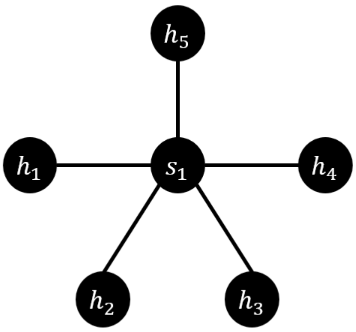
Lab 3: Socket Programming for Chatting Room with Multiple Users

This lab aims to help you get familiar with **socket programming**, which is a commonly used tool for messaging and file transmission in internet. In this lab, we will build **a chatting room** supporting multiple users. You need to program in python. You can refer to https://docs.python.org/3/library/socket.html for more useful information.

1. Chatting room with one server: client-server model with TCP protocol (50 points)
   Requirements:
   a) Use Mininet to build the following topology, which contains 5 hosts.



   b) The workflow of this chatting room is as follows: Each host in $\{h_i\}_{i\in\{1,2,3,4\}}$ inputs a message into its terminal, and also specifies which host $h_{j:j\in\{1,2,3,4\},j\neq i}$ to receive the message. The message from h_i will be sent to h_5, and then redirected to h_j. For example, in h1's terminal, we input "To h2: Hello!". Then h2 will receive the message from h1, and display it on its terminal "Hello! From h1". (In Mininet, the command to open the terminal of host h1 is 'xterm h1'.
   c) You are supposed to build the chatting room using socket programming with TCP protocol. You need to write two python files, namely, **server.py** (running on h_5) and **user.py** (running on $\{h_i\}_{i\in\{1,2,3,4\}}$). You are supposed to run the files on terminals of hosts as

   python server.py   or   python user.py

2. Chatting room with multiple users: client-only model with UDP protocol (50 points)
   Requirements:
   a) Use Mininet to build the same topology as in Problem 1.
   b) The new workflow of this chatting room is as follows: Each client can send the message to the broadcast address. At the same time, they also continuously listen to the messages from the broadcast address. In such way, each client can transmit/receive messages

to/from other clients. All the messages from clients should be displayed on each client's terminal.

c) You are supposed to build the above mentioned chatting room using socket programming with UDP protocol. You need to write one python file, **udpclient.py**. You are supposed to run the file on terminals of hosts as

python udpclient.py

d) For each client, it will receive the messages sent by itself from the broadcast address. Can you filter out these messages, such that each client's terminal will not display the message sent by the client itself?

Your final submission .zip file should include a pdf report, topology.py, server.py, user.py, and udpclient.py. Please put the screenshots of hosts' terminals in your pdf report.