ENTS640 PROJECT REPORT

UDP implementation in Java to ensure reliable data transmission

Group Members:

Aniket P Bhonde UID: 113759758

Prasad Patil UID: 113626715

Table of Contents

PROBLEM STATEMENT

The prime goal of the project is to design a distributed networking JAVA application for effective communication between a client and server. The application should guarantee reliable data transfer between the two communicating parties. The application should employ JAVA's UDP socket and provide reliable data transfer functionality on top of UDP's unreliable communication services by implementing Integrity Checksum. This Integrity Checksum will authenticate and authorize communication access for both the client and server. The protocol will use integrity check, timeout and retransmission, which are required for successful communication between the client and server.
The server should serve the following three text files to the client:

- directors_message.txt
- program_overview.txt
- scholarly_paper.txt

The messages exchanged between the client and server should be text-oriented and consist of human readable character sequences. To calculate the integrity check value at client, the character sequence should be converted to a sequence of 16-bit words. Each 16-bit word should contain ASCII codes for two consecutive message characters. The final calculated 16-bit word is the checksum value. Similarly, the same procedure should be carried out at the server and the integrity checksum value should be recalculated. Then, the computed integrity checksum value and the received integrity checksum value should be compared. If they don't match, then there are some errors in the received message that is, integrity check has failed

UML DIAGRAMS

The project uses four classes in total. Two at the client and the server end each. The UML diagrams for the same are given below. First half of the UML diagram block represents the fields used and second half represents the methods used. Specific signs are used to describe access specifiers for both fields and methods. That is "-" for private and "+" for public.

Client

ClientMain
+main ( ) - static

Client
-PROTOCOL: String - static, final -SEPARATOR: String – static, final -reqMsg: String -fileName: String -checkSum: String -serResCheckSum: String -timeoutCounter: int -timeOut: int
+goToFirstStep ( ) -sendReqMsg (timeout int) -continueQuery ( ) -generateChecksum (div [ ]short): short -characterFitting (msg String): [ ] short -selectFileName ( ): boolean +getProtocol ( ): String +getSeparator ( ): String +setFileName (fileName String) +getFileName ( ): String +setCheckSum (checkSum String) +getCheckSum ( ): String +setTimeoutCounter (timeoutCounter int) +getTimeoutCounter ( ): int +setTimeOut (timeOut int) +getTimeOut ( ):int +setSerResCheckSum (serResCheckSum String) +getSerResCheckSum ( ): String +setReqMsg (reqMsg String) +getReqMsg ( ): String

Server

ServerMain
+main ( ) - static

Server
-PROTOCOL: String – static, final -SEPARATOR: String – static, final -checksumGenerated: String -checksumReceived: String -responseCode: String -contentLength: String -responseMsg: String -responseChecksum: String
+receiveMsg ( ) -getFileContent (receivedMsg String): boolean -checkProtocol (receivedMsg String): boolean -checkSyntax (receivedMsg String): boolean -generateResponse ( ) -charactersFitting (msg String): [ ]short -generateChecksum (div [ ] short): short +getProtocol ( ):String +getSeparator ( ):String +setChecksumGenerated (checksum String) +getChecksumGenerated ( ):String +setChecksumReceived (checksum String) +getChecksumReceived ( ): String +setResponseCode (responseCode String) +getResponseCode ( ): String +setContentLength (contentLength String) +getContentLentgh ( ): String +setResponseMsg(responseMsg String) +getResponseMsg ( ):String +setFileContent (content String) +getFileContent ( ): String +setResponseChecksum (responseChecksum String) +getResponseChecksum ( ):String

DESIGN APPROACH

Design approach describes both client and server operations in brief along with the methods used to perform specific tasks as required for the reliable communication between server and client.

Client

1. Client operation is implemented using 2 classes- ClientMain and Client.
2. ClientMain class contains the main method in which object of class Client is created for running the client operation.
3. The method goToFirstStep() of class Client is called that initiates client's operation. This method internally calls the method selectFileName() which asks the user to enter a filename to be requested from the server. This method is called in a while loop which keeps on asking the user for filename until he enters it correctly.
4. The method selectFileName(), uses a method matcher() of class Pattern from the library "java.util.regex.Pattern" to check syntax of filename and the extension, entered by the user.
5. Further, program uses Protocol (that is ENTS/1.0 Request), [CR+LF], file name entered and [CR+LF] to generate a single string (here, carriage return and line feed are denoted by CR and LF respectively). This string is used in the method characterFitting(). This method is defined to fit 2 consecutive characters of a string into a single 16-bit word and generate array of such 16 but words.
    - If string length N is EVEN, then the length of array is set to N/2 as 2 consecutive characters are fitted into a single 16-bit word
    - To fit 2 consecutive characters in to a single 16-bit word, left-shift operator is used to left shift ASCII value 1st character by 8 and OR that with the ASCII value of next character
    - If string length N is ODD, the array length is set to N/2 +1, and when the array index reaches the last character, the ASCII value of last character is 'OR'ed with O.
    - Array of such 16 bit words are used to generate the checksum
6. generateChecksum() is the method used to calculate integrity checksum value based on array of 16 bit words generated in the previous step.
7. Now we assemble the total request message containing the three desired fields- Protocol (ENTS/1.0 Request), Filename and Integrity checksum value along with CR and LF at the end of each field.
8. Next, timeout is set to1000 milliseconds (1sec) and timeout counter to 0. And sendReqMsg() method is called to send the assembled request message to the server. The string of request message is converted into bytes which are put into data packet created at the client.
9. Now a datagram socket is created in the method sendReqMsg() and the request message is sent. Timeout now starts and client waits for server's response.

- Timeout counter is incremented by 1 each time it fails to receive server's response within the timeout and double the timeout for each failed attempt. After 4 unsuccessful attempts, display communication failure message and ask user if he wants to continue (using the method continueQuery()) sending request. If user enters 'y' then go to the initial step and ask user to enter filename again. If user enters 'n' then terminate the program.
- Reset the timeout counter value to 0 and timeout to 1 second once the response has been received by the client.
- Response received is split at "[CR+LF]" (That is "\r\n"). This helps to separate the received checksum from the packet received and also to calculate checksum of response which is used further to verify the integrity of the message received
- Send Protocol, Response Code, Content Length, Content along with CR and LF after each field to the function characterFitting() as a single string that fits two consecutive characters of the string into a single 16 bit word.
- The method generateChecksum() is again called to calculate the integrity checksum value of response message based on array of 16 bit words generated in the previous step and compare its value with the with the one received from the server.

10. Display the corresponding file content if response code is 0, else display relevant error messages as described below.
    - If response code=1, display 'Error: integrity check failure. The content has one or more bit errors'.
    - If response code=2, display 'Error: malformed request. The syntax of the request message is not correct.
    - If response code=3, display 'Error: non-existent file. The file with the requested name does not exist'
    - If response code=4, display 'Error: wrong protocol version. The version number in the request is different from 1.0'
11. Ask the user whether he/she wants to continue sending the request message (Using the method continueQuery()). Take the corresponding action based on user's response. That is for example, if user says 'y' then for response code 1 resend the same request and for response code 2 ask the user to enter new file name. If user enter 'n', terminate the program.

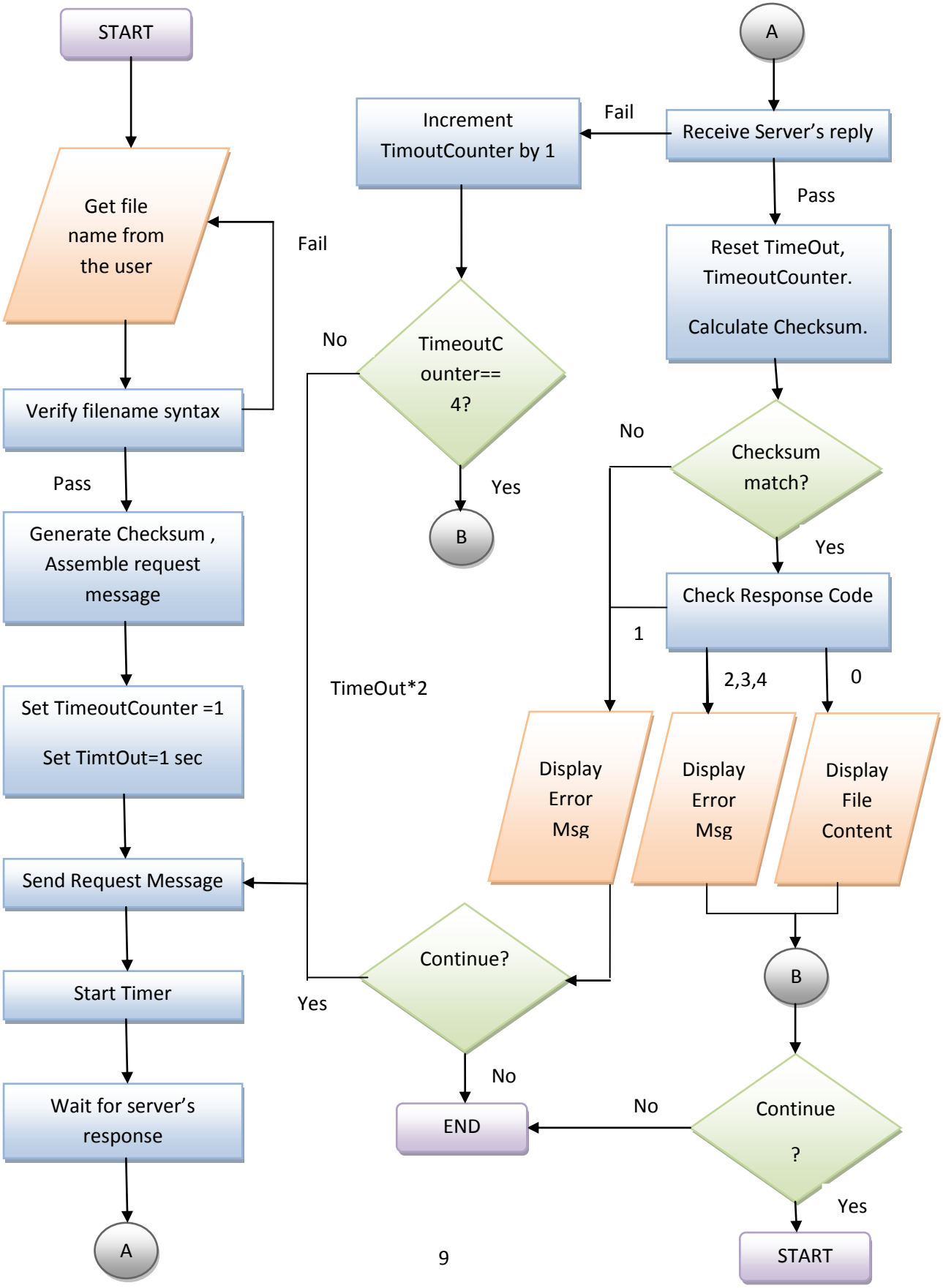
Server

1. Server operation is implemented using 2 classes- ServerMain and Server.
2. ServerMain class has the main method in which object of Server class is created
3. Server creates a datagram socket and waits for client's request message.
4. Received request message from the client is in bytes. Received bytes are converted into a single string and split the received message at [CR+LF] (i.e \r\n). The last split of which contains the checksum generated at client end. This checksum is saved separately which is to be used in further comparison. First two splits contain Protocol and Filename along with the extension. These two splits are combined and put into a single string along with CR and LF at the end of each field. This string is passed to charactersFitting() method.
5. charactersFitting() method is called to fit first 2 consecutive characters of the string into a single 16-bit word and generate array of 16-bit words. Array of such 16-bit words is used to generate integrity checksum value. Checksum is computed using the method generateChecksum().
6. Compare the received checksum value (saved separately in step 4) and the server's locally generated checksum value (generated in the precious step). If the integrity check passes, following actions are performed.
    - Syntax of the request message is checked,
        - If filename is in <filename.extension> format.
        - If number of fields present in the request message are 3.
        - If filename contains anything other than alphanumeric characters and space character.
        - If extension of the filename contains anything other than alphanumeric characters.
        - checkSyntax() method is used to perform all the above checks.
        - If any of these check fail then set response code to 2.
    - Call the method checkProtocol() to check for protocol version number. If version number is not 1.0, set the response code to 4.
    - If syntax and protocol version are correct, Call the method getFileContent() to fetch the file content of the file requested by the client. If file contents are fetched successfully then set response code to 0. Else set it to 3.
7. If integrity check fails, set the response code=1.
8. Call the method generateResponse() to generate the response message based on response code set in the previous steps. If response codes are either 1/2/3/4 set file content to null, else fill the content field with file content obtained in step 6.3 in the same method.
9. generateResponse() intern calls charactersFitting() and generateChecksum() to Calculate integrity checksum value based on protocol (ENTS/1.0 Response), response code, content length, file content and CR+LF at the end of each field.
10. Packet is created consisting of response message and it is sent to the client.
11. Wait for the client's message

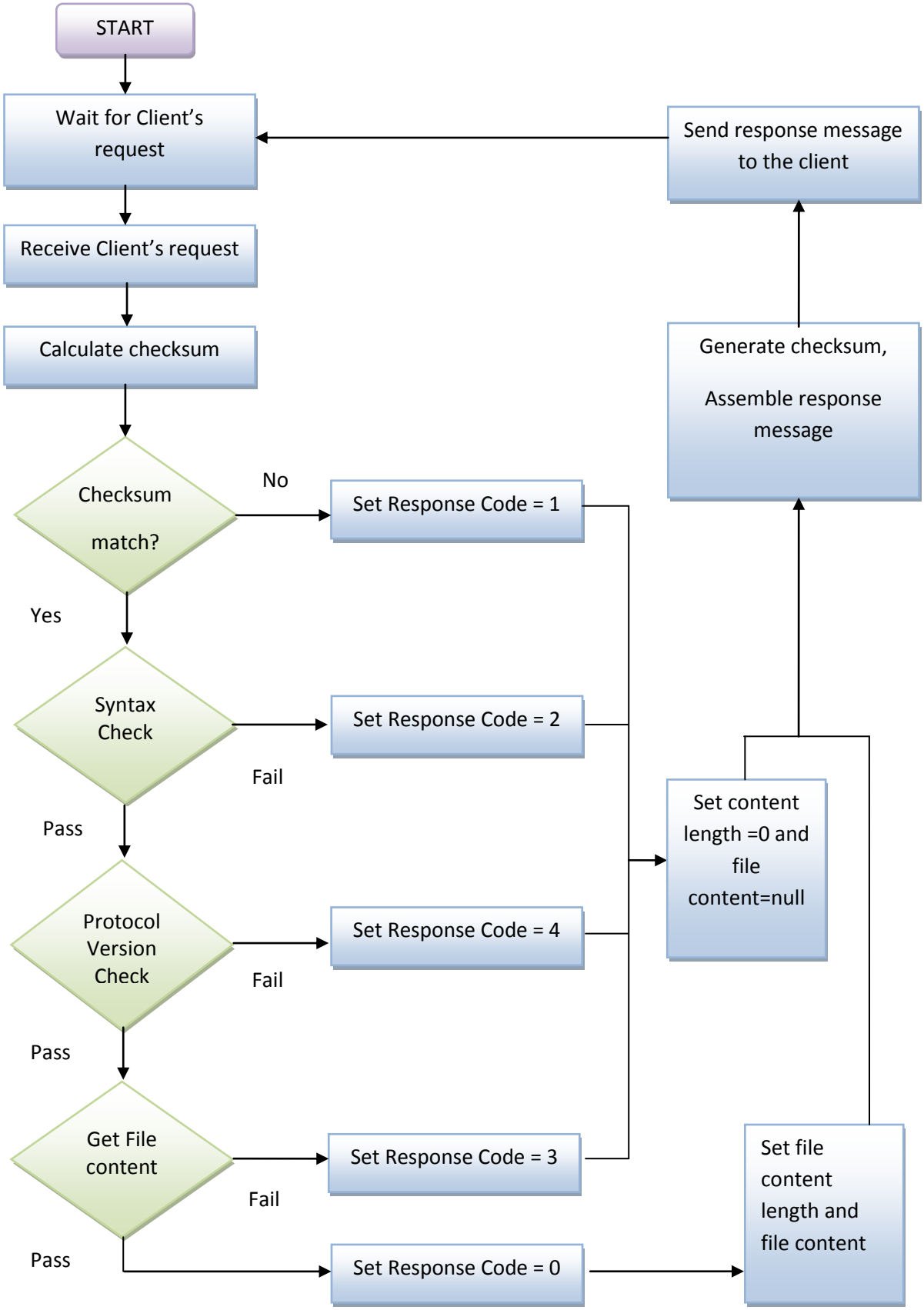
FLOWCHARTS

The section gives brief diagrammatical explanation for client and server operation.

Client



Server


START
Wait for Client's request
Receive Client's request
Calculate checksum
Checksum match?
No
Set Response Code = 1
Yes
Syntax Check
Set Response Code = 2
Fail
Pass
Protocol Version Check
Set Response Code = 4
Fail
Pass
Get File content
Set Response Code = 3
Fail
Pass
Set Response Code = 0
Set content length =0 and file content=null
Set file content length and file content
Generate checksum,
Assemble response message
Send response message to the client

OUTPUTS

The section describes the sample output for each possible test case for the program. Here the output is described as communication occurs between server and the client. That is each step is described as request/response is sent/received by the client. Console's for both client and server are displayed as part of sample outputs.

The path of the directory where all the text files are stored is to be set in the server's java program before taking the sample outputs. getFileContent() is the method of class Server where the path of the directory is set.

Sample Output 1: Requesting correct file name

1. Starting server:

```
Waiting for a message from client..
```

2. Starting client and entering a correct choice:

```
Available files are:
1:directors_message.txt
2:program_overview.txt
3:scholarly_paper.txt

Enter the file name:
directors_message.txt
```

3. Server's console after receiving a request from the client:

```
Waiting for a message from client..
The received message is:
ENTS/1.0 Request
directors_message.txt
21

Integrity check: SUCCESS
Syntax check: SUCCESS
Protocol version check: SUCCESS
File content fetch: SUCCESS

Response message is:
ENTS/1.0 Response
0
1561
Thank you for your interest in the Masters in Telecommunications Program at the
University of Maryland!

Our program offers a unique opportunity to engage in cross-disciplinary coursework
from the Department of Electrical and Computer Engineering and the Robert H. Smith
School of Business. This exceptional combination allows students to enhance their
technical background and gain the necessary business and management skills to
advance their careers.

Our mission is to provide industry-oriented graduate education in
telecommunications that produces graduates who are business-minded, possess the
```

needed technical skills and are well versed in the latest technology trends. We offer a wide range of interesting and relevant technical courses in wireless communications, computer networking and cyber-security. Our thought-provoking business and entrepreneurship courses expose our students to economics, finance, marketing, and management.
The Master of Science degree in Telecommunications is very well respected in the industry. Our recent graduates have landed jobs at companies like Cisco, Juniper Networks, Amazon, Qualcomm, Hughes Network Systems, and Intel. After working in the industry for a few years, many of our alumni advance to engineering managers, product managers, and program managers. There are even directors and vice presidents among our alumni.

I invite you to explore our website or contact us directly at telecomprogram@umd.edu with any questions you may have. We look forward to hearing from you!

Zoltan Safar, Director
22787

Response sent to the client

Waiting for a message from client..

4. Client's console after receiving response from the server:

```
Available files are:
1:directors_message.txt
2:program_overview.txt
3:scholarly_paper.txt

Enter the file name:
directors_message.txt

Attempt 1
Message sent to the server is:
ENTS/1.0 Request
directors_message.txt
21

Waiting for server's response..

Response received from the server.

COMMUNICATION SUCCESSFUL!

File content is:
Thank you for your interest in the Masters in Telecommunications Program at the
University of Maryland!

Our program offers a unique opportunity to engage in cross-disciplinary coursework
from the Department of Electrical and Computer Engineering and the Robert H. Smith
School of Business. This exceptional combination allows students to enhance their
technical background and gain the necessary business and management skills to
advance their careers.

Our mission is to provide industry-oriented graduate education in
telecommunications that produces graduates who are business-minded, possess the
needed technical skills and are well versed in the latest technology trends. We
offer a wide range of interesting and relevant technical courses in wireless
```

```
communications, computer networking and cyber-security. Our thought-provoking
business and entrepreneurship courses expose our students to economics, finance,
marketing, and management.

The Master of Science degree in Telecommunications is very well respected in the
industry. Our recent graduates have landed jobs at companies like Cisco, Juniper
Networks, Amazon, Qualcomm, Hughes Network Systems, and Intel. After working in
the industry for a few years, many of our alumni advance to engineering managers,
product managers, and program managers. There are even directors and vice
presidents among our alumni.

I invite you to explore our website or contact us directly at
telecomprogram@umd.edu with any questions you may have. We look forward to hearing
from you!

Zoltan Safar, Director


Do you want to continue sending request ? (y/n):
```

Sample Output 2: Requesting filename with incorrect syntax

1. Starting server:

```
Waiting for a message from client..
```

2. Starting client and entering filename with incorrect syntax

```
Available files are:
1:directors_message.txt
2:program_overview.txt
3:scholarly_paper.txt

Enter the file name:
abcd.pdf
```

3. Server's console after receiving a request from the client

```
Waiting for a message from client..
The received message is:
ENTS/1.0 Request
abcd.pdf
-5310

Integrity check: SUCCESS
Syntax check: FAILED

Response message is:
ENTS/1.0 Response
2
0

-18892

Response sent to the client

Waiting for a message from client..
```

4. Client's console after receiving response from the server

```
Available files are:
1:directors_message.txt
2:program_overview.txt
3:scholarly_paper.txt

Enter the file name:
abcd.pdf

Attempt 1
Message sent to the server is:
ENTS/1.0 Request
abcd.pdf
-5310

Waiting for server's response..

Response received from the server.

ERROR: malformed request. The syntax of the request message is not correct.

Do you want to continue sending request ? (y/n):
```

Sample Output 3: Requesting non-existing text file

1. Starting server:

```
Waiting for a message from client..
```

2. Starting client and entering non existing text file:

```
Available files are:
1:directors_message.txt
2:program_overview.txt
3:scholarly_paper.txt

Enter the file name:
abcd.txt
```

3. Server's console after receiving the request from the client:

```
Waiting for a message from client..
The received message is:
ENTS/1.0 Request
abcd.txt
788

Integrity check: SUCCESS
Syntax check: SUCCESS
Protocol version check: SUCCESS
File content fetch: FAILED

Response message is:
ENTS/1.0 Response
```

```
3
0

-11517

Rsponse sent to the client

Waiting for a message from client..
```

4. Client's console after receiving the response from the server:

```
Available files are:
1:directors_message.txt
2:program_overview.txt
3:scholarly_paper.txt

Enter the file name:
abcd.txt

Attempt 1
Message sent to the server is:
ENTS/1.0 Request
abcd.txt
788

Waiting for server's response..

Response received from the server.

ERROR: non-existent file. The file with the requested name does not exist.

Do you want to continue sending request ? (y/n):
```

Sample Output 4: Making a request with wrong protocol version

1. Starting the server

```
Waiting for a message from client..
```

2. Starting the client and requesting a file name

```
Available files are:
1:directors_message.txt
2:program_overview.txt
3:scholarly_paper.txt

Enter the file name:
program_overview.txt
```

3. Server's console after receiving a request from the client

```
Waiting for a message from client..
The received message is:
ENTS/2.0 Request
program_overview.txt
-8528

Integrity check: SUCCESS
```

```
Syntax check: SUCCESS
Protocol version check: FAILED

Response message is:
ENTS/1.0 Response
4
0

14418

Rsponse sent to the client

Waiting for a message from client..
```

4. Client's console after receiving response from the server

```
Available files are:
1:directors_message.txt
2:program_overview.txt
3:scholarly_paper.txt

Enter the file name:
program_overview.txt


Attempt 1
Message sent to the server is:
ENTS/2.0 Request
program_overview.txt
-8528

Waiting for server's response..

Response received from the server.

ERROR: wrong protocol version. The version number in the request is different from
1.0.

Do you want to continue sending request? (y/n):
```

Sample Output 5: When server does not respond

1. Starting the client and requesting a file name without starting the server.

```
Available files are:
1:directors_message.txt
2:program_overview.txt
3:scholarly_paper.txt

Enter the file name:
scholarly_paper.txt

Attempt 1
Message sent to the server is:
ENTS/2.0 Request
scholarly_paper.txt
```

```
-30662

Waiting for server's response..

No response from the server

Attempt 2
Message sent to the server is:
ENTS/2.0 Request
scholarly_paper.txt
-30662

Waiting for server's response..

No response from the server

Attempt 3
Message sent to the server is:
ENTS/2.0 Request
scholarly_paper.txt
-30662

Waiting for server's response..

No response from the server

Attempt 4
Message sent to the server is:
ENTS/2.0 Request
scholarly_paper.txt
-30662

Waiting for server's response..

No response from the server

Reached maximum time outs. COMMUNICATION FAILURE!!

Do you want to continue sending request ? (y/n):
```

Sample Output 6: Entering invalid file names

1. Starting the client and entering invalid filenames:

```
Available files are:
1:directors_message.txt
2:program_overview.txt
3:scholarly_paper.txt

Enter the file name:
*&*asdsa.txt
Invalid file name.
File name cannot contain special characters or it can not start with a number.
Filename can only be alphanumeric

Available files are:
1:directors_message.txt
2:program_overview.txt
```

```
3:scholarly_paper.txt


Enter the file name:
callitmagic.txt%$
Invalid file extension.
File extension can only be alphanumeric

Available files are:
1:directors_message.txt
2:program_overview.txt
3:scholarly_paper.txt

Enter the file name:
callitmagic.pdf.txt
Invalid file name.
Please enter the filename in <filename.extension> format

Available files are:
1:directors_message.txt
2:program_overview.txt
3:scholarly_paper.txt

Enter the file name:
```

Sample Output 7: Integrity check failure

1. Starting server:

```
Waiting for a message from client..
```

2. Starting client and making valid file name request:

```
Available files are:
1:directors_message.txt
2:program_overview.txt
3:scholarly_paper.txt

Enter the file name:
scholarly_paper.txt
```

3. Server's console after receiving a request from the client:

```
Waiting for a message from client..
The received message is:
ENTS/2.0 Request
scholarly_paper.txt
-198

Integrity check: FAILED

Response message is:
ENTS/1.0 Response
1
0

29821
```

```
Response sent to the client

Waiting for a message from client..
```

4. Client's console after receiving a response from the client:

```
Available files are:
1:directors_message.txt
2:program_overview.txt
3:scholarly_paper.txt

Enter the file name:
scholarly_paper.txt

Attempt 1
Message sent to the server is:
ENTS/2.0 Request
scholarly_paper.txt
-198

Waiting for server's response..

Response received from the server.

ERROR!! Integrity check failure! Server response has one or more bit errors.

Do you want to continue sending request ? (y/n):
```

CONCLUSION

As we know, User Datagram Protocol is an unreliable protocol for the data transmission. However using,

- Integrity Checksum at both client and server end
- Timeout at the client end, when it does not receive response from the server after particular set time and,
- Retransmission as needed

We can ensure the reliable data transmission. The design approach, flowchart and the sample outputs prove reliable communication between two parties using UDP. Hence, we can say that UDP can be implemented as reliable data transmission protocol with added complexity at the server and the client end.