



POLISH-JAPANESE ACADEMY
OF INFORMATION TECHNOLOGY

UDP Protocol Design Implementation Port Knocking

Revision 1.0

Table of Contents

Table of Contents	ii
Revision History	iii
Approved By.....	iii
1. Introduction.....	1
1.1 Purpose	1
1.2 System Overview.....	1
1.3 Design Map	1
2. Design Considerations	1
2.1 Assumptions	1
2.2 System Environmen.....	
2.3 Design Methodology	
2.4 Risks and Volatile Areas	
3. User Interface Design	2
3.1 Application Controls	2

Revision History

Version	Name	Reason For Changes	Date
1.0	Vikram Mandal	Initial Revision	10/01/2021

Approved By

Approvals should be obtained for project manager, and all developers working on the project.

Name	Signature	Department	Date
Kazimierz Zielinski		PJAIT Information Technology	

1. Introduction

1.1 Purpose

This design will detail the implementation of the requirements as defined in the Software Requirements Specification – UDP Port Knocking (SKJ) 2020

1.2 System Overview

This project is intended to create a custom protocol which utilizes UDP protocol for sending client packets to the server and after wards the server responds back with a randomly selected port through UDP protocol. Later on, further communication is established through this randomly generated protocol using TCP protocol for some request- response.

1.3 Design Map

Summarize the information contained within this document or the family of design artifacts. Define all major design artifacts and/or major sections of this document and if appropriate, provide a brief summary of each. Discuss any significant relationships between design artifacts and other project artifacts.

2. Design Considerations

All design considerations were handled in current project considering that there are not any changes in original design specification.

2.1 Assumptions

The ports which are used are available in the host machine.

2.2 System Environments

- Client
IP Address, Port1, Port2, Port3 ...
- Server
Port1, Port2, Port3 ...

2.3 Design Methodology

➤ 2.3.1 Server

To design the server, we need to open a Datagram socket to accept UDP packets at a given port, to do this we need to create separate threads to handle parallel incoming connections at

each UDP port inside **service()** method.

After we are done with all incoming packets we search for “bye” message in the incoming packet to stop receiving any further packets from the client. When we receive a packet with the “bye” message we end the while loop and call the **process()** method to process the accumulated packets, generate a random TCP port and send the random port number as a response datagram to the original client from where the request had come. For this we take the help of **Map<InetAddress,AddressStore> packetCountsSent** which stores the details of each incoming datagram(its origin and the packets).

After successfully sending the random port number, we initialize the TCP server with this random port number and wait for requests from the client. After successful requests-response TCP communication we close the TCP server.

➤ 2.3.2 Client

To design the client, we get the IP address and ports to knock through the program arguments. For each input IP and port combination, we create a datagram using the input data received from the standard input stream. (**System.in ()**). These datagrams are then sent to the intended UDP server.

After successfully sending the datagram, the client receives the response from the UDP server with the random TCP port number generated. Using this random TCP port, we further communicate with the TCP server for some request response.

2.4 Risks and Volatile areas

As this application is based on Threads, so we need to be careful while creating multiple server and client threads, we might face **OutOfMemoryError** in case we run out of the memory allocated to our program.

3. User Interface Design

Server is provided with the ports to initialize UDP server through the java program argument which is then extracted to initialize the UDP servers threads.

Similarly, The Ip and ports are passed through the program arguments where the client is intended to knock on.

Later on the inputs from the standard input stream are used to create request datagram packets.

3.1 Application Controls

- We can initialize single or multiple clients to start communicating with the designated server.
- The project is created using maven build tool, so we can easily create a **jar** file for easy installation in JVM.
mvn clean install
- The project can be deployed to a nexus repository using command
mvn clean deploy
- To run the Server program from JAR in port 1025
java -cp ./KnockKnock-1.0-SNAPSHOT.jar com.syc.portknock.Server 1025

- Now to run the Client program to knock on the server running in port 1025
`java -cp ./KnockKnock-1.0-SNAPSHOT.jar com.syc.portknock.Client 192.168.0.100 1025`