Precise Time Synchronization over GPRS and EDGE

Petr Kovar, Karol Molnar

Brno University of Technology, Dept. of Telecommunications, Purkynova 118,
612 00 Brno, Czech Republic
kovapetr@phd.feec.vutbr.cz, molnar@feec.vutbr.cz

Abstract. Modern communication services are more and more focused towards real time mobile multimedia applications like mobile video and Voice over IP. To assure sufficient Quality of Service control, appropriate measurements are needed. One of the most difficult tasks is to measure the exact value of one way delay, because very precise time synchronization between host and peer is needed. Most multimedia applications use JavaME platform and GPRS or EDGE technologies without any additional services or hardware. This leads to the very challenging task: To assure precise time synchronization using application layer and high latency networks.

Keywords: Time offset, time synchronization, EDGE, GPRS, JavaME

1 Introduction

Precise time synchronization over data networks is not a new problem. In 1985, the first version of Network Time Protocol was released [1]. The Network Time Protocol is marginally used on low latency networks or at least on networks with constant traffic delay. Using GPRS or EDGE services is much more problematic, because traffic delay is very variable and is depending on actual network load.
Other problem arises when JavaME platform is used. JavaME is platform designed for usage on resource constrained devices ant it uses sandbox model for assuring better compatibility end security. Lower network layers cannot be accessed directly, this can be made only with Java API.

2 Theoretical expectations

2.1 Available directions

Most publications focused on the problem are counting with high, bud invariable delay [2][3]. The solution of this problem should be, at first, to propose whole new standard. The second possibility is to analyze Network Time Protocol and its adaptation, if it is possible, to application layer and networks with very variable

latency. For the following work, the second possibility was chosen. The very important condition is that accuracy in ones or few tens of milliseconds is sufficient for the job.

2.2 Network Time Protocol

As were said, the Network Time Protocol specification was proposed in 1985 [1]. We have the third version of NTP specification at the moment [4].
For exact definition of the clock offset between host and peer, the NTP protocol specifies the clock offset as following function:

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

where T_i, T_{i-1}, T_{i-2} and T_{i-3} are timestamps described by Fig. 1.
Roundtrip delay is defined as

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

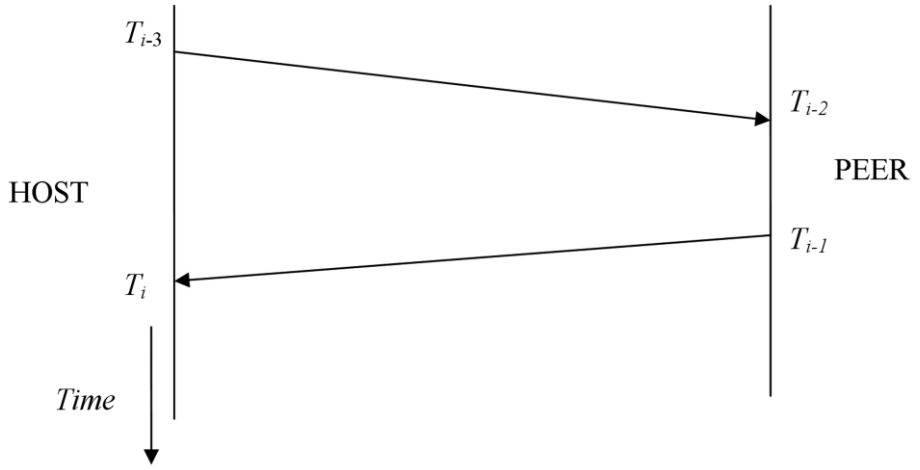

Fig. 1. Network Time Protocol principle

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 layer and thanks for that, it is able to recognize exact time when the packet was received or sent.

2.3 Network Time Protocol application layer adaptation

Software designated for JavaME platform, which is ideal for developing universal and highly compatible mobile applications, is not able to access network layer directly. Network communication is possible only using JavaME API libraries. Because of that, it is necessary to adapt NTP to work on application layer. In this case, probably the biggest problem is that it is not possible to detect exact moment in which is packet sent or received. For this case, much more corresponding traffic scheme is shown on Figure 2.

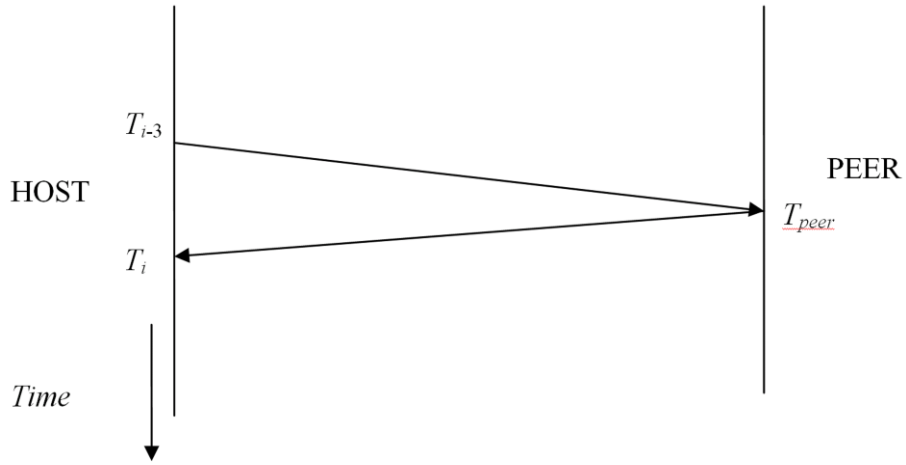

Fig. 2. Network Time Protocol adapted to application layer

When it is not possible to detect exact moment of packet receiving or sending, the time variables T_{i-2} and T_{i-1} are merged to only one variable, T_{peer}, because when application is optimized to count with application layer limitations, it is not useful to try to differentiate the moments when the packet was received and sent. For this case, clock offset and roundtrip delay are defined as

$$\theta_1 = \frac{\left(T_{peer} - T_{i-3}\right) + \left(T_{peer} - T_i\right)}{2} = T_{peer} - \frac{T_{i-3} + T_i}{2}, \tag{3}$$

$$\delta_1 = \left(T_i - T_{i-3}\right) - \left(T_{peer} - T_{peer}\right) = T_i - T_{i-3}. \tag{4}$$

2.4 High latency networks - two-way algorithm

Time synchronization in high latency networks with very high jitter, as is in GPRS or EDGE, is very challenging task. The roundtrip delay for GPRS/EDGE 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. If this happens, it is absolutely necessary to perform measurements not only in one direction, but in both directions. Also, it is necessary to perform the second way measurement as earliest as possible, because network load and other conditions influencing traffic delay can change in any second. With regard to this fact, the two way algorithm is proposed (Fig. 3).

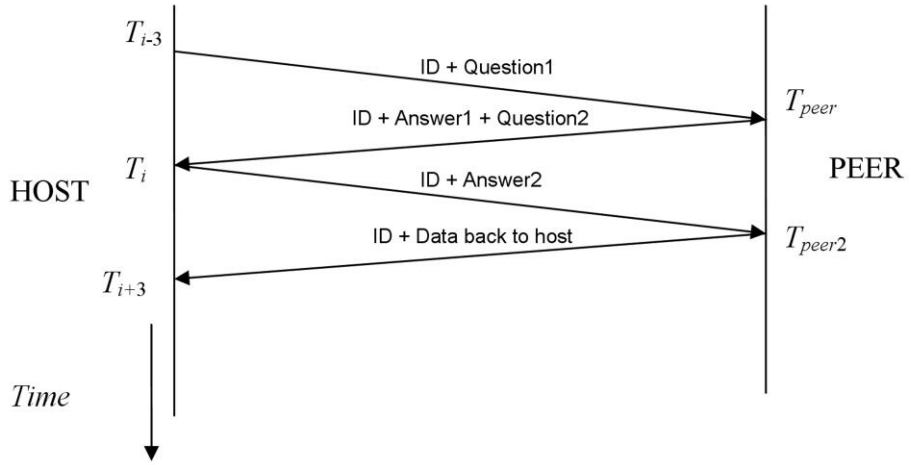

Fig. 3. Network Time Protocol adapted to application layer with two way measurements

Using this algorithm, it is possible to obtain two measurements at almost one time, the first for host – peer – host direction and the second for peer – host – peer direction. The first clock offset remains as (3) and the second clock offset can be enumerated as

$$\theta_2 = \frac{(T_i - T_{peer}) + (T_i - T_{peer2})}{2} = T_i - \frac{T_{peer} + T_{peer2}}{2}. \quad \textbf{(5)}$$

The second roundtrip delay is equal to

$$\delta_2 = (T_{peer2} - T_{peer}) - (T_i - T_i) = T_{peer2} - T_{peer}. \quad \textbf{(6)}$$

Using this two way algorithm, we believe that it is possible to eliminate traffic delay variation and obtain clock offset value in precision of tens. Also, two way algorithm helps us to determine values that are encumbered with errors and should be disqualified.

3 Obtained results, discussion

3.1 Java application

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.

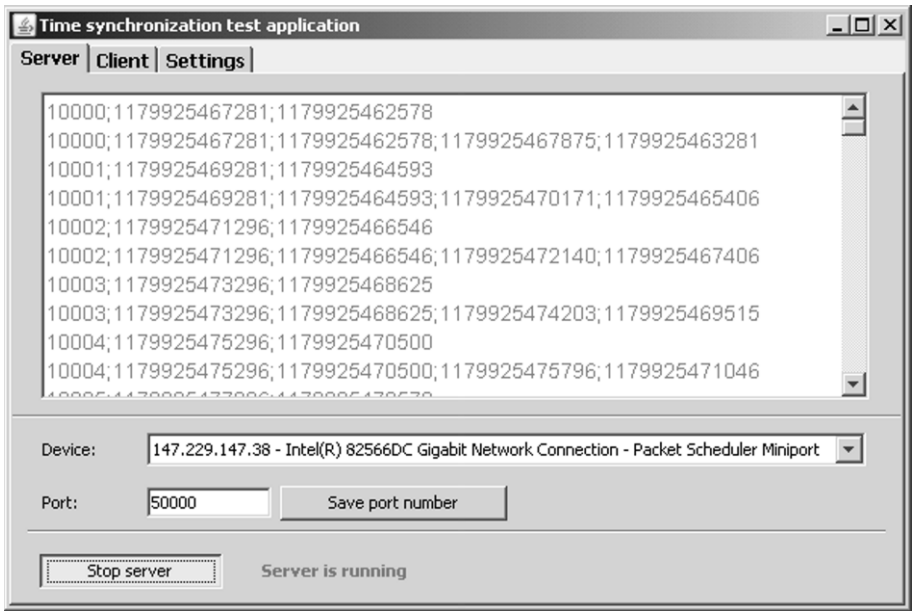

Fig. 4. Time synchronization test application, server side

The UDP connection was used, because the TCP error correction methods are not acceptable here. Even if all packets don't carry the same amount of information, we decided to set all packets to same size, filling spare place with blank data.

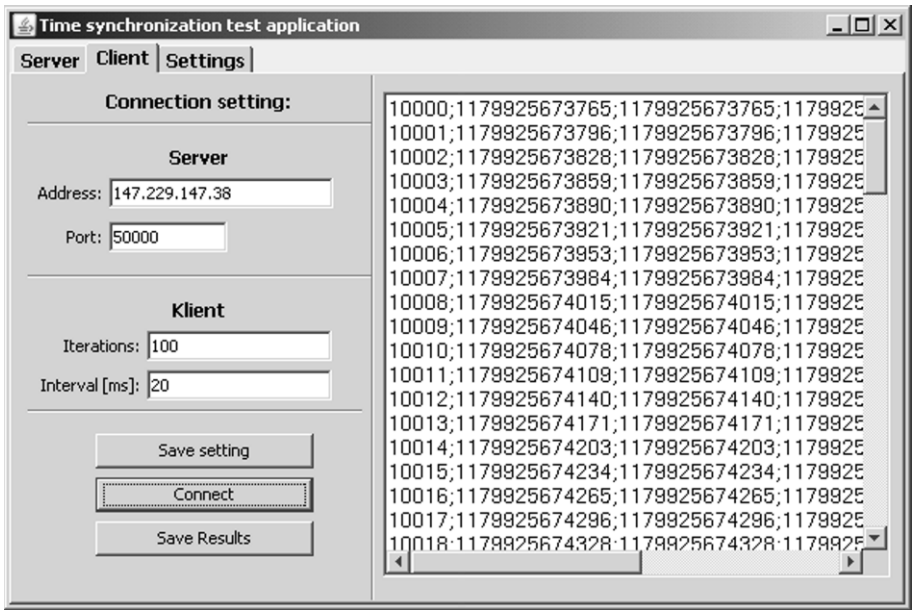

Fig. 5. Time synchronization test application, client side

3.1 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 evaluation methods that should lead to proper results.

In Figure 7, there is a typical distribution of the results achieved by a 100-cycle measurement. The right time offset value is 3816ms and is marked by line. 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.

After deep analysis of the problem, we believe that most time offset measurement errors arise from situation, when traffic delay in forward direction is different from traffic delay in backward direction. With increasing difference, the error size is increasing, too. With two-way measurement algorithm we can partially eliminate this problem, because some of the error encumbered samples can be identified and disqualified.

Many tests were carried out and proved that, for precision in ones, only about 100 iterations are needed. This exceeded out theoretical expectations. With low network load, only about 15 - 25% samples have to be disqualified. Tests also proved that it is

not effective to start one iteration when another was not finished yet. This can partially overload the network and lead to unusable results. The sufficient inter-iteration interval is about 2000 ms.

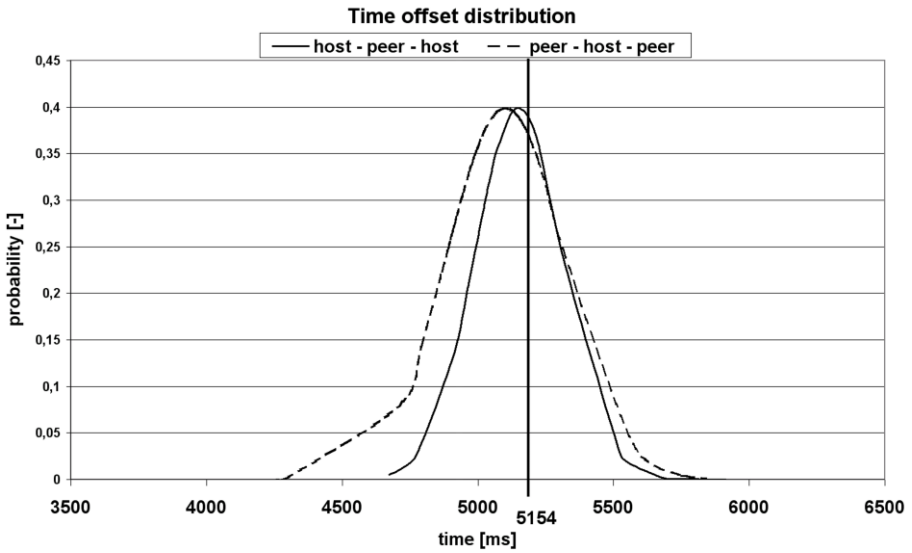

Fig. 6. Two-way measurements algorithm results distribution for 100 iterations.

To achieve still sufficient precision in few tens of milliseconds, 20-iteration measurement is needed, but this still needs about 40 seconds.

4 Conclusion and following work

Time synchronization over high latency network with huge delay variation is very challenging, but not impossible task. Using proposed methods, the success rate partially depends on network load, when the biggest influence has an additional traffic generated by GPRS-connected side of communication. Because of this, we recommend that time synchronization should be only task performed on GPRS-connected terminal in one time.

The following work on this topic will be focused on proposing other optimalizations, which will reduce the time needed for measurement with sufficient precision. The target time is about 15 seconds. When finished, algorithm will be used to determine characteristics of communication channel and will help with setting optimal traffic parameters before its start.

Acknowledgments: The paper was written with support of the MŠMT 1K04116, GAČR 102/06/1569, MŠMT MSM0021630513 and FRVŠ 1835/G1 project.

References

1. Mills, D.L.: Network Time Protocol, Sterling, 1985 Internet Engineering Task Force [online] <http://www.ietf.org/rfc/rfc0958.txt>
2. Syed, A.A., Heidemann, J: Time Synchronization for High Latency Acoustic Networks, INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings, Barcelona, page 1 – 12, ISSN: 0743-166X, ISBN 1-4244-0221-2
3. Elson, J., Estrin, D.: Time synchronization for wireless sensor networks, Parallel and Distributed Processing Symposium., Proceedings 15th International Volume , Issue , Apr 2001, page1965 – 1970, ISSN: 1530-2075, ISBN: 0-7695-0990-8
4. 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>