Desktop Monitoring over Local Network

B.Tech. Major Project Report

BY

Niharika Ahuja (13559)
Anil Kumar (13530)
Armaan Jain (13581)
Anil Chaurasiya (13568)
Abhinandan Jain (13532)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY
HAMIRPUR-177005, HP (INDIA)

Aug-Dec, 2016

Desktop Monitoring over Local Network

A Major Project Report

submitted in partial fulfilment of the requirements for the award of the degree

of

Bachelor of Technology

in

COMPUTER SCIENCE & ENGINEERING

BY

Niharika Ahuja (13559)
Armaan Jain(13581)
Anil Kumar(13530)
Anil Chaurasiya(13568)
Abhinandan Jain(13532)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY
HAMIRPUR-177005, HP (INDIA)

CERTIFICATE

I hereby certify that the work which is being presented in the B.Tech. Major Project Report entitled **"Desktop Monitoring over Local Network",** in partial fulfillment of the requirements for the award of the **Bachelor of Technology in Computer Science & Engineering** and submitted to the Department of Computer Science & Engineering of National Institute of Technology Hamirpur HP is an authentic record of our own work carried out during a period from August 2016 to November 2016 under the supervision of **Dr. T.P.Sharma, Associate Professor**, **CSE Department**.

The matter presented in this report has not been submitted by us for the award of any other degree elsewhere.

Signature of Candidate
Niharika Ahuja (13559)

Signature of Candidate
Armaan Jain(13581)

Signature of Candidate
Anil Chaurasiya(13568)

Signature of Candidate
Anil Kumar(13530)

Signature of Candidate
Abhinandan Jain(13532)

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Date:

Signature of Supervisor(s)
Dr. T.P.Sharma
Associate Professor, CSED
Project Supervisor

Head
Computer Science & Engineering Department
National Institute of Technology Hamirpur HP

Acknowledgement

“Mentoring is a brain to pick, an ear to listen and a push in the right direction". Hard work, dedication and sincerity have always played a key role in any venture. It is not just the brain that matters, but also the one which guides it. The completion of the project took endless hours of toil, had its moments of joy, frustration and anxiety but in the end everything seems to have made sense.

At this stage of life, it is often difficult to understand the wide spectrum of knowledge unless someone enlightens the right path. Hence, I take this opportunity to express my heartfelt gratitude to my mentor **Dr. T.P.Sharma**, who had unending faith in me. I would like to thank them for their immense interest, valuable guidance, and constant inspiration throughout the period of project. I would be grateful to them for providing me this opportunity to work under their esteemed guidance.

I thank my parents for their moral support, motivation and inspiration to carve out this project and above all, god for removing all hurdles in the way.

Abstract

The report presents Desktop Monitoring System over local network in which the user’s activities can be recorded without even letting them know this. Viewing their computers in real time from the admin’s desktop using Desktop Application. This monitoring system works invisibly, without slowing down their PCs. IP theft can also be caught using this system. Screenshots are sent from the user’s desktop to the admin’s desktop which can be used as an evidence later in case of theft or any illegal work done by any of the student or employee.

Table of Contents

List of Figures

CHAPTER-1

Introduction

1.1 Application Description

We usually come across areas where an admin or department or company head needs to monitor user work. This monitoring helps the authority to know about any unusual activity or any activity not supposed to be done in office premises, which is done by the employee. A desktop application is developed which acts as the interface of the admin dashboard, so that to monitor the work of employee more easily and effectively. Remote desktop monitoring needs a working internet connection that too having a high bandwidth. Well such system when monitoring on a large number of PC's proves to have a quite reasonable load on the network. Since, the system works by sending constant image snapshots of computer screen whenever the admin requests for the same using desktop application. These constant image transfers from the employee computers to the desktop of the admin. So, we propose an On Demand Desktop Monitoring system that monitors a PC on authority demand[1].

1.2 Driving factor for the proposed Application

There are large slots dedicated to laboratories for programming and application development in our curriculum. But they prove to be less productive than they actually should be. This is due to the ignorant nature of the students and also the lack of monitoring. The teachers cannot just continuously monitor each and every student's desktop. The driving factor behind this proposed application is to make this monitoring easy and effective.

There is a desktop application running on the teacher's desktop. The application itself monitors the PC's which are connected to the local network. Now, the teacher can select among the various options available and then click on Get Shot. The snapshot of the student's screen is sent to the teacher's desktop using socket programming.

The server will work in the following fashion:-

1. Wait until a client connects.

2. Send the size of the file that will be send.This is needed so the client knows how long he should read from the server.
3. Send the name of the file.
4. Send the file in (file size/buffer size) amount of chunks.
5. Close connection upon completion.

The client works like the server, only it receives the information and data.

1. Connect to the server.
2. Read the file size from the server.
3. Read the file name from the server.
4. Read the file from the server, chunk by chunk.
5. Close connection.

1.3 Problem Statement

The problem is of observing student activities during classes and exams. The impact is increased amount of unethical conduct or playing games during classes. A successful solution would be a system that enables the teacher to monitor students' activities and react on unethical conduct in a timely manner.

This concept can be used in a MNC where Boss does not have time to keep an eye on every activity of employee. It can even be used in Cyber Cafe where admin can check if any of the users is using the system for malfunctioning.

CHAPTER-2

Technologies Involved

2.1 Go Language

Go, or as its easily google-able moniker, Golang, is a typed systems programming language created by Google. It was designed to be in the vein of C/C++ with Garbage Collection, simpler to read and write unlike C++, strong opinions about how code should be built and strong primitives to enable effective concurrency. It was originally designed as a thought experiment between Robert Griesemer (V8 engine, Google Distributed File System), Rob Pike (Unix Team) and Ken Thompson (B Programming Language, c predecessor). Some of the features of Go language are:-

1. It is a statically typed, fast compiled language designed for modern development.
2. Produces binary executables, like C++.
3. Powerful at Concurrency with inbuilt primitives and features to allow systems to be built to scale and effectively use the computing power available.
4. Simple but Opinionated
5. Empowered by the open source community; all users are encouraged to put their go projects on github.com when setting up their environment
6. Automatic type inference (similar to dynamic languages, var x int =3 or x := 3 or y := “James” are all valid)
7. Automatic documentation. Like JDoc or PHPDoc, but no comments required, and even no commands. You can automatically have your code documented at the same level of quality as the standard library and inbuilt golang API’s.
8. Strict compilation checking, unused packages and variables will prevent compilation.
9. Strong standard library
10. Awesome at building API’s/backend
11. Cross platform
12. Useful tools for code linting, code generation, building, testing, etc.

Go doesn't have the following features:-

1. Classes -Golang has Structs, and struct functions, but there is deliberately no traditional class structure or inheritance. Objects follow an Interface simply by having the correct function
2. Generics. This is divisive to newcomers, but decisions made to build fast and smart binaries, no generic functions exist. This means if you have an getArea(number) function, you must have a getFloatArea(float) and getIntArea(int) if you wish to cover multiple types. Note that interfaces will allow alternatives to sidestep this issue with still some extra code but better organization.
3. Full memory control (due to garbage collection system)
4. Vendoring is still early and can be enabled via an experimental flag. You can pull packages but updating them independently; there are some community tools but I would expect something like a package.json to be added in the next release of Golang[7].

Advantages of using Go language:-

1. Golang combines a clean, fast, typed language that is particularly strong for web development and has built into its ethos and community one very clear message: **There is no need of a web development framework.** The standard library provides everything you need to build a web server, do routing, html templates, file serving, res/req handling, databases, etc..
2. It is very opinionated about how code should be written. For example the default linting tool decides exactly where there should be new lines (no more than one additional between function, etc.), automatically formatting brackets; code will not execute if you start your curly brace on a new line, **there are no semi-colons** due to the insistence on this rule. This means code is very readable, code can be easily worked on between teams and people can easily move between projects and be effective.
3. Compilation is very fast (usually a second or two), and produces a single binary. This means in terms of artifact handling for deployment you only need to push a new binary and run it. There are currently **many** tools to enable application server orchestration

(Puppet, Docker and the like), because a traditional Java Application might take someone half a day to setup just to run a hello world on a server.

4. Whilst “The Cloud” has enabled us to quickly build, clone and dynamically add servers to handle load of web applications, it often ends up with people horizontally scaling to cater for often poor performing applications. 10 applications servers to run a simple web app because peak loads require it is a very expensive cost to customers. Using a typed compiled language gives huge benefits in this domain. Concurrency is baked nicely into the language. Go has the very simple ‘go’ keyword, which is to say “go do this function in the background”, channels, which allow for buffered references of data for processing as well as parallelism by running go-routines on multiple cores asynchronously. These tools are not libraries or packages but instead primitives in the flow of the language/compiler. This means developers can write programs that with relatively little easy **take full use of the CPU available** and build non blocking, beautiful programs.
5. It’s **open source** and encourages all code and libraries built to solve problems to be part of the open source community. This would provide **an opportunity to make some excellent tools and libraries that could be shared with the larger software development community.**
6. Golang is currently the **goto programming language for dev ops development.** If we wish to not just support dev ops for clients but continue building tools that are flexible and usable for the larger dev ops community, Golang is an excellent language for doing so.

The Go language has built-in facilities, as well as library support, for writing concurrent programs. Concurrency refers not only to CPU parallelism, but also to asynchrony: letting slow operations like a database or network-read run while the program does other work, as is common in event-based servers. The primary concurrency construct is the goroutine, a type of light-weight process. A function call prefixed with the go keyword starts a function in a new goroutine. The language specification does not specify how goroutines should be implemented, but current implementations multiplex a Go process's goroutines onto a smaller set of operating system threads, similar to the scheduling performed in Erlang. While a standard library package featuring most of the classical concurrency control structures (mutex locks, etc.) is available, idiomatic concurrent programs instead prefer channels, which provide send messages between goroutines. Optional buffers store messages in FIFO order and allow sending goroutines to proceed before their messages are received. Channels are typed, so that a channel of type chan T can only be used to transfer messages of type T. Special syntax is used to operate on them; <-ch is an expression that causes the executing goroutine to block until a value comes in over the channel ch, while ch <- x sends the value x (possibly blocking until another goroutine receives the value). The built-in switch-like select statement can be used to implement non-blocking communication on multiple channels; see below for an example[7].

Go has a memory model describing how goroutines must use channels or other operations to safely share data. The existence of channels sets Go apart from actor model-style concurrent languages like Erlang, where messages are addressed directly to actors (corresponding to goroutines); the actor style can be simulated in Go by maintaining a one-to-one correspondence between goroutines and channels, but the language allows multiple goroutines to share a channel, or a single goroutine to send and receive on multiple channels. From these tools one can build concurrent constructs like worker pools, pipelines (in which, say, a file is decompressed and parsed as it downloads), background calls with timeout, "fan-out" parallel calls to a set of services, and others. Channels have also found uses further from the usual notion of interprocess communication, like serving as a concurrency-safe list of recycled buffers, implementing coroutines (which helped inspire the name goroutine), and implementing iterators. Concurrency-related structural conventions of Go (channels and alternative channel inputs) are derived from Tony Hoare's communicating sequential processes model. Unlike previous concurrent programming languages such as Occam or Limbo (a language on which Go co-designer Rob Pike worked), Go does not provide any built-in notion of safe or verifiable concurrency. While the communicating-processes model is favored in Go, it is not the only one: all goroutines in a program share a single address space. This means that mutable objects and pointers can be shared between goroutines[7].

2.2 Nmap

Nmap ("Network Mapper") is a free and open source (license) utility for network discovery and security auditing. Many systems and network administrators also find it useful for tasks such as network inventory, managing service upgrade schedules, and monitoring host or service uptime.

Nmap uses raw IP packets in novel ways to determine what hosts are available on the network, what services (application name and version) those hosts are offering, what operating systems (and OS versions) they are running, what type of packet filters/firewalls are in use, and dozens of other characteristics. It was designed to rapidly scan large networks, but works fine against single hosts. Nmap runs on all major computer operating systems, and official binary packages are available for Linux, Windows, and Mac OS X. In addition to the classic command-line Nmap executable, the Nmap suite includes an advanced GUI and results viewer, a flexible data transfer, redirection, and debugging tool (Ncat), a utility for comparing scan results, and a packet generation and response analysis tool (Nping)[6]. Some of its features are:-

1. **Flexible**: Supports dozens of advanced techniques for mapping out networks filled with IP filters, firewalls, routers, and other obstacles. This includes many port scanning mechanisms (both TCP & UDP), OS detection, version detection, ping sweeps, and more.
2. **Powerful**: Nmap has been used to scan huge networks of literally hundreds of thousands of machines.
3. **Portable**: Most operating systems are supported, including Linux, Microsoft Windows, FreeBSD, OpenBSD, Solaris, IRIX, Mac OS X, HP-UX, NetBSD, Sun OS, Amiga, and more.
4. **Easy**: While Nmap offers a rich set of advanced features for power users, you can start out as simply as "nmap -v -A *targethost*". Both traditional command line and graphical (GUI) versions are available to suit your preference. Binaries are available for those who do not wish to compile Nmap from source.
5. **Free**: The primary goals of the Nmap Project is to help make the Internet a little more secure and to provide administrators/auditors/hackers with an advanced tool for exploring their networks. Nmap is available for free download, and also comes with full source code that you may modify and redistribute under the terms of the license.
6. **Well Documented**: Significant effort has been put into comprehensive and up-to-date man pages, whitepapers, tutorials, and even a whole book.
7. **Supported**: While Nmap comes with no warranty, it is well supported by a vibrant community of developers and users. Most of this interaction occurs on the Nmap mailing

lists. Most bug reports and questions should be sent to the nmap-dev list. You can also find Nmap on Facebook and Twitter.

8. **Acclaimed**: Nmap has won numerous awards, including "Information Security Product of the Year" by Linux Journal, Info World and Codetalker Digest. It has been featured in hundreds of magazine articles, several movies, dozens of books, and one comic book series.
9. **Popular**: Thousands of people download Nmap every day, and it is included with many operating systems (Redhat Linux, Debian Linux, Gentoo, FreeBSD, OpenBSD, etc). It is among the top ten (out of 30,000) programs at the Freshmeat.Net repository. This is important because it lends Nmap its vibrant development and user support communities.

2.3 Scrot

Scrot (SCReenshOT) is an open source, powerful and flexible, command line utility for taking screen shots of your Desktop**,** Terminal or a Specific Window manually or automatically by Cron job. Scrot is similar to Linux 'import' command, but uses 'imlib2' library to capture and save images. It supports multiple image formats (JPG, PNG, GIF, etc), which you can specify while taking screen shots by using the tool. Some features of Scrot are:-

1. With scrot we can take screen shots easily without any additional work.
2. We can also optimize the quality of the screen shots image (with the -q switch, followed by a quality level between 1 and 100. The default quality level is 75.
3. It is very easy to installation and use.
4. We can capture a specific window or a rectangular area on the screen with the help of switch.
5. Can get all screen shots in a particular directory and also can store all screen shots in a remote PC or network server.
6. Can monitor all Desktop PC in admin absent and prevent to unwanted activities[2].

CHAPTER-3

Design and Diagrams

3.1 Context Diagram

A **system context diagram** (SCD) in software engineering and systems engineering is a diagram that defines the boundary between the system, or part of a system, and its environment, showing the entities that interact with it. This diagram is a high level view of a system. It is similar to a block diagram. System Context Diagrams represent all external entities that may interact with a system. Such a diagram pictures the system at the center, with no details of its interior structure, surrounded by all its interacting systems, environments and activities. The objective of the system context diagram is to focus attention on external factors and events that should be considered in developing a complete set of systems requirements and constraints.

Figure 3.1 represents the context free diagram of our system.

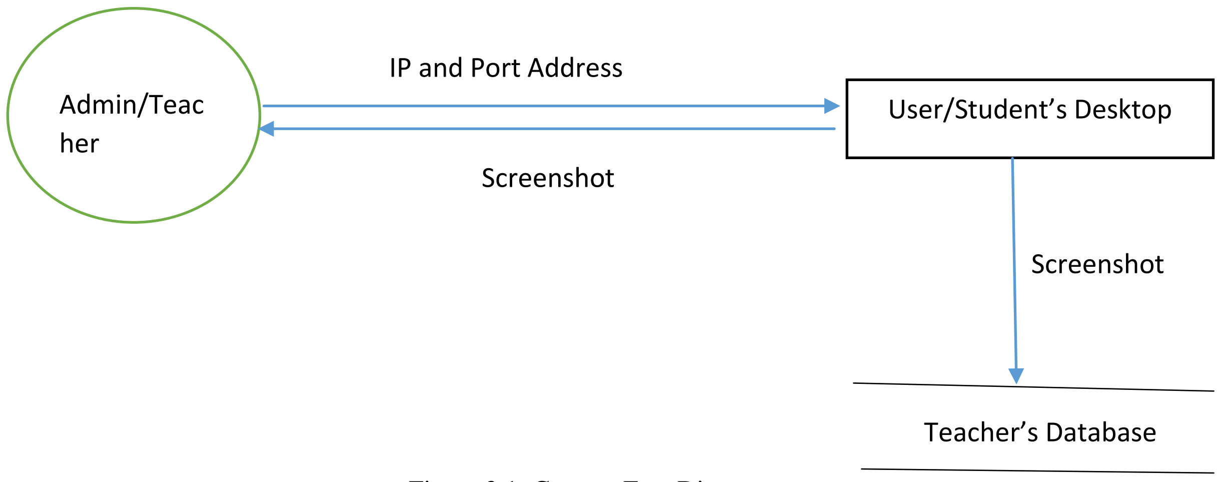


Figure 3.1: Context Free Diagram

3.2 Data Flow Diagram(DFD)

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. A DFD is often used as a preliminary step to create an overview of the system, which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).

A DFD shows what kind of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of process or information about whether processes will operate in sequence or in parallel (which is shown on a flowchart). Figure 3.2 represents the DFD for our system. Using the port scanning script, the available monitors on the network are stored in the database. Also, the screenshots taken from student's desktop are stored in another database.

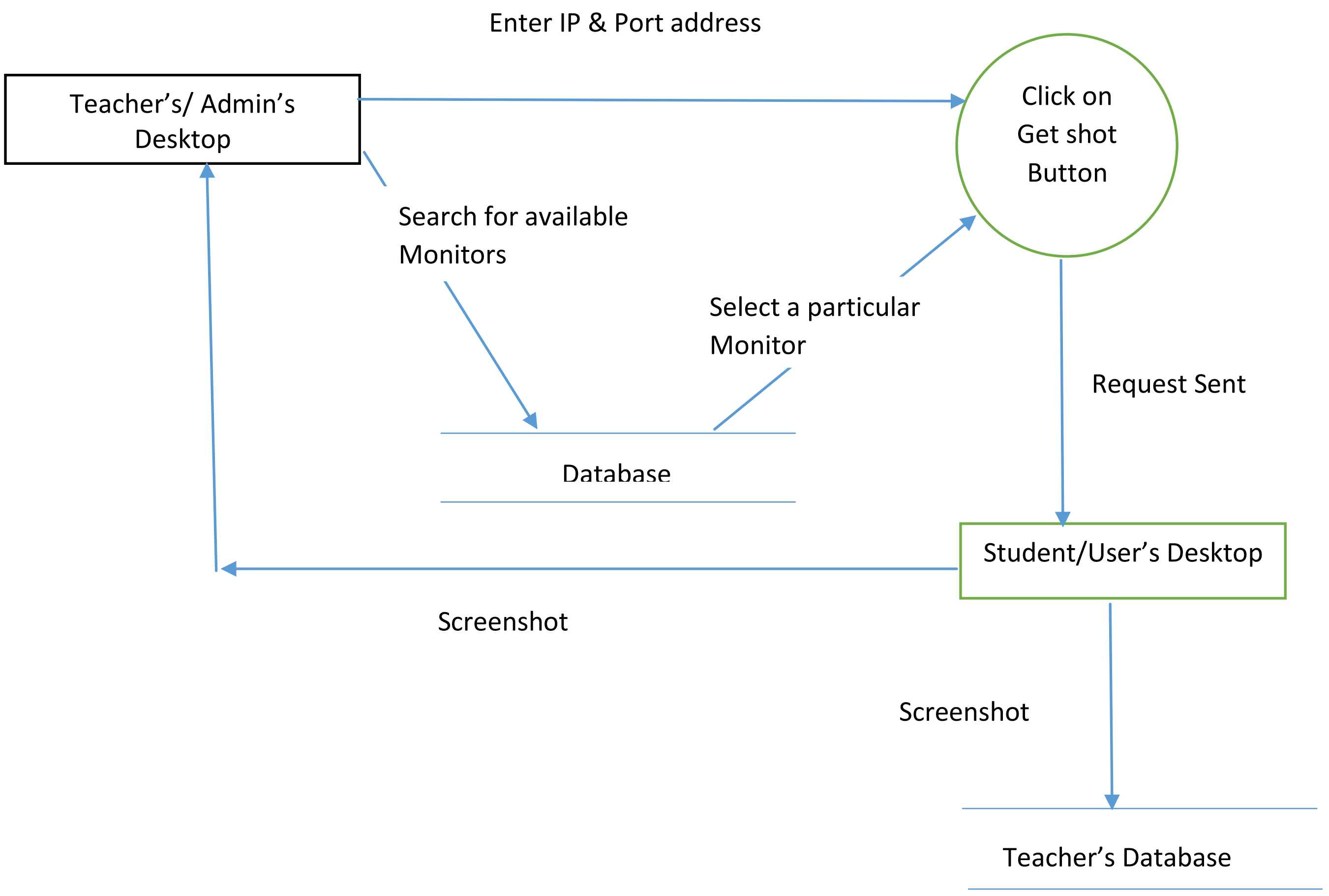


Figure 3.2: Data Flow Diagram

3.3 E-R Diagram

An entity–relationship model (ER model) describes inter-related things of interest in a specific domain of knowledge. An ER model is composed of entity types (which classify the things of interest) and specifies relationships that can exist between instances of those entity types. Figure

3.3 represents the ER diagram in which there are only 2 entities Teacher and Student, student having 2 attributes including port and IP address.

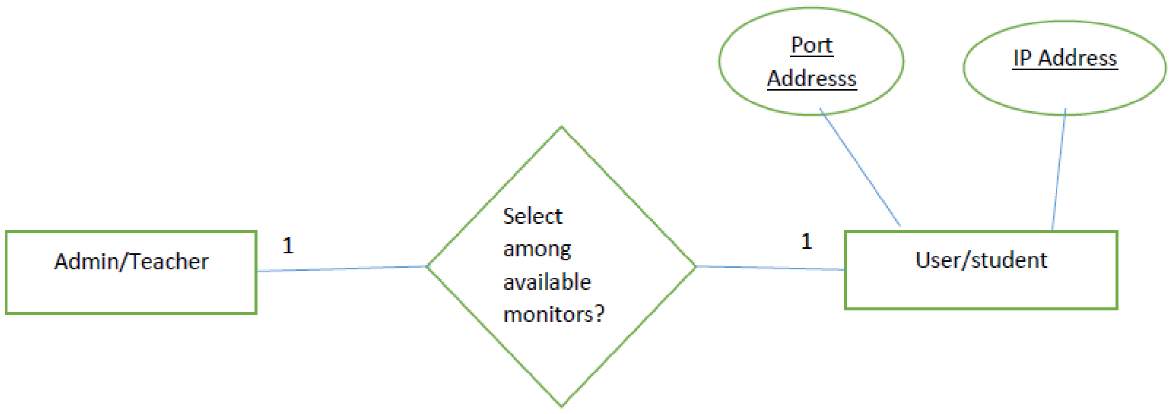


Figure 3.3: ER Diagram

3.4 Use Case Diagram

Use case diagrams are usually referred to as behavior diagrams used to describe a set of actions (use cases) that some system or systems (subject) should or can perform in collaboration with one or more **external users** of the system (actors). Each use case should provide some observable and valuable result to the actors or other stakeholders of the system.

Use case diagrams are used to specify(external) **requirements**, required usages of a system under design or analysis (subject) to capture what the system is supposed to do, the **functionality** offered by a subject, what the system can do. Figure 3.4 represents the use case diagram of our system. The main functions in our system are shown in this figure.

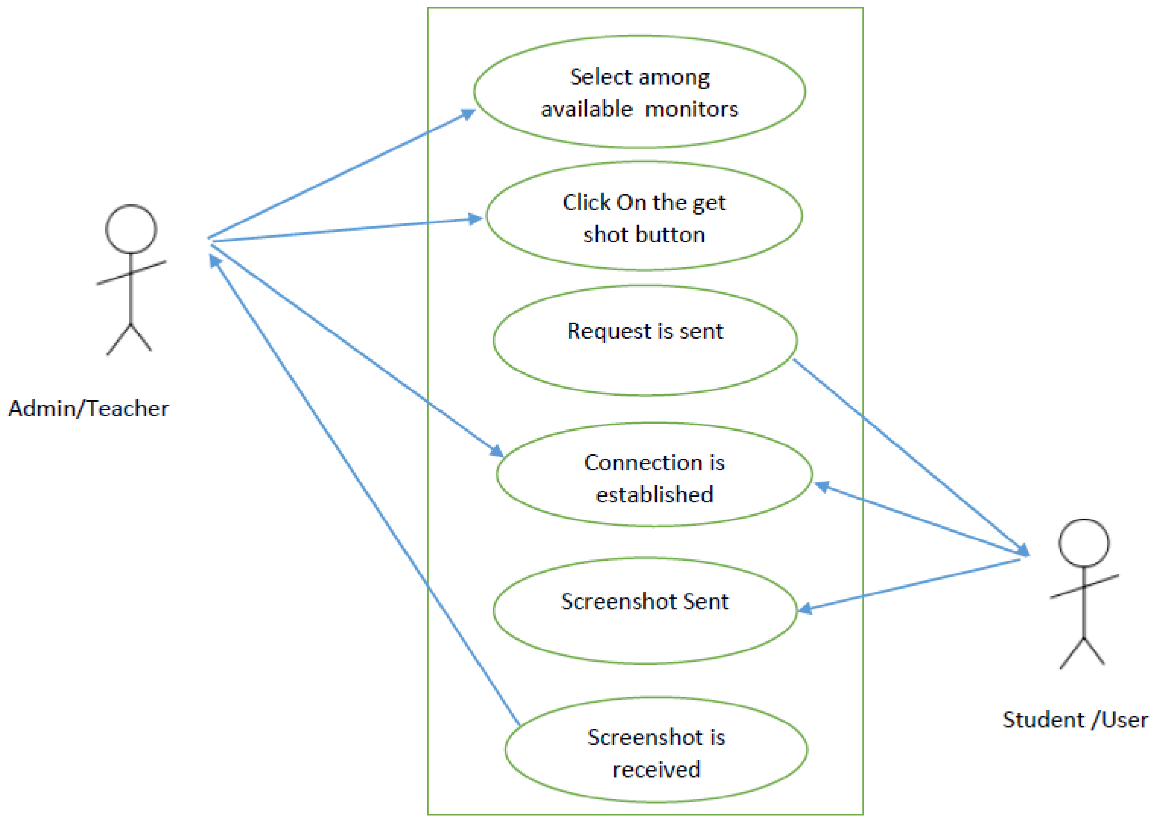

Select among available monitors
Click On the get shot button
Request is sent
Connection is established
Screenshot Sent
Screenshot is received
Admin/Teacher
Student /User

Figure 3.4: Use case Diagram

3.5 Sequence Diagram

A sequence diagram is an interaction diagram that shows how objects operate with one another and in what order. It is a construct of a message sequence chart.

It shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner. Figure 3.5 represents the sequence diagram of our system.

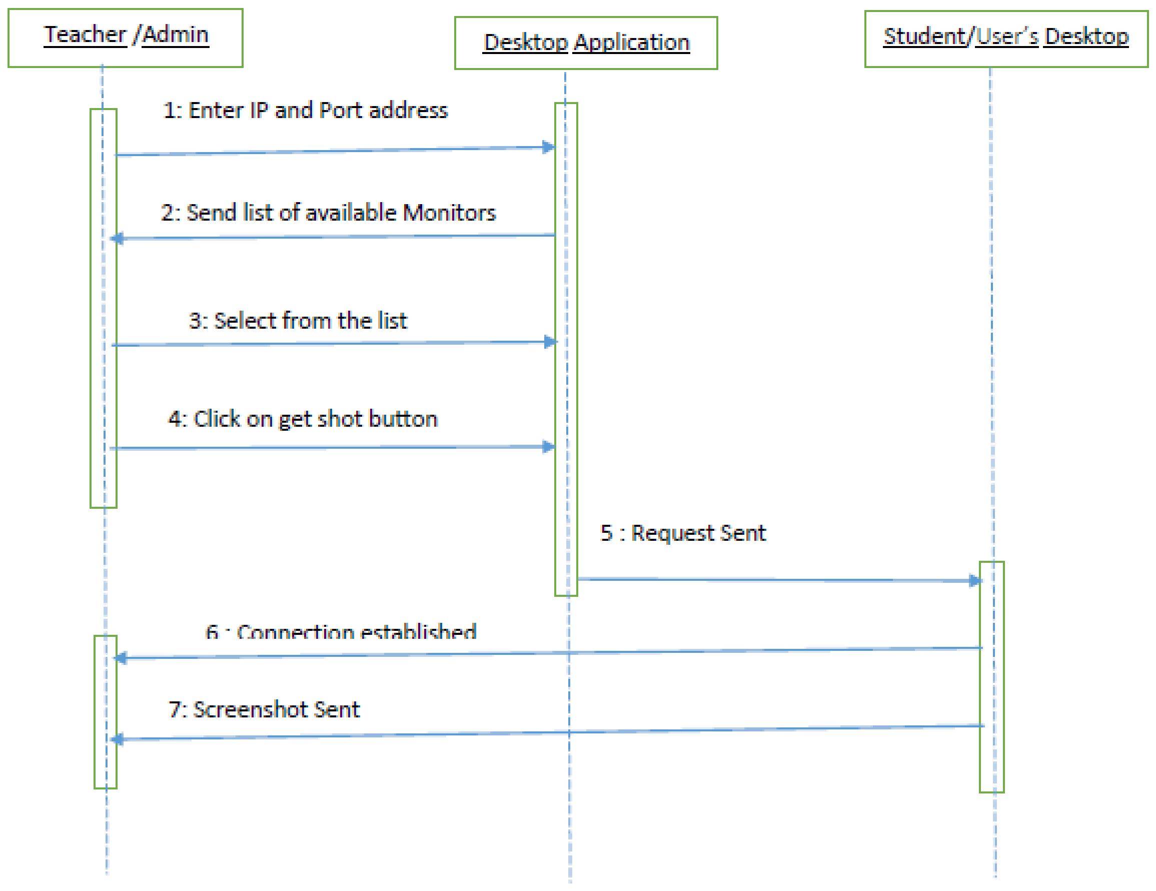

Teacher /Admin
Desktop Application
Student/User's Desktop
1: Enter IP and Port address
2: Send list of available Monitors
3: Select from the list
4: Click on get shot button
5 : Request Sent
6 : Connection established
7: Screenshot Sent

Figure 3.5: Sequence Diagram

3.6 Activity Diagram

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. It is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system. So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent. Activity diagrams deals with all type of flow control by using different elements like fork, join etc. Figure 3.6 represents the activity diagram of our system.

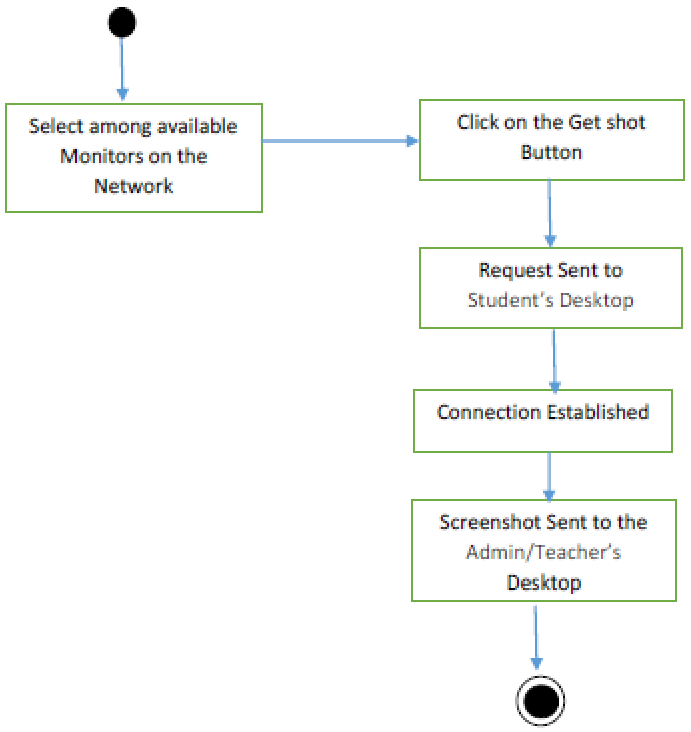

Select among available Monitors on the Network
Click on the Get shot Button
Request Sent to Student's Desktop
Connection Established
Screenshot Sent to the Admin/Teacher's Desktop

Figure 3.6: Activity Diagram

CHAPTER-4

Implementation

The application contains 3 major modules i.e.

1. To invoke request to the hosts for the screenshot.
2. Hosts- Computers on the network which are being monitored.
3. Screenshot- Script which takes screenshots on hosts and which receives screenshots on the admin side.

There are two entities in the application-:

1. Administrator
2. The host being monitored

4.1 Administrator

The administrator is the entity around which application is built, to facilitate monitoring of certain hosts of interest on the network to the administrator. The administrator can establish connection with multiple hosts of interest on the network and then can invoke request to get the real time screenshot from any of these hosts while maintaining a asynchronous communication with them. The major function code in this module is-

1. Connection Establish() -To establish the connection with multiple hosts,a separate thread is created for each connection with a host.
2. InvokRequest() -After establishing connection this function can be used to invoke request to get the real time snapshot of the connected hosts along with the timestamp.
3. ReceiveSnap() -This is the function which serves the admin's request of screenshot by receiving screenshot sent over by the host in ".png" format.
4. ShowSnap() -After being received by the ReceiveSnap() this function shows up the screenshot of the host on the admin's screen.

4.2 Major Features

The major features of this module is that it uses asynchronous communication with the hosts to handle multiple hosts efficiently with all the hosts thus enabling scalability of the application and also to request the snapshot anytime from any of the hosts.

Along with these a port sensing script is also embedded within these module which separates the intended hosts form other hosts on the network, this script senses the hosts which are running a corresponding program of PC monitoring and stores them as a list in a file for the admin to connect with any of these available hosts.

Also it maintains a separate thread for each connected hosts which avails concurrent interaction with all the connected hosts. Figure 4.1 represents the detailed working of our system.

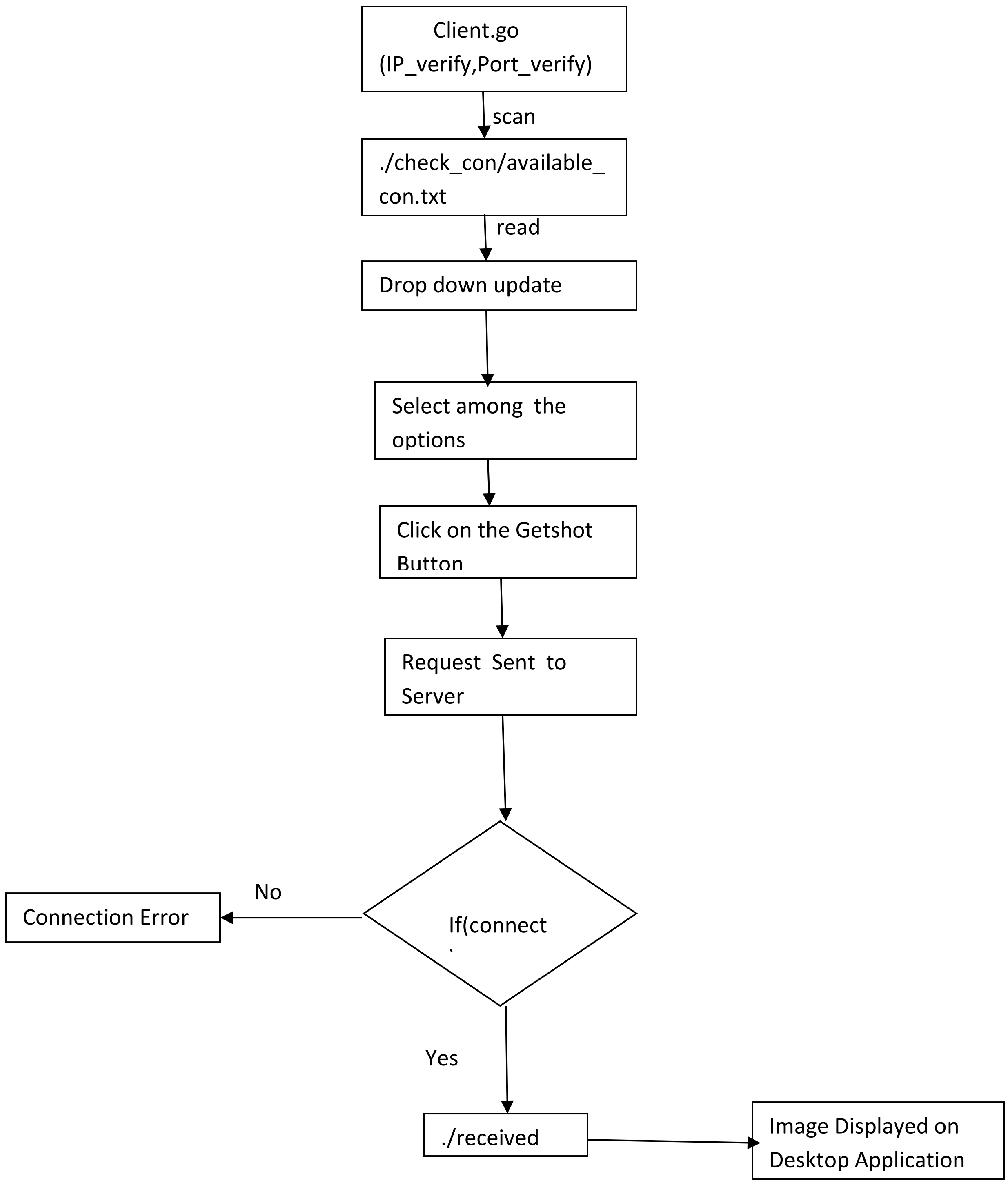


Figure 4.1: Detailed Working

4.3 Hosts

These are the hosts which are being monitored on the network, more broadly a separate process running on each of the machine under admin's control which serves the request of admin to send real time screenshots over the network. It works in the following manner-

1. Wait until the admin connects
2. Send the size of the image that will be send. This is needed so the admin knows how long he should read from the host
3. Send the name of the image along with the timestamp
4. Send the file(image) in (file size/buffer size) amount of chunks
5. Close connection upon completion

It setups asynchronous communication with the admin, and then whenever admin requests for the screenshot, it sends the screenshot in chunks of buffer size to the admin. Upon successfully sending the screenshot it closes the connection.

It puts an identifier to the screenshot being sent by appending its IP address and timestamp with the screenshot for easy and efficient receiving at the admin side.

4.4 Scrot Script

Scrot (SCReenshOT) is an open source, powerful and flexible, command line utility for taking screen shots of your Desktop, Terminal or a Specific Window manually or automatically by Cron job. Scrot is similar to Linux 'import' command, but uses 'imlib2' library to capture and save images. It supports multiple image formats (JPG, PNG, GIF, etc), which you can specify while taking screen shots by using the tool.

Scrot script is used to capture the screenshot on the hosts side, along with the features it provide it makes the process of sending and receiving these screenshot a little bit more easier.

4.5 Multithreading

The connection with the multiple computers on the network is each put into a separate thread by the admin to achieve concurrency.

Concurrency-Large programs are often made up of many smaller sub-programs. For example a web server handles requests made from web browsers and serves up HTML web pages in response. Each request is handled like a small program.

It would be ideal for programs like these to be able to run their smaller components at the same time (in the case of the web server to handle multiple requests). Making progress on more than one task simultaneously is known as concurrency. Go has rich support for concurrency using go routines and channels.

4.6 Goroutines

A goroutine is a function that is capable of running concurrently with other functions. To create a goroutine we use the keyword `go` followed by a function invocation:

```
package main

import "fmt"

func f(n int) {
  for i := 0; i < 10; i++ {
    fmt.Println(n, ":", i)
  }
}

func main() {
  go f(0)
  var input string
  fmt.Scanln(&input)
}
```

This program consists of two goroutines. The first goroutine is implicit and is the main function itself. The second goroutine is created when we call go f(0). Normally when we invoke a function our program will execute all the statements in a function and then return to the next line following the invocation. With a goroutine we return immediately to the next line and don't wait for the function to complete. This is why the call to the Scanln function has been included; without it the program would exit before being given the opportunity to print all the numbers.

Goroutines are lightweight and we can easily create thousands of them. We can modify our program to run 10 goroutines by doing this:

```
func main() {
  for i := 0; i < 10; i++ {
    go f(i)
  }
  var input string
  fmt.Scanln(&input)
}
```

You may have noticed that when you run this program it seems to run the goroutines in order rather than simultaneously. Let's add some delay to the function using time. Sleep and rand.Intn:

```
package main

import (
  "fmt"
  "time"
  "math/rand"
)

func f(n int) {
  for i := 0; i < 10; i++ {
    fmt.Println(n, ":", i)
    amt := time.Duration(rand.Intn(250))
    time.Sleep(time.Millisecond * amt)
  }
}

func main() {
  for i := 0; i < 10; i++ {
    go f(i)
  }
  var input string
  fmt.Scanln(&input)
}
```

f prints out the numbers from 0 to 10, waiting between 0 and 250 ms after each one. The goroutines should now run simultaneously.

4.6 Features

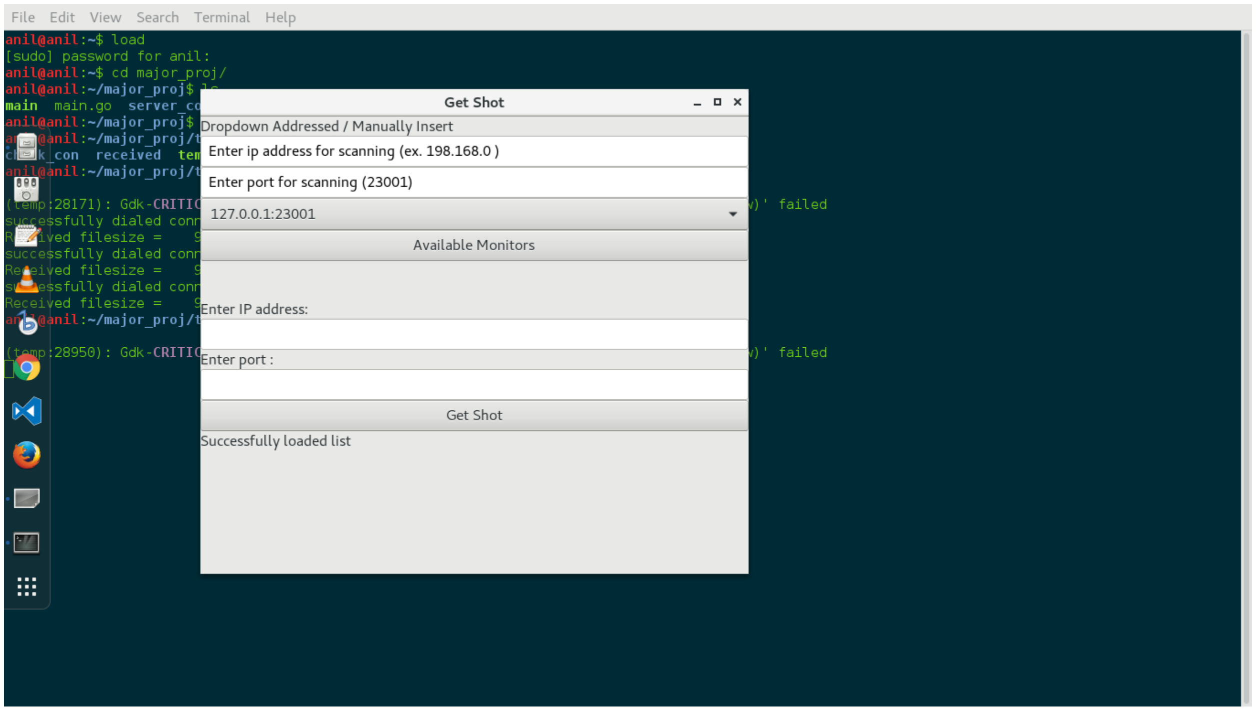
Thus the major features of the final built desktop application are:-

1. Remote PC monitoring- The application enables the administrator to monitor multiple hosts on the network with the ease of sitting in their offices, or the lab itself as long as they are on the same network.
2. Real time Screenshot- The admin can request the multiple real time screenshots from the connected hosts to monitor their activities.
3. Asynchronous Communication- The application establishes an asynchronous communication with all the host without the overhead of handshakes and acknowledgements and does the task quite efficiently thus is suitable for the application and provides scalability to the application.

Automated and Manual Input-The user have the option to enter the ip address and port of any of the computer manually or can chose from the available connected host on the network and can thus get the screenshot thus making the application easy to use.

CHAPTER-5

Application Screenshots



```
anil@anil:~$ load
[sudo] password for anil:
anil@anil:~$ cd major_proj/
anil@anil:~/major_proj$ ls
main  main.go  server_co
anil@anil:~/major_proj$
anil@anil:~/major_proj/t
check_con  received  tem
anil@anil:~/major_proj/t

(temp:28171): Gdk-CRITIC                                    v)' failed
successfully dialed conn
Received filesize =    9
successfully dialed conn
Received filesize =    9
successfully dialed conn
Received filesize =    9
anil@anil:~/major_proj/t

(temp:28950): Gdk-CRITIC                                    v)' failed
```

Dropdown Addressed / Manually Insert

Enter ip address for scanning (ex. 198.168.0)

Enter port for scanning (23001)

127.0.0.1:23001

Enter IP address:

Enter port :

Successfully loaded list

Figure 5.1: The screenshot above shows the main screen of the application.

Components of our proposed application are:-

1. Initiation

Upon initiation of the application, it gets the list of all the available PCs on the network and stores their corresponding IP addresses and port number in a file.

2. Drop Down Menu

The dropdown menu "Available Monitors" shows the list of all the online PCs whose IP addresses are stored on the startup of application. The admin can chose from any of these available host to get and then can get a screenshot.

3. Manual Text Field

There are two text fields in the application:-

1. Enter IP address-Here a recognized IP address of any of the PC on the network can be entered for the retrieval of the screenshot.
2. Enter Port-Here corresponding port number to the IP address has to be entered.

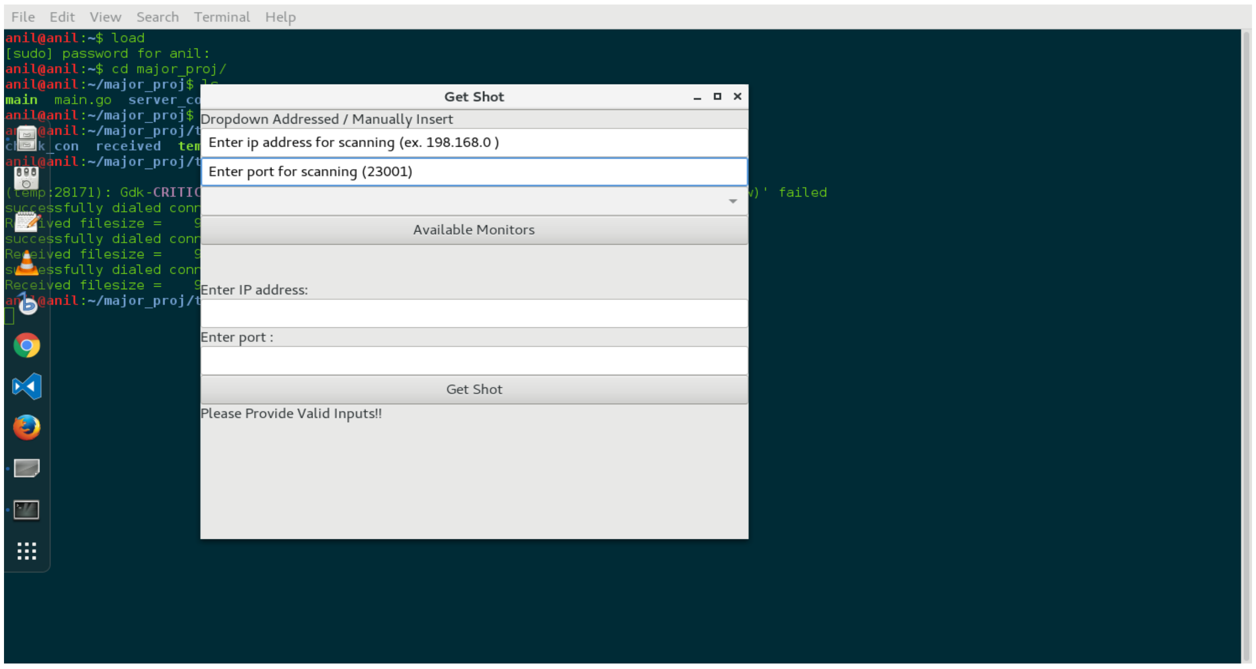
4. Get Shot

After successfully choosing one of the available PC's or entering a valid IP address and port number, this button can be used to get the screenshot which pops up in a separate window on the admin's screen.

5. Status Label

Below the Get Shot button, there is a status label which shows the error like-

Enter valid inputs on entering an unrecognized IP address as shown in Figure 5.2. And it also shows success on the retrieval of the screenshot.



```
anil@anil:~$ load
[sudo] password for anil:
anil@anil:~$ cd major_proj/
anil@anil:~/major_proj$ ls
main  main.go  server_co
anil@anil:~/major_proj$
anil@anil:~/major_proj/t
check_con  received  tem
anil@anil:~/major_proj/t

(temp:28171): Gdk-CRITIC                                      w)' failed
successfully dialed conn
Received filesize =    9
successfully dialed conn
Received filesize =    9
successfully dialed conn
Received filesize =    9
anil@anil:~/major_proj/t
```

Dropdown Addressed / Manually Insert

Enter ip address for scanning (ex. 198.168.0)

Enter port for scanning (23001)

Enter IP address:

Enter port :

Please Provide Valid Inputs!!

Figure 5.2 Error on entering Invalid Input

CHAPTER-6

Conclusion

This application is our effort to provide a monitoring facility to people like faculty members in educational institutions, lab heads, managers in the organization to monitor the students or employees in a timely fashion by the medium of real time screenshots. The application addresses the issue of scalability and reduced overhead, because the administrator doesn't have to look for those long video feeds to catch any unethical activity instead can request screenshot of the suspected individual any time.

With features such as real time screenshot capturing, concurrent connection with multiple computers on the network, a list of all the available computers to chose from as well as option of manual input, the application is easy to use with a really simple interface thus successfully serving its purpose.

It is a major improvement of the tradition video feed techniques which is cumbersome and time consuming to monitor and track the unethical activities, it provides an efficient and an easy way to monitor the intended audience on demand without having to undergo the pain of constantly looking at the video feed for monitoring purpose.

CHAPTER-7

Future Scope

The application is a light version of PC monitoring over the local internet. It has a major scope and can be extended to serve the other needs of the institutional or organizations head by adding functions like input unit controls like cursor control or privilege to simulating a key press on the keyboard to close the not intended activities on the culprit's computer.

In addition data mining techniques can be added to the application, which can fetch the history of the browser of the host on the network categorized into various classifications which can then be used to plot a graph for the specific individual to track the actual productive work he/she has performed and the non productive aspect too.

References

[1] PC Control Over Internet | NevonProjects, *Nevonprojects.com*, 2016.

[2] S. Singh and V. Posts, "Scrot: A Command Line Tool to Take Desktop/Server Screenshots Automatically in Linux", *Tecmint.com*, 2016.

[3] G. Vlasselaer, "Golang transfer a file over a TCP socket - Mr.Waggel", *Mrwaggel.be*, 2016.

[4] Remote System Monitoring and Controlling via Web based Mobile or Desktop Application," electrofriendscom. [Online]. Available: http://electrofriends.com/projects/computer-programming/remote-system-monitoring-controlling-web-based-mobile-desktop-application

[5] "Activity Monitor - Employee Monitoring Software, Network Monitoring | SoftActivity," SoftActivity. [Online]. Available: https://www.softactivity.com/activity-monitor/. [Accessed: 13-Dec-2016].

[6] "Nmap," Nmap: the Network Mapper - Free Security Scanner. [Online]. Available: https://nmap.org/. [Accessed: 13-Dec-2016].

[7] J. O'Toole, "Golang and why it matters," Medium, 2016. [Online]. Available: https://medium.com/@jamesotoole/golang-and-why-it-matters-1710b3af96f7. [Accessed: 13-Dec-2016].