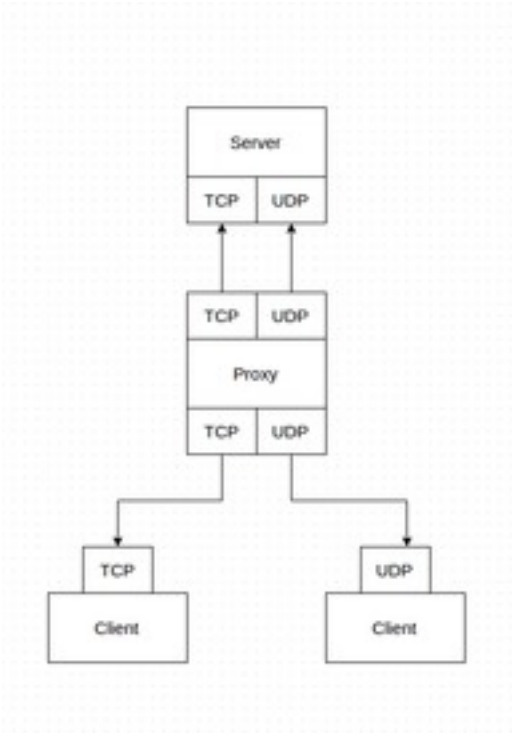
Foundations of Computer Networks
Siddharth Rajendra Prasad Kalluru



The main server has a class which runs an instance of the main server, it has two thread classes which can be instantiated from the main server, TCP server thread and UDP server thread. This allows the access of both clients with TCP protocol and clients with UDP protocol. Both the server threads can be instantiated from the main server thread. The server holds the UTC time which can be accessed from the clients, via a get command or set from the client side, if the credentials provided by the clients are authenticated then it can be set via the clients.
A client can run TCP protocol or a UDP protocol, as shown in the image. To connect to the main server, the client request has to go through a proxy server, as direct access to the main server is not allowed. A TCP client connects to the proxy server through TCP protocol, which sends a request to the TCP thread of the main server, vice-a-versa if a UDP client connects to the proxy server through UDP protocol then the proxy server connects to the UDP thread of the main server.
The proxy server runs an instance of the main class, and two threads running in parallel, which take connection request from a TCP client and connects to the TCP thread of the server. Another thread takes in a request from UDP client and tries to connect it to the UDP thread of the main server.
The client can run only one of the protocols, either TCP or UDP. If a client is running a TCP protocol, it sends a TCP request to the proxy server, which validates the types of request and sends it to the main server, and gets a response or an error from the server, depending on the request from the client and then forwards the same to the client. Same goes for the UDP client, which sends a UDP request to the proxy server, having a request for the main server, which is forwarded to the main server via the proxy server. This request returns a response from the main server to the proxy server which then forwards it to the UDP client.
If the proxy server already holds a connection to the main server from a client, say a TCP connection from a client to the main server is already established via proxy, and meanwhile the proxy server gets a second request from another client for a TCP connection again, then the proxy server refuses the second client for a connection, giving an error that a connection on the TCP thread has already been

established, so it will have to wait until the first connection is dropped.
The message format is as such:

The message format is given as such,

- Type of message: it holds the type of message from the client. Either it wants the value of the time at the server, which would be described by 'g' meaning the client wants to get the value of the server time. The client can also set the value of the server time, which would mean it would need to give the right credentials to authenticate and then set the value. To set the server time, values of UTC time which the client needs to set, the user-name and the password are mandatory.
- UTC Time: When the client does a get, then this value needs to be empty, however when a set time is done then this value is mandatory, otherwise the server would return an error
- Username: When a client wants to get the value of the server time, then this value needs to be empty, however when a client wants to set the value of the server time, then this value becomes mandatory for authentication
- Password: Same goes for password as well, as it is important for authentication. When a client does a get, then this value needs to be empty as no authentication is required and the request is just read only. But when client wants to set the value of the server time, then the client is required to sent the values of the user-name and password to authenticate its set request.
- Message from the server: The message from the server is returned in both the request cases, get or set. If the client doe a get server time, then the server returns a response or an error to the proxy server which is then forwarded to the client which made the request from the proxy. If the get request returns a successful value of the server time, it then returns the response to the client via proxy. But if the get returns an 'e',i.e. error then this error is also forward to the client via proxy.

When the client does a set, if the new UTC time is set via the client, then the server returns a successful response to the client via proxy which holds the new time set and also the old time which was overwritten by the client via the proxy to the client which had made the set request. If the set request is not completed, such as there was an authentication error, or another client was currently setting the value in that case the server returns an error to the client currently wanting to set a new UTC time via the proxy.

The length of the get request:

- type of message: it is a character 'g' or 's' and it takes 2 bytes of message space. In case of get time, it is 'g' and all other values are set as empty or '#'
- UTC time: It is a string which is sent to the server, in case of get requests. It is specified in String format and the length if “String-length*2” bytes
- User-name: User-name is send in string format to the server, whenever a set request is made. The length of the user-name is as “String-length*2” bytes
- Password: Password is send in string format to the server, whenever a set request is made. The length of the password is as “String-length*2” bytes
- Message from the sever: This part of the message is returned from the server to the client,

depending on the request from the client and how its processed at the server side. This is a string format message and in case of a get request from the client it is the current UTC time at the server, and when a server gets a “set” request from the client and the time is set successfully at the server end, then this is returned as the new time set and the old time which is overwritten by the client. However, if the get or set request returns an error then this part of the message passes the error to the client, which is also in string format. The length of Message from the server is “String-length*2” bytes.

Total length of the message: 2 bytes + 2*(length of UTC time in string) + 2*(length of user-name in string) + 2*(length of password in strings) + 2*(length of message from server in string)