Project Report On

# Password Cracker based on Distributed Computing

By
Mr. Shaswat J.Babhulgaonkar
115011103

**Problem Definition:**

Develop a distributed password breaker system which work as follows:

- Develop a client program and a server program
- When run, the server will generate a random password with following properties:
- Password length: 5
- Password may contain numbers (0-9) or Capital letters (A-Z)
- The server will generate an MD5 hash, H with the password and current date
- When a client sends a REQ packet, the server will reply with H and a range of 1,000,000 password to test, e.g., 000000 – 999999.
- The server will send different range to different client
- The client will generate a random password, generate MD5 hash, H* with the password and current date. Then, it will compare H* with H. If matches, the password is found and it will send a SUCCESS packet to the server with the recovered password.
- If the client could not find the password in its range, it will send a RETRY packet to the server and the server will send a new range to the client
- A client can send maximum 3 RETRY packets. After that, it will terminate
- The client and server program should print different values in the screen.

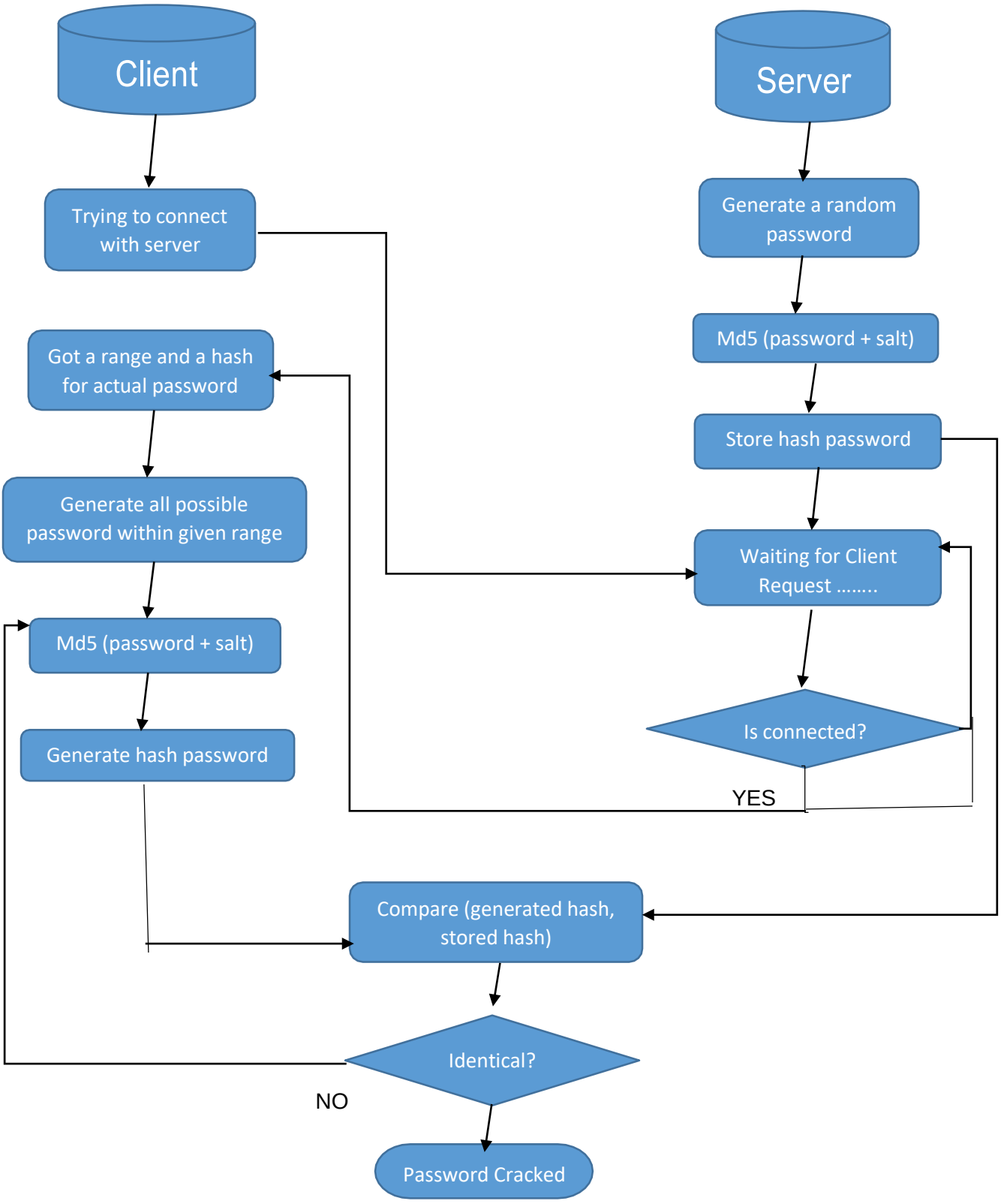
**Introduction:**

The goal of this project is to create a distributed system that can run across the entire Internet. We know password cracking is an entirely parallel application. It consists of a set of operations on small chunks of data, and there's no data that needs to be shared between different password cracking nodes as they crack. They just receive a work unit allocation, try all of the passwords in that unit, and tell the server if any of them was a match.

**Procedure:**

My password breaking system consists of two programs, the worker client and the server. The server creates a password cracking job and is responsible for dividing the job into parts and allocating those parts to worker clients. The clients and server communicate with each other using the sockets API to send UDP packets. Here the Server sends the actual hash password and a range to the client. The Client after get connected to the server receive the hash and the range and try to generate all possible passwords within the range and hash it to compare it to the actual hash password. If any generated password hash matches to the actual given hash then the client successfully crack the password.

**Work flow:**


Client
Server
Trying to connect with server
Generate a random password
Md5 (password + salt)
Got a range and a hash for actual password
Store hash password
Generate all possible password within given range
Waiting for Client Request .....
Md5 (password + salt)
Is connected?
Generate hash password
YES
Compare (generated hash, stored hash)
Identical?
NO
Password Cracked

**Code Description:**

Server Architecture:

As per problem definition the server will generate a random password with 5 character. This password may contain numbers (0-9) or Capital letters (A-Z). It will then generate an MD5 hash of combining the password and the system date. Now the server is ready to distribute this hash among the worker clients for break.

The Server should be multi-threaded to facilitate communication with the client. Each time a new request for connection from client comes the server should create a new thread to communicate with that server. The client should be independent and should be able to find out everything it needs to know from the server. The server should recognize the two different clients as they contact it. It should keep track of the clients and which jobs they're currently working on or what job they've requested.

Client Architecture:

The Worker client after successfully get connected to the server it will get a range at which it will try to break the actual password. In order to making password within given range following mapping is done….

Total possible alpha-numeric characters: 36 (numbers (0-9) and capital letters (A-Z)).
Password length: 5
Possible combinations: $36^5$ (60466176)

In order to map password within one million range for each client we can divide the range with Base 36 system.

In Base 36 system all possible combinations: start 00000 to ZZZZZ.

Some conversion from decimal to base 36 is given in following table….

| Range No | 1,000,000 range in decimal | 1,000,000 range in base 36 |
|---|---|---|
| 0 | 000,000 to 999,999 | 00000 to 0LFLR |
| 1 | 1,000,000 to 1,999,999 | 0LFLS to 16V7J |
| 2 | 2,000,000 to 2,999,999 | 16V7K to 1SATB |
| 3 | 3,000,000 to 3,999,999 | 1SATC to 2DQF3 |
| 4 | 4,000,000 to 4,999,999 | 2DQF4 to 2Z60V |
| 5 | 5,000,000 to 5,999,999 | 2Z60W to 3KLMN |
| 6 | 6,000,000 to 6,999,999 | 3KLMO to 4618F |

**Rules for running project:**

This project consists of two program named Server.java and Client.java. These two java programs are compiled using java compiler and corresponding bat files are given.

This project is built in java language and hence to run the project JDK is first needed to install.

For windows pc simply run the bat file. For Linux the two programs should first compile and then run from Terminal.

The Server Program should run first and then the client program. To break the password the server should always in live.

Please DO NOT CLOSE the command prompt of the server while any client is connecting with the server or make the server in “select command prompt” state.

**Summary:**

When the server starts running it generate a hash with a combination of a random password and system date. Then it awaits for incoming client request. The client received a range from the server. If it fails to get the password within the range, it sends a RETRY packet to server .When a client successfully gets the password, it prints the password, and also notifies the server about it. But if it does not, it terminates with a failure message.