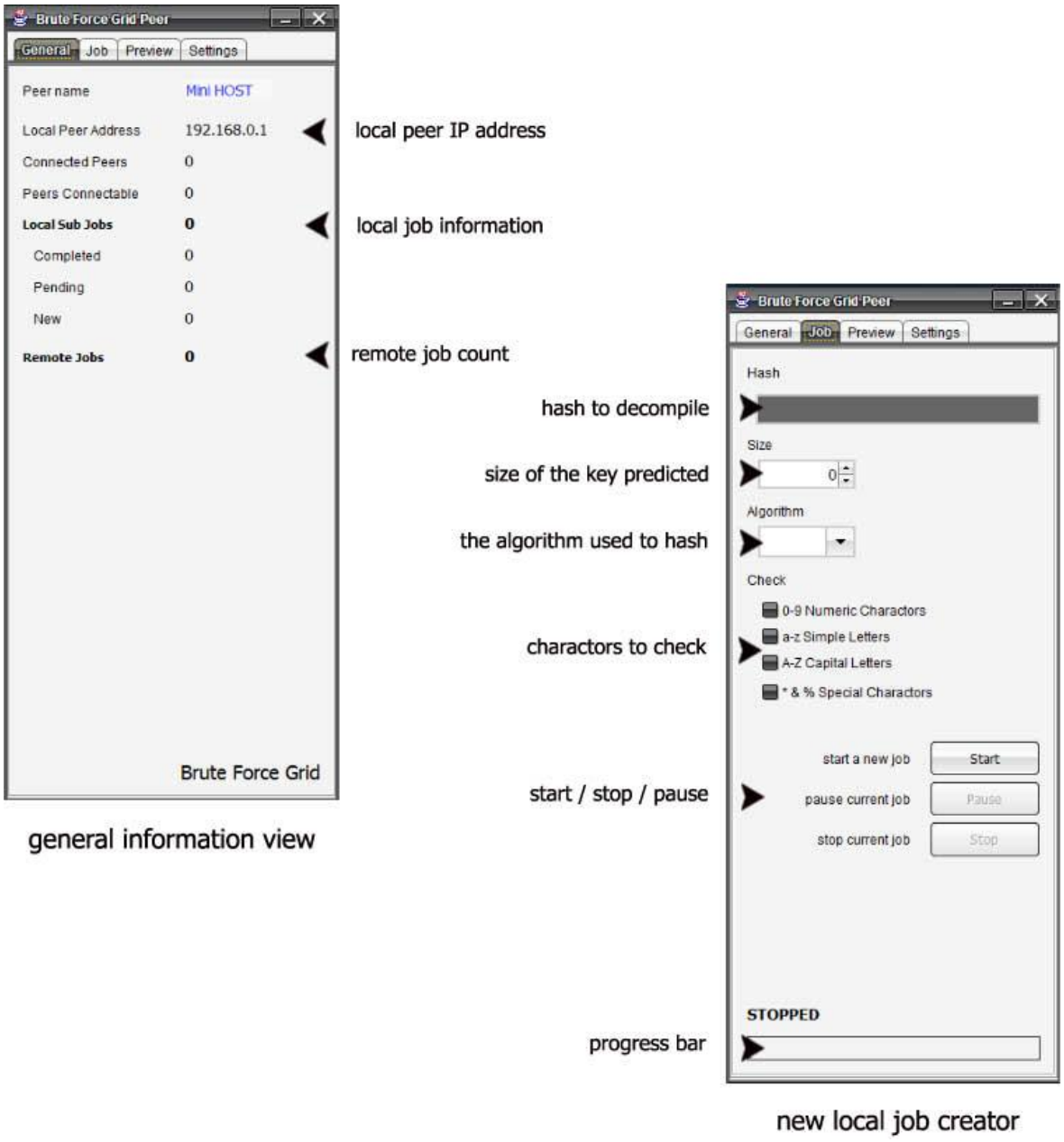

Figure 3.40 – General view and new local job creator

Figure 3.41 – Peers view and sub jobs view

Figure 3.42 – IP list to check and Console preview

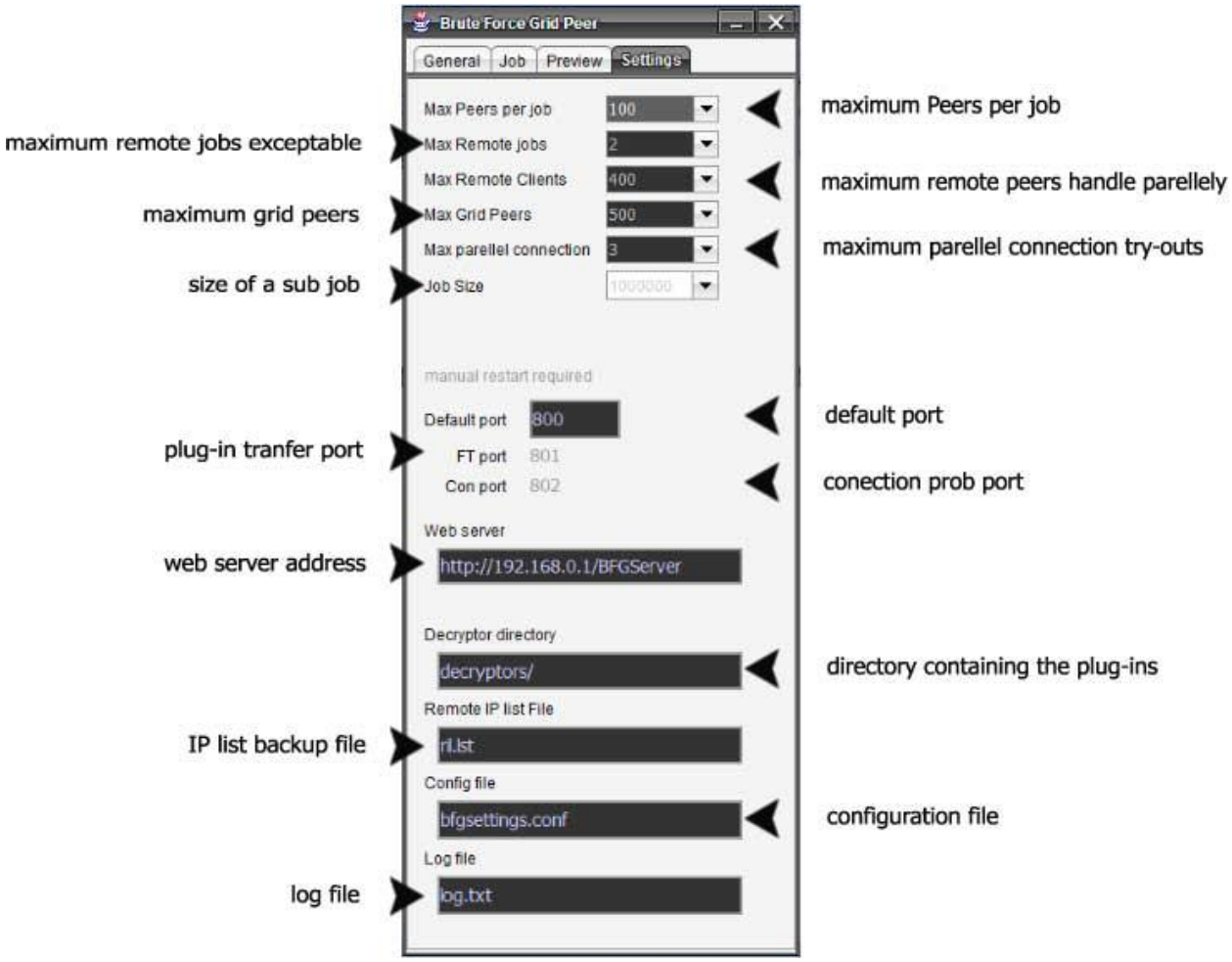

Figure 3.43 – BFG setting center

3.4 Application Plug-in (MD5)

Since brute forcing takes a lot of computations, we decided to have it as the application which is run on to of the basic framework. So the user's task would be to derive the clear text password from a given encrypted hash key.

We can sub-divide the functionality of the plug-in software as follows

- Taking all the user inputs and store in the program.
- Create a data array depending on user inputs.
- Generate all the possible text combinations depending on the map created.
- Generate the hash key by sending it through the hash function.
- Hash comparison and give the final result back to the local host.

3.4.1 Storing the user inputs passed by the GUI to the program.

The user inputs are,

- The four conditional Booleans values,
  - **Symbolic** - consider symbolic values when generating the combinations.
  - **Upper** - consider uppercase characters when generating the combinations.
  - **Lower** - consider lowercase characters when generating the combinations.
  - **Numeric** - consider numbers when generating the combinations.

- Other user inputs,
  - **passLen** (integer) – The length of the password.
  - **segSize** (integer) – Size of a segment allocated for each node/host. (Number of combinations a node should process).
  - **segNo** (integer) – The segment number assigned for that particular host.
  - **passHash** (String) - The hash key code of the real password.

3.4.2 Create a data array depending on user inputs.

When developing the brute force plug-in, one of the challenges we had to face is handling the characters using the ASCII table. When generating that combination, our plan was to use the ASCII values of the characters to do the text operations; like incrementing the ASCII value of a character by one to get the next one. But we came across a problem when tackling special characters since there are in four places in the standard ASCII table.

We used loops and jumping methods to the catch the all special characters but this made the code was very complex and inefficient.

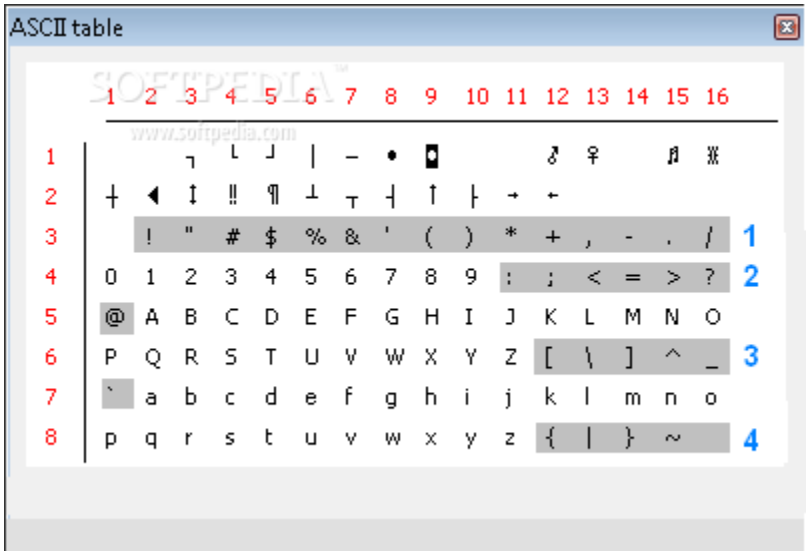

Figure 3.44 – ASCII Table overview

So we had to seek for a different approach when dealing with the characters. There we came to a solution of creating our own character map, a char array with indexed values. This enhancement worked fine and it helped us to reduce the code size and increased the efficiency of the program. This Map is created dynamically depending on the four conditional Boolean variables. Depending on what conditions are true, it creates the character map and stores the values. This way we only had to work with the index values of the characters not the ASCII value of them thus by allowing us to work without any dependency of the ASCII character map.

For example ***Symbolic*** – false, ***upper*** – true, ***lower*** – false, ***Numeric*** – true

Figure 3.45 – Dynamically created Map array

3.4.3 Generate all the possible text combinations depending on the map created.

- We created a special function which returns the string component of the given element.

  "string functionJump(double val, int arraySize, char arrayCharactor[], int passsize)"

  The input parameters are,

  - **double val** – The position of the element to be retrieved, nth element of all the possible combinations. (eg: 100th).
  - **int arraySize** – size of the character array.
  - **char arrayCharactor[]** – The character array passed.
  - **Int passsize** – The size of the password (for validations purposes).

- Above function is looped and called to get the string combinations within that segment range assigned for that local host.

3.4.4 Generate the hash key by sending it through the hash function.

In this phase, the string combinations retrieved in step 3 are sent though the hash function in order to get the hash key code.

MD5 plug-in consists of four c ++ code files apart from the main function. Those are,

- md5.cpp - C++ implementation of the MD5 Message-Digest
- md5.h – the header file
- md5wrapper.cpp - wrapper-class to create a MD5 Hash from a string.
- md5wrapper.h – the header file.

The hashing function,

*"md5wrapper md5;""hash = md5**.getHashFromString (<<combination>>)**;"*

3.4.5 Hash comparison.

Then the generated hash key is compared with the original hash code. If it retrieves the correct password it sends the notification to the local host, or else it computes all the combination in the given range and informs the local host.

3.4.6 The DFD Diagram for the Brute force plug-in

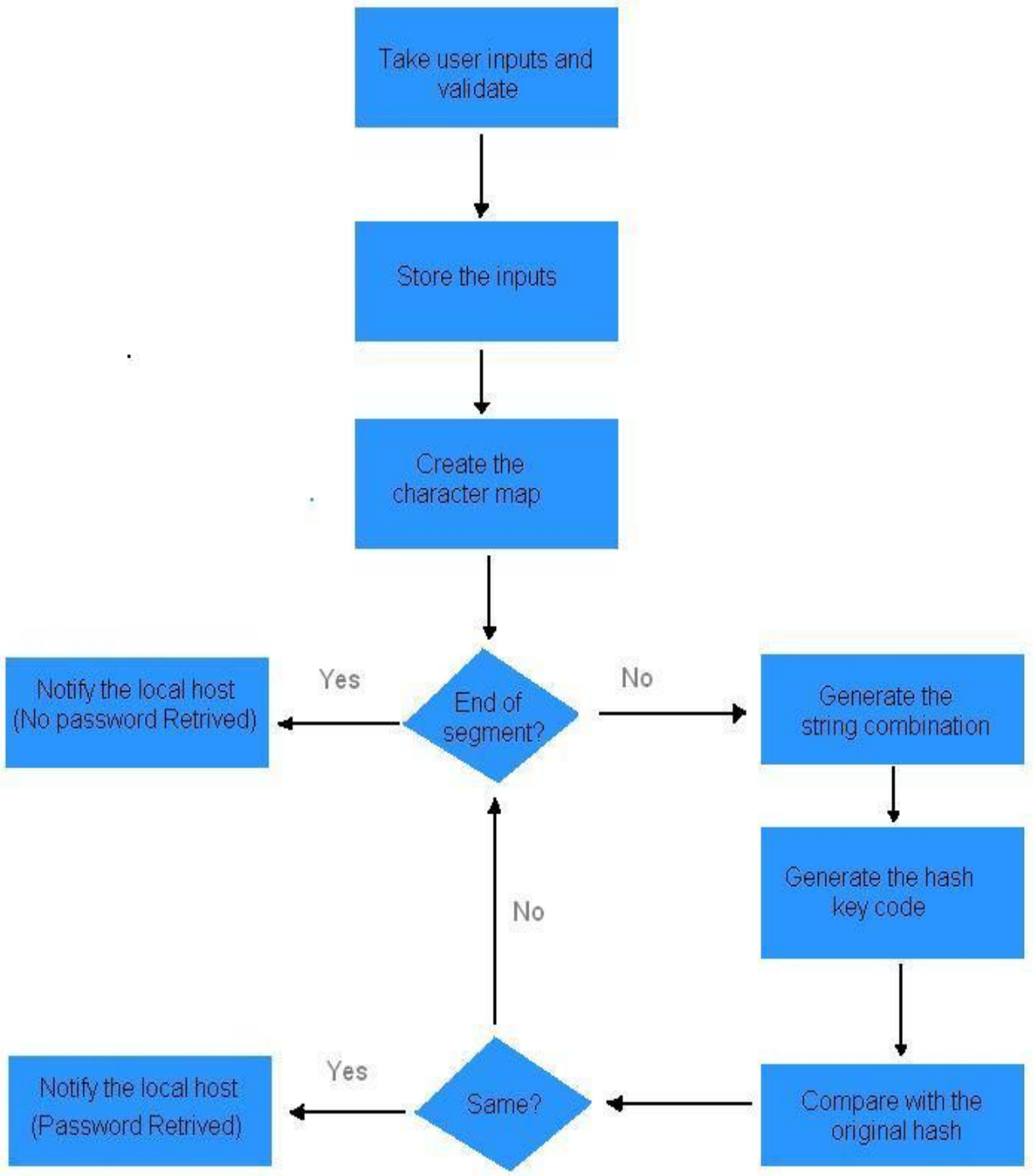

Figure 3.46 – DFD Diagram of the MD5 plug-in

3.5 Web server

In this system the server is a very simple model because we are intended to make the project system as decentralized as possible. The peer IP addresses are stored in the My SQL database when a peer is probed. And when a peer requests the IP lists, it checks the database and produces the list of IP addresses.

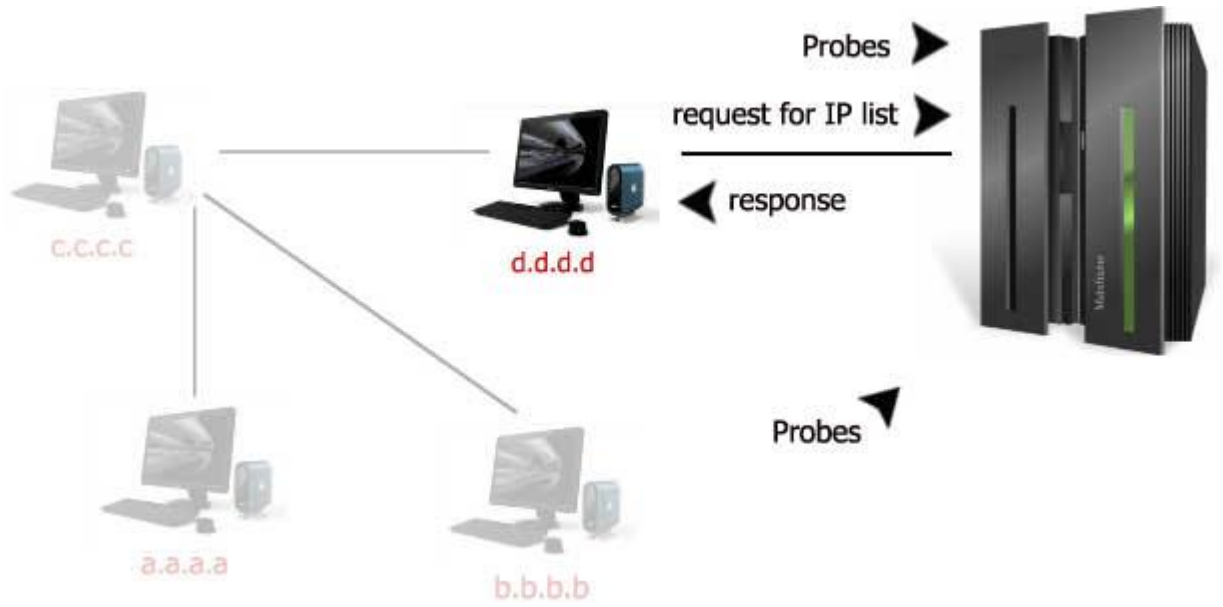

Figure 3.47 – Probe and IP list Request

We have put a mechanism to timeout and delete the IP entries if a specific host didn't probe for a particular amount of time. This is checked each time a peer request for IP list.

The web server consists of following modules:

- DB Connection

  This module deals with the database connection, SQL query execution and database closing. That is, this module provides the manipulation mechanism to the web server.

- Poke

  This is the page called by the application host in order to inform that the peer is active and up. So that information can be return to other peers, if the request is made by them.

- Active IPs

  This searches the DB and retrieves the up IP lists. Then the time is compared to get at what time the entry is made of. If the time period exceeds the IP entry is removed from the database.

- checksum

  This module is used to check the hash of the plug-in downloaded. When a request is made, this module produces the specific hash related to it and returns.

4. Results

We did a lot of testing under different environments and platforms to make sure the implementation works well in most cases with no errors and exceptions. We are happy to say we were able to see the desired outcomes, thus proving our objectives of this project and making it a success.

4.1 Outputs

Sample password retrieval is showed below,

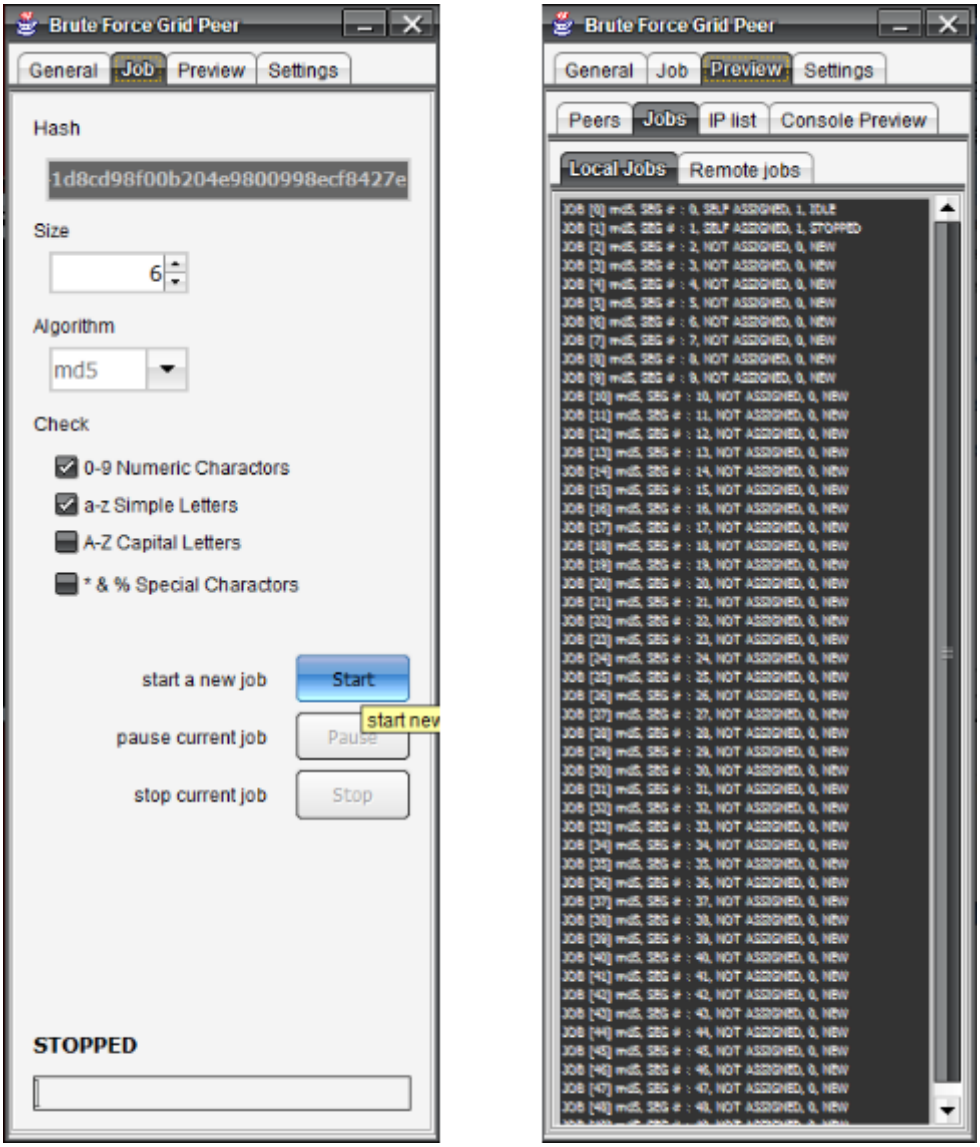

Figure 4.1 – Screenshots 1

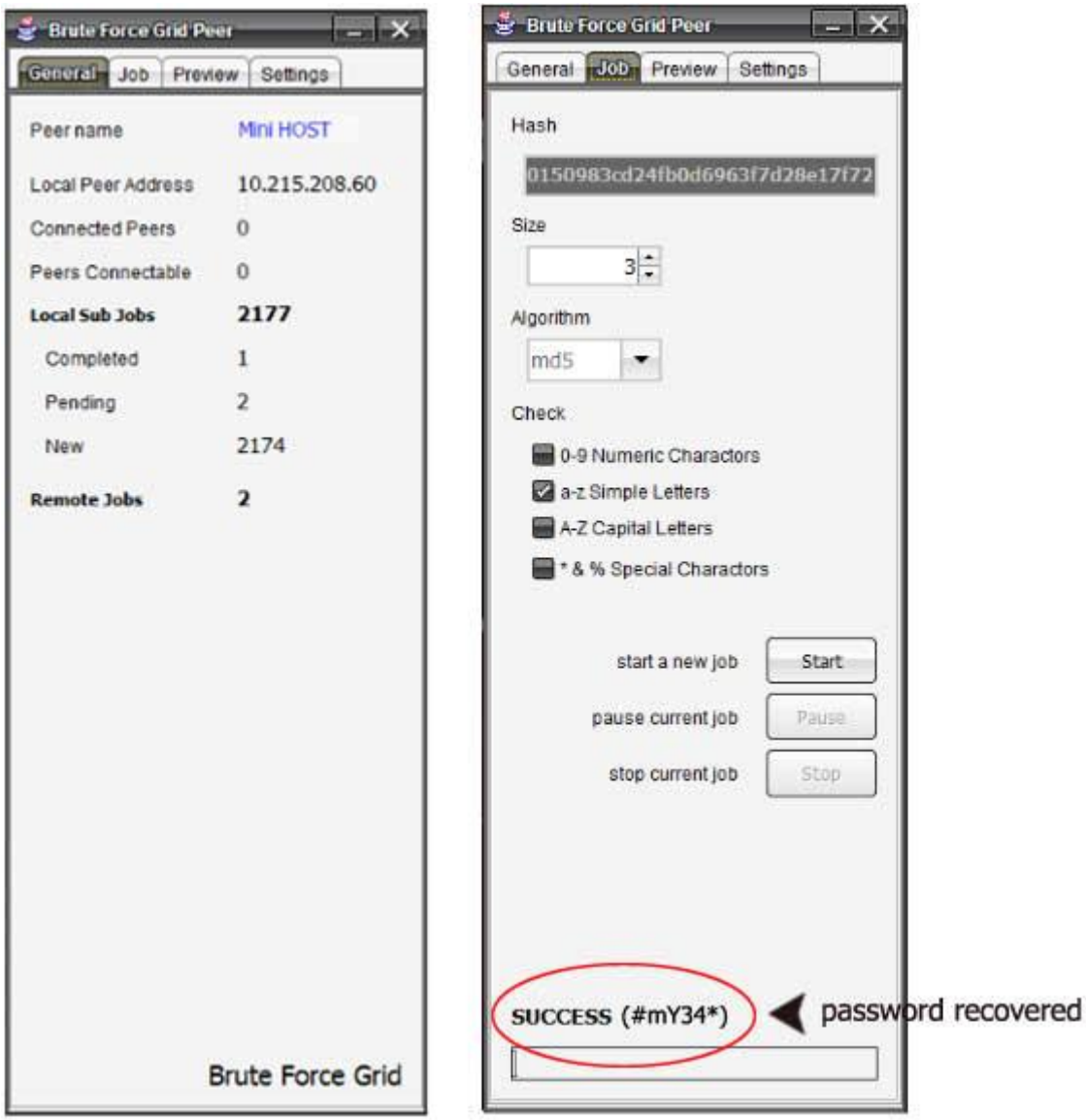

Figure 4.2 – Screenshots 2

4.2 The Experiments and the Results

The results of the testing can be analyzed under four major test cases. They are,

- The performance deviation with the implementation (application).
- The performance deviation with the segment size.
- The performance deviation with the password size.
- The performance deviation with the password type.
- The performance deviation with the number of nodes in the framework.

4.2.1 The performance deviation with the implementation

In this test case the main focus is to differentiate the performance variation with the application, the performance difference between our solution and some of other leading products which are available in the market. We choose four password recovering applications to compete with our solution.

Those are,

- Cain and Able - #1
- John the Ripper - #2
- L0phtcrack - #5
- Pwdump - #8

* Right hand side shows the current ranking of the application. [19]

The same password (“@xM1”) is given to all application and the time taken for all of them to recover the password is recorded.

Solution	Time (in seconds)
Cain & Able	**12.36**
John the Ripper	**25.09**
L0phtcrack	**123.65**
Pwdump	**240.88**
BFG (our)	**Y**

Table 4.1 - The performance deviation with the implementation

4.2.2 The performance deviation with segment size

This test case is related to the one of the node parameters set by the local host, the segment size. We recovered the same password only changing the segment size assigned by the job starter while keeping the other constrains the same. The graph clearly plots out the out put we came up with.

Password = “hel12”

Segment size	Time
1000	**8min 54sec**
10000	**5min 43sec**
100000	**4min 05sec**
1000000	**2min 36sec**
10000000	**0min 18sec**

Table 4.2 - The performance deviation with segment size

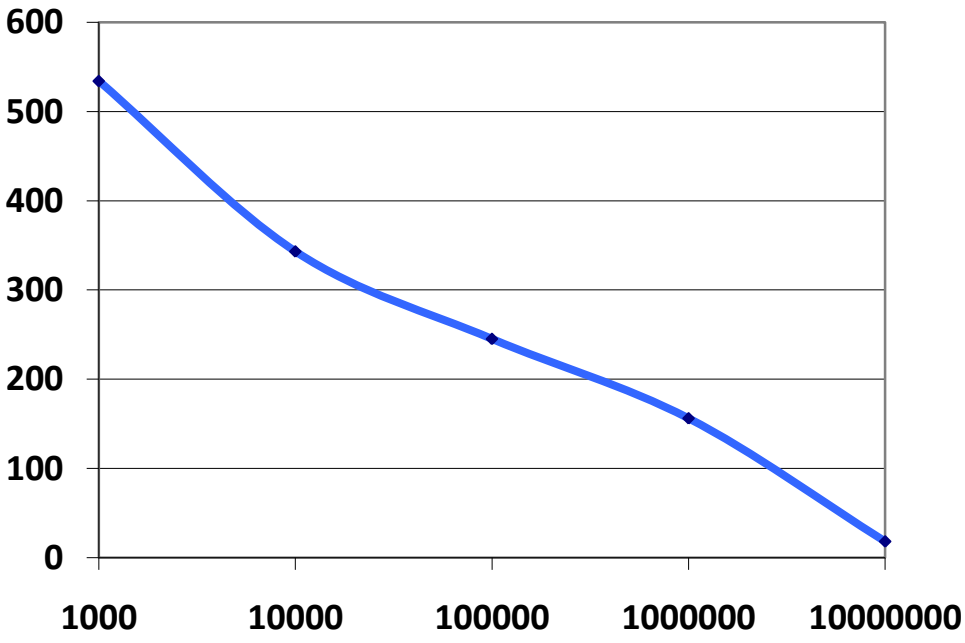

Figure 4.3 - The performance deviation with segment size graph

4.2.3 The performance deviation with password size

In this test case we wanted to check how the performance varies having the password size as the changing parameter while keeping other constrains constants. The time take to recover the range of passwords with different sizes (same type) is recorded. We used the lowercase as the password type. The graph plots out the outputs clearly.

Outputs:

Password size	Time
abc (3)	**0min 4sec**
abcd (4)	**0min 8sec**
abcde (5)	**0min 11sec**
abcdef (6)	**1min 50sec**
abcdefg (7)	**34min 32sec**

Table 4.3 - The performance deviation with password size

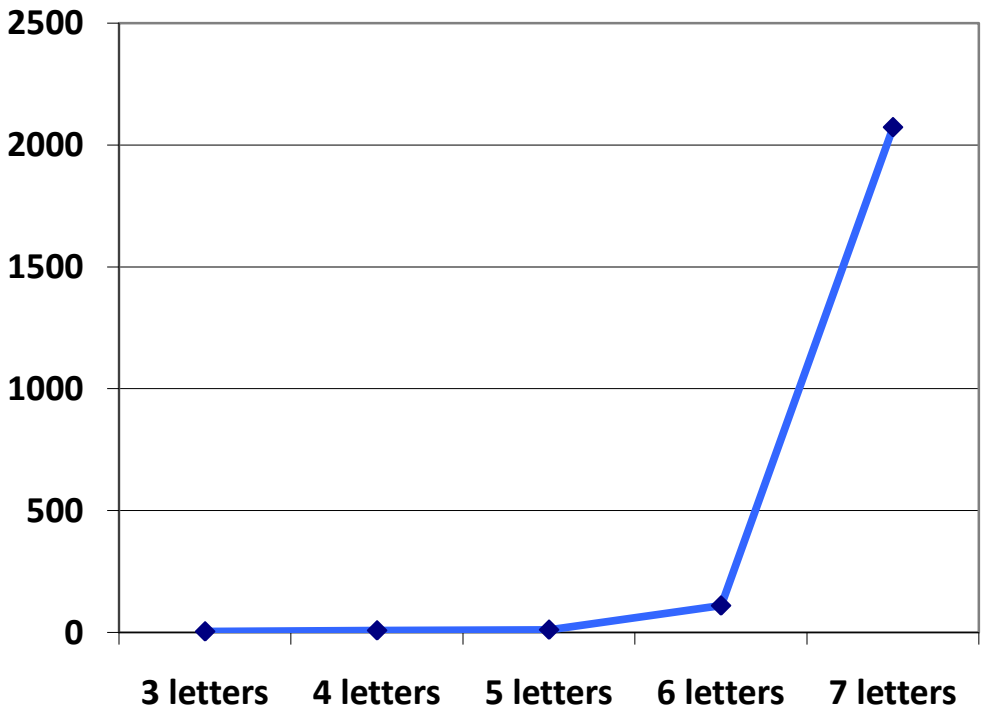

Figure 4.4 - The performance deviation with password size graph

5.2.4 The performance deviation with password type

This test case involves with the password type that the user is interested and how it affects to the performance. Different passwords (type) with the same lengths are recovered while maintaining the other factors the same and the time taken to recover is recorded. The graph plots out the out puts clearly.

* The 1st letter of each range is taken to minimize the searching delay, so it is easy to detect the time variation according to the type

3 peers, segment size - 1000000

Password	Time
aaaa (lower)	**0min 6sec**
aaAA (lower & upper)	**0min 20sec**
aAA1 (lower, upper & numeric)	**0min 53sec**
aA1! (lower,upper,numeric & symbolic)	**4min 19sec**

Table 4.4 - The performance deviation with password type

5.2.5 The performance deviation with number of nodes available

Checking how the performance varies depending on the number of nodes available in the framework is the focus of this phase. We recovered the same password with different number of nodes in the framework. The plots out the out puts clearly.

Password = “helloxy”

Number of nodes	Time
1	**40min 32sec**
3	**16min 44sec**
5	**10min 38sec**
7	**6min 12sec**
9	**4min 45sec**

Table 4.5 - The performance deviation with number of nodes available

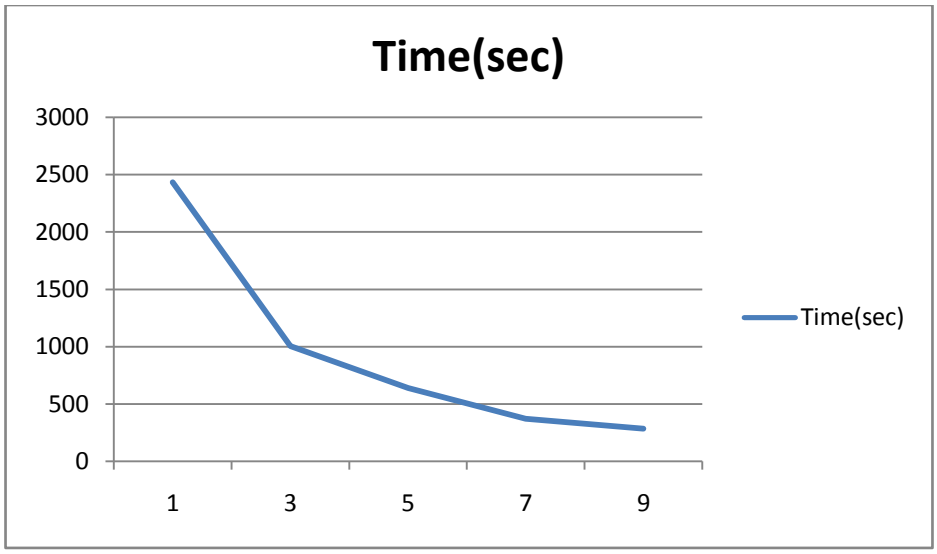

Figure 4.5 - The performance deviation with password size graph

5. Conclusion

5.1 The Analysis

The outputs shown for the experiments done under all the test cases helped us to come up with the conclusions about the performance of our implementation in different aspects.

5.1.1 The performance deviation with the implementation

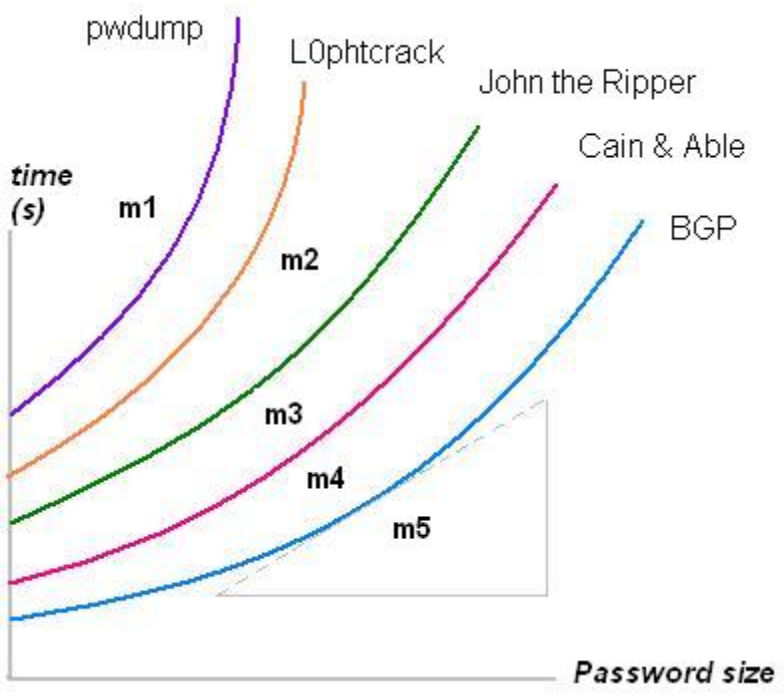

Figure 5.1 - The performance deviation with the implementation

According to the graph, we can see that BFG is the most-efficient solution in the given context by recovering the password in the least amount of time. It has the lowest gradient (m5), so in larger password sizes it shows a better performance than the others. So a user can go for BGP to gets his tasks done quickly and in an efficient manner.

5.1.2 The performance deviation with segment size

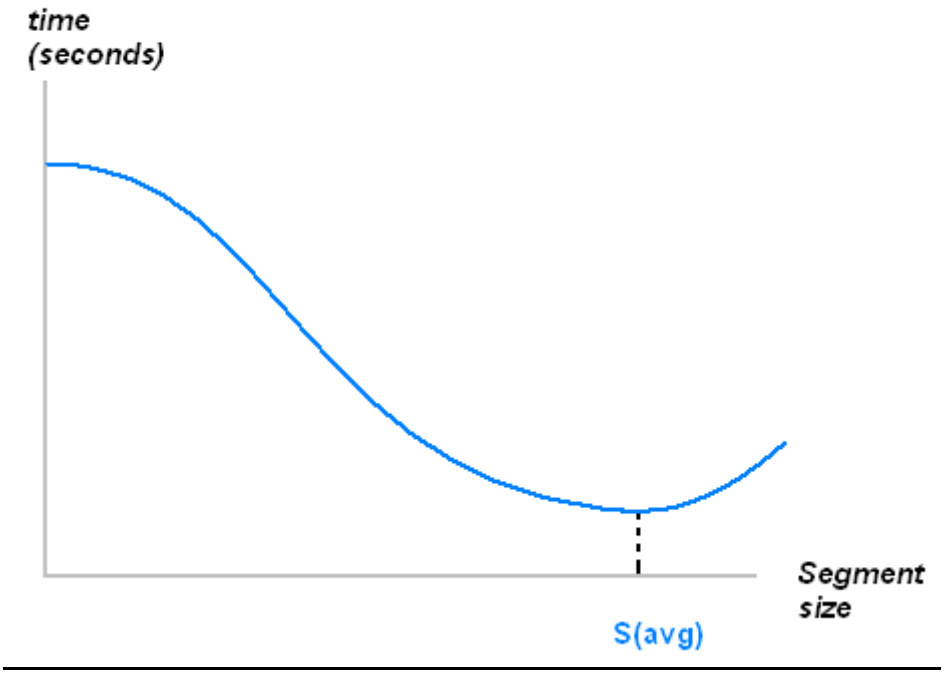

Figure 5.2 - The performance deviation with segment size

As the graph shows, in lower segment sizes the time taken to recover a password is large. This matches the least CPU scenario, thus it has a large number of combinations to compute, and lower numbers of segment allocations is the result of this case. When the segment size increases the process time taken to recover the password decreases, thus bringing the application to its maximum performance (s_{avg}). After this point if the segment size is increased the time also increases. The reason for this is the overhead of the framework implementation. Signal and message passing of the operations is done more than the actual computations done for the brute forcing in this case.

5.1.3 The performance deviation with password size

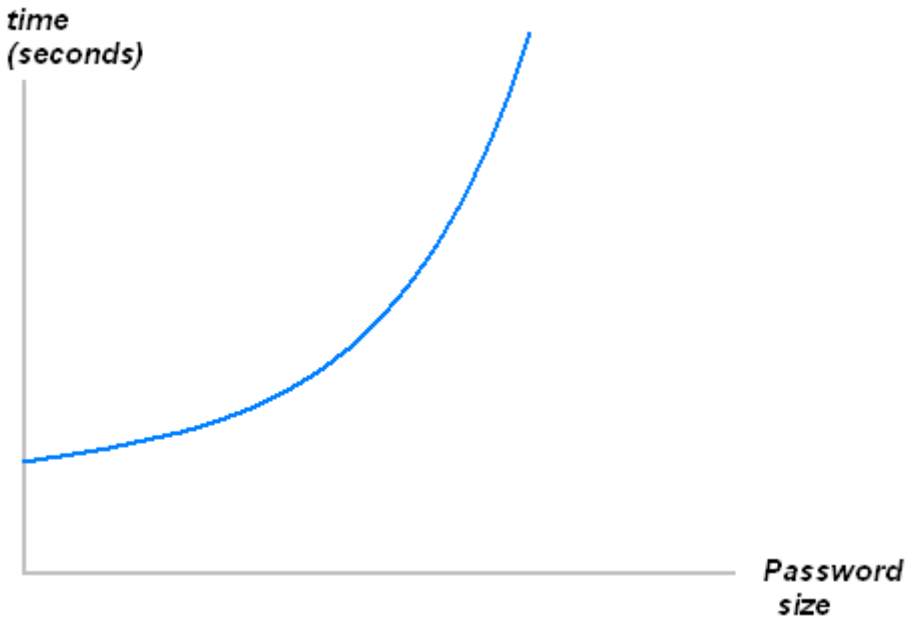

Figure 5.3 - The performance deviation with password size

According to the figure the time increases when the password size increases. But since the password is the user input and the cannot be compromised, we cannot say that “the performance increases when the password size is low”, but comparing with the other products BFG shows a remarkable performance difference at higher password sizes. The conclusion is short passwords are recovered quickly and more time is taken for the longer passwords.

5.1.4 The performance deviation with password type

This test case deals with the performance deviation with the type of the password. Again since this is a user input and cannot be compromised, we cannot say that “the performance increases when the password type is not mixed”, but again comparing with the other products BFG shows a good performance difference with mixed password. The conclusion is more time is taken to recover when the password is a mixed-typed one. Worst case is four types mixed passwords; Symbolic, Numeric, Uppercase and Lowercase. Here the application has to do a full keyboard search.

5.1.5 The performance deviation with number of nodes available

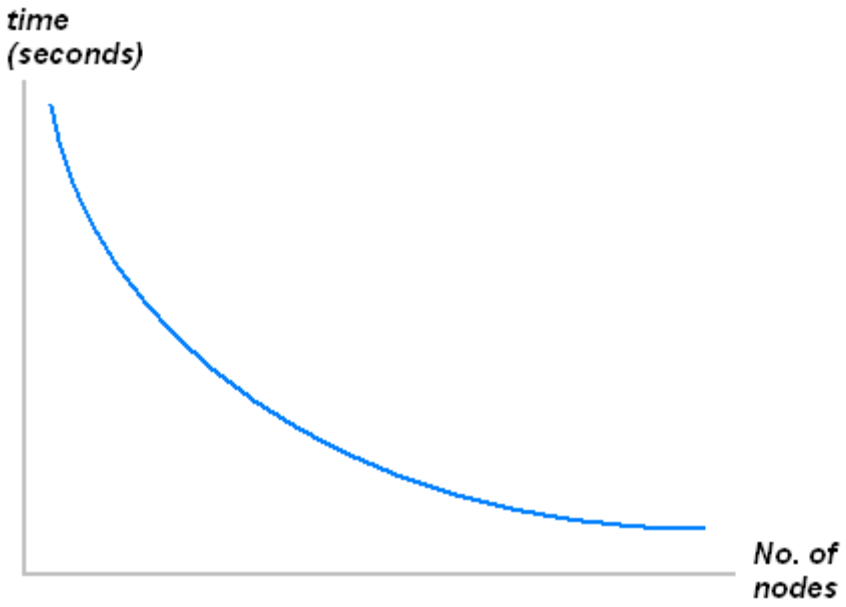

Figure 5.5 - The performance deviation with number of nodes available

How the performance varies with the number of hosts in the in the framework is discussed under this test case. As the graph shows the performance increases as the number of nodes in the framework increases. So user can have more nodes to get a better efficiency.

5.2 The Reasons why BFG is unique

- The efficiency of BFG.
- BFG is a dynamic implementation.
- It supports custom developments.
- BFG is simple
- BFG is a totally free application.

5.2.1 The efficiency

BFG is a unique because of its powerful performance. User can get his password recovered easily and efficiently. This efficiency is highlighted at longer and more complex passwords.

5.2.2 The implementation

BFG is very versatile and can be run in different platforms like LAN, WAN and public networks like internet. So it is a dynamic implementation and can be run anywhere with no architectural complexities found in other distributed solutions.

5.2.3 Supports custom developments

The support for custom developments is another good feature of BFG. Users can develop custom plug-ins and deploy them on the frame work and get their tasks done easily. This makes the framework more dynamic and less application-oriented.

5.2.4 The Simplicity

Simplicity and user-friendliness are important features of BFG. The simple and easy user interface covers up the most complex operations are done under the network level thus giving the user a refreshing feeling with ease and power.

5.2.4 Free!!

BFG is a totally free. Anybody can use to get their tasks done without any charges.

5.3 The Contribution

Peer to Peer network is been used to share files between peers that runs the P2P application. But there has been no actual implementation on the process sharing field using this P2P architecture. There are many written papers and research sheets on this field. But still there hasn't been an implementation in a distributed manner that can be used to have an overlay P2P network on the public internet.

Out project can be the first step towards the field of P2P process sharing, parallel processing technology. We give the software freely and we invite people to join hand in the field of developing this framework to provide efficient and high performance P2P for process sharing.

5.4 Future Enhancements

- Optimize the plug-in.
- Enhance the plug-in database of BFG.
- Use UDP connections for TCP.
- Strengthen the security by using encryption for the signal passing.
- Enhance to make the framework totally application-independent.

5.4.1 Optimizing the plug-in

The current plug-in will be more optimized to do the brute forcing faster. Use of library function will be minimized and the code will be improved with simpler functions with less redundancy and less CPU cycles.

5.4.2 Enhancing the plug-in database.

The current database holds only for the MD5 hashing. In future the plug-in database will be enhanced with brute forcing for more hashing algorithms like MD2, MD4, SHA1, SHA2, SHA 128, SHA 512, DES etc.

5.4.3 Use of UDP when making connections.

We have used CP to transfer signals and other information to the peers in the network. This is much inefficient when we consider about the overheads that includes in the TCP data transfer. We can include simple error checking and sequencing with UDP and use the UDP to send the information. Using UDP we can broadcast the common signals that are being uncast in the current application

5.4.4 Use Encryption for signal passing.

The system security can be increased by using the SSH encrypted connections to transfer important signals and passwords, recovered from the application.

5.4.5 Making the framework application-independent.

Making the framework application-independent will be a big challenge as well as a huge achievement. Not only for password recovering, for all the user-defined custom tasks will be processed by the framework. Dynamic user inputs will be taken, dynamic GUI will be built according to the user inputs and user developed plug-in for his task will be run, thus making the framework totally application-independent

References

[01] Anonymous, What is Hashing, www.webopedia.com, none, October 20, 2008.[Online] Available: http://www.webopedia.com/TERM/h/hashing.html, [Accessed: October 24th 2009]

[02] Anonymous, Overlay Network. Network Overlays, www.overlay-networks.info, none, 2008.[Online] Available: http://www.overlay-networks.info/info.html, [Accessed: October 24th 2009]

[03]. S. Brin, "Extracting Patterns and Relations from the World Wide Web," *Selected papers from the International Workshop on The World Wide Web and Databases*, London, 1999, pp. 172–183.

[04]. D.S. Milojičić, et al., "Peer-to-Peer Computing," *Technical Report HPL-2002-57R1*, HP Laboratories, Palo Alto, California, 2002.

[05]. A.W. Loo, "The Future of Peer-to-Peer Computing," *Communications of the ACM*, vol. 46, no. 9, Sep. 2003, pp. 56–61.

[06]. M. Schrage, "Piranha processing - utilizing your down time," *HPC wire (Electronic Newsletter),* Aug. 1992.

[07]. A. Davies, "Computational intermediation and the evolution of computation as a commodity," *Applied Economics*, vol. 36, no. 11, June 2004, pp. 1131–1142.

[08]. D. De Roure, M.A. Baker, N.R. Jennings, and N.R. Shadbolt, "The evolution of the Grid," *Grid Computing: Making the Global Infrastructure a Reality*, F. Berman, G. Fox, and A.J.G. Hey, eds., John Wiley & Sons, Chichester, 2003, ch. 3, pp. 65–100.

[09]. F. Berman, G. Fox, and A.J.G. Hey, "The Grid: Past, Present, Future," *Grid Computing: Making the Global Infrastructure a Reality*, F. Berman, G. Fox, and A.J.G. Hey, eds., John Wiley & Sons, Chichester, 2003, ch. 1, pp. 9–50.

[10]. H. Stockinger, "Defining the grid: a snapshot on the current view," *The Journal of Supercomputing*, vol. 42, no. 1, Oct. 2007, pp. 3–17.

[11].S. Androutsellis-Theotokis and D. Spinellis, "A Sur-vey of Peer-to-Peer Content Distribution Technologies," *ACM Computing Surveys*, vol. 36, no. 4, Dec. 2004, pp. 335–371.

[12]. G.M. Amdahl, "Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities," *AFIPS Conference Proceedings*, AFIPS Press, vol. 30, Apr. 1967, pp. 483–485.

[13]. J.L. Gustafson, "Reevaluating Amdahl's Law," *Communications of the ACM*, vol. 31, no. 5, May 1988, pp. 532–533.

[14]. D.S. Milojičić, F. Douglis, Y. Paindaveine, R. Wheeler, and S. Zhou, "Process Migration," *ACM Computing Surveys*, vol. 32, no. 3, Sep. 2000, pp. 241–299.

[16]. I. Foster and A. Iamnitchi, "On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing," *Proceedings of the Second International Workshop on Peer-to-Peer Systems* (IPTPS '03), Berkeley, 2003, pp. 118–128.

[17]. V. Donaldson, F. Berman, and R. Paturi, "Program Speedup in a Heterogeneous Computing Network," *Journal of Parallel and Distributed Computing*, vol. 21, no. 3, June 1994, pp. 316–322.

[18]. C. Perez, "Technological Revolutions, Paradigm Shifts and Socio-Institutional Change," *Globalization, Economic Development and Inequality:* An Alterna-tive Perspective, E. Reinert, ed., Edward Elgar, Chel-tenham, 2004, pp. 217–242.

[19] Anonymous, Top 10 password crackers, sectools.org, none, 2008.[Online] Available: http://sectools.org/crackers.html, [Accessed: October 25th 2009]