| | |
|---|---|
| **SE424: Distributed Systems** | **Recitation 10** |
| **Semester 1 5785** | **12 Jan 2025** |
| **Lecturer: Michael J. May** | **Kinneret College** |

# Network Time Protocol

Today we will do some work with the Network Time Protocol (NTP), including writing an NTP client and NTP server.

## 1 NTP Trace

We'll start by examining a packet trace from a real NTP conversation from a Windows 7 computer. The NTP request is sent from the computer in the following packet:

```
▾ Internet Protocol Version 4, Src: 10.9.27.20, Dst: 20.101.57.9
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 76
    Identification: 0x92e2 (37602)
  ▸ 000. .... = Flags: 0x0
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 128
    Protocol: UDP (17)
    Header Checksum: 0x3534 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 10.9.27.20
    Destination Address: 20.101.57.9
▾ User Datagram Protocol, Src Port: 123, Dst Port: 123
    Source Port: 123
    Destination Port: 123
    Length: 56
    Checksum: 0x412f [unverified]
    [Checksum Status: Unverified]
    [Stream index: 389]
  ▸ [Timestamps]
    UDP payload (48 bytes)
▾ Network Time Protocol (NTP Version 3, client)
  ▸ Flags: 0xdb, Leap Indicator: unknown (clock unsynchronized), Version number: NTP Version 3, Mode: client
    [Response In: 7366]
    Peer Clock Stratum: unspecified or invalid (0)
    Peer Polling Interval: 17 (131072 seconds)
    Peer Clock Precision: -23 (0.000000119 seconds)
    Root Delay: 0.000000 seconds
    Root Dispersion: 1.000000 seconds
    Reference ID: NULL
    Reference Timestamp: Feb 18, 2024 09:06:45.345629399 UTC
    Origin Timestamp: NULL
    Receive Timestamp: NULL
    Transmit Timestamp: Feb 19, 2024 07:14:48.063632099 UTC
```

Note that the request includes the transmit timestamp (Dec 27, 2014 21:43:07.114120000 UTC) in the request. The first timestamp is the timestamp when the computer last did a synchronization (Dec 23, 2014 20:52:39.450120000 UTC). The two middle fields will be filled in by the server later (they are zero as sent by the client).

The server (`time.nist.gov`) responds with the following message:

 

```
Internet Protocol Version 4, Src: 20.101.57.9, Dst: 10.9.27.20
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 76
    Identification: 0x3864 (14436)
  > 000. .... = Flags: 0x0
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 109
    Protocol: UDP (17)
    Header Checksum: 0xa2b2 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 20.101.57.9
    Destination Address: 10.9.27.20
User Datagram Protocol, Src Port: 123, Dst Port: 123
    Source Port: 123
    Destination Port: 123
    Length: 56
    Checksum: 0x018e [unverified]
    [Checksum Status: Unverified]
    [Stream index: 389]
  > [Timestamps]
    UDP payload (48 bytes)
Network Time Protocol (NTP Version 3, server)
  > Flags: 0x1c, Leap Indicator: no warning, Version number: NTP Version 3, Mode: server
    [Request In: 7353]
    [Delta Time: 0.073576000 seconds]
    Peer Clock Stratum: secondary reference (3)
    Peer Polling Interval: 17 (131072 seconds)
    Peer Clock Precision: -23 (0.000000119 seconds)
    Root Delay: 0.001678 seconds
    Root Dispersion: 0.031769 seconds
    Reference ID: 25.66.230.3
    Reference Timestamp: Feb 19, 2024 07:01:40.058507399 UTC
    Origin Timestamp: Feb 19, 2024 07:14:48.063632099 UTC
    Receive Timestamp: Feb 19, 2024 07:14:47.901504699 UTC
    Transmit Timestamp: Feb 19, 2024 07:14:47.901508299 UTC
```

Note that the server sends back the time that the client sent its message (Dec 27, 2014 21:43:07.114120000 UTC), what time it received the message (Dec 27, 2014 21:43:10.096164000 UTC), and when it responded (Dec 27, 2014 21:43:10.096174000 UTC). These correspond to the $T_1, T_2, T_3$ values mentioned in class. The time that the message is actually received by the client isn't shown (that's $T_4$).

## 2 NTP Client in Java

The classic NTP protocol version 3 is the standard out there today (RFC 1305 - `https://tools.ietf.org/html/rfc1305`). There is also a version 4 (RFC 5905 - `https://tools.ietf.org/html/rfc5905`) out there where is more modern and has some security features not found in NTPv3.

## 3 Downloading the Classes

We're going to use the Apache Commons `NTPUDPClient` class for the tool. The class can be downloaded from `https://commons.apache.org/proper/commons-net/download_net.cgi`. You'll probably want to download the `commons-net-3.11.1-bin.zip` file (it has the sources inside the ZIP archive).

**Using Maven** You can add the dependency to your project using Maven by adding the following dependency: commons-net:commons-net:3.11.1

**Using Gradle** You can add the dependency to your build.gradle file by adding: implementation 'commons-net:commons-net:3.11.1'

**Without Maven, manually** If you can't use Maven, you'll need to do the following steps:

1. Adding Dependency: Once you downloaded the library, unzip it in the project directory and add it in as an external JAR to the project (Project → Properties → Java Build Path → Libraries → Add External JARs...)
2. Link Sources: To link the sources JAR to the binaries JAR: Right Click on the `commons-net-3.11.1.jar` under Referenced Libraries → Properties → Java Source Attachment → Browse (if the JAR is in the workspace) or External location → External File... (if it's not in the workspace).

### 3.1 What to do

Use the `NTPUDPClient` class to query an NTP server for the time. You can use one of the following servers (they should work without a problem):

- `timeserver.iix.net.il`
- `ntp.netvision.net.il`
- `time-a-g.nist.gov`

**What to output** Your goal is to print out the following data fields:

1. $t_1$, $t_2$, $t_3$, $t_4$ in a human readable format
2. The round trip time from your computer to the NTP server
3. The clock stratum of the server
4. The offset from the server's time.

All of the above fields can be found using the `NTPUDPClient` and `TimeInfo` classes.

## 4 Important Methods and Classes

You'll need to use the following methods and classes to make things work:

`InetAddress` To convert the DNS name to an IP and address and store the IP address returned.

`NTPUDPClient.getTime(InetAddress)` Does the work of retrieving the NTP time request.

`TimeInfo` A class returned by the `NTPUDPClient` class which contains the response from the NTP server.

`NtpV3Packet` A class which represents the contents of the message returned inside the `TimeInfo` object.

`TimeStamp` A class which represents the time stamp format in NTPv3 (it's a rather complicated integer format).

`NtpV3Packet.getOriginateTimeStamp()` Figure it out!

`NtpV3Packet.getReceiveTimeStamp()` Figure it out!

`NtpV3Packet.getTransmitTimeStamp()` Figure it out!

`TimeInfo.getReturnTime()` Figure it out!

`TimeStamp.getNtpTime(long)` Figure it out!

`TimeStamp.toDateString()` Converts from the NTPv3 time stamp format to a human readable string

`TimeInfo.computeDetails()` Look it up!

`TimeInfo.getOffset()` Look it up!

`TimeInfo.getDelay()` Look it up!

# 5 NTP Server in Java

Once you have your NTP client working with the Apache Commons classes, we'll take a look at a simple NTP server (also writable with the Apache Commons library). Since NTP has some niggly complexities which you need to write correctly to make the packets work and understandable, we'll use a prepared simple NTP server from Apache Commons (`SimpleNTPServer.java`) which can be downloaded from `https://commons.apache.org/proper/commons-net/examples/ntp/SimpleNTPServer.java` or from Moodle.

## 5.1 What to do

Once you have your simple NTP server up and running, perform the following experiments:

1. Start up the NTP server on port 123 or a custom port (*e.g.* 6644, 5000, etc) if you can't listen on 123 for some reason.
2. Point your NTP client from the previous section at it. You may need to modify the client to use `NTPUDPClient.getTime(InetAddress, int)` if you haven't done so already.
3. Compare the output from the simple NTP server and the official ones. Which clock seems to be more accurate? Why?

 