# Lab 7: Understanding the Network Time Protocol

**Introduction**

This lab focuses on the giving overview of network time protocol. An implementation will be required to demonstrate network time protocol.

**Objectives**

To understand the concept of network time protocol.

**Tools/Software Requirement**

MS Visual Studio 2013

**Description**

SNTP, as its name implies, is a protocol for transferring date and time information. The main purpose is for time synchronization. For example, Windows uses this (occasionally!) to keep your computer's clock updated, but it could be used on a LAN with one machine acting as a server, to make sure all client machines are perfectly 'synced' with the server, and therefore each other, in time critical applications. It can also be used for validating times as I did in my aforementioned article. In fact, any application that uses dates and/or times could find a use for SNTP. It uses [UTC](http://en.wikipedia.org/wiki/Coordinated_Universal_Time) (Coordinated Universal Time) for all its data, and .NET conveniently provides methods to easily convert between UTC and local time. It uses [UDP](http://en.wikipedia.org/wiki/User_Datagram_Protocol) on port 123, but there are some (nonstandard servers) that operate using TCP/HTTP on different ports. As they are non standard, they are ignored here.

SNTP is a simple system (in the normal unicast mode) that consists of one packet of bytes being sent by a client, and one packet then being received. Each packet consists of 48 bytes (68 if Key Identifier and Message Digest are used). The list below explains each byte's meaning.

* Byte 0: This contains three values.
  1. The **Leap Indicator** which is contained in bits 7 and 6. This indicates whether there is to be a [leap second](http://en.wikipedia.org/wiki/Leap_second) added or removed.
  2. The protocol **Version Number** to use in bits 5, 4, and 3. Version 3 and version 4 are in common usage although NTP version 4 has yet to get an RFC. Previous versions are now commonly considered obsolete.
  3. The **Mode** in the remaining bits 2, 1, and 0. In unicast mode (which is all I'm considering in this article), we set this to 3 to indicate that we are a client, and check that it is 4 on receipt to make sure the data has come from a server.
* Byte 1:

The **Stratum**, or how far we are away from the primary reference source.

The value 0 is unspecified (this is the actual clock source). 16 to 255 are reserved for future use. A stratum of 1 is considered a primary source such as an atomic clock, GPS, radio etc. If a server synchronizes itself with a stratum 1 server, it is a stratum 2 as it's one step more away. This carries on all the way up to 15.

* Byte 2:

**Poll Interval**, the time in seconds between the server re-syncing with its source.

To stop servers being overrun with constant requests, polling is recommended to be carried out infrequently. We need to remember this for our own clients and should probably not check the same server more than every 64 seconds (the recommended 'default' value). The actual time is calculated by 2 ^ value.

* Byte 3:

**Precision**, the precision of the server's clock. This is calculated by 2 ^ value.

* Bytes 4 - 7:

**Root Delay**, the round trip delay to the primary reference source (in stratum 1, if it's not already a stratum 1 server) from the server, and back again. This is a 32 bit fixed point value, 16 for the integer part and 16 for the fractional part giving fine precision.

* Bytes 8 - 11:

**Root Dispersion**, the nominal error relative to the primary reference source. 32 bits as in Root Delay above.

* Bytes 12 - 15:

**Reference Identifier**, this identifies the reference in a variety of ways depending on the version being used and the stratum. If it's a stratum 1 source, this is 4 characters identifying the type of clock. If it's a stratum 2 to 15 (secondary), then:

* 1. If version 3: Each byte represents an octet of the IP address of the server's reference source.
  2. If version 4: This should be the integer 32 bits of the latest transmit timestamp of the reference source, although in all my tests, the IP address was here as in version 3!
* Bytes 16 - 23:

**Reference Timestamp**, the time at which the server's clock was last corrected. This is a 64 bit fixed point value, 32 for the integer part and 32 for the fractional part giving extremely fine precision. In fact, this precision is far greater than can be handled in .NET which can be accurate to 10 nanoseconds at best.

* Bytes 24 - 31:

**Originate Timestamp**, the time at which the request departed the client for the server. We don't set this in the client. Instead, we use the transmit timestamp, and the server copies this into the originate timestamp in its reply. 64 bits as in reference timestamp.

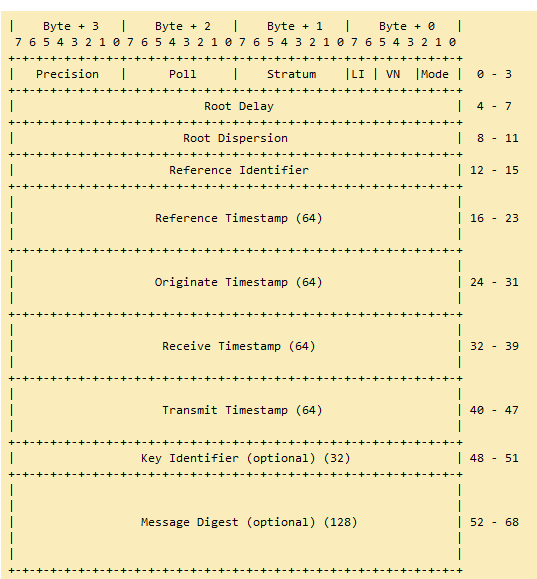
* Bytes 32 - 39:

**Receive Timestamp**, the time at which the request arrived at the server. 64 bits as in reference timestamp.

* Bytes 40 - 47:

**Transmit Timestamp**, the time at which the reply departed the server for the client, or the request departed the client for the server. 64 bits as in reference timestamp.

Here is the code structure

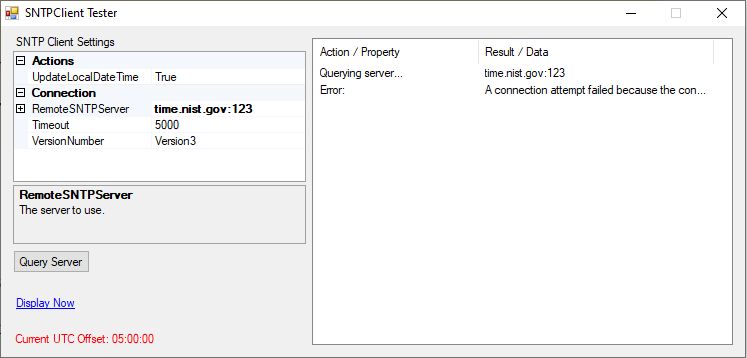


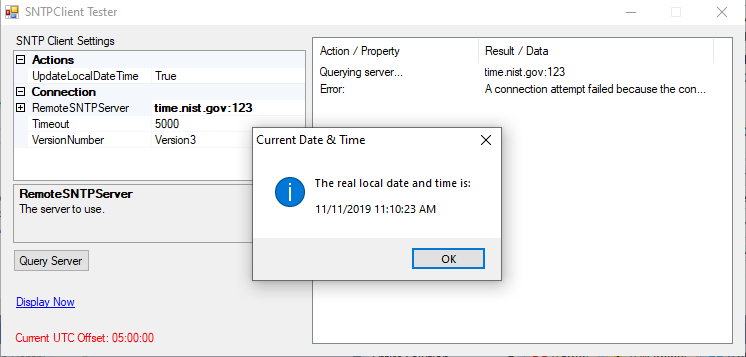
**Lab Tasks**

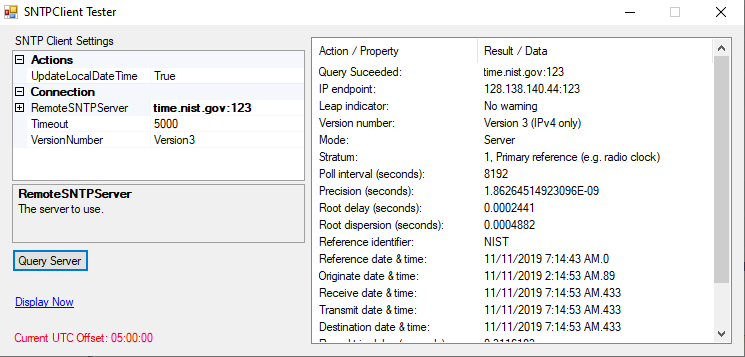
1. Run the code (uploaded on LMS) to get the time from the time server, observe the changes in different header fields on receipt of message.
2. Edit the code to get the time from ten servers and display, also calculate the time difference with multiple time servers.

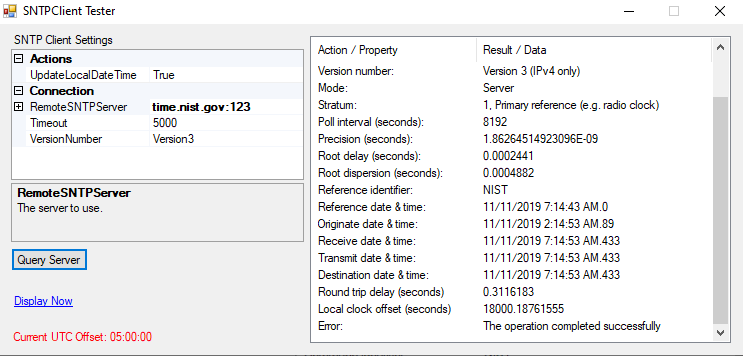
**Deliverables**

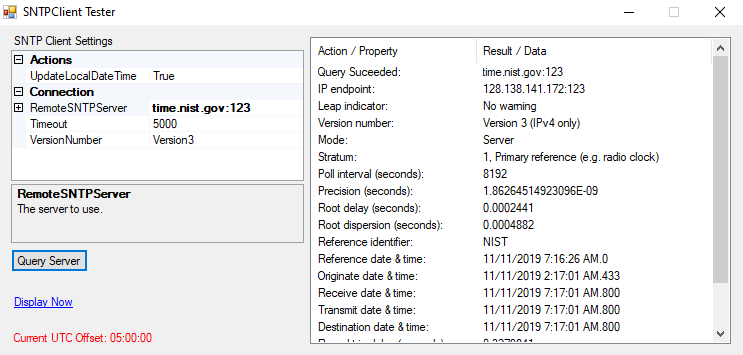
Submit the project files and screenshots of your output on LMS.

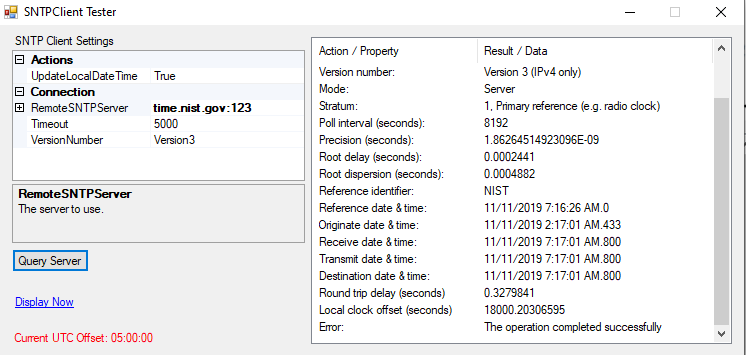


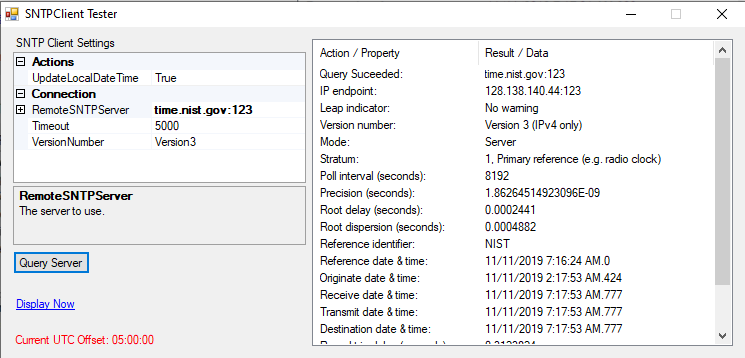










’

