ADAPTATION OF NETWORK TIME PROTOCOL ON APPLICATION LAYER AND HIGH-LATENCY NETWORKS

Petr Kovář
Doctoral Degree Programme (1), FEEC BUT
E-mail: xkovar25@stud.feec.vutbr.cz

Supervised by: Karol Molnár
E-mail: molnar@feec.vutbr.cz

ABSTRACT

The key problem of modern mobile packet – oriented networks is adaptation of QoS, Quality of service. For complex and reliable control of QoS, the appropriate measurements are needed. For exact and authentic measurement of absolute size of traffic delay, it is absolutely necessary to detect correct time offset between mobile station and corresponding server. According to the consequential research, the service should be functional with any Java – enabled mobile phone and in any situation when the signal quality for data transfers is sufficient. Only way how to achieve this goal is to adapt existing standards, or propose new one, which will be able to provide efficient time synchronization over high – latency data networks using application layer only.

1. INTRODUCTION

In low – latency data network, the Network Time Protocol (NTP) is commonly used. NTP is very complex and sophistical protocol and for its proper use, it is necessary to access lower network layers. Using JavaME, which is Java platform designed for use on resource – constrained devices, it is not possible to adapt whole NTP protocol implementation, because of its complexness and because the Java platform does not allow to access lower network layers. The solution is to adapt NTP principles to application layer and high – latency networks.

2. APPLICATION

In the first part of the paper, the NTP protocol specification will be analyzed, and Java and JavaME limitations will be presented. Than the solution and adaptation of NTP principles on Java platform will be proposed together with some test results.

2.1. NETWORK TIME PROTOCOL

Network Time Protocol specification was proposed in 1985 [1]. At the moment, we have the third version of NTP specification [2].

For exact definition of the clock offset between host and peer, the NTP protocol specifies the clock offset as following function:

$$\theta = \frac{(T_{i-2} - T_{i-3}) + (T_{i-1} - T_i)}{2}, \quad (1)$$

where T_i, T_{i-1}, T_{i-2} and T_{i-3} are timestamps described by figure 1.

Roundtrip delay is defined as

$$\delta = (T_i - T_{i-3}) - (T_{i-1} - T_{i-2}). \quad (2)$$

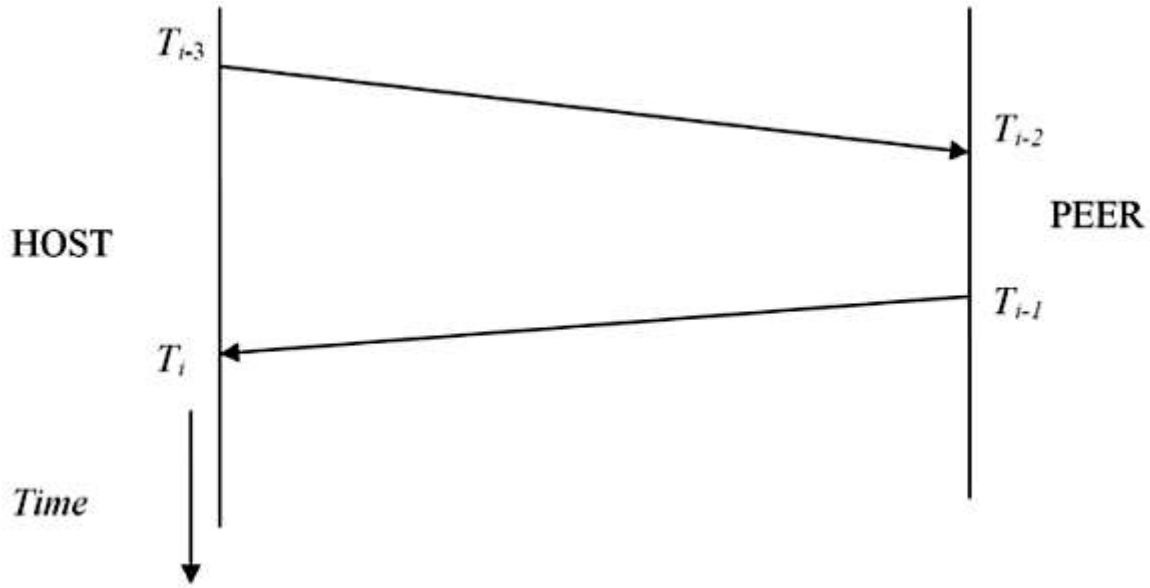

Figure 1: NTP timestamps.

It is obvious that T_{i-3} is system clock of host in which he sends request to peer. T_{i-2} is system clock of peer when the request arrives. T_{i-1} is system clock of peer when the answer is send and T_i is the time of its arrival at host system. NTP is implemented on network transport layer [3], [4] to recognize exact time when the packet was received or sent.

When working with Java platform, it is impossible to determine the exact moment of packet sending or receiving, because Java application cannot access the transport layer. Also using variables T_{i-2} and T_{i-1} has no purpose, because T_{i-2} and T_{i-1} variables have same value when programming on application layer. The variable T_{peer} is used instead. The following formula adapts the mentioned conditions for the clock offset:

$$\theta = \frac{(T_{peer} - T_{i-3}) + (T_{peer} - T_i)}{2} = T_{peer} - \frac{T_{i-3} + T_i}{2}. \quad (3)$$

The formula for the roundtrip delay is simplified to the form

$$\delta = (T_i - T_{i-3}) - (T_{peer} - T_{peer}) = T_i - T_{i-3}. \quad (4)$$

2.2. HIGH LATENCY NETWORKS - GPRS

Time synchronization in high latency networks with very high jitter, as is in GPRS, is very challenging task. The roundtrip delay for GPRS usually varies between 400 and 1200 ms. The delay in the host to peer direction can also vary from peer to host direction very heavily.

For the first analysis, both the server and the client side of the Java based test application using the modified NTP system was implemented on JavaSE platform. After successful tests and proper algorithm investigation, the JavaME version of the client will also be developed and tested.

The application designed for JavaSE platform consists of two parts, the server side and the client side. The UDP connection was used, because the TCP error correction methods are not acceptable here. The server side (Figure 2) can listen on user – specified UDP port and client sends UDP packet with own timestamp data.

Figure 2: Server-side part of application.

After receiving the time packet, the server adds its own timestamp to the payload and sends the packet back to the client. For compensation of the difference between the upload and the download data rates, and possibly delay variations of the GPRS system, another measurement is made instantly, as it is displayed in Figure 3.

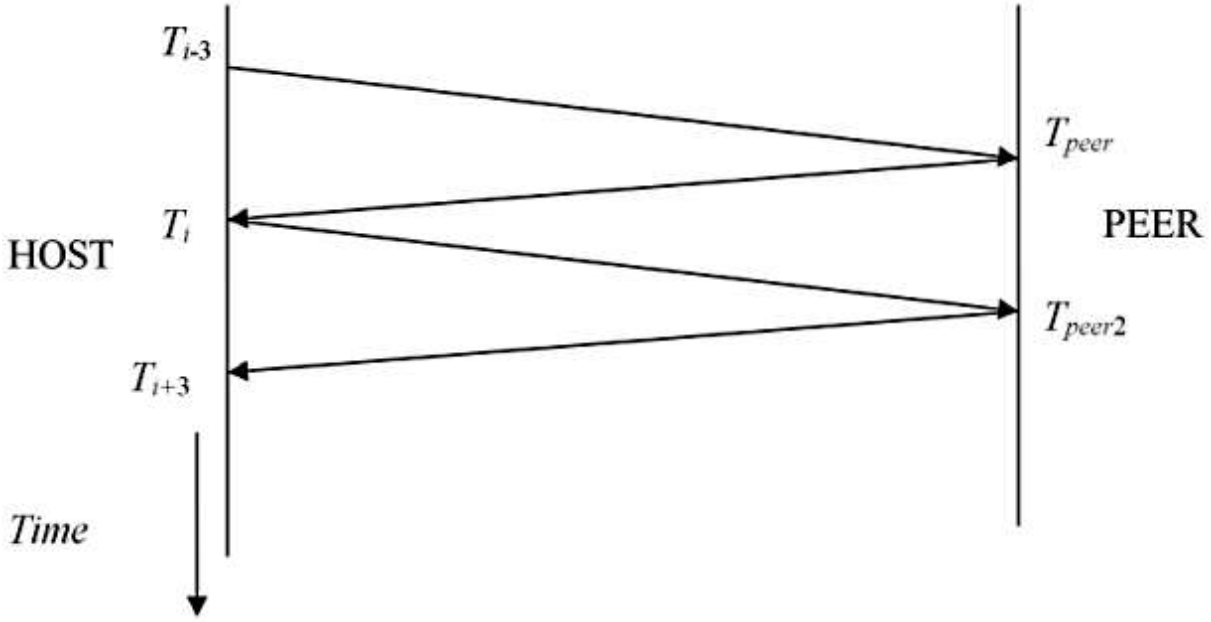

Figure 3: Two – way measurement algorithm.

Using this algorithm, it is possible to obtain two measurements at one time, the first for host – peer – host direction and the second for peer – host – peer direction. With adapted clock offset formula, the second clock offset can be enumerated as

$$\theta = \frac{(T_i - T_{peer}) + (T_i - T_{peer2})}{2} = T_i - \frac{T_{peer} + T_{peer2}}{2}. \quad (5)$$

With this two–way measurement, we expect that it is possible to correct errors which can be caused by GPRS nature resulting in a simple arithmetic mean at least.

2.3. MEASUREMENTS

The server – client application used to collect samples was successfully developed and tested. The next part is to analyze the results and to propose statistical methods that should lead to proper results.

Couple of measurements was conducted to validate our theoretical expectations. These tests proved that GPRS is not very suitable for precise time offset definition, but it is believed that a proper method can be developed to express it much more precisely.

In Figure 4, there is a typical distribution function of the results achieved by a 100-cycle measurement. The right time offset value is 3816ms and is marked by line. Distribution graph shows us, that some of the results are very inaccurate and whole measurement is encumbered with random errors, which mainly pushes the time offset to lower values than it actually is. Also the graph shows us, that systematic faults are also very probably involved, pushing the peak of distribution to wrong position. The most difficult task is to analyze the results in way, which will separate only inadequate samples and obviates affects of systematic fault.

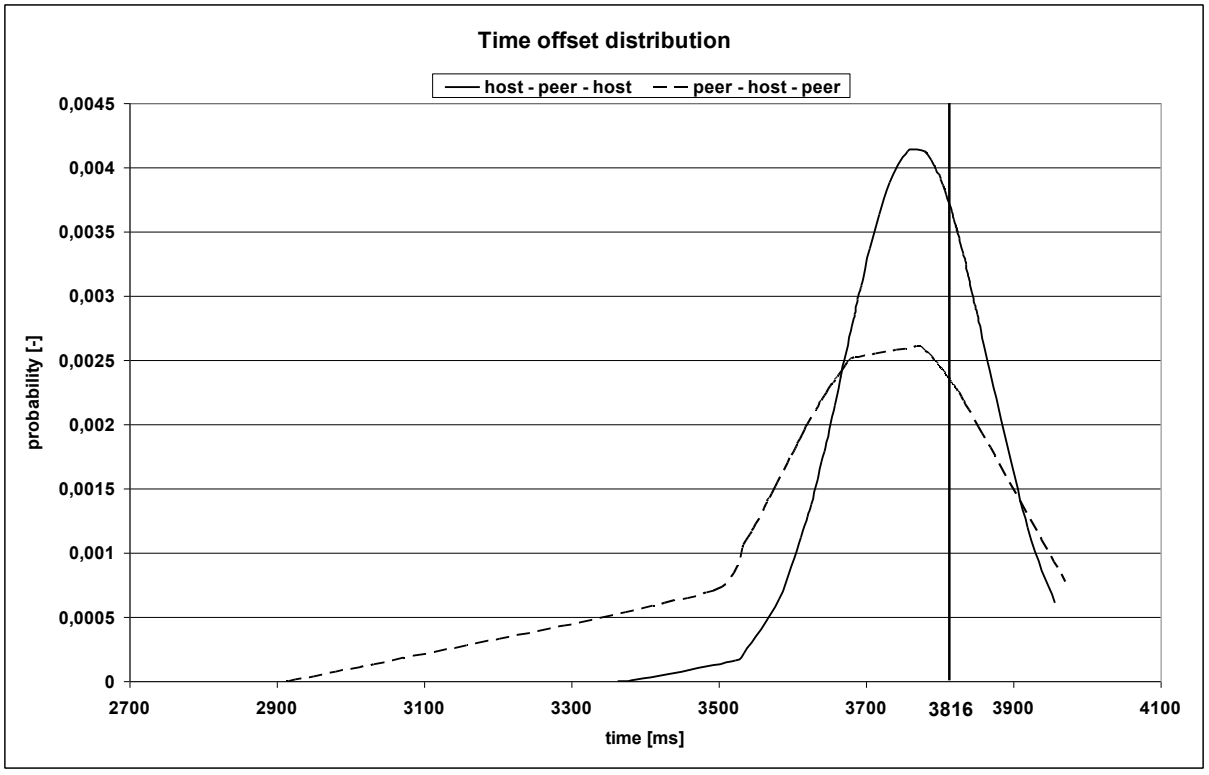

Figure 4: Two – way measurement algorithm results

Using only the arithmetic mean, the results are not quite precise and the difference is about 80 milliseconds. The way how to achieve better results is to remove samples with large dispersion (the best results were achieved when samples with dispersion larger than 15000 were removed) and using median instead a simple arithmetic mean. With this improve-

ment, it is possible to calculate the time offset with accuracy of approximately 30 milliseconds.

2.4. CONCLUSION

The performed analysis proved that our adaptation of the NTP protocol on the application layer is possible. While using high-latency network, achieving of exact time offset is very challenging, but not impossible task. This method proved, that is partially susceptible to reduce impact of difference between upload and download times. Results are showing that inaccurateness in two or three tens of milliseconds could be achieved relatively simply, but reaching accuracy in ones of milliseconds can be very difficult.

The following work on this topic will be focused on deep analysis of GPRS standards and of its specialties. We hope it will have a positive influence on the precision of the measurement and systematic fault and it is expected, that it will have biggest influence in suppression of upload and download times difference.

Another research work will be focused on finding out statistical methods which will be more suitable for the task. Finally, the JavaME version of the application will be developed and tested.

2.5. ACKNOWLEDGEMENT

This paper was written in support of the GAČR project 102/06/1569 and FRVŠ 1835/G1.

REFERENCES

[1] Mills, D.L.: Network Time Protocol, Sterling, 1985 Internet Engineering Task Force [online] http://www.ietf.org/rfc/rfc0958.txt

[2] Mills, D.L.: Network Time Protocol (Version 3) – Specification, Implementation and Analysis, Sterling, 1992 Internet Engineering Task Force [online] ftp://ftp.rfc-editor.org/in-notes/rfc1305.pdf

[3] ISO 7498:1984 Open Systems Interconnection - Basic Reference Model

[4] Internetworking Technology Handbook, Cisco Systems, 2007 [online] http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/index.htm