Time Keeping And Retrieval Protocol

Data Communication and Networks

Protocol Specification

March 2017

by
Zachary Migliorini
Rochester Institute of Technology
zmm2962@rit.edu

Table of Contents

Description and Options

The class tsapp is designed to handle both client and server functionality, using either UDP or TCP for transport. The server will always be ready to accept either a UDP or a TCP connection, giving the client the option of choosing the transport protocol. The server can hold a time value, and an authorized username and password. The client can request the time, or attempt to set the time. Setting the time requires the client to provide the username and password which the server has on file. Failure to do so results in the time being unchanged. Output is only done through the client application, and is simply the time currently on the server, as of the last client message being sent and processed, and the round-trip-time in milliseconds.

Shared Options

The command-line arguments passable to tsapp running as either the client or server are variable in number, and are as follows:

“-c”	Run this instance of tsapp as a client
“-s”	Run this instance of tsapp as a server
“-T <time>”	Set the time to <time>
“--user <username>”	Set the username to <username>
“--pass <password>”	Set the password to <password>
“port”	The client uses this as the UDP/TCP port to connect to, server uses it as the UDP destination port

Client Options

The client can take a variable number of command-line arguments, specific to client runs, depending on desired operation. They are as follows:

“-u”	Run using UDP
“-t”	Run using TCP
“-z”	Report the time in UTC formatted
“-n <num>”	Send the request to the server and print a response <num> times
“server address”	The address of the server to connect to, e.g. “localhost”

Server Options

The server can take a variable number of command-line arguments, depending on desired operation. They are as follows:

“port2”	The port to be listening on for a TCP connection.

Client Messages

The client application can send two types of messages. The first is a message requesting the time, which is simply “GET”. The second is a set request, formatted as follows: “SET <time> <username> <password>”. <time> is the desired time to set the server to, formatted as a UNIX timestamp. <username> is the username, and <password> is the password.

Server Messages

The server application can respond with three types of messages, one for a valid message from the client, and two for error reporting. On a valid GET message, the server will simply respond with the time. On a valid SET message, the server will set the time, and respond with the new set time. The format for both of these is simply “<time>”, where time is a UNIX timestamp representing the server time. Upon the client providing incorrect credentials, the server will respond with “Error: invalid credentials.” Upon the client providing an invalid message, i.e. neither a GET or a SET with the proper format, the server will respond with “Error: bad message.”

Output

The output is two pieces of information. The first is the current time on the server, as of the last client message being sent and processed by the server, and the second is the round-trip-time in milliseconds. They are formatted as follows:

If the “-z” argument has been passed to the client, requesting UTC time:
“Server time:“ <UTC time>
<RTT>

If the “-z” argument has not been passed:
“Server time:” <UNIX time>
<RTT>

If an error has been found by the server:
<Error message>
<RTT>