
SNTPclient.java

By rac

Published: 15.01.2008 - 09:46

1.
package hsz.ITDP.rac.uebungen.SNTPclientServer;

2.

3.
import java.io.IOException;

4.
import java.net.DatagramPacket;

5.
import java.net.DatagramSocket;

6.
import java.net.InetAddress;

7.
import java.text.DecimalFormat;

8.
import java.util.*;

9.

```
10. public class SntpClient {
11.
12.     /**
13.      * @param args
14.      */
15.     public static void main( String [] args) {
16.
17.         String sntpHost = "lisa.hsz-t.ch";
18.
19.         int sntpPort = 123;
20.
21.         try {
22.
23.             byte[] data = generatePacket();
24.
25.             InetAddress addr = InetAddress .getByName( sntpHost );
```

```
sntpPort );  
  
23.     DatagramPacket send = new DatagramPacket ( data, data.length, addr,  
  
24.     DatagramSocket ds = new DatagramSocket ();  
  
25.     System .out.println("NTP request sent, waiting for response...n");  
  
26.     DatagramPacket recv = new DatagramPacket (data, data.length);  
  
27.     ds.receive(recv);  
  
28.     // Immediately record the incoming timestamp  
  
29.     //double destinationTimestamp = (System.currentTimeMillis()/1000.0) +  
2208988800.0;  
  
30.     byte[] recvdata = recv.getData();  
  
31.  
  
32.     long TransmitTimestampInt = 0;  
  
33.     long TransmitTimestampFrac = 0;
```

34.

35.

```
for(int i=0; i<4; i++)
```

36.

```
{
```

37.

```
short b = recvdata[40+i];
```

38.

```
if((b & 0x80)==0x80) b = (short) (128 + (b & 0x7f));
```

39.

```
else b = (short) b;
```

40.

```
TransmitTimestampInt += b * Math .pow(2, (3-i)*8);
```

41.

```
}
```

42.

```
for(int i=4; i<8; i++)
```

43.

```
{
```

44.

```
short b = recvdata[40+i];
```

45.

```
if((b & 0x80)==0x80) b = (short) (128 + (b & 0x7f));
```

46.

```
        else b = (short) b;

47.        TransmitTimestampFrac += b * Math .pow(2, (3-i)*8);

48.    }

49.    Date TS = getTimeDate(TransmitTimestampInt,
TransmitTimestampFrac);

50.    System .out.println("TransmitTimestamp: "+TS);

51.    ds.close();

52. }

53. catch ( IOException ioe) {

54.    System .out.println( ioe );

55. }

56. }

57.

58.
```

```
static private byte[] generatePacket()
```

```
59.
```

```
{
```

```
60.
```

```
//Define packet fields
```

```
61.
```

```
byte LI = 0;
```

```
62.
```

```
byte VN = 4;
```

```
63.
```

```
byte Mode = 3;
```

```
64.
```

```
short Stratum = 0;
```

```
65.
```

```
byte Poll = 0;
```

```
66.
```

```
byte Precision = 0;
```

```
67.
```

```
double RootDelay = 0;
```

```
68.
```

```
double RootDispersion = 0;
```

```
69.
```

```
byte[] ReferenceIdentifier = {0, 0, 0, 0};
```

```
70.
```

```
double ReferenceTimestamp = 0;
```

71. double OriginateTimestamp = 0;

72. double ReceiveTimestamp = 0;

73. double TransmitTimestamp = ([System](#) .currentTimeMillis()/1000.0) + 2208988800.0;

74. byte[] p = new byte[48];

75. p[0] = (byte) (LI << 6 | VN << 3 | Mode);

76. p[1] = (byte) Stratum;

77. p[2] = (byte) Poll;

78. p[3] = (byte) Precision;

79. int l = (int) (RootDelay * 65536.0);

80. p[4] = (byte) ((l >> 24) & 0xFF);

81. p[5] = (byte) ((l >> 16) & 0xFF);

82. p[6] = (byte) ((l >> 8) & 0xFF);

83.

```
p[7] = (byte) (l & 0xFF);

84.    long ul = (long) (RootDispersion * 65536.0);

85.    p[8] = (byte) ((ul >> 24) & 0xFF);

86.    p[9] = (byte) ((ul >> 16) & 0xFF);

87.    p[10] = (byte) ((ul >> 8) & 0xFF);

88.    p[11] = (byte) (ul & 0xFF);

89.    p[12] = ReferencelIdentifier[0];

90.    p[13] = ReferencelIdentifier[1];

91.    p[14] = ReferencelIdentifier[2];

92.    p[15] = ReferencelIdentifier[3];

93.    for(int i=0; i<8; i++)

94.    {

95.        double base = Math.pow(2, (3-i)*8);
```

```
96.         p[16+i] = (byte) (ReferenceTimestamp / base);

97.         short b = p[16+i];

98.         if((b & 0x80)==0x80) b = (short) (128 + (b & 0x7f));

99.         else b = (short) b;

100.        ReferenceTimestamp = ReferenceTimestamp - (double) (b * base);

101.

102.    }

103.    p[7] = (byte) ( Math .random()*255.0);

104.    for(int i=0; i<8; i++)

105.    {

106.        double base = Math .pow(2, (3-i)*8);

107.        p[24+i] = (byte) (OriginTimestamp / base);

108.
```

```
        short b = p[24+i];

109.        if((b & 0x80)==0x80) b = (short) (128 + (b & 0x7f));

110.        else b = (short) b;

111.        OriginateTimestamp = OriginateTimestamp - (double) (b * base);

112.

113.    }

114.    p[7] = (byte) ( Math .random()*255.0);

115.    for(int i=0; i<8; i++)

116.    {

117.        double base = Math .pow(2, (3-i)*8);

118.        p[32+i] = (byte) (ReceiveTimestamp / base);

119.        short b = p[32+i];

120.        if((b & 0x80)==0x80) b = (short) (128 + (b & 0x7f));
```

```
121.         else b = (short) b;

122.         ReceiveTimestamp = ReceiveTimestamp - (double) (b * base);

123.

124.     }

125.     p[7] = (byte) ( Math .random()*255.0);

126.     for(int i=0; i<8; i++)

127.     {

128.         double base = Math .pow(2, (3-i)*8);

129.         p[40+i] = (byte) (TransmitTimestamp / base);

130.         short b = p[40+i];

131.         if((b & 0x80)==0x80) b = (short) (128 + (b & 0x7f));

132.         else b = (short) b;

133.
```

```
TransmitTimestamp = TransmitTimestamp - (double) (b * base);
```

```
134.
```

```
135.
```

```
    }
```

```
136.
```

```
    p[7] = (byte) ( Math .random()*255.0);
```

```
137.
```

```
    return p;
```

```
138.
```

```
    }
```

```
139.
```

```
    static private Date getTimeDate(long integerPart, long fractionalPart)
```

```
140.
```

```
    {
```

```
141.
```

```
        TimeZone UTC = new SimpleTimeZone (0,"UTC");
```

```
142.
```

```
        Calendar c = new GregorianCalendar (1900, Calendar .JANUARY,1,0,0,0);
```

```
143.
```

```
        c.setTimeZone(UTC); //set time zone to 0-longitudinal (London Greenwich) because time from  
        SNTP server is GMT (=UTC when no summertime)
```

```
144.
```

```
        Date startOfCentury = c.getTime();
```

```
145.
```

```
integerPart = integerPart * 1000; //get milliseconds

146. System.out.println("integerPart = " + Long.toHexString(integerPart));

147. System.out.println("fractionalPart = " + Long.toHexString(fractionalPart));

148. float fPart = fractionalPart; //first assign long value to float

149. fPart = fPart / 4294967296L; //shift value left so only fractional part remains; division by
2^32

150. //now fPart is a number < 1

151. fPart = fPart * 1000; //get milliseconds (drop higher precision)

152. System.out.println("fPart = " + Float.toString(fPart) + "[ms]");

153. //now 0 <= fPart <= 999 (number of milliseconds)

154. long msSinceStartOfCentury = integerPart + (long)fPart; //add integer and fractional part

155. Date now = new Date (msSinceStartOfCentury+startOfCentury.getTime());

156. return now;

157.
```

```
}
```

```
158.
```

```
159.
```

```
}
```

[< DemoNtpMessage.java up](#)