An Authentic Approach to Client-Server Communication Based on an Algorithmic Time-Bound Validation

Vishal Jain
Geetanjali Institute of Technical Studies
Monika Bhatt
Geetanjali Institute of Technical Studies
Mayank Patel (mayank999_udaipur@yahoo.com)
Geetanjali Institute of Technical Studies https://orcid.org/0000-0002-8580-4184
N.S.Rathore
Geetanjali Institute of Technical Studies

Research Article

Keywords: Message, server, client, algorithm, validation

Posted Date: February 12th, 2024

DOI: https://doi.org/10.21203/rs.3.rs-3948802/v1

Additional Declarations: The authors declare no competing interests.

Abstract

In sensitive communication, the genuineness of the data (messages) plays the most important role. Often, communication is misunderstood if the data are not validated before they are sent. This occurs because the data to be delivered can be typed incorrectly or because the data are fetched incorrectly (not in order) from the source where they are already stored. Therefore, in this paper, we introduce a new approach to client-server communication based on time-bound validation of the messages. This paper introduces an algorithm to overcome this issue. With respect to the algorithm, we propose a tricky approach that can be used by the client and the server to ensure the exchange of validated data (messages). The delivery of reviewed data ensures that no incorrect or misleading data are sent to the opposite side. The concept of a timer is therefore used in this research. The paper focuses on the validation of data from one of two sources: 1) a text file or 2) a keyboard (user input). By implementing the proposed algorithm, the server and the client can communicate with each other without any false interpretation of the data delivered. In this way, highly sensitive communication can occur successfully without any misunderstanding, which often arises because of the lack of reviewed data.

1 Introduction

Interprocess communication is the foundation of client-server communication. Any network that allows the two processes to communicate with each other employs interprocess communication to play the most important role. The two processes can be performed on the same machine or on different machines. The process that starts communication by sending the request is termed the client process. A process that accepts a request and fulfils the demands of other processes is called a server.

While communication occurs between processes, the genuineness (correctness) of the data becomes very important. Misspelled or meaningless data can lead to misunderstandings between two parties. A false interpretation of the data can cause severe discordance between the two sets of data. When the two parties are the countries, companies or organizations, the accuracy and the genuineness of the data become the most important factors in communication.

Most often, it is seen that the communicating processes send the data without reviewing them. Whether this is a time issue or something else, they do not even think that it takes time to review the data and press the key in hurry to send the data. This often depicts the nonprofessional behaviour of two parties. Sometimes it becomes a severe issue between the two if the received data are misinterpreted.

In this paper, we provide a tricky yet simple approach to interprocess (client-server) communication in the form of an algorithm that is based on time-bound validation of the data. In this case, both communicating parties are given an option to review the data that they want to send. Upon choosing the review option, they are given a time span of 60 seconds to review their data before they are sent. Upon completion of the 60 seconds (after review), they can either accept or deny sending the data. The algorithm also provides an option to send the data without reviewing (validating) it for faster

communication. By implementing the algorithm (opting the validation option), two parties can ensure the genuineness of the data to be delivered. In this way, the two processes (two parties) can communicate with each other in an authentic way.

A number of studies have been performed to address this issue, but the solutions that they are providing are mostly encryption methods or something else. Many encryption methods exist for protecting the data. Our paper does not focus on protecting the data, so we have not given any new idea of encryption here.

Instead of protecting the data, we focused on the situation where we can validate the data that are going to be delivered. A time-bound validation of the data by the two parties can ensure that the authenticated data are being delivered.

The Java programming language is used to implement the algorithm. Two Java source codes have been developed for the server and the client processes. Server and client programs are executed on the same machine (local host) for testing purposes. We used the concepts of socket programming, file handling and Java's time and date packages. The lack of extra hardware and the lack of an increase in the budget can be considered the headlines of this research. These results are exactly what we expected. The future scope of this research is quite fascinating, as making this approach easier and more difficult can make communication easier and easier.

2 Literature Review

Many efforts have been made in the field of networking. Thousands of researchers are addressing different networking issues. Data security, compression, encryption, topologies, network types, etc., are factors upon which much research has been performed. Many related studies are in progress as well. Many studies tell us new approaches to communication in a specific topology. A few studies introduce new types of topologies that depict the arrangement of devices in a network in different ways. Many studies are addressing peer-to-peer networks and ad hoc networks as well. Bluetooth, PAN, LAN and WAN issues have been the subjects of hundreds of studies for several decades.

Data security is considered the biggest issue in every type of network. Therefore, considerable effort has been expended in terms of encryption, data protection, compression, etc. We have gone through a number of research articles that have contributed to a similar type of research. We witnessed that many of them are suggesting the need to grow supporting infrastructure, e.g., firewalls, antiviruses, spell checkers, translators, etc. Very few studies have shown that they are dealing in a similar way as we are suggesting, e.g., working on application software or system software (operating system) implementing an algorithm.

According to many articles, a lack of research on software or algorithms prevents us from broadening the use of alternative networks. Many algorithms for data protection, compression, encryption, captioning, barcoding, QR codes, steganography, etc., are available because of those studies, but validation is the factor upon which much work has yet to be done.

A few studies that are similar to ours address topological advancement. Many articles guide how to solve ring topology issues, whereas many of them discuss optical backbone networks. Many studies have been performed on star topology, whereas efforts have also been made to analyse the rate and demand in various topologies. One article provides a new algorithm to solve the problem of data loss in ring topology when a participating node crashes accidently. Different issues are addressed by a number of other significant studies.

We came to know that very few efforts have been made to address this issue. Therefore, we decided to work on the trustworthiness (validation) of the data instead of protecting it. We developed an algorithm. By implementing this algorithm, communication processes can ensure trustworthy communication. They can completely rely upon the data they are sending and receiving.

Vishal Jain et al. (2022). A Modified Approach on Ring Topology to Enhance the Network Efficiency. Journal of Comparative Effectiveness Research. This paper addresses an interesting aspect of technological scenarios: the fundamental concept of ring topology has the drawback of one-way networking, and if one of the device connections is faulty or disconnected by any means, that is, we can have another solution so that the device is still connected and the work is not interrupted [11].

Murugan, Karthick & Vasuki.J. T, & Arulvizhi.M,. (2014). Performance analysis of ring topology in an optical backbone network African Journal of Science and Research Vol 4. 26–29. The introduction of optical networks has been one of the most significant changes in the telecommunications industry in recent years. With the increasing deployment and growth of optical transport networks, solving the classic network design problem of optimizing quality and scalability has become increasingly critical. The fundamental design objective in an optical network involves optimizations of the following two metrics. The first objective is to minimize the total network cost. The second is to maintain an acceptable quality of service (QoS). The SDH (synchronous digital hierarchy) is a strictly planned system in the optical backbone network [1].

Different networking issues are addressed by the two articles referred to above. The article titled “A Modified Approach on Ring Topology to Enhance the Network Efficiency” discusses a new approach to communicating in ring topology. This article describes what can be done in case of the failure of any participating node so that the data transmission is not affected. However, data validation is not needed to ensure hassle-free communication. The articles that point to the issue of data (message) validation guide different approaches.

The approach that we suggest is a trickier yet simple one that is not exactly found in any of the articles that we have gone through yet. Our approach to data validation is different and advantageous because it is inexpensive and effective.

3 Validation, Cryptography and Client-Server Communication

3.1 Encryption vs Validation

Encryption is the process of converting simple text (plain text) into ciphertext. A ciphertext is a converted form of plain text that cannot be understood by anyone without reconverting it. The process of constructing plain text from ciphertext is called decryption. Symmetric vs. asymmetric is a comparison that is often used in the world of cryptography and computer security. Symmetric encryption involves using a single key to encrypt and decrypt data, while asymmetric encryption uses two keys—one public and one private—to encrypt and decrypt data. The RSA, AES, DES, IDEA, message digest, Diffie-Hellman key exchange, etc., are the algorithms for cryptography.

Data validation is a process of ensuring that the data to be sent or received are genuine and that sending or receiving parties (processes) can reply upon them. Data type checks, code checks, range checks, format checks, consistency checks, uniqueness checks, etc., are the methods used for data validation. The three steps of data validation are as follows:

1. Determine the data sample
2. Database validation
3. Data formatting

3.2 Client-Server Communication (A General Approach)

Multiple clients are often welcomed by the server for interaction. This interaction is made possible by the creation of connections between them. Two approaches taken by the client and server to establish connections between them are connection oriented or disconnected (connectionless). The server creates a connection to a port that can be any of the available 65536 ports (2^{16} ports). This process results in a server socket. The server is listening for connections to clients in this socket at the moment. A port number is a 16-bit integer.

The client is also creating the server socket to specify the identity of the server, i.e., its IP address and port number on which it will create a server connection. If the TCP protocol is used for the connection, then a path is established between them; otherwise, (in the case of UDP), no path is pre-established between the devices. Often, communication is accepted and initiated upon receipt of a client's request.

When the TCP protocol is applied, there are two types of streams in which data can be passed: input and output. Datagrams are used to store data when referring to the UDP. TCP and UDP are used at the transport layer. TCP ensures guaranteed and ordered delivery of the data, whereas UDP does not guarantee data delivery. Moreover, this approach does not guarantee the order of the data.

4 Methodology

We, the authors, ultimately developed a new approach to validate the data before delivery to the opposite side. The algorithm developed for this purpose is implemented in the following way:

1. Two source files in java were prepared. The server program was written in the first file, whereas the client program was written in the second.
2. The java.net, java.util and java.io packages were imported into both source codes to use some necessary predefined classes of networking, input–output and utility classes such as Scanner and Date. The Socket class and the ServerSocket class of the java.net package were used to create the sockets. Only the Socket class was used in the client's source code, whereas the ServerSocket and Socket classes were used in the server's source code.
3. Sockets were successfully created. We used different port numbers on both sides for binding the socket, as a port could not be used by more than one process for the purpose of sending the data.
4. We started both processes (executed both the java programs) at the local host, i.e., at the same machine, for testing purposes first. After successful testing at the local host, we started server and client processes at different machines as well. This attempt at client-server communication was also successful.
5. To make the server listen and accept the client requests, we used the listen() and accept() methods of the ServerSocket class.
6. Upon receiving a request on the server socket, the accept() method created and returned a new dedicated socket for the connected client. In that way, the server also obtained client information (client's IP address and port number). The accept() method helps the server keep track of the connected clients.
7. The server and the client both had two of the streams associated with their sockets. The two streams were the input and the output streams. The data sent by the client were transmitted to the server through an input stream from the server. In contrast, server data that should have been delivered to a client were taken over by an output stream from the server. The same applies to the client as well. Therefore, the output stream of the server behaves as the input stream of the client, whereas the input stream of the server behaves as the output stream of the client.
8. The flush() method of the OutputStream class was used to ensure that the data were sent from the sockets on both sides, i.e., at the server and the client.
9. Upon receiving data at the socket from the opposite side, the readUTF() method was used to fetch the incoming data. To write the data on the output stream, the writeUTF() method was used. The data were returned by the readUTF() method in the form of a string so that whenever needed to convert the data, the appropriate type of casting was performed. For the purpose of type casting, predefined methods of the Java library were used. parseInt(), parseFloat(), and parseDouble() are some of those predefined methods.
10. To store predefined server messages, a text file was created on the HDD of the local host. We could manage to extract the messages from this file when the server chose the option to obtain the message from the file instead of getting it from the keyboard. To store and fetch the data from the file, we used Java's file handling concept. To store the data in the file, we used the FileOutputStream

class of the Java library, whereas the FileInputStream class was used to retrieve the data from the file.

11. For time-bound validation, we had to work on the system time. We had to retrieve the data and time for this purpose. For this purpose, the Date class of the java.util package was used. getHours(), getMinutes() and getSeconds() are the predefined methods of the Date class that were used to work on the exact time-bound validation.
12. To ensure a timer of 60 seconds, the use of the getMinutes() method became fruitful. The logical use of the time returned by the methods made it possible for the server and client to review or validate their data within 60 seconds before sending.
13. To make the data validation optional, we made it a user choice. When the server or client chose a certain value of 5, only data validation (data review by the user) took place. When a value other than 5 was chosen, the data were sent without review. The option of not reviewing the data were also maintained to ensure faster delivery of the data, regardless of the genuineness of the data.
14. We made the source of the data optional out of the two. The server and client could choose their data either from a text file or from the keyboard. Some predefined messages were stored in the text file for the convenience of the server and the client.
15. The data exchanged between the server and the client were considered genuine when the data were prevalidated given a time of 60 seconds. Reviewing the data for 60 seconds made the data authentic and trustworthy. Trustworthiness of the data without any misunderstanding between the client and server (two processes) was the objective of this research, and this could be achieved because of the time-bound validation of the data before delivery.

To ensure the genuineness of the data and perform a time-bound validation, we developed and implemented the following:

4.1 Algorithm for the Server Process

Step-1: Begin

Step-2: Start the server.

Step-3: The server obtains the connection request from the client and accepts it.

Step-4: The server selects an option for the source of the data to be delivered to the client. The two options are a) a text file and b) a keyboard (user input).

Step-5:

Step-5.1: If server inputs 1:

The text file becomes the source of the server message, and the data of the file becomes the server message.

Step-5.1.1: If the server message is not equal to “stop”:

Make the time-bound validation of the message optional for the server in such a way that the server can either select reviewing the message or not before its sending.

Step-5.1.1.1: If the server inputs 5:

A timer of 60 seconds starts for reviewing the message. After an expiry of 60 seconds, the server is prompted to input 7 (for sending the message) or to input 8 (for not sending/rejecting the message).

Step-5.1.1.1.1: If the server inputs 7:

The message is delivered to the client through the dedicated socket for the connected client. Go to step 6.

Otherwise, if the server inputs 8:

The message is rejected and not delivered to the client. Go to step 4.

Step-5.1.1.2: If the server inputs 6:

The server message is delivered to the client without review (validation). Go to step-6.

Step-5.1.2: If the server’s message is equal to “stop”, proceed as follows:

Go to step 9.

Step-5.2: Else if the server inputs 2:

The server message is read from the keyboard (user input). Goop step 5.1.1.

Step-6: Let the server read the data from the input stream (client’s message) and store it as a variable.

Step-7: Print the value of the variable. The server can also utilize the variable (client’s message) in other operations.

Step-8: Repeat step-4.

Step-9: End

4.2 Algorithm for the Client Process

Step-1: Begin

Step-2: Start the client.

Step-3: The client sends a request to the server and obtains a successful connection after the server accepts the request.

Step-4: The client reads data (server's message) from an input stream associated with its socket.

Step-5: The server message is displayed by the client. The client can utilize it in other operations as well.

Step-6: Make the client select an option for the source of the data to be delivered to the server. The two options are a) a text file and b) a keyboard (user input).

Step-7:

Step-7.1: If client inputs 1:

The text file becomes the source of the client message, and the data in the file becomes the client message.

Step-7.1.1: If the client message is not equal to "stop":

Make the time-bound validation of the message optional for the client in such a way that the client can either select reviewing the message or not before its sending.

Step-7.1.1.1: If the client inputs 5:

A timer of 60 seconds starts for reviewing the message. After an expiration of 60 seconds, the client is prompted to input 7 (for sending the message) or to input 8 (for not sending/rejecting the message).

Step-7.1.1.1.1: If the client inputs 7:

The message is delivered to the server through the dedicated socket for the connected server. Go to step 8.

Otherwise, if the client inputs 8:

The message is rejected and not delivered to the server. Go to step 6.

Step-7.1.1.2: If the client inputs 6:

The client message is delivered to the server without review. Go to step 8.

Step-7.1.2: Else if the client's message is "stop":

Go to step-11.

Step-7.2: Else if the client inputs 2:

The client message is read from the keyboard (user input). Goop step 7.1.1.

Step-8: Let the client read the data from the input stream (server's message) and store it as a variable.

Step-9: Print the value of the variable. The client can utilize the variable (server's message) in other operations as well.

Step-10: Repeat step 6.

Step-11: End

Table 1
Client and server statistics

	IP Address	Port Number	Timer (in seconds)	Success Rate After Implementation of Algorithm
Client	127.0.0.1	4545	60	99%
Server	127.0.0.1	5555	60	90%

5 Working Model of the Proposed Algorithm

5.1 Case Study

A computer network researcher always looks for the best place to implement his new idea of networking. For this purpose, nothing can't be better than a computer lab having an LAN facility with a star topology. We already had a computer network lab with the said requirements, so we chose to implement our idea there. This research originated from the fact that we often face the issue of data genuineness. As a part of an educational institute, the genuineness of the data becomes the most important thing for us in official communications (intra- or inter-organizational). We chose a computer with 8 GB of RAM and 1 TB of HDD for this purpose while testing the idea of using a single machine (to run client and server processes on the same machine/local host). To test the idea on different machines, we chose 2 computers with 8 GB of RAM and 1 TB of HDD each.

For the server process, we built a Java source code referred to as myserver.java and another Java source code referred to as myclient.java for the client process. For initial testing purposes, we kept the server and the client processes on a single machine named the 'local host'.

We opened two command prompts (CMDs), one for the server and another for the client, to run the codes. First, we compiled and ran the server code. Thereafter, the client's code was executed.

We could successfully run both source codes as long as we kept the concepts of socket programming, input—output, system-time, and file handling. As a consequence, both the client and server processes communicate with each other, as described in the methodology section.

6 Results

Results images are available in the Figures carousel.

Conclusion

This research provides a solution to the problem of the genuineness of the data. We developed two algorithms for the server and client to validate the data before they were sent. Time-bound validation is the objective of this research. In this study, we propose that a time-bound validation of the data (messages) can ensure the trustworthiness of the communication. It becomes the most essential when communication occurs between two highly sensitive sides, i.e., countries, companies, organizations, etc.

To implement both algorithms, Java is used to create client and server programs. In the client and server codes, the concepts of socket programming, file handling, system time and input–output are used. To execute the programs (to start server and client processes), a command prompt is also used for the supporting tool.

The provision of the review data ensures that there is no incorrect or misleading information transmitted to the other party. In this research, the concept of a timer has therefore been applied. This paper will examine whether the following data from one of these sources can be verified: 1) a text file or 2) a keyboard user input.

With the implementation of the proposed algorithm, a server and its client can communicate with each other without false interpretations of what they have received. This allows very sensitive communications to occur in a manner that does not cause any misunderstandings due to data that are not subject to review.

Given that only a software approach can overcome the issues of false interpretation of the data, no additional hardware is required for support. Due to the lack of additional hardware, no cost (budget) issues arise.

As an inexpensive but effective solution, this algorithm has great potential for the future. Many contributions to this approach can be made in the future. As the algorithms become more advanced, the percentage of successful data interpretation may increase. A person (programmer) with sound knowledge of network programming can put forth effort into adding something extra in this approach. The use of the Java programming language is not a constraint here. Any programming language providing the required library for interprocess communication can be used to develop client and server programs.

By setting commands for data collection (from databases or text files), storing data, and applying connection-oriented (TCP-based) and connectionless (UDP-based) approaches in socket programming, a network researcher can easily improve the approach.

Our research helped us to overcome the problem of data validation (data genuineness) in our institute's LAN, as trustworthiness of the data exchanged between the two parties is the most important factor for us when it is either at the interdepartmental level or at the central level. Proper visibility and extensive use of the research can overcome the issues of the other networks as well.

This research can play a very significant role in solving some network issues quite effectively. It is highly feasible for use by any network because it deals with software only.

Declarations

Acknowledgement

There is a great saying that “Not only books make the world educated but also needs and experiences make the world educated too”. Believing the facts and experiencing the things in life, we are therefore very grateful to all those who were directly or indirectly associated with our research. We are heartily thankful to the management and administration of our institute for motivating us to contribute something to the field of research. We are deeply thankful to our colleagues and family members for their support, patience and guidance. We would like to salute all those researchers whose innovations and research articles made us informative, inspired, motivated and educated. Last but not least, we bow down to the almighty god for all their blessings.

References

1. J.T.Vasuki, Dr.D.Mohana Geetha ,Bonfring International Journal of Research in Communication Engineering titled on “Analysis of Cost and Demand Rate for Various Network topologies in Optical BackBone Network” Special issue, Vol.2, P.No.159-164.
2. H.G.Perros, connection-oriented networks: SONET/SDH, optical networks, ser. Lecture Notes in optical networks, willey, Hoboken, NJ, 2005.
3. E. Modiano and A. Narula –Tam “Survivable routing of logical topologies in WDM networks” In Proceedings of IEEE INFOCOM, 2001.
4. A. A. M. Saleh, “Transparent Optical Networking in Backbone Networks,” OFC ’00, patient March 5-10, 2000, ThD7.
5. ITU-T Rec. G.803, Architecture of transport networks based on the synchronous digital hierarchy (SDH)., Geneva: International Telecommunications Union, March 2000, retrieved 2010-11-03.
6. Available at http://www.openreachcommunications.co.uk/optical network topologies/ring topology.
7. E. Dotaro and A. Jordan, “Optical Backbone Topology Design under Traffic Aggregation and Flexibility Constraints”, “OFC’01, March 19-22, 2001, TuG6.
8. Bradley Mitchell. "Introduction to Computer Network Topology". About.com. Retrieved January 18, 2016.
9. IEEE 802.11 Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE Inc., New York (2007)
10. TKN EDCA IEEE 802.11e extension (2006), http://www.tkn.tu-berlin.de/research/802.11e_ns2
11. Vishal Jain, Monika Bhatt, Mayank Patel, Ajay Kumar Sharma, Mr Bhupendra Kumar Teli and Mr Abhishek Gupta, A Modified Approach on Ring Topology to Enhance the Network Efficiency, url=

https://www.researchgate.net/publication/362411923

12. Kosek, K., Natkaniec, M., Vollero, L., Pach, A.R.: An Analysis of Star Topology IEEE 802.11e Networks in the Presence of Hidden Nodes. In: ICOIN 2008, Korea (2008)
13. Patel, M., Choudhary, N. (2017). Designing an Enhanced Simulation Module for Multimedia Transmission Over Wireless Standards. In: Modi, N., Verma, P., Trivedi, B. (eds) Proceedings of International Conference on Communication and Networks. Advances in Intelligent Systems and Computing, vol 508. Springer, Singapore. https://doi.org/10.1007/978-981-10-2750-5_17
14. Abolhasan, M., Wysocki, T., Dutkiewicz, E.: A review of routing protocols for mobile ad hoc networks. Ad Hoc Networks 2(1), 1–22 (2004)
15. Kumar, S., Raghavan, V.S., Deng, J.: Medium Access Control Protocols for Ad-Hoc Wireless Networks: A Survey. Elsevier Ad-Hoc Networks Journal 4(3), 326–358 (2006)

Figures

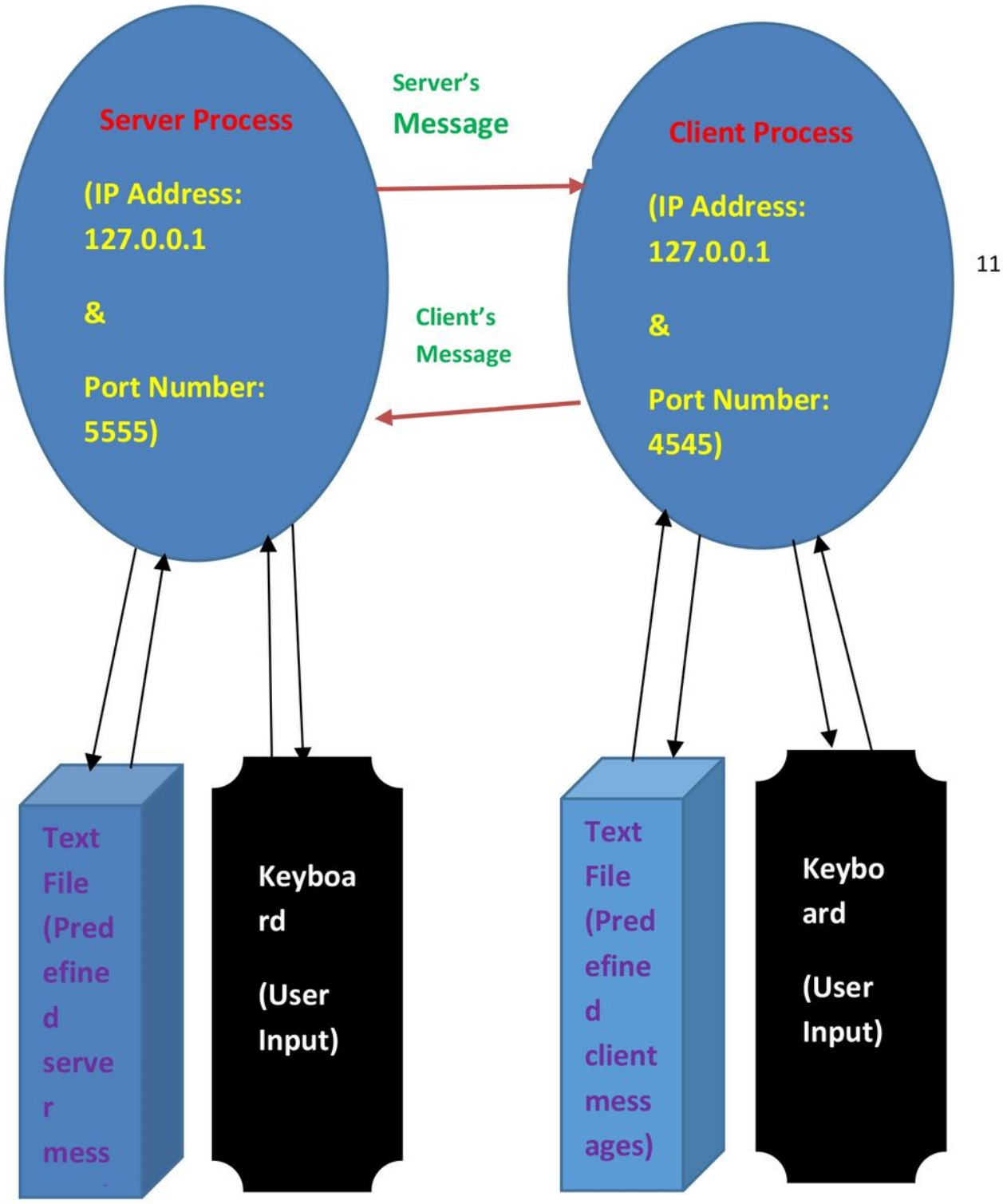

Figure 1

A scenario of a local host during the implementation of the algorithm.

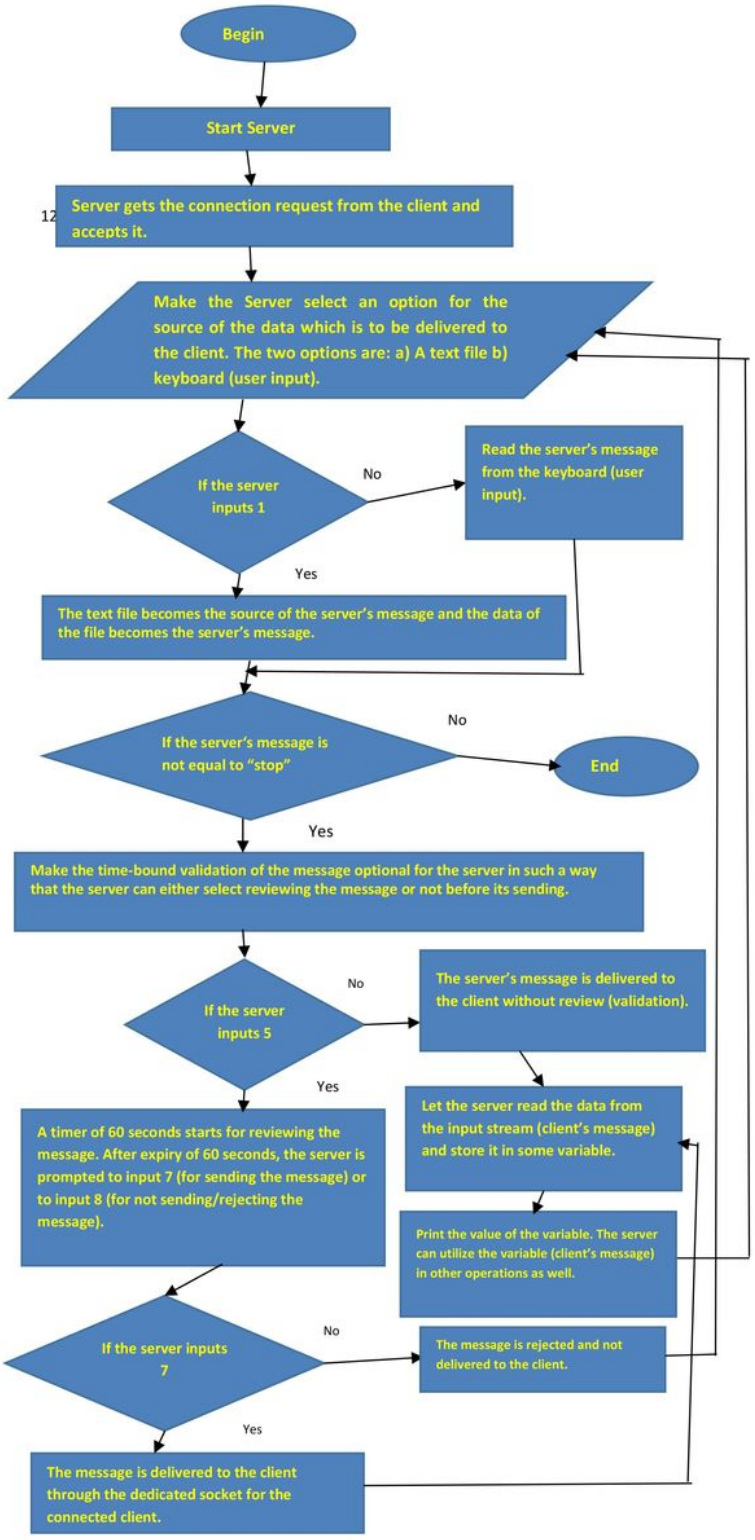

Figure 2

Flowchart of the server process

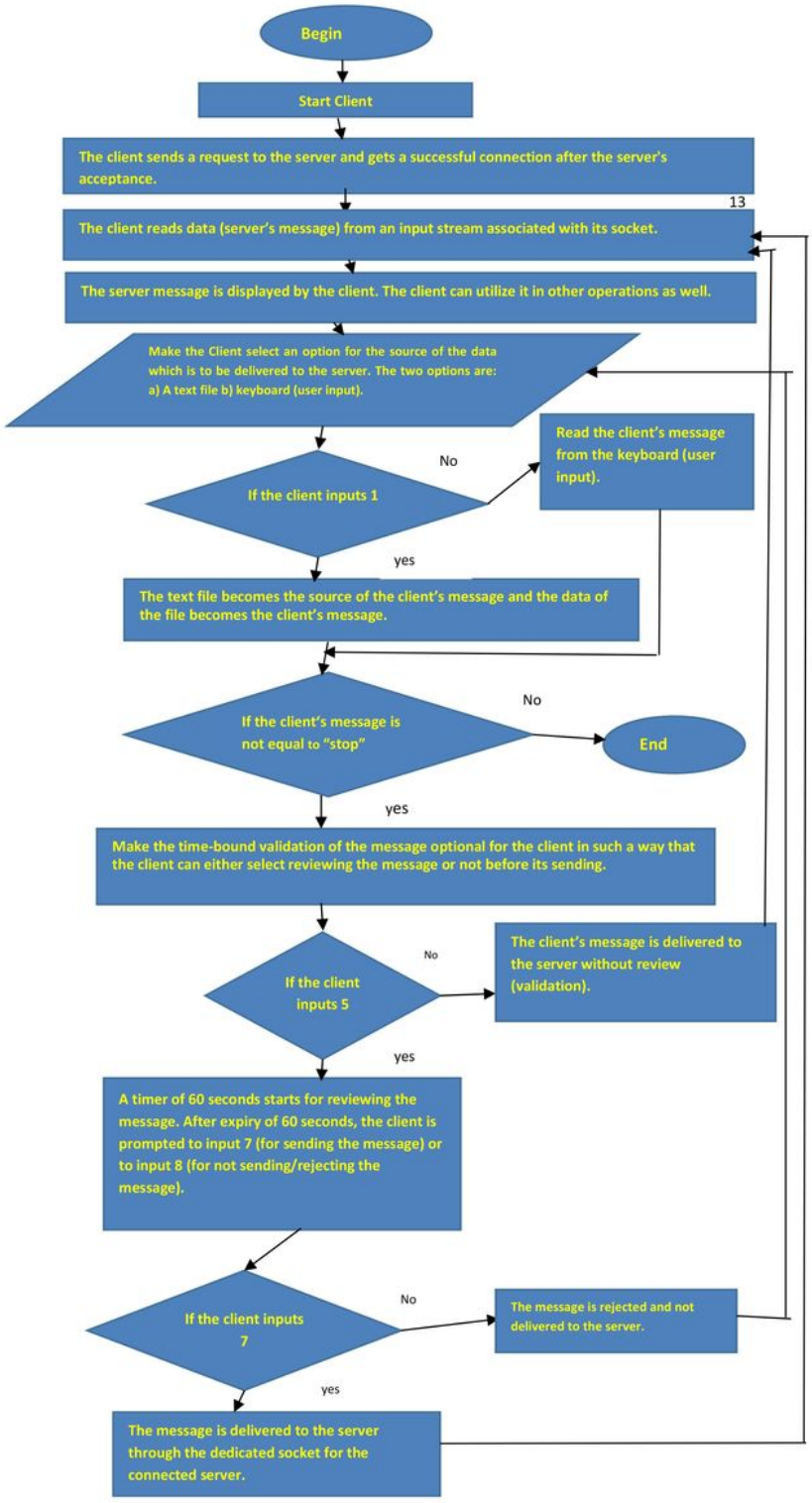

Figure 3

Flowchart of the client process

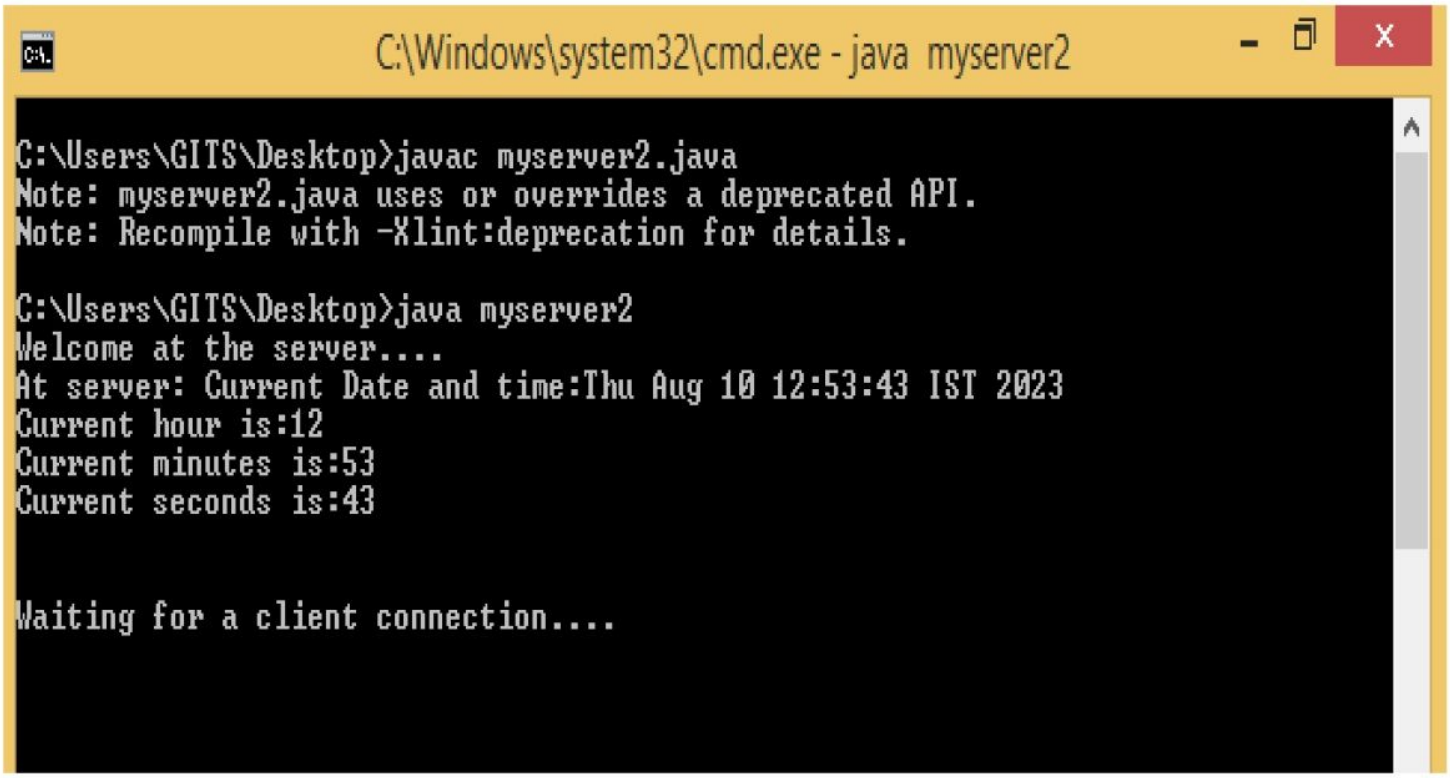

Figure 4

Server starts and waits for client connections

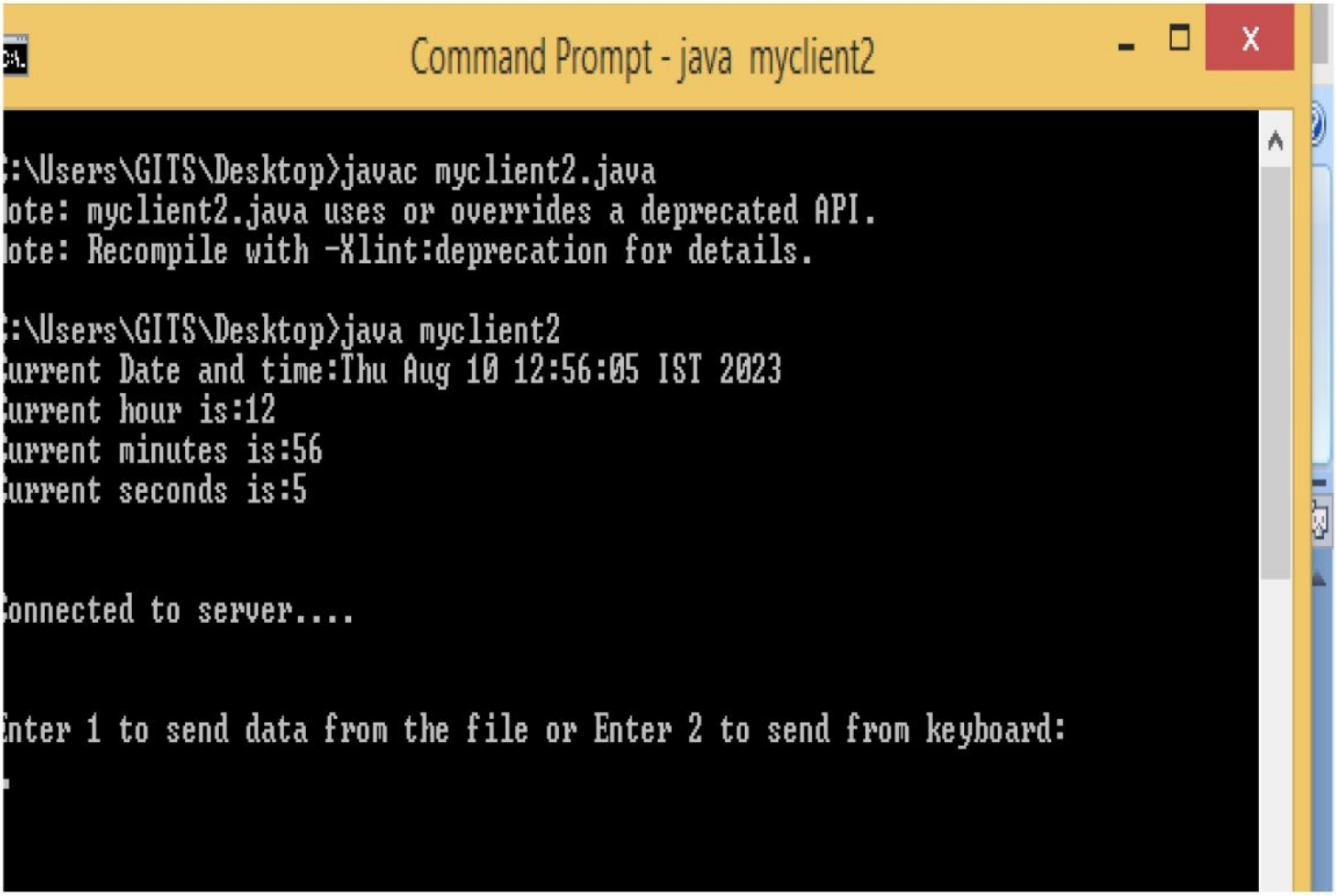

Figure 5

The client is connected to the server and is asked to enter 1 if he or she sends a message from a file or 2 if he or she does not.

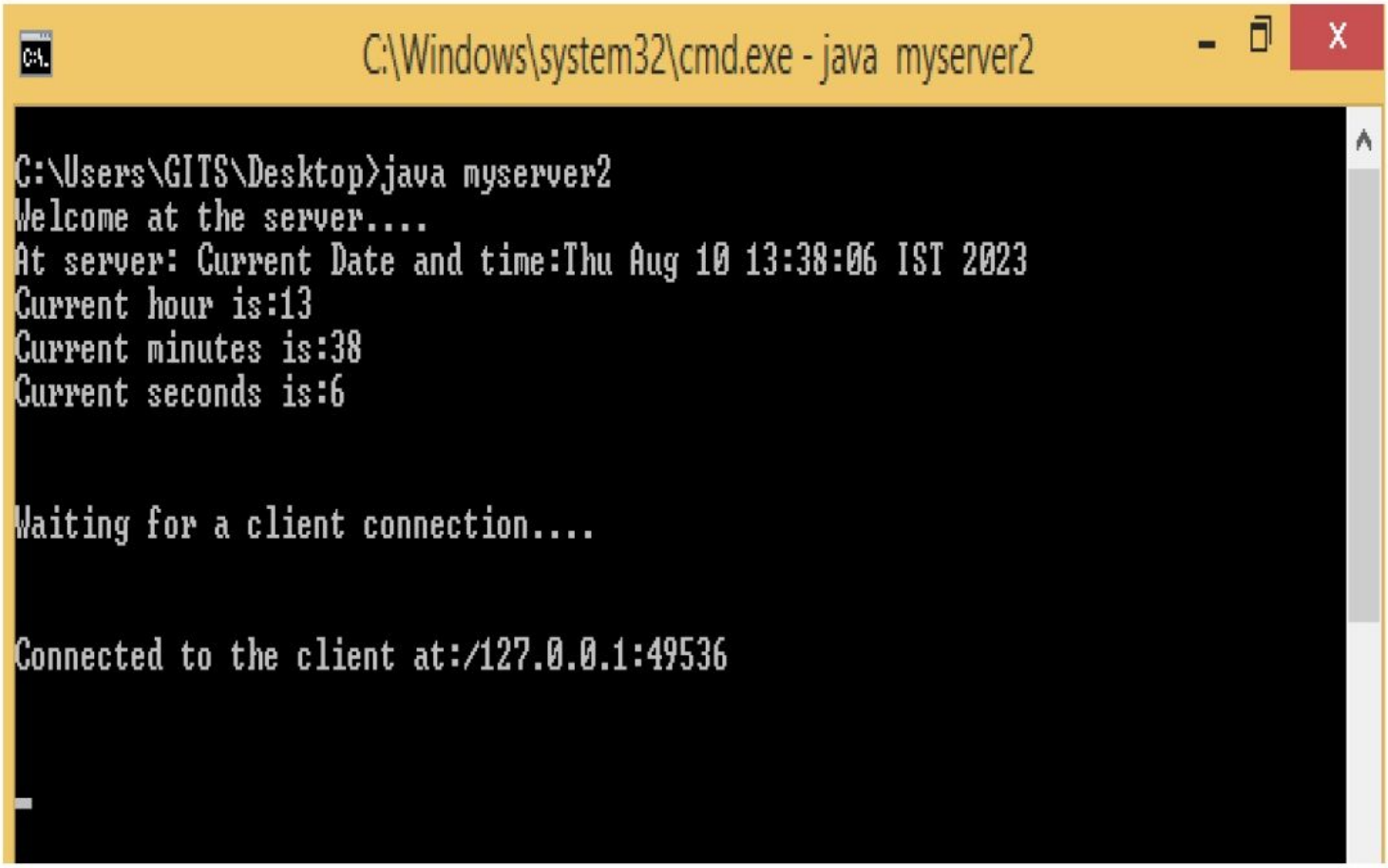

Figure 6

The server obtains the client connection. He or she displays the client's IP address and port number to verify the client's identity

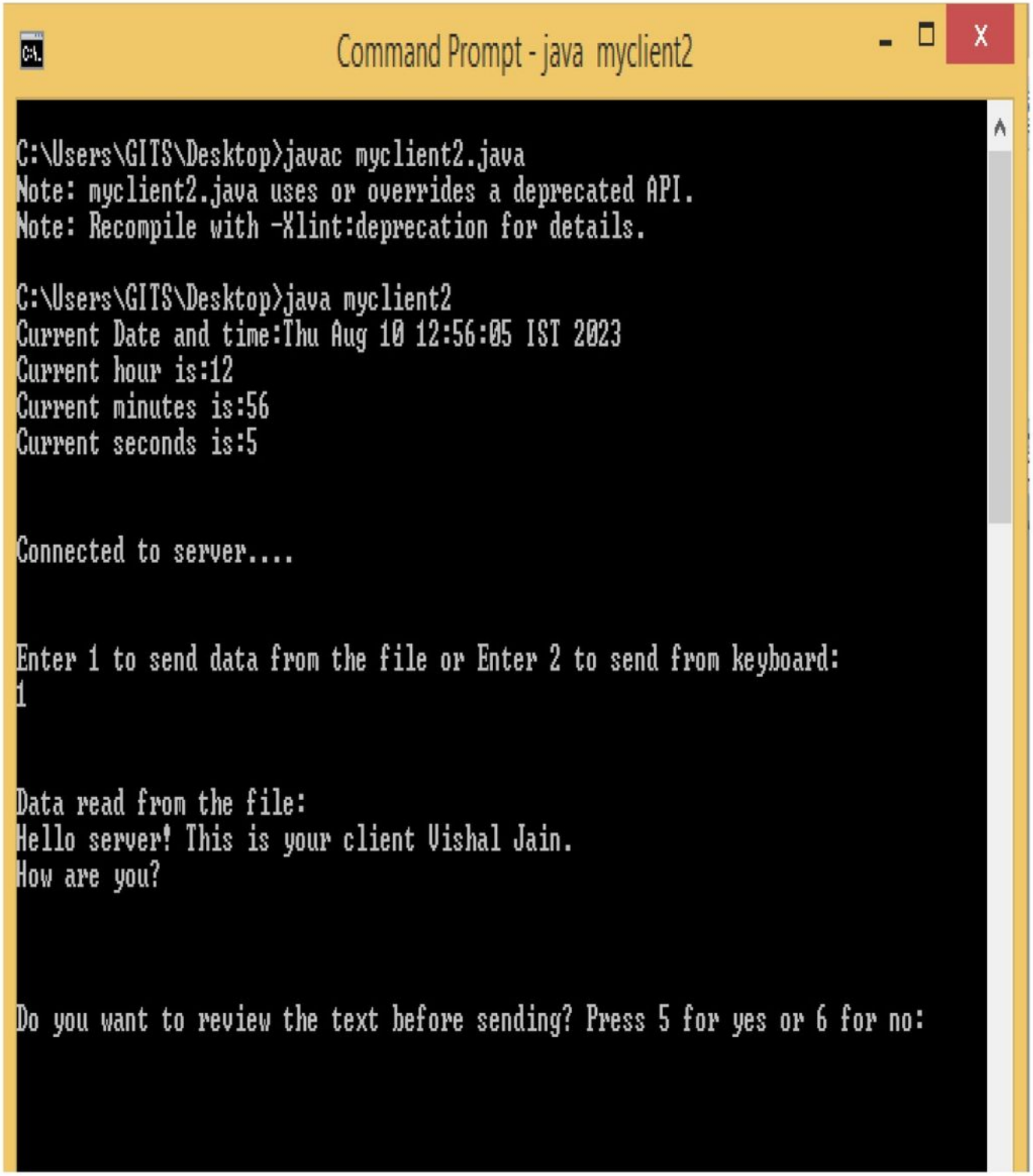

Figure 7

The data in the file are displayed, and the client is asked to verify the file before it is sent to the server.

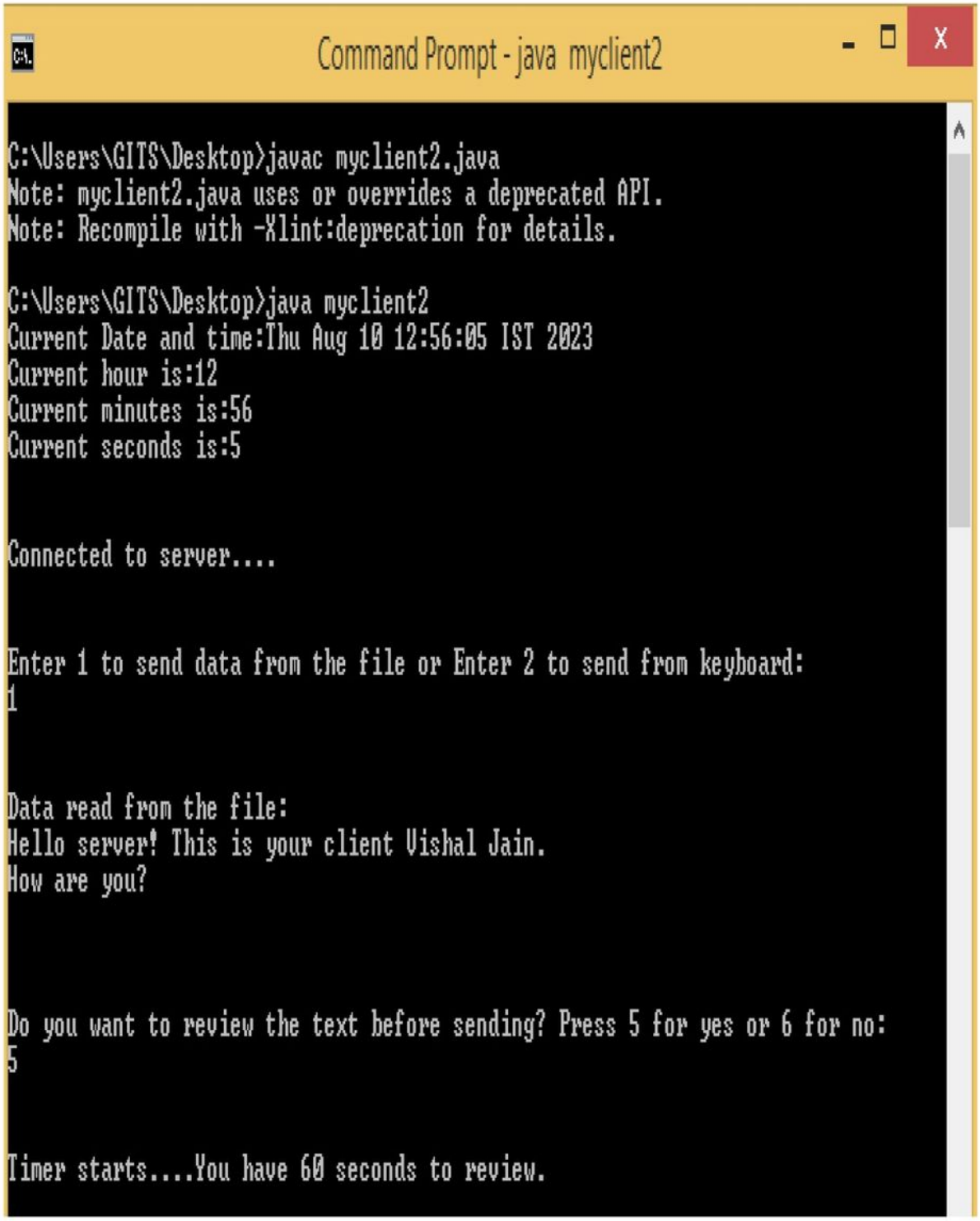

Figure 8

User input 5 and a timer of 60 seconds starts to review the text before sending

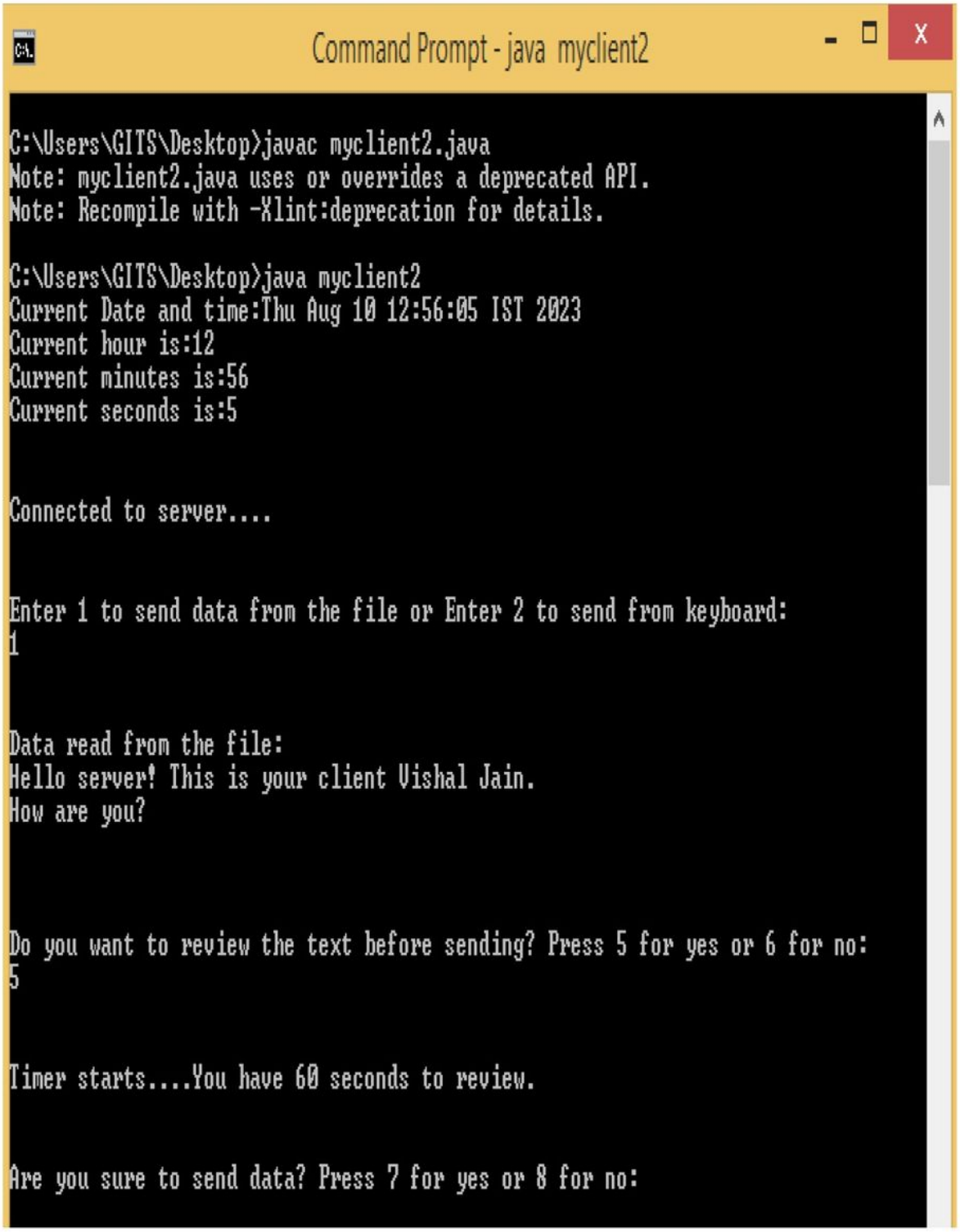

Figure 9

After 60 seconds, the client is asked to send the data to the server

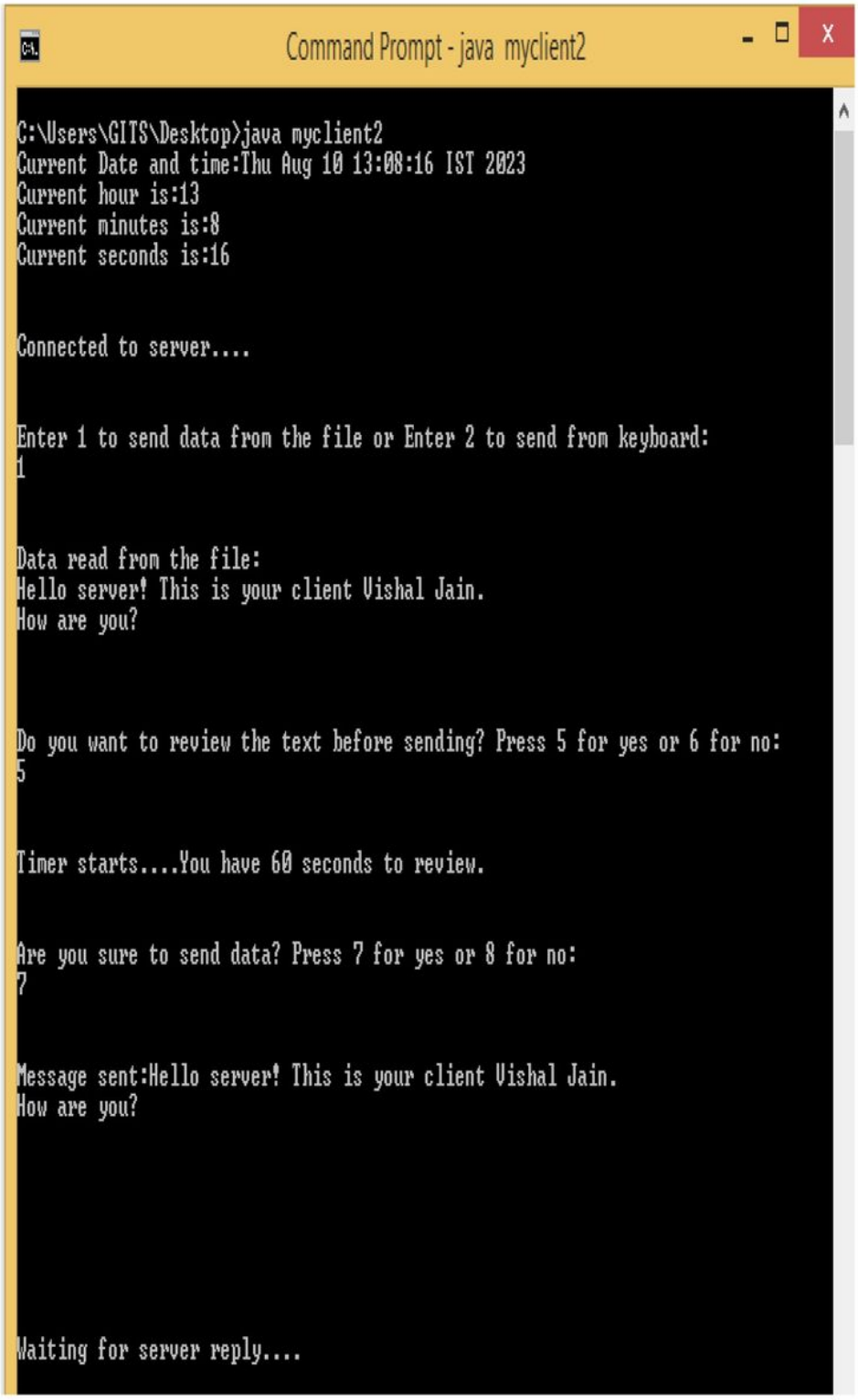

Figure 10

Client input 7, where the message is transferred to the server. The client waits for the server's reply

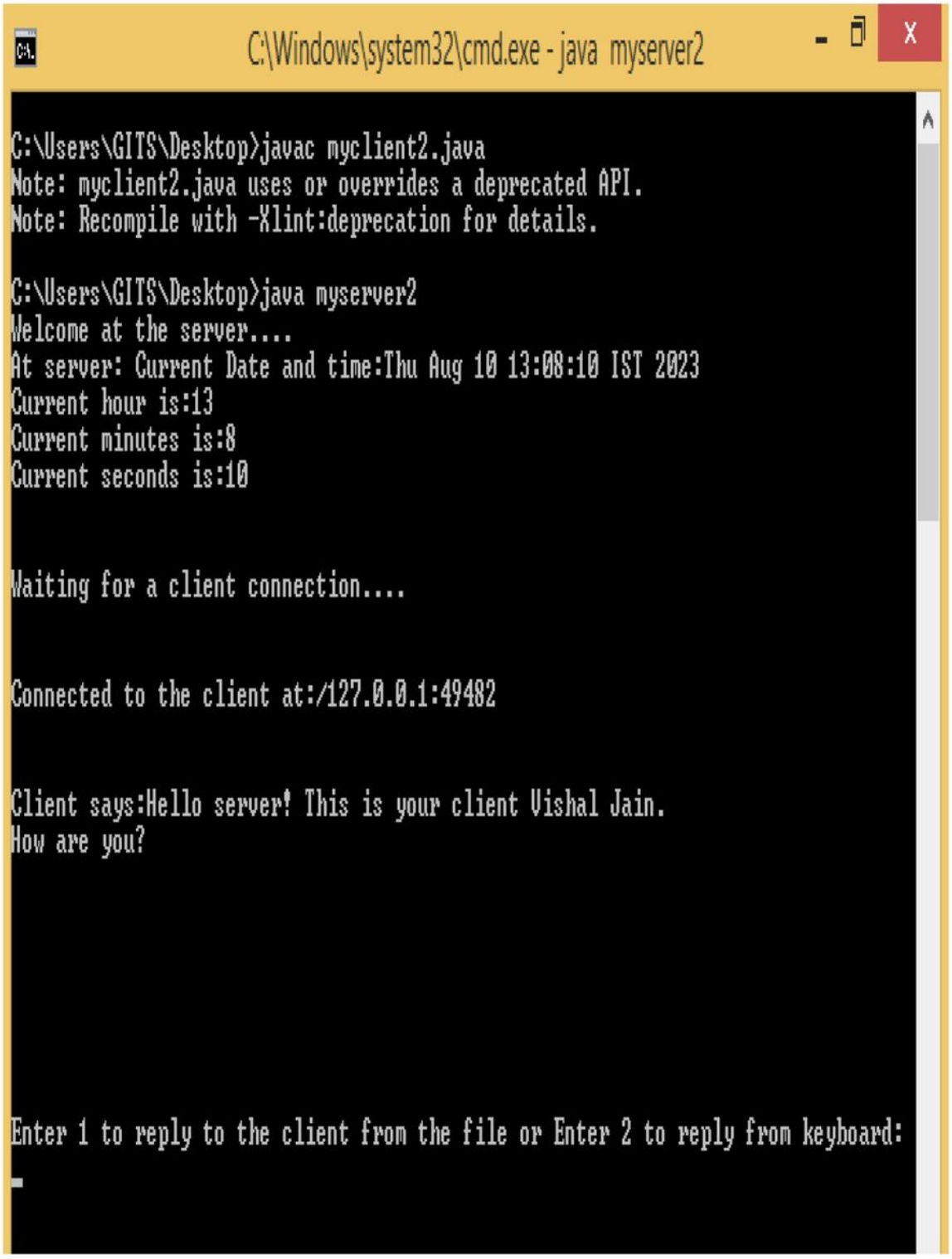

Figure 11

The server receives the client's message, and he is asked to reply to the client

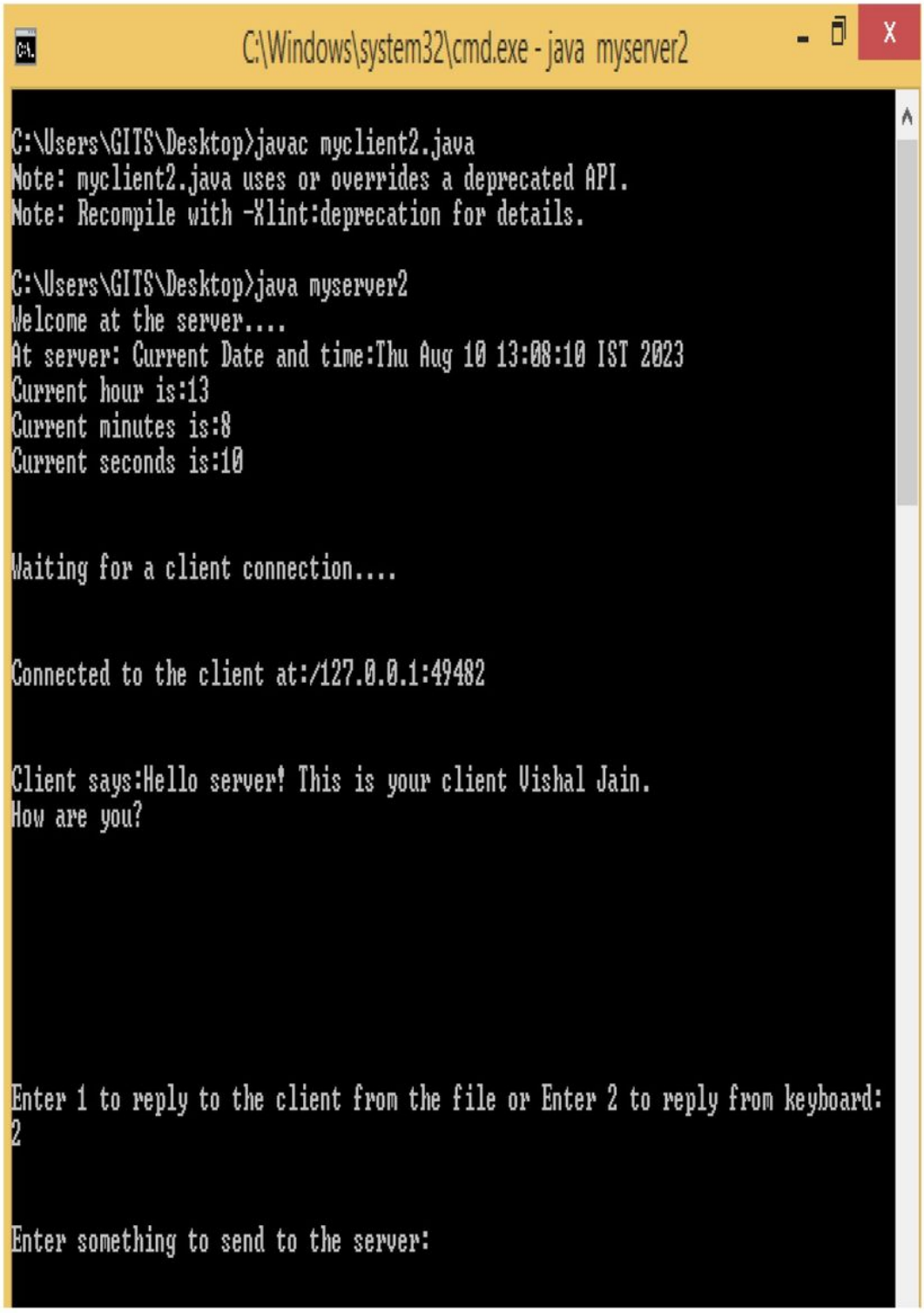

Figure 12

Server input 2. He is asked to enter the message from the keyboard

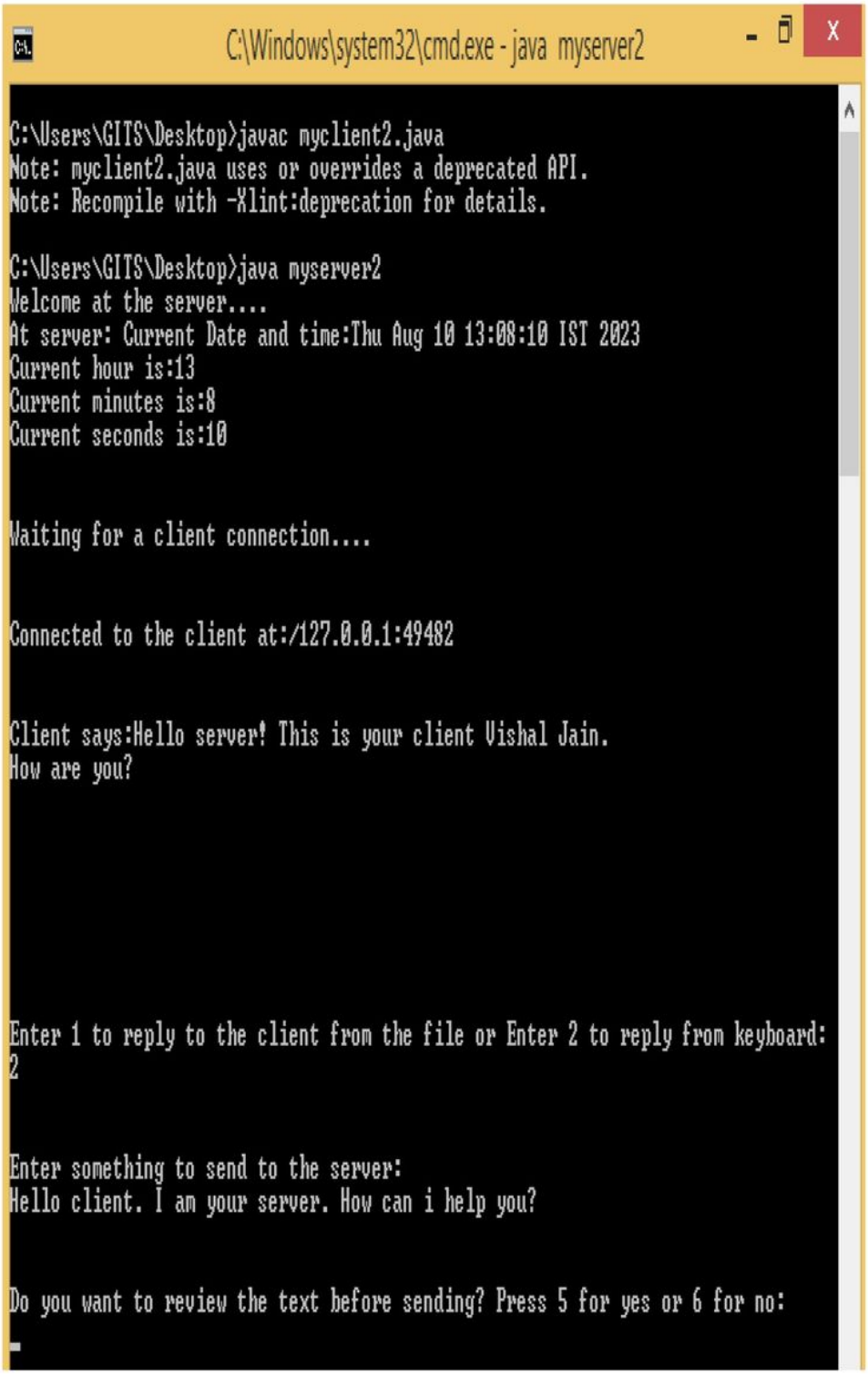

Figure 13

The server sends some input messages, and the user is asked whether to review the message before it is sent to the client

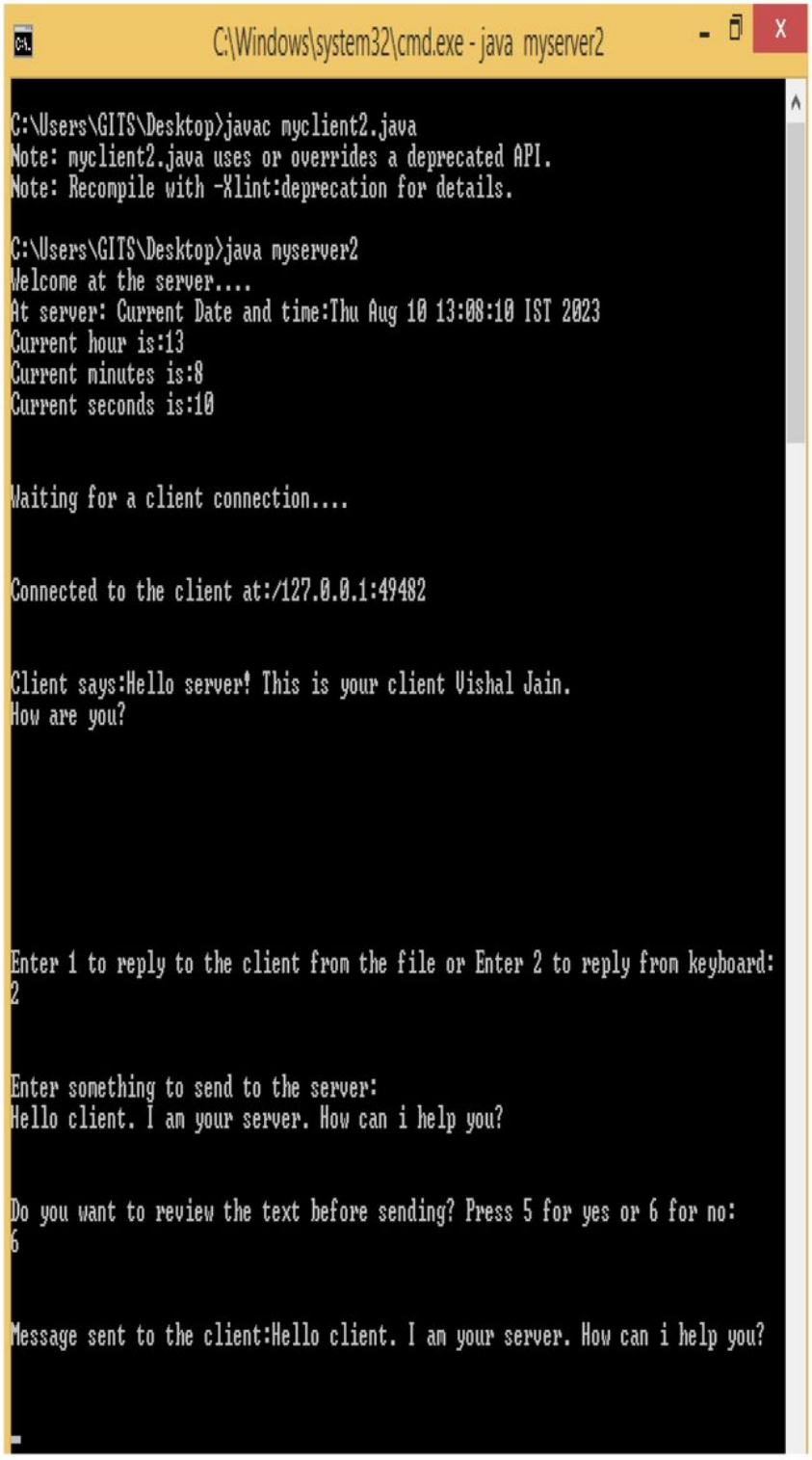

Figure 14

Server inputs 6 and the message is delivered to the client without review (no 60 seconds timer to review)

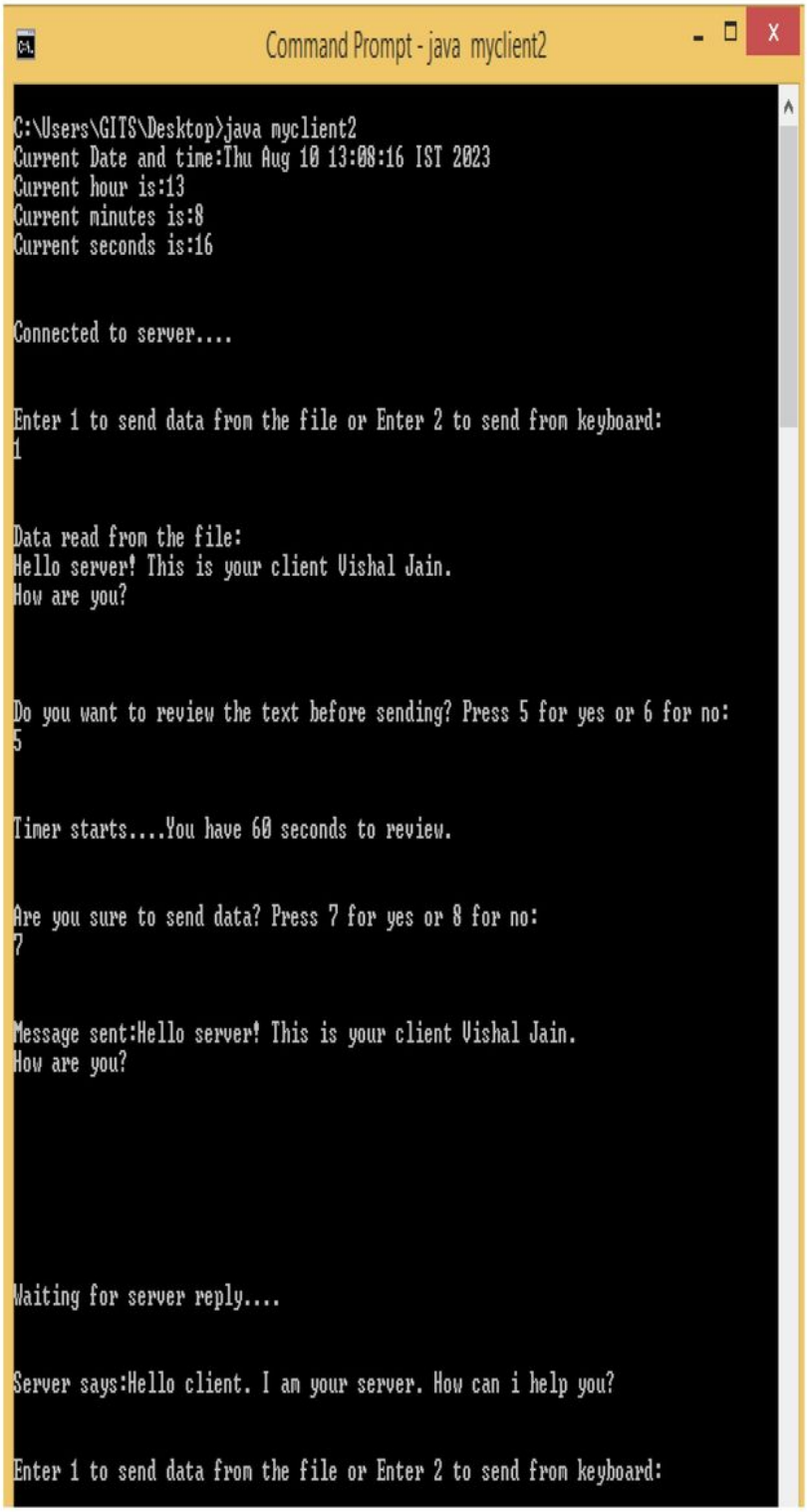

Figure 15

The client receives the server's message, and he is asked to return the data.

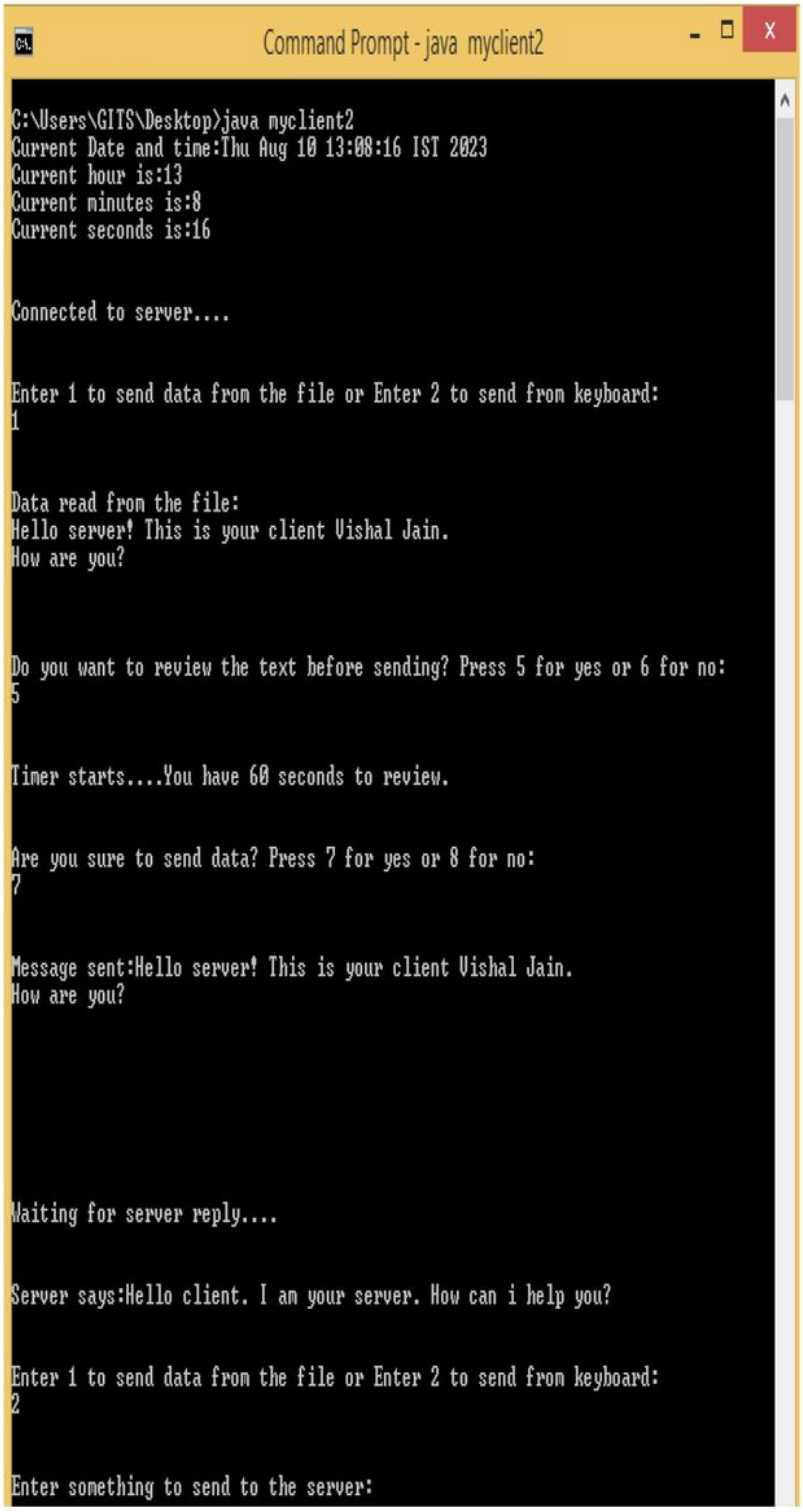

Figure 16

Client input 2: The participant is asked to enter a message from the keyboard

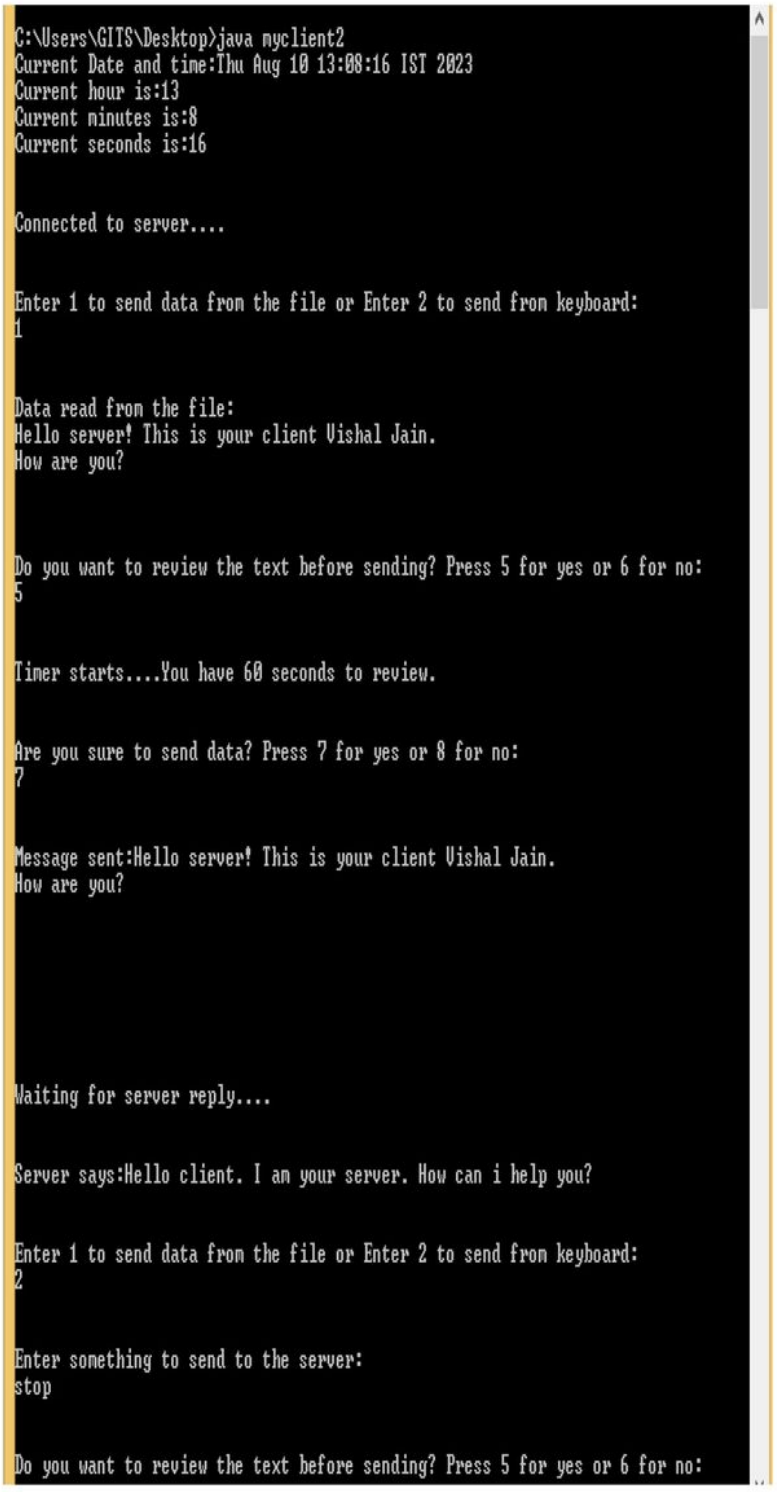

Figure 17

Client input “stop”, which he is asked to review before sending

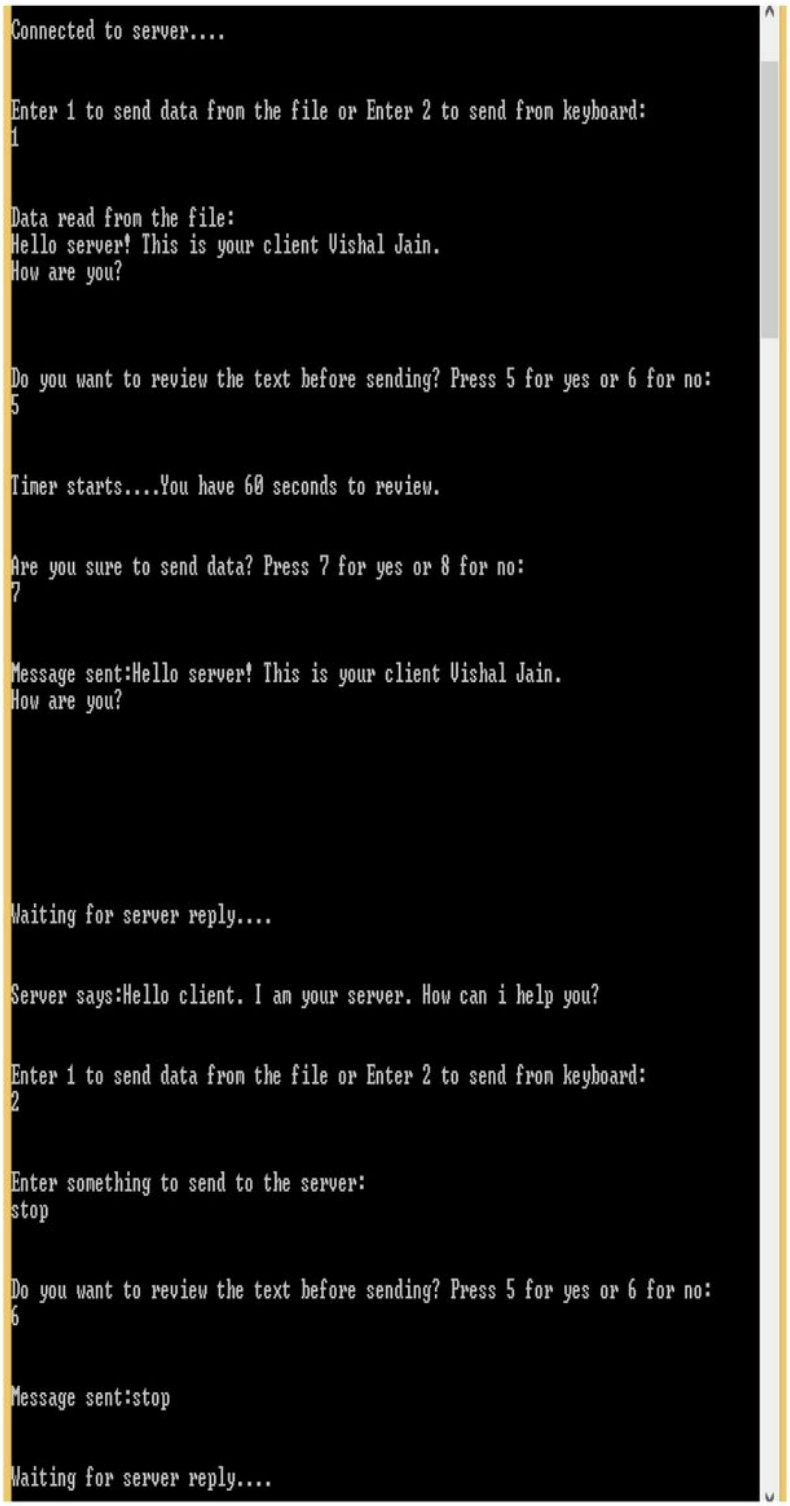

Figure 18

Client input 6. The message is delivered without review by the client. The client waits for the server's reply

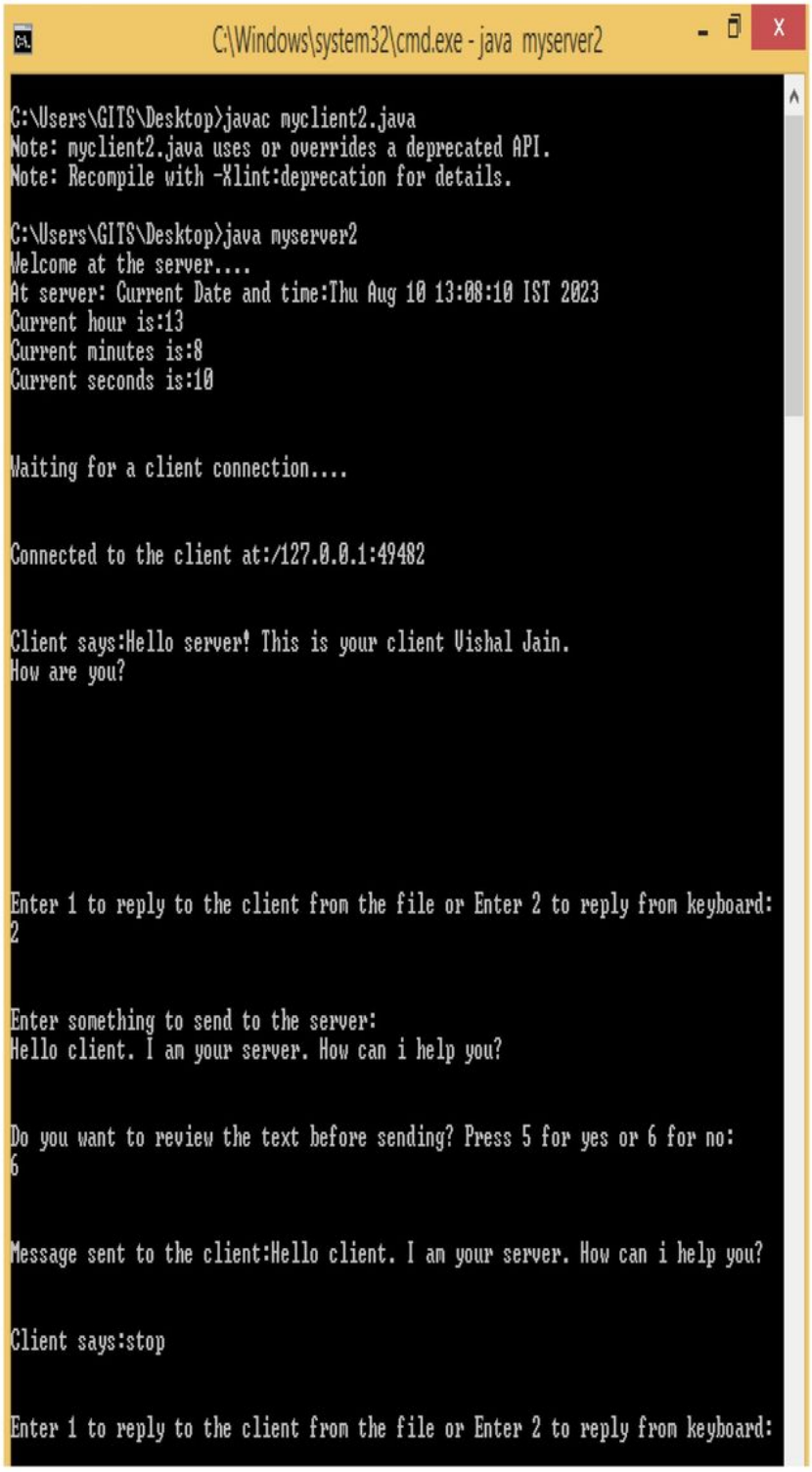

Figure 19

The server receives a client message, and he is asked to reply

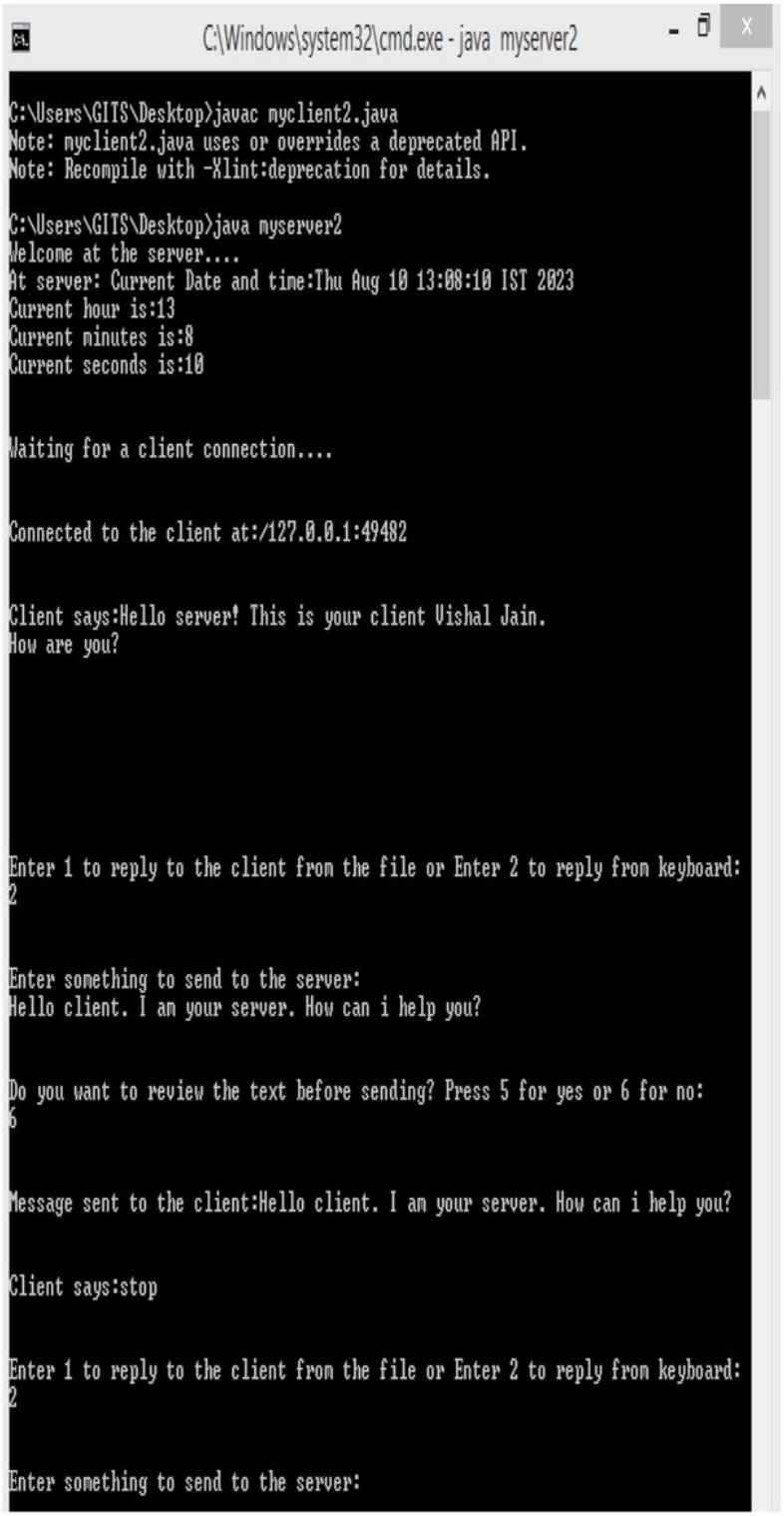

Figure 20

Server input 2. He is asked to input from the keyboard

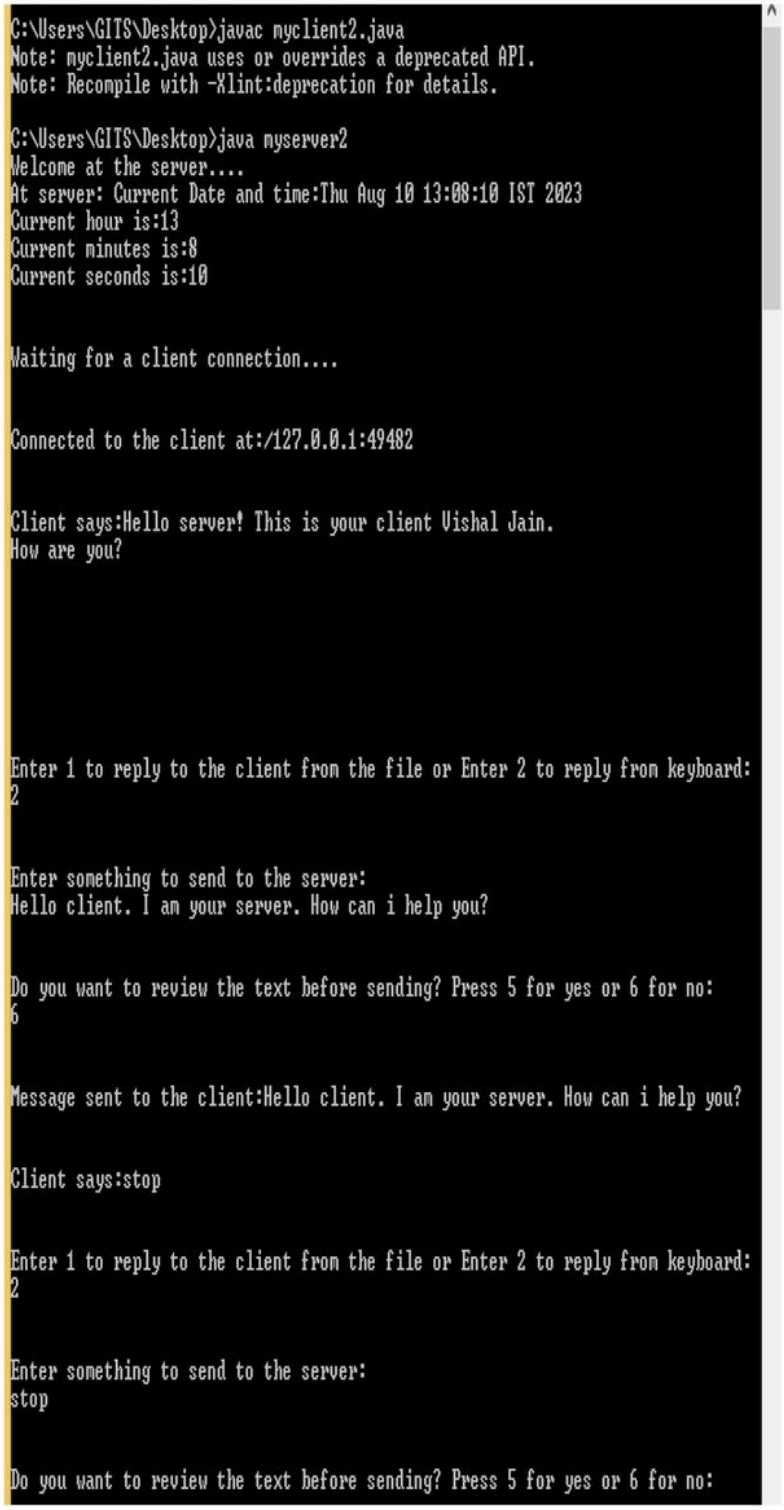

Figure 21

Server inputs “stop”, which he is asked to review before sending

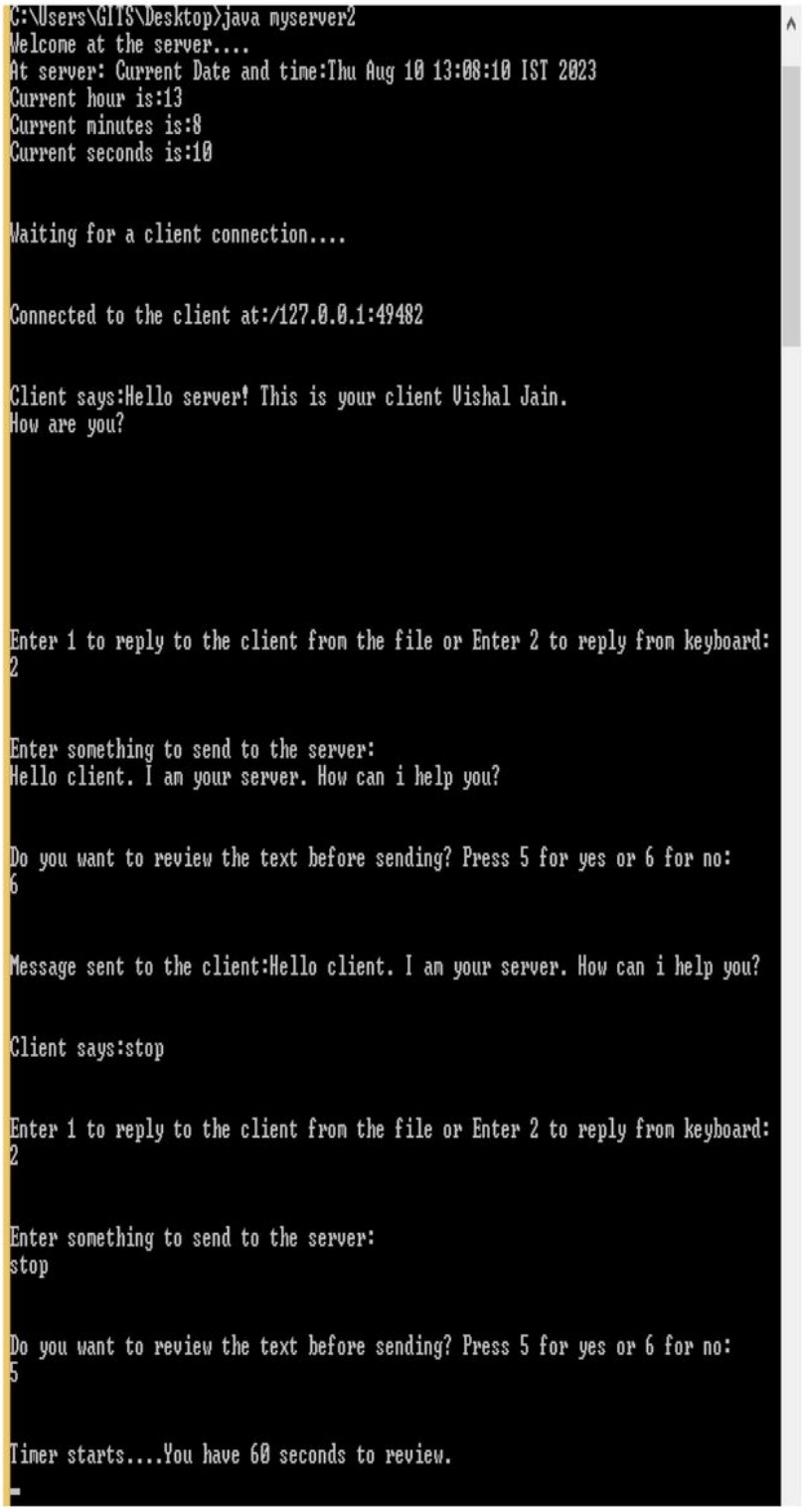

Figure 22

Server input 5 and a timer of 60 seconds starts to review it

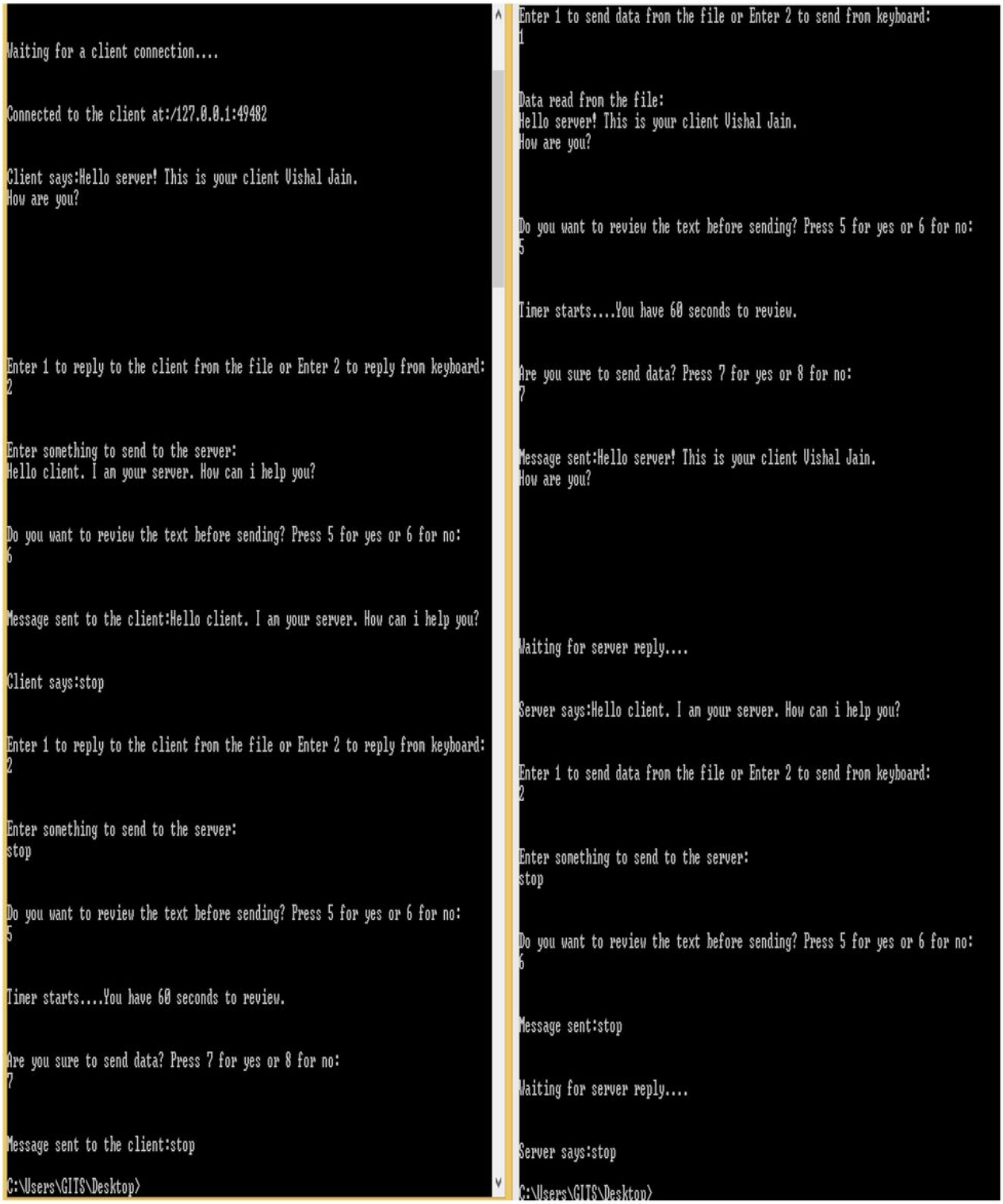

Figure 23

After 60 seconds, the server is asked to send the message to the client or not. Server inputs 7, and the message is delivered to the client. The client and server are disconnected because both parties have replied to each other

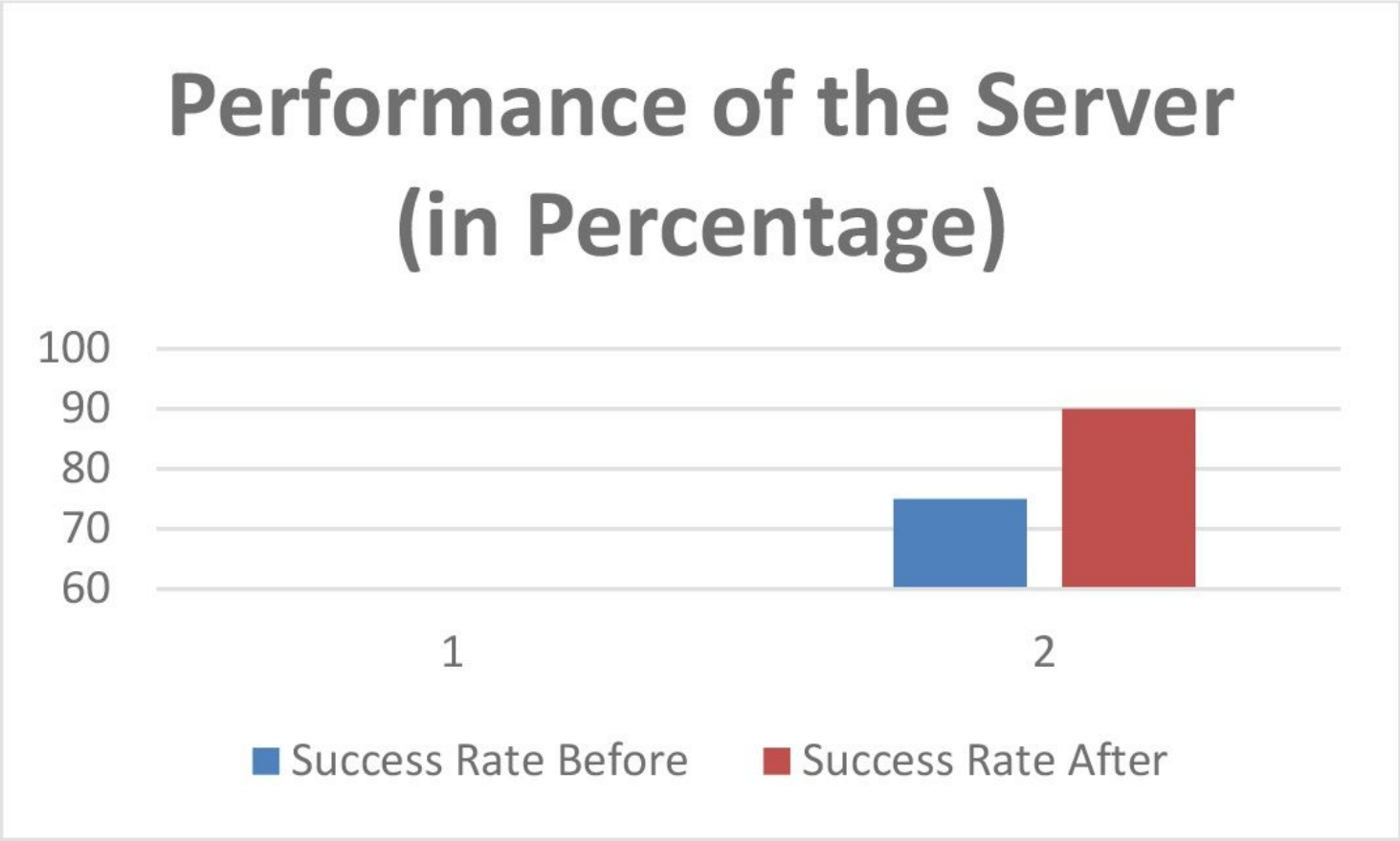

Figure 24

Performance of the server before and after the proposed algorithm was implemented

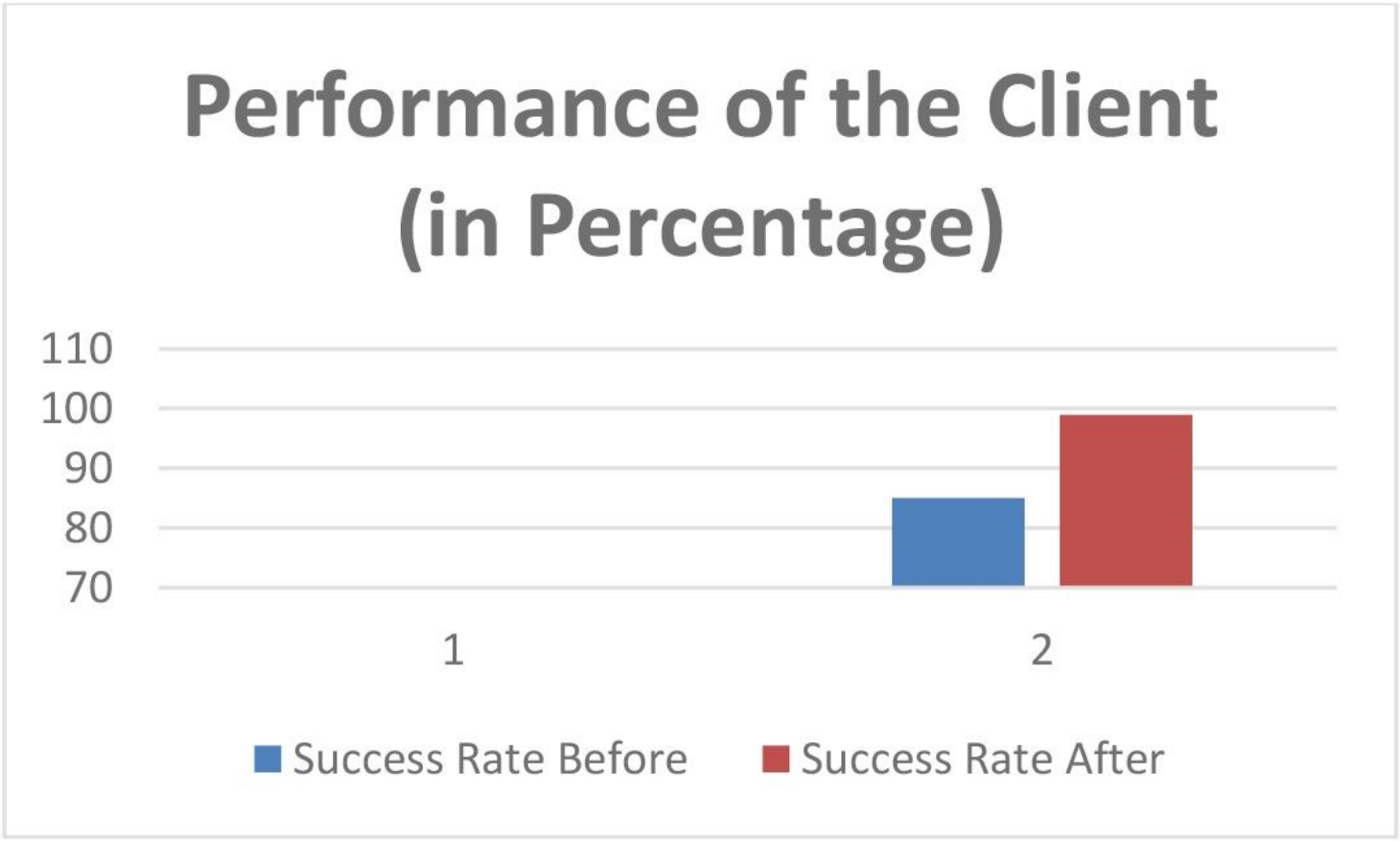

Figure 25

Performance of the client before and after the proposed algorithm was implemented