# Protocol – UDP Client/Server Time Application

## 1) Summary

The application implements a UDP server (port 27015) and a UDP client. The client sends different time-related requests; the server processes them according to a request code and returns a textual response.

## 2) Transport Layer

• Transport protocol: UDP  
• Server port: 27015  
• Reliability: No guarantee of delivery/order; the client/server must handle missing packets (timeouts, etc.).

## 3) Message Structure

Request (Client → Server):  
Byte 0 : RequestCode (1..13)  
Byte 1 : PayloadLength  
Byte 2.. : Payload (if any)  
  
Response (Server → Client):  
Text string (e.g. "2025-08-25 13:47:05").

## 4) Request Codes

|  |  |  |
| --- | --- | --- |
| Code | Name | Description |
| 1 | GetTime | Full date and time: YYYY-MM-DD HH:MM:SS |
| 2 | GetTimeWithoutDate | Time only: HH:MM:SS |
| 3 | GetTimeSinceEpoch | Seconds since 1.1.1970 |
| 4 | GetClientToServerDelayEstimation | Server's GetTickCount() value |
| 5 | MeasureRTT | Server's GetTickCount() value for RTT measurement |
| 6 | GetTimeWithoutDateOrSeconds | Time without seconds: HH:MM |
| 7 | GetYear | Year only: YYYY |
| 8 | GetMonthAndDay | Month and day: MM DD |
| 9 | GetSecondsSinceBeginingOfMonth | Seconds since beginning of month |
| 10 | GetWeekOfYear | Week number of the year |
| 11 | GetDaylightSavings | 1 if DST is active, 0 otherwise |
| 12 | GetTimeWithoutDateInCity | Time in supported cities or UTC if unsupported |
| 13 | MeasureTimeLap | Measure time between two consecutive requests |

## 5) Wireshark Screenshots

Attach here screenshots of the captured messages in Wireshark and explain how they match the message structure defined above.

## 6) Theoretical Questions

A. Could the client or the server get 'stuck'?  
Yes, since UDP does not guarantee delivery. If there is no timeout defined, the client may wait forever for a reply. Suggested fix: use receive timeout/retries.

For example – if after 5 seconds the client is not receiving any response, he will send the request again.  
  
B. What does the average value in the ClientToServerDelayEstimation represent?  
It represents **the average difference between consecutive GetTickCount values returned by the server**, i.e., the spacing between client requests as observed by the server. This reflects the Client→Server **delay**, not the RTT.