**External Project Report on Computer Networking (CSE3034)**

**INSTITUTE OF TECHNICAL EDUCATION AND RESEARCH**

**(FACULTY OF ENGINEERING)**

i

**SIKSHA ‘O’ ANUSANDHAN (DEEMED TO BE UNIVERSITY), BHUBANESWAR,**

**ODISHA**

Establishing a peer-to-peer network of three machines (H1, H2, DS) using UDP



**Submitted by**

|  |  |
| --- | --- |
| **Name: Abhisek Swain** | **Reg. No.: 2141019415** |
| **Name: Aswini Sanket Patnaik** | **Reg. No.: 2141016180** |
| **Name: Mithilesh Swain** | **Reg. No.: 2141019416** |
| **Name: Nikita Mohapatra** | **Reg. No.: 2141013257** |
| **Name: Suparna Biswas** | **Reg. No.: 2141002133** |

**B. Tech. CSE 5th Semester (Section P )**

# Declaration

We, the undersigned students of B. Tech. of **Computer Science** Department hereby declare that we own the full responsibility for the information, results etc. provided in this PROJECT titled “**Establishing a peer-to-peer network of 3 machines (H1,H2,DS) using UDP”** submitted to **Siksha ‘O’ Anusandhan (Deemed to be University), Bhubaneswar** for the partial fulfillment of the subject **Computer Networking (CSE 3034)**. We have taken care in all respect to honor the intellectual property right and have acknowledged the contribution of others for using them in academic purpose and further declare that in case of any violation of intellectual property right or copyright we, as the candidate(s), will be fully responsible for the same.

|  |  |
| --- | --- |
| **Aswini Sanket Patnaik**  **2141016180** | **Abhisek Swain**  **2141019415** |
| **Nikita Mohapatra**  **2141013257** | **Mithilesh Swain**  **2141019416** |
| **Suparna Biswas**  **2141002133** | |

12/01/2024

**DATE:**

BHUBANESHWAR

**PLACE:**

# Abstract

This project involves setting up a peer-to-peer network with three machines (H1, H2, DS) using UDP in Java. H1 connects to DS to fetch data and time, and H2 then retrieves this information from H1. The focus is on practical implementation, emphasizing the efficiency of UDP for fast data exchange. Time synchronization between nodes is achieved through DaytimeServiceProvider (DS), offering insights into network communication, protocol selection, and the importance of time-related functionalities. Participants gain hands-on experience in UDP protocols and network synchronization for effective peer-to-peer communication.

# Contents

|  |  |  |  |
| --- | --- | --- | --- |
| **Serial**  **No.** | **Chapter No.** | **Title of the Chapter** | **Page No.** |
| 1. | 1 | Introduction | 1  2 |
| 2. | 2 | Problem Statement |  |
| 3. | 3 | Methodology | 4 |
| 4. | 4 | Implementation | 5  7  8  9  9 |
| 5. | 5 | Results and interpretation |  |
| 6. | 6 | Conclusion |  |
| 7. |  | References |  |
|  |  |  |  |

**Introduction**

The project focuses on establishing a **peer-to-peer network** involving three machines (**H1**, **H2**, **DS**) through **UDP** and Java programming. The primary objective is for **H1** to connect to the daytime service provider (**DS**) to retrieve data and time, and subsequently, **H2** retrieves this information from **H1**. The implementation employs a fast, connection-less mechanism for efficient data exchange, utilizing **UDP** with a dynamic port.

This comprehensive approach not only provides insights into network communication and protocol selection but also underscores the significance of time-related functionalities. Participants gain a practical understanding of **peer-to-peer communication** in diverse scenarios, emphasizing speed, protocol implementation, and the critical role of synchronized time management in networked systems.

# Problem Statement

**Explanation of problem with identification of element/object to be entered through console by the user and the result to be reflected in the form of file content/database/ in the console.**

In this problem first of all we have to make a **DS** Server using **UDP** protocols i.e. using **DatagramSocket** and **DatagramPacket**. **DatagramPacket** does two things.

**First** is make the message ready to send the particular clientAddress containing port number to imply to which application of the client.

**Second** is create an empty message area or say array in order to receive the data in the form of bytes from the clients or hosts.

Also it accepts the messages in the form of bytes so we have to convert the stringtypes or any other types to byte convertible form.

**DatagramSocket** is mainly used to establish a connection between two or more peers, it contains two metho**DS** i.e. **receive()** & **send()** which is used to receive and send the messages( already made ready by Datagram Packet) between peers.

So, here in **DS** Server (1ST Machine) it firsts create a connection and make the Server to receive the message (made to get ready to accept) by **DatagramPacket** ,and after receiving the required **DATE & TIME** from your LocalSystem it than sen**DS** it to **H1** host using Socket’s **send()** method. Here no input to be entered in console, it will display whether the packet has been sent to **H1** or not as an acknowledgement.

Now **H1** host is created, which will do 3 things , first is to send request to **DS Server** in order to access it by using **send()** method of Socket. **Second** is to receive the **DATE & TIME** from the **DS** and display it in the Console using **receive()** method of Socket. **Third** is again this **DATE & TIME** will be sent to **H2** after printed in the console of **H1** by using **send()** method of Socket.

Now finally the last host **H2** is created which only does 2 things, first is to send request to **H1** in order to access **DATE & TIME** by using **send()** method of Socket. **Second** is to receive the **DATE & TIME** from the **H1** and display it in the Console using **receive()** method of Socket

**Highlighting the constraints**.

**1.Port Conflicts:**

The code assumes fixed port numbers for **DS(2445), H1(2446),** and **H2(2447).**

Ensure these port numbers are available and not in use by other processes to prevent **BindException**.

**2.Limited Error Handling:**

The code lacks extensive error-handling mechanisms. Real-world scenarios may require more robust error handling for network failures, packet loss, or unexpected data formats.

**3.Synchronization:**

The code uses separate machines for **DS, H1, and H2** but doesn't implement advanced thread safety mechanisms. In more complex applications, additional synchronization may be necessary for shared resources.

**5.Data Validation:**

The code assumes a specific format for requests and responses without thorough data validation. Production code should include proper validation and error-checking for incoming data

# Methodology

**I. Algorithm/Pseudocode**

DS Machine Algorithm:

1.Initialize **UDP socket(Datagram)** to listen for incoming requests & then run

the server

2. Listen for incoming requests from H1 Machine

3. On request, fetch **date and time** from LocalHost

4. Send the **date and time** back to H1

5. Always keep the Server Machine on i.e. in Infinite

H1 Algorithm:

1. Initialize UDP socket for communication with DS

2. Send a request to DS to fetch **date and time**

3. Receive **date and time** from DS

4. Receive the request from H2 machine(host/client)

5. Send the same date and time received from DS Server to H2

4. Close the **UDP socket.**

H2 Algorithm:

1. Initialize UDP socket for communication with H1

2. Send a request to H1 to fetch data and time

3. Receive data and time from H1

4. Close the UDP socket

# Implementation

I. Program

**DS CODE:(System 1):-**

****

Here,DS server has been created and now it is being run in the console which stats

Server is activated and now H1 can send requests:-

**H1 HOST: (System 2)**

****

Now H1 machine code has been run and it has successfully asked request from DS

Server , and it has printed O/P on the console, so now H2 will send request to H1

**H2 Code(System 3):-**

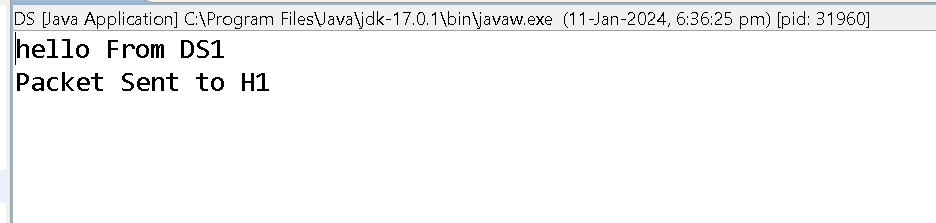


H2 has sent request and received the same data from H1

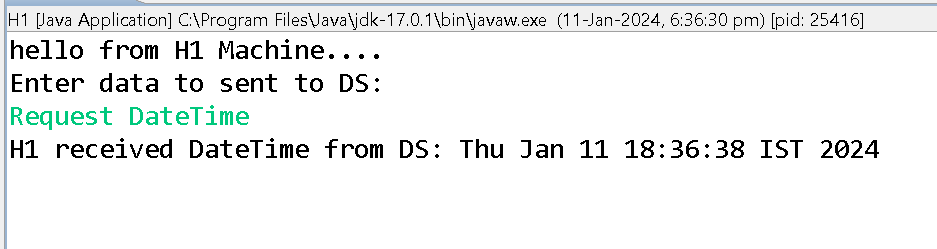
# Results & Interpretation

Output screenshots with respect to inputs justifying the outcomes alongwith small explanations.

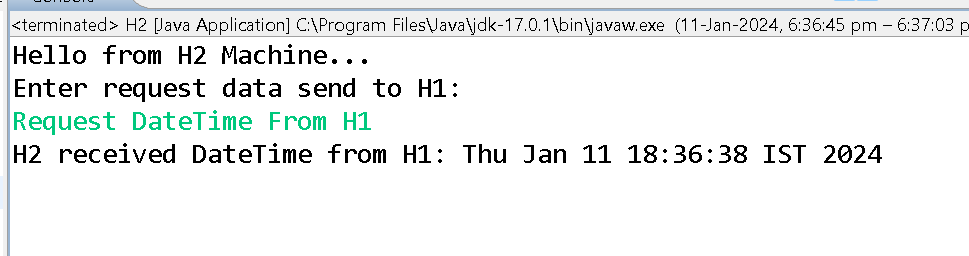
**DS CODE OUTPUT:-**

****

**H1 CODE OUTPUT:-**

****

**H2 CODE OUTPUT:-**

****

# Conclusion

# In summary, the Java code presented establishes a simple peer-to-peer network involving three machines using UDP – H1, H2, and a Daytime Service (DS) provider. This project demonstrates the basic principles of network communication, where H1 connects to DS to fetch date and time, and H2 subsequently retrieves this information from H1. Despite its educational value, the code exhibits certain constraints, such as limited error handling, assumptions about localhost communication, and thread safety concerns.

# For practical deployment, enhancements are necessary, including robust error handling, addressing port conflicts, scalability considerations, and improved data validation. Further, the code's modularity could be refined for increased flexibility and maintainability. This project provides a solid foundation for understanding UDP-based peer-to-peer networking but should be extended and refined to meet the standarDS of real-world applications.

**References**

(as per the IEEE recommendations)

**[1] Computer Networks, Andrew S. Tannenbaum, Pearson India.**

[2] Java Network Programming by Harold, O’Reilly (Shroff Publishers).

**[3]**[www.computer.org](http://www.computer.org) **(IEEE website ) & Oracle**