Project Two Report

Paul Jayme | 300222932

RSA Challenge – Estimates the strength of an algorithm or application against exhaustive key-search

BlowFish Algorithm – variable key length – symmetric block cipher

“Bag of Tasks” Architecture – single key manager and dynamic set of clients

TWO PROGRAMS: KeyManager and Client – Evaluate Performance

KEY MANAGER:

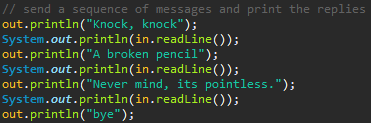
* Handles Clients joining system
* Allocates them key spaces to search & report results
* NO GUI – invoked from CMD Line | java KeyManager initial-key keysize ciphertext
* Initial-key: starting point for the search | integer
* Keysize: defines the key spaces | expressed in no. of bytes
* Ciphertext: text encoded as a base64 string
* Print a RANDOM PORT No. – to be used by clients
* When key has been found – PRINT message on CMD indicating success + PRINT KEY OR FAILURE MESSAGE
* Keep track of how long it takes to either find the key or come to the end of the key space and print this when it exists – TIMER COUNTER

CLIENT:

* Java client

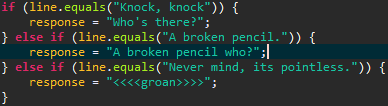
# Task One:

EchoClient.java:

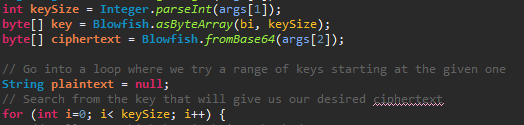


* When an EchoClient is launched it establishes its communication channels then sends a sequence of messages to the EchoServer. Here the code has been modified to a different knock-knock joke. The server must then recognize what the client is saying so it also has to be modified from that end.

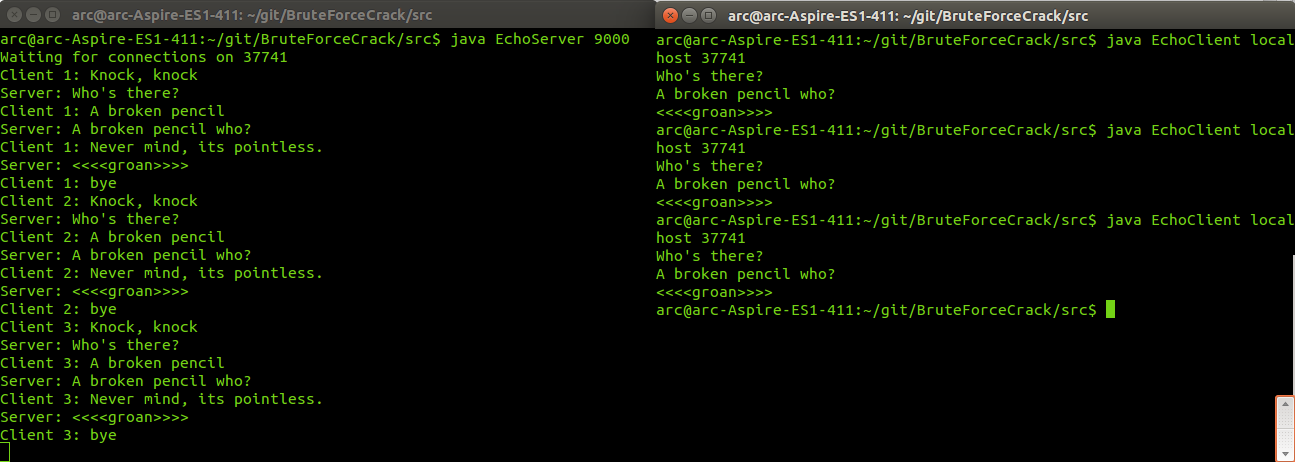
Connection.java:



* When a new thread is created by the EchoServer class it uses the Connection class as its main Thread class (since Connection extends Thread). Hence when “Connection.start()” is called, the program executes the “run()” method of the Connection class which launches the knock-knock joke interaction between the client and the server.



* Search was initially searching based on a fixed number of keys (100 keys) – “i < 100”.
* The conditional statement defined in the for-loop has now been changed to handle a dynamic / user-defined number of keys – “keySize” which is entered as the arguments of the program prior to its launch.



*Above is a screenshot of the system functioning correctly - the left terminal represents the “Manager” / Server who is waiting for connections on port 37741. The right terminal represents the multiple clients that have connected to the Server. The moment a client connects to the server the full interaction is displayed on the terminal of the Server. Ultimately, the sequence of running the system starts with launching the server followed by clients connecting to the server.*

# Task Two: Control Flow

Requirements:

1. Clients only need to be aware of location of Manager
2. Clients can join or leave but will complete the work they have been requested
3. Clients request work from the key manager and return results to it - DYNAMIC
4. Connections between clients and the Masters only exist long enough to request work or to return results
5. When the key is found, the key manager will shutdown

## Key Manager:

1. Launches and begins listening for incoming connections from Clients.
2. When a connection is received from a Client – the Manager creates a new Connection object which acts as a Thread servicing for the new Client Connection.
3. Manager then starts the Thread by calling c.start() – thread begins executing its task.