# NETDOG - CONFIGURATION MANAGEMENT AND MONITORING SYSTEM


**Mini Project 2018**

Done By

**ASWIN BABU K**

Guided By

**Prof. Baby Syla L**

**Assistant Professor**

**Department of Computer Applications**

**College of Engineering**

**Trivandrum-695016**

# ABSTRACT

The NetDog Project started with the aim to develop a software system that can bring up and down computers on the network. It has now added a number of abilities to its feature list, including the ability to execute scripts or commands across machines on the network and the ability to copy files to machines on the network without relying on third party protocol. The system is developed to be fully extensible so that developers can extend the system as per their requirements through plugins.

# ACKNOWLEDGEMENT

If words are considered as symbols of approval and tokens of acknowledgment, then let words play the heralding role of expressing our gratitude.

First of all I would like to thank God almighty for bestowing us with wisdom,courage and perseverance which had helped us to complete this project ***NET-DOG - CONFIGURATION MANAGEMENT AND MONITORING SYSTEM***. This project has been a reality as a result of the help given by a large number of personalities.

I would like to remember with gratitude **Prof. Jose T Joseph**, Head Of Department Department of Computer Applications, College of Engineering, Trivandrum for the encouragement and guidance rendered.

I express our sincere thanks to **Prof. Baby Syla**,Assistant Professor, Department of Computer Applications, College of Engineering Trivandrum for his valuable guidance, support and advices that aided in the successful completion of my project.

Finally, We wish to express our sincere gratitude to all our friends, who directly or indirectly contributed in this venture.

ASWIN BABU K

# Contents

# List of Figures

# Chapter 1

# Introduction

The netdog project started with the aim of designing a program which can easily bring computers up and down remotely. The idea came from the realization that, quite a few computers were left powered up when the college lab closes for the day. Thus the initial name "Project Green".

A number of features were added to the feature list, most importantly tracking computers even if their IP addresses changed, the ability to execute commands or scripts on a specified range of machines with a single command, and the ability to copy files to a range of machines without requiring protocols such as SFTP or FTP.

The abilities does not end there and netdog can provides many notable features such as early warning of HDD failure on the machines on the network. The exhaustive list of features is listed in later section. Once completed, netdog will be a completely extensible system to which features an be easily added through plugins.

## 1.1 Licence

NetDog is licensed under the MIT License. A copy is attached below

**MIT License**

# Chapter 2

# Requirement Analysis

## 2.1 Purpose

The purpose of this system is to build a configuration management and monitoring system that is dead simple to use. Highly efficient and feature rich configuration management systems such as Ansible exists. But the problem is that they have very steep learning curve. This is also the case with monitoring systems such as Nagios which require considerable amount of setup from the part of admins.

The learning curve makes these systems unsuitable to small office environments, schools or colleges where the systems are managed by regular employees. NetDog is dead simple, with zero learning curve. As long as a person is aware of the command he/she wishes to execute on the remote machine, the person is ready to use NetDog.

## 2.2 Overall Description

NetDog has a client server architecture. The server is responsible for issuing commands to the clients and is to be used by the administrator of the network. The client application is to be run on machines on the network to be administered.

After installing NetDog server or client on a machine, a unique public-private key pair for the machine is generated which is then used for uniquely identifying the machine and securing data transmission between the client and server.

Once the server program is up and running, it listens on the port 1337 for connections from clients. Once the client program is up, it starts listening on port 1994. These ports serve dual purpose of facilitating communication and allowing the identification of server and clients from the rest of the machines on the network. Both NetDog server and client are daemons. They are system services which remain in memory and automatically starts during system boot.

When a client starts for the first time, it looks for active servers on the network. When it finds one, it starts the pairing procedure. During the pairing process, the client sends its hostname and public-key. The server in turn provides the client with it's public key. These keys are then used for identification and encrypted communication between machines.

The client and server uses public key encryption to identify and secure the communication between them. The network admininstrator can issue commands from the server machine which will then be sent to all the clients on the network. A client can also contact the server occasionally, for example if the client detects undseriable network traffic or if the system is overheating.

NetDog is capable of shutting down all the clients on the network at once by remotely executing the shutdown command. It is also able to power up systems which support remote Wake-On-LAN feature, by sending magic packets.

### 2.2.1 Product Functions

- Bring all computers on the network up and down remotely
- Execute commands/scripts remotely on machines
- Copy files to remote machines without third party protocols
- Track and identify clients through IP changes
- Secure client-server communication using public key encryption
- List all machines on the network which are not clients (detect intruders)
- Centralized logging of all data regarding clients
- Web interface

### 2.2.2 Hardware Requirements

- Intel Pentium IV or equivalent CPU
- 512 MB or more RAM
- 100 mbps Network Interface Card

### 2.2.3 Software Requirements

- Linux
- Python 3
- pip

## 2.3 Functional Requirements

The system is should be designed to accept communication requests from many clients at once. For this, a multi threaded server is necessary. Also, network outages can occur during the operation. The server should be resilient enough so that, it checks the status of the connection every once in a while and restarts the communication process once the network is up and running again. Also, the functions should extensively log themselves so that in case of a system failure, the culprit can be easily found.

## 2.4 Performance Requirements

The system would need a gigabit ethernet controller for the server to make sure that it can properly handle connections from multiple clients on the network. The client can work satisfactorily well even on old hardware such as a 10 mbit network card. There are no specific requirements for the CPU or the rest of the hardware. The machine must be powerful enough to run a recent version of Linux, which means any usable computing hardware would do.

# Chapter 3

# Design And Implementation

## 3.1 Overall Design

NetDog follows the client-server architecture. What this means is that, NetDog has both a client part and a server part. The server part is meant to be run on the machine used by the system administrator, while the client part runs on the rest of the machines on the network.

Both the server and client listens for connections from each other, allowing the server to execute commands on clients and the client to report back any progress. But the server has an additional capability that the clients lack, an easy to use web interface that can be used by the administrator to control the machines and configure the server.

## 3.2 User Interfaces

NetDog is a very easy to use system. One of the main aims while designing the system was to abstract as much lower level details of the system as possible from the user. There is no user interface for the client machines. All of the system is controlled from the server. The server provides a command line interface and a very convinient web interface for administration. The whole system can be operated and configured from these interfaces.

## 3.3 Hardware Interfaces

The system tries hard not to reinvent the wheel. Thus, it uses existing Linux subsystems as much as possible. As a side effect, NetDog can work with any networking hardware supported by Linux. If the clients on the network can succesfully communicate over the network using their network interface cards, you are good to go.

### 3.4 Communications Interfaces

The system uses the TCP/IP system for communication between server and clients. On top of TCP, the system uses a home grown higher level "NetDog" protocl for coordinating the client machine and servers. It is the NetDog protocol that ensures efficient communication and security. The protocol specification is under heavy development and the exact specification will be relased in further revisions of this document.

### 3.5 System Design

On a lower level, the design of NetDog is a bit different. NetDog is splitted into three modules, the server module, the client module and the web server module. The three of them has distinct functions. But the interesting fact is, both the client module and the server module share the same underpinnings. All the functions they use are from a library called libgreen, that was built specifically for use in NetDog.

#### 3.5.1 NetDog Server

The NetDog server module acts as the server. The server module resides in the file "netdog_server.py" It uses functions part of the libgreen library to setup a server working on the netdog protocol.

It listens for connections from clients on port 1337 and also provides the web interface. The server is completeley multithreaded and this allows it to perform tasks involving hundreds of machines at a surprisingly quick rate.

#### 3.5.2 NetDog client

The NetDog client module is the part of NetDog that is run on the client machines. The client module listens for connections from the server at port 1994. Once a connection is received, it is handled as per the command it contains.

#### 3.5.3 Web Server

The webserver module provides the web interface for use by the administrator. Written using the bottle library, it resides in the file "web_server.py". The web server also uses the libgreen library extensively, to serve its clients. It is not directly invoked from the file, rather the contents of "web_server.py" is imported into "netdog_server.py" and started as a separate thread.

### 3.5.4 NetDog protocol

The NetDog client and server operates using the NetDog protocol. NetDog protocol is a custom built protocol, that has been developed on top of the Sockets library part of standard Python installation. The mechanism as well as the policies for the protocol is defined inside the libgreen library.

The NetDog protocol defines the structure of messages passed between the server and the clients. The message has primarily three parts:

- Hostname, which is the hostname of the sender
- encrypted flag, whether the message is encrypted or not
- data, which is further divided into command and payload

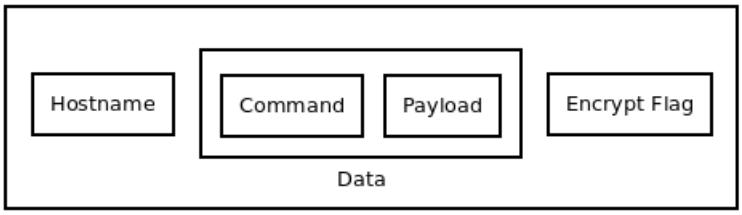


Figure 3.1: NetDog protocol message structure

### 3.5.5 DataFlow Diagram

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modelling its process aspects. A DFD is often used as a preliminary step to create an overview of the system without going into great detail, 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, how the data will advance through the system, and where the data will be stored. It does not show information about process timing or whether processes will operate in sequence or in parallel, unlike a traditional structured flowchart which focuses on control flow, or a UML activity workflow diagram, which presents both control and data flows as a unified model.

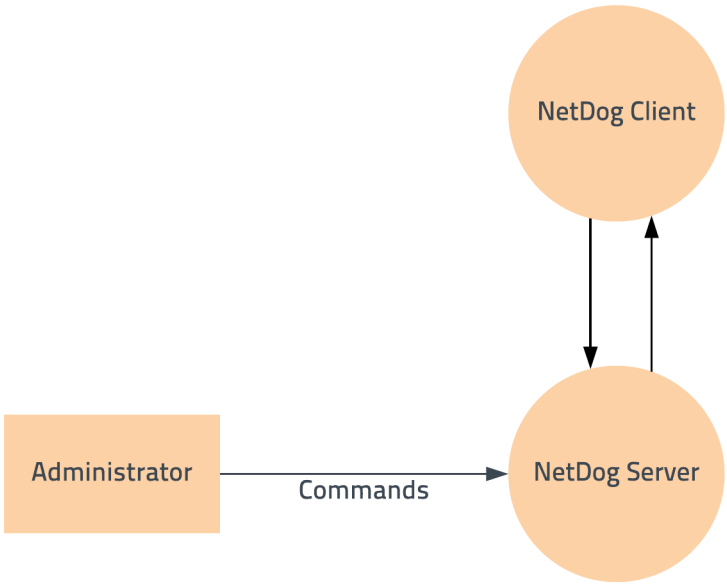


Figure 3.2: Level 0 Data Flow

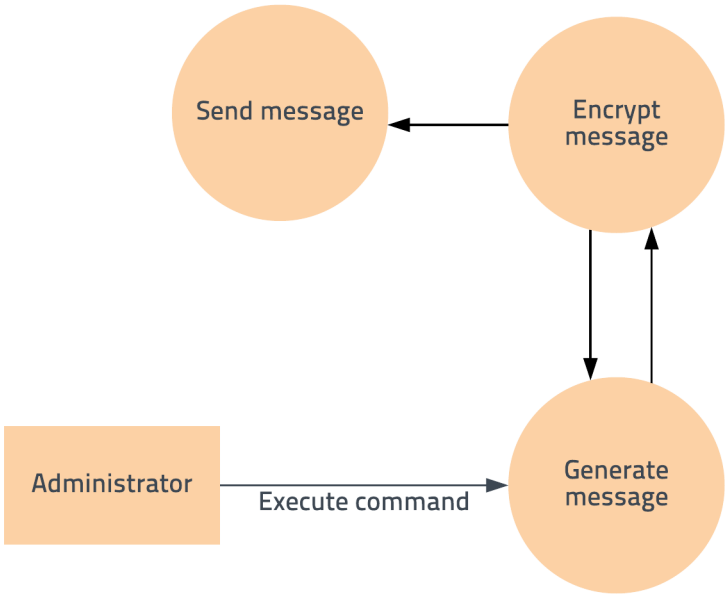


Figure 3.3: Level 1 Data Flow

### 3.5.6 Screenshots

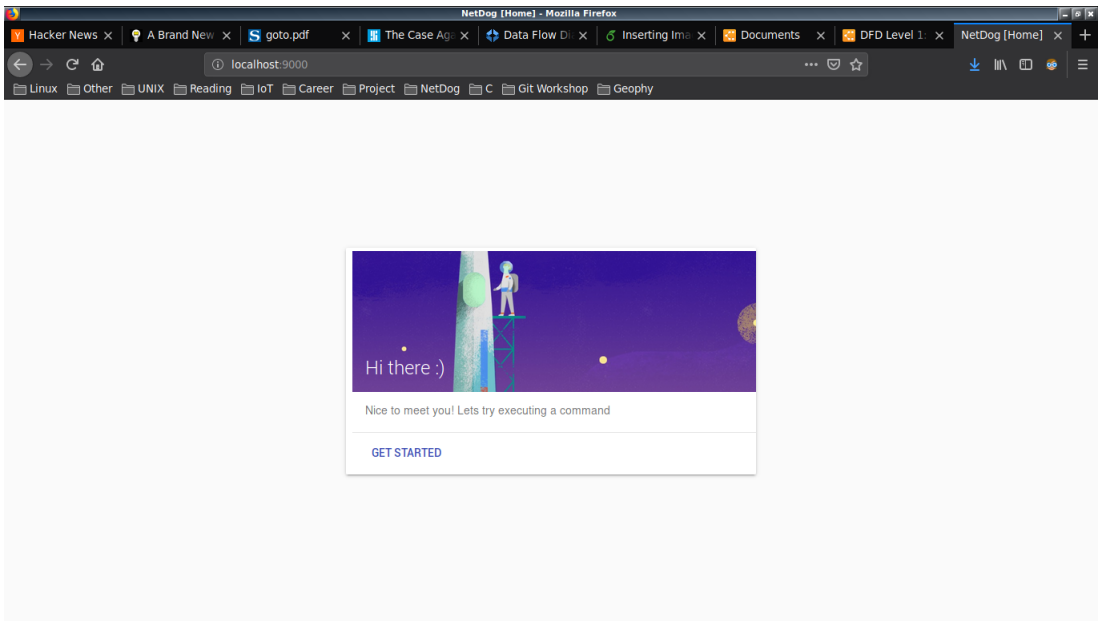


Figure 3.4: NetDog Home Page

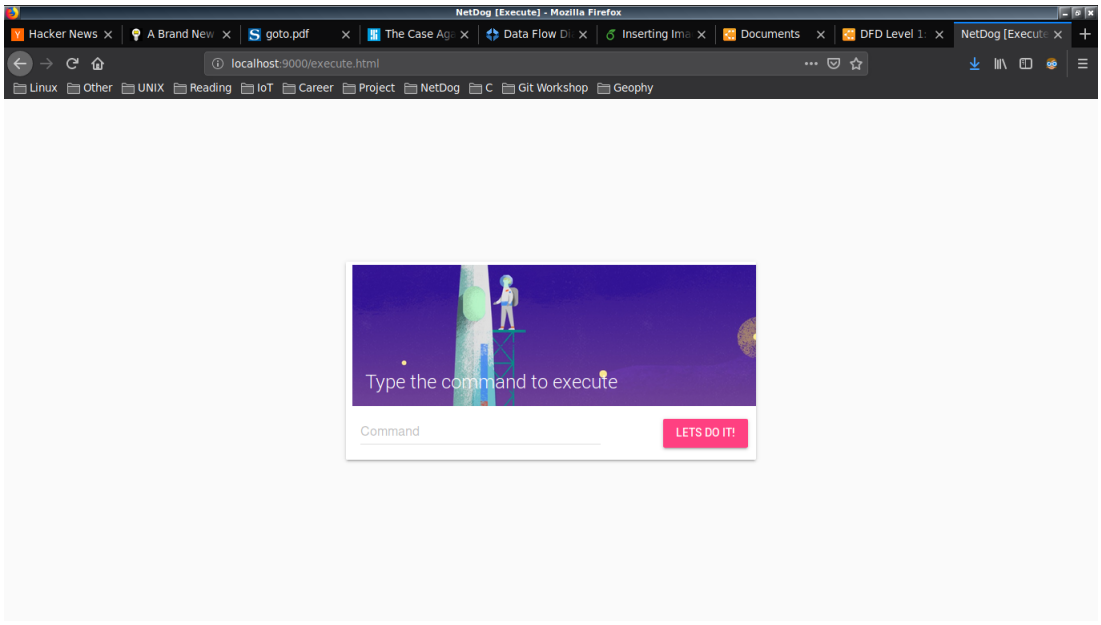


Figure 3.5: NetDog Execute command page

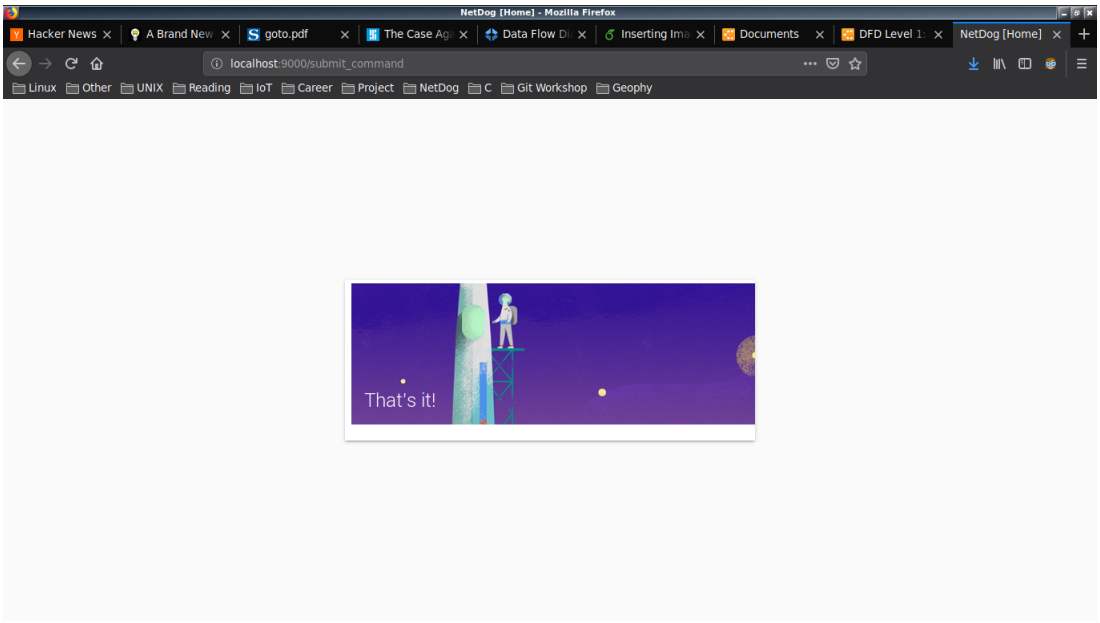


Figure 3.6: NetDog success page

# Chapter 4

# Coding

**Algorithm 1** NetDog Server

1: Initialize the system
2: Setup Network with Server value as True
3: Start the server thread
4: Start web server

**Algorithm 2** NetDog Client

1: Initialize the system
2: Start the server thread
3: Setup the network

**Algorithm 3** NetDog Client

1: Initialize the system
2: Start the server thread
3: Setup the network

# Chapter 5

# Testing and Implementation

## 5.1 Testing methods done for the project

System testing is the stage of implementation which is aimed at ensuring that the system works accurately and efficiently before live operation commences. Testing is the process of executing the program with the intent of finding errors and missing operations and also complete verification to determine whether the objective are met and the user requirements are satisfied.

The ultimate aim is quality assurance. Tests are carried and the results are compared with the expected document. In that case of erroneous results,debugging is done. Using detailed testing strategies a test plan is carried out on each module. The test plan defines the unit,integration and system testing approach. The test scope includes the following: A primary objective of testing application systems is to assure that the system meets the full functional requirements, including quality requirements(Non functional requirements).

At the end of the project development cycle, the user should find that the project has met or exceeded all of their expectations as detailed in requirements. Any changes, additions or deletions to the requirements document, functional specification or design specification will be documented and tested at the highest level of quality allowed within the remaining time of the iproject and within the ability of the test team.

The secondary objective of testing application systems will be do:identify and expose all issues and associated risks, communicate all known issues are addressed in an appropriate matter before release. This test approach document describes the appropriate strategies, process, work flows and methodologies used to plan, organize, execute and manage testing of software project "Real-time Traffic Congestion Analyzer for Road Safety"

### 5.2 Unit Testing

| Sl No | Procedures | Expected result | Actual result | Pass or Fail |
|---|---|---|---|---|
| 1 | Generate signature | Signature generated | Same as expected | Pass |
| 2 | Verify signature | Signature verified | Same as expected | Pass |
| 3 | Encrypt message | Message encrypted | Same as expected | Pass |
| 4 | Decrypt message | message decrypted | Same as expected | Pass |
| 5 | Read configuration | Configuration read | Same as expected | pass |

Table 5.1: Unit test cases and results

### 5.3 Integration Testing

| Sl No | Procedures | Expected result | Actual result | Pass or Fail |
|---|---|---|---|---|
| 1 | Client-Server pairing | Pairing works | Same as expected | Pass |
| 2 | Web server start | Web server started | Same as expected | Pass |
| 3 | Server send commands | Commands sent | Same as expected | Pass |
| 3 | Client receives commands | Command received | Same as expected | Pass |

Table 5.2: Unit test cases and results

### 5.4 System Testing

| Sl No | Procedures | Expected result | Actual result | Pass or Fail |
|---|---|---|---|---|
| 1 | Server startup | Server starts | Same as expected | Pass |
| 2 | Client startup | Client starts | Same as expected | Pass |

Table 5.3: Unit test cases and results

# Chapter 6

# Conclusion

NetDog is a configuration management and monitoring system, that has been developed with naive users in mind. The server and clients will automatically find and pair with each other over the network without the intervention of the users.

The system provides considerable decrease in the learning curve compared to existing configuration management systems such as Ansible, Puppet and provides zero configuration monitoring compared to monitoring tools such as Nagios which require initial setup.

The system is released under MIT license so that anyone can modify, adapt or even sell the source part as part of their product.

.