**Peer-to-Peer Distributed Computing Framework**

**Project ID:16-023**

**Project Proposal**

**B.Sc. Special (Honors) Degree in Information Technology**

**Submitted on 10/03/2016**

**Peer-to-Peer Distributed Computing Framework**

**Project ID: 16-023**

**Authors:**

|  |  |  |
| --- | --- | --- |
| **Student ID** | **Name** | **Signature** |
| IT13037116 | Dharmapala P M |  |
| IT13002466 | Sri Darshanun S |  |
| IT13071608 | K Lumeshkantha |  |
| IT13029364 | Ismail M I |  |

**Supervisor**

………………………………….

**Dr. Darshana Kasthurirathna**

**Co-Supervisor**

………………………………….

**<*Name of the Co-supervisor*>**

# DECLARATION

We declare that this is our own work and this project proposal does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of our knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

………………………………….. ……………………………………….

Dharmapala P M K Lumeshkantha

………………………………….. ……………………………………….

Sri Darshanun SIsmail I M

# ABSTRACT

Peer-to-peer computing is a distributed application architecture that partitions workload and tasks between peers. Owing to the recent growth of the internet, resource sharing over public internet infrastructure has been in the rise. We have witnessed firsthand how the Bittorrent protocol changed the way people shared files and the effect it has had on the bandwidth of the internet. Even so, among the many peer-to-peer systems that exist today, resource sharing has been limited to the realm of file sharing. Another observation that we have made over the recent past is how much personal computers have advanced in computational capabilities and how common they’ve become. Volunteer distributed computing projects like BOINC have taken advantage of this trend and created their very successful platform on this basis. It is worth noting that these platforms are dependent on a central server for coordination and network management. We can safely say that as of this writing there does not exist a peer-to-peer, server-less, distributed computing platform. Although past research has been done about CPU time sharing over peer-to-peer networks, a working implementation of such a system is yet to be found. It is the purpose of this research to design and implement such a system. The planned procedure is to design the network layer separately at first, build the execution layer on top of that and create a user agent software. Ideally, the outcome would be a peer-to-peer distributed computing framework that’s globally scalable and popular among the general population as a way to access high performance computing.

**TABLE OF CONTENT**

[DECLARATION iii](#_Toc445288044)

[ABSTRACT iv](#_Toc445288045)

[LIST OF TABLES vi](#_Toc445288046)

[LIST OF FIGURES vi](#_Toc445288047)

[1. INTRODUCTION 1](#_Toc445288048)

[1.1 Background 1](#_Toc445288049)

[1.2 Literature Review 2](#_Toc445288050)

[1.3 Research Gap & Research Problem 5](#_Toc445288051)

[2. OBJECTIVES 7](#_Toc445288052)

[2.1 Main Objectives 7](#_Toc445288053)

[2.2 Specific Objectives 7](#_Toc445288054)

[3. RESEARCH METHODOLOGY 10](#_Toc445288055)

[3.1 Design a distributed, peer-to-peer, server-less network that can manage arbitrary arrivals and departures of random nodes 10](#_Toc445288056)

[3.2 Research and develop a method to break down a generic computational task into smaller independent units of work. 12](#_Toc445288057)

[3.3 Develop a distributed computing algorithm to share the workload among many computational nodes, execute and aggregate the results. 14](#_Toc445288058)

[3.4 Develop the network and distributed computing modules to be fault tolerant 14](#_Toc445288059)

[4. DESCRIPTION OF PERSONAL AND FACILITIES 15](#_Toc445288060)

[5. REFERENCES 16](#_Toc445288061)

# LIST OF TABLES

[Table 1.1: Feature comparison 6](#_Toc445287977)

[Table 1.1: Personal and facilities……………………………………...……………………..18](#_Toc445287977)

# LIST OF FIGURES

Figure 1.1: DNS Architecture 2

Figure 1.2: Gnutella Architecture 3

Figure 1.3: Kazaa Architecture 4

Figure 1.4: BitTorrent Architecture 5

[Figure 3. 1: Peers arranged in a virtual GUID ring 9](file:///C:\Users\Jear\Desktop\Project-Proposal.docx#_Toc445242248)

[Figure 3. 2: A peer referencing a subset of peers in the Network 9](file:///C:\Users\Jear\Desktop\Project-Proposal.docx#_Toc445242249)

[Figure 3. 3: Peer locating 10](#_Toc445242250)

[Figure 3. 4: Peer locating algorithm 10](#_Toc445242251)

[Figure 3. 5: Joining network 11](#_Toc445242252)

[Figure 3. 6: Leaving network 1](#_Toc445242253)2

Figure 1.1: Gnatt chart 15

# INTRODUCTION

## Background

Firstly, Computational power of personal computers has advanced since the birth of the first personal computer in 1975 and it has become a most commonplace equipment in many households. Processing power has become cheaper as transistor technology sees constant improvement and mass production of processors has resulted in extremely low cost to the consumer. It has been observed that the number of transistors in integrated circuits approximately doubles every two years [1]. As a result the average personal computer of today is more powerful than the most expensive and advanced supercomputer decades ago.

Secondly, personal computer prices have dropped drastically over the years [2]. This has resulted in more people purchasing computers for their personal use.

Put together the facts of how affordable a computer is and how capable its processor is and it is very common to see most of the time these powerful machines are powered on and idling away in an office or a home. For example, there could be a couple of moderately powerful computers dedicated to the task of merely searching a library catalog. There could be half a dozen high performance computers in a small office that are solely used for word processing and spreadsheets. There are hundreds of computers in computer labs of academic institutions that are powered on and in idle state throughout the day. We are personally guilty of purchasing the highest performance computer out there that money can buy and spending most of the time surfing the web on it.

There is a lot of potential if we are able to harness the idle processing power of these machines. The idea is to develop a framework to share the processing power of your machine over a peer-to-peer network. The system would act as one giant supercomputer by utilizing several idle computing power. By making our processing power available to others, we encourage others to do the same.

A distributed computing system can be centralized or decentralized. Centralized systems have been utilized for a long time in master slave based systems. All the individual nodes of the system are directly dependent on the centralized server and the central server is responsible for managing and coordinating the rest of the network. While this architecture is fundamentally good and rather easy to implement than the decentralized system it gives rise to a single point of failure. For example, in the BitTorrent protocol, the peer discovery is done by using the centralized server called the BitTorrent tracker. A government authority can force a BitTorrent tracker to shut down, causing the whole network to come down as a result.

## Literature Review

1.2.1. Domain Name System

The Domain Name System was introduced in 1983 as a solution for the growth of the internet and being not able to manage the hosts.txt file. The hosts.txt file which contains mappings between host IP addresses and user friendly names was copied around the internet on regular basis. So managing this file was not an easy task and maintaining accurate host lists was impossible.

DNS introduced a hierarchical systems structure to store records about hosts on the internet and as a part of the research, they considered not to have a dedicated host which contains all information about other hosts. According to this, a single point of failure is not possible as well as the name servers are responsible for host names in its zone only. The figure below depict how zones are separated in the internet.

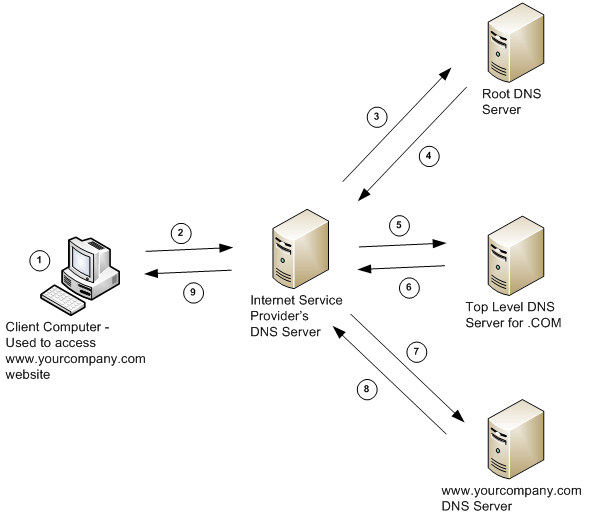


Figure 1.1: DNS Architecture

Apart from this structure DNS introduced caching of data. The mappings between host names and IP addresses as well as the information will be saved for a limited period of time in this structure to drastically reduce reaction period of time when the same information is requested by any other subject.

1.2.2. Usenet

Usenet is an outdated non-centralized computer network service which internet provided in early 80’s for discussions and sharing of files. This is still being used although its popularity has decreased. Usenet transferred messages via newsgroups as well as it used Unix-to-Unix and TCP/IP based protocols for transmission. Users were able to post messages in newsgroups and they will be transmitted to hosts according to newsgroups subscribed. Unfortunately this service is no longer visible over the unbearable growth of spam.

1.2.3. Napster

Peer-to-peer technology experienced a revival with the introduction of Napster which allowed to share music files among users. Napster used a dual architecture for distribution of files and a combination of peer to peer and client-network architecture. The file exchanges are handled by the peers on the network and file indexing is done via a simple-client model. Registration is automatically done by the Napster client and all usernames are used in conjunction with a password to identify users uniquely. When the user needs to get hold of a music file, first Napster server identifies the user and returns the location of the file if the requested content is available in the Napster server database.

1.2.4. Gnutella

Gnutella is a decentralized peer-to-peer service in which searching and downloading is completed by peers on the network querying own neighbors. The Gnutella network is referred to as GNet and the nodes which participate in the network are called servants. A servant should get the host address of at least one other servant to get connected to the network. It is possible to find servant addresses by either using the GWebCache protocol or using the peer address which was saved while the servant was connected previously. GWebCache protocol retrieves host addresses by sending HTTP queries to GWebCache servers and the servant is connected to the network after obtaining the host address. The diagram shown below depicts the handshake between a servant which needs to be connected to the network and the servant which is already in the network.



Figure 1.2: Gnutella Architecture

The servant issues a query message to all its neighbors when a particular file needs to be found. This query message is transmitted across the network from neighbor to neighbor until the file which was in search is found. Also if the file is available with any node in the network, the requesting servant is able to download the file.

1.2.5. Kazaa

Kazaa is another example for peer-to-peer file sharing. Kazaa uses the same type of technology which made Napster famous, although a decentralized system was used unlike Napster. The protocol being used in transmission is FastTrack and file sharing is basically done by contacting other similar nodes in the network. Kazaa divides the users in the network into two groups called Super nodes and Ordinary nodes where super nodes are powerful computers with fast network connections, quick processing capabilities and higher bandwidth, but the owners does not have any clue that their machine is used as a supper node in the network. How the machines are organized in the Kazaa architecture is shown below.

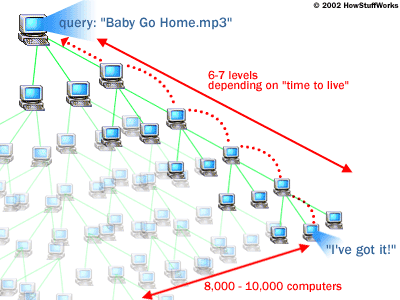


Figure 1.3: Kazaa Architecture

When users get Kazaa deployed in their machines, a list of super nodes will be notified. Also every time the user launch the Kazaa application, the machine gets registered with the central server and then select a super node from a currently active list. The requests sent by users for the purpose of download and upload are tunneled via super nodes. The super nodes communicate with every other super node in the system and every super node communicate with other regular nodes to retrieve the requested data. Also Kazaa has an inbuilt ­­feature called ‘Time to Live of 7’ which allow the search request to propagate seven levels in to the network. If the file is found, it is transferred to the requester from the file owner using HTTP protocol directly, but it doesn’t have to be transmitted through a super node.

1.2.6 BitTorrent

BitTorrent is one of the most common communication protocols used in peer to peer file distribution which a large amount of data is allowed to share over the internet reliably and quickly. The initial version and the most recent version of BitTorrent was released respectively in 2001 and 2003.

BitTorrent system has no central dedicated server since peers in the network share information among them. A BitTorrent client needs to be deployed on devices which use BitTorrent to fulfil sharing files. [μTorrent](https://en.wikipedia.org/wiki/%CE%9CTorrent), [Xunlei](https://en.wikipedia.org/wiki/Xunlei) and [qBittorrent](https://en.wikipedia.org/wiki/QBittorrent) are some popular clients which are currently in use. To perform a torrent download the required torrent should be selected from the web and opened with a BitTorrent client, afterwards the BitTorrent client gets connected to the tracker from which it receives a list of peers transferring pieces of the required torrent file and performs the download. The group of nodes which has pieces of the master file is called a swarm and when parts of the required file is being downloaded, these pieces are uploaded to other nodes simultaneously, so all these nodes will have its own complete master file.

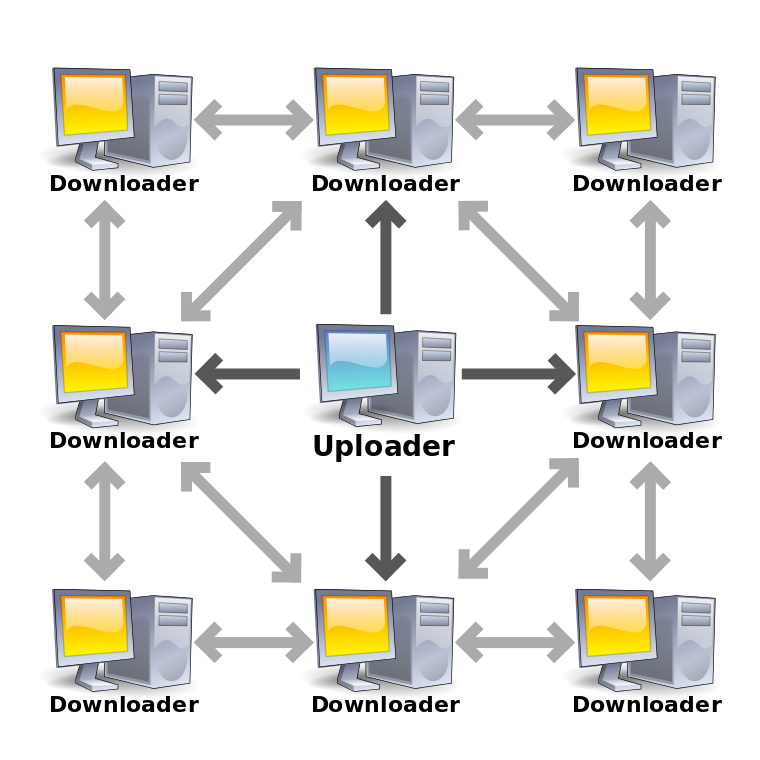


Figure 1.4: BitTorrent Architecture

Also BitTorrent protocol always acquire the rarest piece of the file first, so this can be used as a cost effective efficient way to publish digital files when other systems require more bandwidth and robust server systems.

## Research Gap & Research Problem

There are various systems that are dedicated for distribution and sharing of digital media using P2P networking technologies.

The three different types of file sharing systems are Non-public file sharing, Public file sharing and Academic file sharing. Some examples of Non-public file sharing are Anonymous P2P, Friend-to-friend and Darknet Private P2P. Public file sharing networks and services include Gnutella, eDonkey, BitTorrent, RetroShare and KickassTorrents. Academic file sharing systems includes Sci-Hub, Library Genesis and ICanHazPDF. These peer-to-peer systems do not provide a computational resource sharing for public networks.

Though there are several systems for file sharing, there is yet no a decentralized mechanism for computational time sharing. The existing peer-to-peer centralized systems for computational time sharing, such as BOINC, does not provide the capability for an average user to perform their own tasks.

Table 1.1 shows a feature comparison between existing products and the proposed framework.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Features** | **DNS** | **Usenet** | **Napster** | **Gnutella** | **Distributed.net** | **BitTorrent** | **BOINC** | **Our Solution** |
| **Decentralized** | **✓** | **✓** |  | **✓** |  | **✓** |  | **✓** |
| **Computational Time Sharing** |  |  |  |  |  |  | **✓** | **✓** |
| **Parallel Processing** |  |  |  |  | **✓** |  | **✓** | **✓** |
| **Authentication** |  | **✓** | **✓** |  |  |  | **✓** |  |
| **Fault Tolerance** | **✓** | **✓** | **✓** | **✓** | **✓** | **✓** | **✓** | **✓** |
| **Result Validation** |  |  |  |  | **✓** |  | **✓** | **✓** |
| **Reward System** |  |  |  |  |  |  |  | **✓** |

Table 1.1: Feature comparison

The proposed system is intended to work on a decentralized organizational structure and work as a general purpose system.

# OBJECTIVES

## Main Objectives

1. Research and design a distributed peer-to-peer computing framework.

2. Develop a user agent software to demonstrate the functionality of the framework.

## Specific Objectives

1. **Design a distributed, peer-to-peer, server-less network that can manage arbitrary arrivals and departures of random nodes**

This is the network layer of the proposed computing framework. The framework application will run on top of the peer-to-peer network seamlessly. The defining characteristic of this system that sets it apart from other similar systems is that it has no central server to manage and coordinate the peer-to-peer network. This becomes rather problematic when the network is scaled up into thousands of peers. Therefore, a mechanism must be designed to:

1. *Peer identification*

Each peer must be uniquely identified within the network for successful communication to take place.

1. *Peer discovery (peer location)*

There has to be a mechanism for a node to locate the other peers in the network so that it can communicate with them.

1. *Booting the network*

The network must be started up by using just one node initially.

1. *Joining an existing network*

A peer must be able to join an existing network that has been already booted. There must exist a mechanism for an arbitrary peer to find the network and become a part of it.

1. *Leaving the network*

When a peer wants to leave the network, there must be a way for the node to notify the rest of the network its intention to leave the network, so that the network can adapt to the leaving of the node.

1. **Research and develop a method to break down a generic computational task into smaller independent units of work.**

In order to take advantage of the processing power of several computers to simultaneously solve a problem, the problem must be broken down into smaller units. The divide and conquer strategy can be employed to solve such a problem in parallel. The framework should be able to break down a generic computational problem into small units. The so-called embarrassingly parallel problems can be separated rather easily [3].

The proposed system should be able to break down a generic parallel problem into independent work units. The system must be able to accommodate any problem and distribute it over the network.

1. **Develop a distributed computing algorithm to share the workload among many computational nodes, execute and aggregate the results.**

The algorithm must take into account factors like the CPU load of the computer. Ideally it should utilize the idle processing time of the computer. The algorithm should be able to decide how many nodes to deliver the workload to and select the best nodes to deliver them depending on load and capacity. Once the work has been delivered to the computer, it will be completed and the results sent back to the originator of the workload. A mechanism must exist to validate the results. Validation can be done by redundantly calculating the results on another node. Once the results have been delivered to the workload originator, it should be aggregated to get the final outcome.

1. **Develop the network and distributed computing modules to be fault tolerant**

The system should continue to operate properly in the event of the failure of some of its components. The fault tolerant aspects addressed will be:

1. The failure of a computational node without gracefully leaving the network
2. The failure of a workload allocator
3. Vandalism of results by malicious peers

# RESEARCH METHODOLOGY

## Design a distributed, peer-to-peer, server-less network that can manage arbitrary arrivals and departures of random nodes

A node in a peer-to-peer network has to be able to establish communication with the other nodes in the network. The node must be able to locate the target node in the P2P network to be able to do so. Each peer has a routing table to find the other peers in the network. The routing table in a peer does not contain routing information about all the peers in the network. Rather, it contains the routing information about a few peers in the network.

A peer-to-peer network in our system may be scaled up to millions of peers and each peer cannot possibly maintain a complete list of all the nodes in the rest of the network. A table of that size would take up too much resources in the peer and updating a table of that size as the peers join and leave the network is not practical.

Therein lies the sub-objectives of the first objective

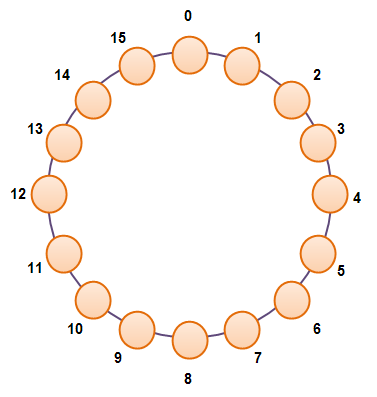
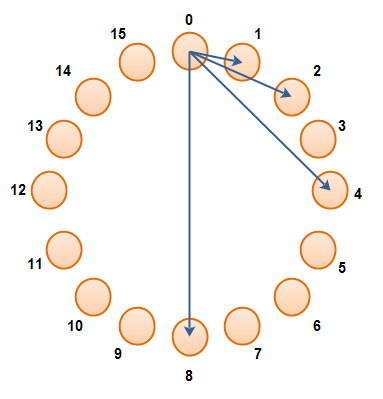
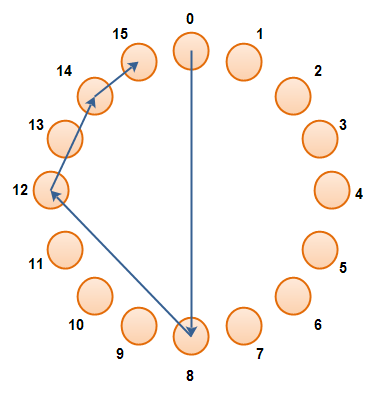
1. Peer Identification (GUID)
2. Peers create a GUID themselves – create a smart algorithm
3. Peers are assigned a GUID when they join an existing network – one peer already in the network generate a GUID.
4. Peer Location

Figure 3. 1: Peers arranged in a virtual GUID ring

­

Figure 3. 2: A peer referencing a subset of peers in the Network

Once each and every peer in the network have been assigned a GUID (Globally Unique ID), it can be used to uniquely identify a specific peer in the network. When a peer is being looked up in the network, the GUID is used to address the peer. The GUIDs connect the peers in a virtual P2P ring. The GUIDs assigned to the peers does not indicate the geographical relationship between the peers. For example, the peer with GUID 1 and the peer with GUID 2 can be located halfway across the globe from each other, while the peer with GUID 1 and the peer with GUID 200 could be located in the same neighborhood.

1. Finding peers

Peer 15 being found by peer 0. It only takes log(n) steps. The algorithm makes it feasible to locate the peers in extremely large networks with hundreds and thousands of peers.

Figure 3. 3: Peer locating

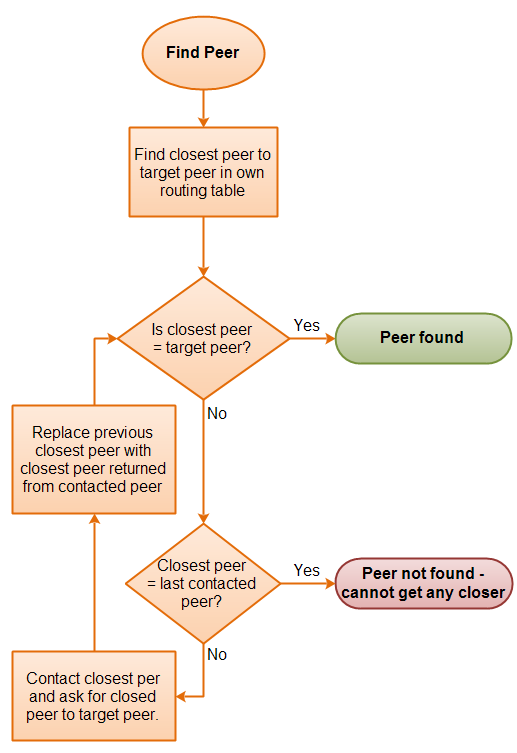


Figure 3. 4: Peer locating algorithm

1. Booting and joining a P2P network

The boot peer is alone in the network at first until the first peer requests to connect to the boot peer by sending a join message. A new GUID is generated by the boot peer and sent to the joining peer.

|  |
| --- |
| A peer contacting the boot peer to join the network. |

Figure 3. 5: Joining network

A peer signals its intent to gracefully leave the P2P network by sending a leave message to all the peers that are in its routing table. The next step is to remove that peer’s GUID from all the routing table of the peers that received the message.

|  |
| --- |
| A leaving peer sends leave messages to all peers in its routing table. |

Figure 3. 6: Leaving network

The leaving peer can close down the network connections safely once the leave message has been received by all the peers in its routing table.

## Research and develop a method to break down a generic computational task into smaller independent units of work.

In parallel computing it is a must to separate the workload or the problem into a number of parallel and independent tasks. Obviously, not every problem can be separated into parallel tasks. Most of the computational problems require an order of execution and the end result depends on the order of execution. Often there is much dependency and need of communication between parallel tasks and the results between them.

There are, however a few instances where the tasks is perfectly parallel (The so-called embarrassingly parallel problem). For this type of tasks, little or no effort is needed to divide the problem into a number of parallel tasks. This type of problems are easy to solve on large server farms. For example, the most prominent distributed computing platform BOINC has many projects that utilize the parallel nature of extremely large computational problems [4]. Some of them are:

**Einstein@Home:** Analyzes LIGO detector data for continuous gravitational wave sources

**Climate Prediction:** Finds ways to improve climate prediction models

**Distributed.net:** Attempts to crack the RC5-72 cipher and find the optimal Golomb rulers of length 28

**Folding@Home:** Understand protein folding, misfolding, and related diseases, with a minor emphasis in protein structure prediction.

Some examples for the embarrassingly parallel problems are:

1. Cryptographic brute force searching
2. Facial recognition systems
3. Image rendering
4. Video rendering
5. Marching squares algorithm
6. Event simulation in particle physics

Parallel slowdown is another phenomenon that must be taken into account when discussing parallel processing. Parallel slowdown is a phenomenon in parallel computing where parallelization of a parallel algorithm beyond a certain point causes the program to run slower (take more time to run to completion) [5].

This phenomenon results as from communication bottlenecks. There is a certain point when adding processing nodes to a parallel processing system, each node has to spend less time doing actual useful processing and more time communicating.

Several technologies exist to distribute parallel problems for processing

**OpenMP:** Supports multiprocessing and shared memory in C++.

**OpenCL:** This framework supports writing programs that can execute across heterogeneous platforms.

**CUDA:** This is an API and a platform created by nVIDIA. This platform enables programmers to use GPUs for general purpose processing

## Develop a distributed computing algorithm to share the workload among many computational nodes, execute and aggregate the results.

1. Sharing the workload among peers

Once the parallel problem has been separated into smaller parallel tasks, the system should find the peers to submit the tasks. This part of the algorithm should take into account factors such as load, performance and user settings of the peer. If the load is too much on a certain peer, it may not accept the next task. Computers with lower performance should be given less amount of tasks to perform. Users may choose to limit the amount of work performed by adjusting the settings in the client software.

2. Execution

Execution of the task depends on the nature of the problem. For example rendering an image is quite different from trying brute force combinations. For the system to be a general purpose execution platform, each kind of problem needs to be solved on its own execution module. The system should be designed so that the client software can plug-in these modules to itself dynamically when it needs to execute the task given to it. The system should have a well-defined interface for accepting the plugins. Any third party can write these modules and distribute it with the task units to the executing peers.

## Develop the network and distributed computing modules to be fault tolerant

To provide a reliable and robust environment it is important that means of resilience come with the system. The major concern in a peer-to-peer computation time sharing system is the reliability, as a single node failure fails all running applications on the node [6].

## C:\Users\Jear\Desktop\gnatt.JPGGnatt chart

Figure 3. 7: Gnatt chart

# DESCRIPTION OF PERSONAL AND FACILITIES

|  |  |  |
| --- | --- | --- |
| **Member** | **Component** | **Task** |
| Dharmapala P M | Design a distributed, peer-to-peer, server-less network with arbitrary node management | Peer identification, peer discovery, bootstrapping a new network, joining an existing network, leaving a network gracefully |
| Sri Darshanun S | Research and develop a method to break down a generic computational task into smaller independent units of work | Develop the module to break down a task into parallel independent tasks |
| K Lumeshkantha | Develop a distributed computing algorithm to share the workload among many computational nodes, execute and aggregate the results | Define and create an interface to accept pluggable executor modules in the user agent software.  Develop a pluggable module to solve such a problem. |
| Ismail M I | Develop the network and distributed computing modules to be fault tolerant | Achieve fault tolerance from node failures.  Achieve system security from malicious peers |

Table 4.1: Personal and facilities

# REFERENCES

|  |  |
| --- | --- |
| [1] | Wikipedia, "wikipedia.org," [Online]. Available: https://en.wikipedia.org/wiki/Moore's\_law. |
| [2] | Statista, "statista.com," [Online]. Available: http://www.statista.com/statistics/203759/average-selling-price-of-desktop-pcs-worldwide/. |
| [3] | "Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/Embarrassingly\_parallel. |
| [4] | "wikipedia.org," [Online]. Available: https://en.wikipedia.org/wiki/List\_of\_distributed\_computing\_projects. |
| [5] | "Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/Parallel\_slowdown. |
| [6] | T. Tangmankhong, et al., "Peer-to-peer fault tolerance framework for a grid computing system," in *International Joint Conference on Computer Science and Software Engineering (JCSSE)*, 2012. |