Delay Characteristics and Server Update Optimization of Multiplayer Gaming in Mobile Environment

Gábor Kiss[1], János Levendovszky[1], Sándor Molnár[2], Long Tran-Thanh[1]
[1]Dept. of Telecommunications, Budapest University of Technology and Economics (BME), Hungary
[2]Dept. of Telecommunications and Media Informatics, BME, Hungary
{kissg,levendov,ttl}@hit.bme.hu, molnar@tmit.bme.hu

Abstract— **In this paper a novel approach to provide satisfactory multiplayer gaming quality in mobile environment is presented. The paper has two contributions: (i) evaluation of the delay characteristics of multiplayer games in mobile environment based on extensive measurements to verify whether HSDPA access can provide a satisfactory gaming quality; (ii) improving the gaming quality with new server update time optimization algorithms taking advantage of the statistical delay characteristics.**

Keywords- Gaming quality, HSDPA network, server update time optimization

I. INTRODUCTION

Multiplayer gaming is a rapidly growing segment of the computer game industry [3]. Most of the video games (e.g. FPS - First Person Shooter; RTS - Real Time Strategy) supports a lot of simultaneous players which requires increasing network and computational resources, due to the sharp rise in the number of players [1]. With the appearance of new generation networks capable of servicing multimedia streams, the players will have facilities to play games from different locations. This implies that game servers should serve players with highly different access parameters (e.g. players from 3G wireless networks and players having optical based connections can experience different RTT) and provide uniform gaming quality [2].

With emergence of HSDPA access one of the concerns lies with the question is whether HSDPA is capable of providing the required gaming quality. As a result, one of the main aims of the paper is to answer this question by performing RTT measurements by pinging a server via HSDPA access. Preliminary investigation can also be found in [16], [17]. The experiments were performed on different periods of the day (during busy hours and early morning) and in different environment (downtown, outskirt and rural area) with three mobile service providers by pinging a server via HSDPA access and measuring RTT delays. A thorough statistical analysis on the measurements has revealed that in the case of underloaded cells the required gaming quality can be ensured if being in Cell_DCH state [14]. However, the delay resulting from being in Cell_FACH does not provide a satisfactory level of QoS. Furthermore, in the case of highly loaded cells scheduling may have a fundamental impact on the delays. As a result, further algorithmic means are necessary to improve gaming quality, such as optimizing the server update period.

In order to achieve this goal we propose server update time algorithms. These algorithms can improve either (i) the tail probability of the maximal idle time; or (ii) the probability of missing an update period. Our objective is to choose an appropriate server update time subject to minimizing the probability of the maximal idle time exceeding a certain threshold (which threshold is associated with the "psycho-physically approved quality of the game"). The performance of the new methods has been evaluated by extensive simulations based on the statistics obtained from the measurements.

The paper treats these topics in the following order. In Section I and II the technical challenges of gaming in mobile environment are outlined. The HSDPA delay state graph is described in Section III. In Section IV the measurement setup and environment is discussed followed by an extensive statistical analysis in Section V, which reveals the delay characteristics of HSDPA. Our new server update optimization has been introduced in Section VI. Furthermore, an adaptive update algorithm is also introduced in Section VII. The performance of the algorithms is shown in Section VIII. Finally, the paper is concluded our summary in Section IX.

II. EFFECT OF NETWORK LATENCIES ON GAMING QUALITY

In this section the effect of delays originating from mobile network accesses are analyzed.

In a LAN environment the latency is typically small and exhibits homogenous characteristics. However, in mobile network environment the following attributes must be taken into account:

- clients connect to the game server from different mobile service providers, which can causing different latency, delay, delay variation;
- client connect from different mobile cells;
- clients may experience a large delay variation within a game session depends on the mobile cell usage, traveling speed, distance from the mobile antenna;

These attributes show that delay and jitter can have extremely high impact on quality in mobile networks. The general characteristics and classification of 2G and 3G delays are discussed in [8] and [9]. However, the measured delay and jitter varies at different operators based on their system version, configuration, traffic load, etc.

The RTT values are typically between 200ms and 500ms in 2G and between 60ms and 120ms in 3G networks [10], [11], [7], [12]. Even though there is a continuous improvement of delay characteristics in mobile networks, the delay variation is still typically higher than in fixed environment.

Results about the tolerated delay and jitter for FPS games show that 139ms is the maximum delay for mobile real-time games, however, for hard-core gamers 60ms round trip time with a 8% of packet loss can only be tolerated [4]. In [5] a delay bound of 150ms is defined for Halflife, whereas in [13] 300ms delay maximum is given for RTS games, e.g., Age of Empires. In [10] a maximum acceptable end-to-end delay was evaluated between 100ms and 200ms. In [6] the effect of latency on online Madden NFL Football was studied and the authors concluded that there is little impact from latency on client performance with latencies as high as 500ms. However, with latencies higher than 500 ms the performance can degrade by almost 30 percent.

These values can be met in LAN environments, but 2nd and newer generation mobile networks only partially satisfy these requirements as depicted by *Table 1* [8], [9], [14], [15]:

TABLE I. MOBILE NETWORK DELAY CHARACTERISTICS

TCP down	GPRS 42.47 kbps	EDGE 211.78kbps	UMTS 353.31 kbps	HSDPA 14.4 Mbps
TCP up	22.79 kbps	99.55 kbps	57.27 kbps	355.15 kbps
Best case latency	200 msec	140 msec	103 msec	60-80 msec

As can be seen from the tolerated delay and packet loss values and the performance of the mobile networks, only HSDPA (HSxPA) is potentially capable of serving real time games, but the delay performance of it is barely enough for the needs of hard-core gamers.

In the following section we tried to obtain answers for the following questions:

- What is the delay characteristic of the HSDPA network?
- What is the delay variation, packet loss probability of the network?
- What is the impact of network load and usage on delay characteristic?
- What kind of methods can be developed in order to ensure quality improvement for keeping delay and jitter in the tolerable limits, if the performance of HSDPA leaves its best-cases values?

III. THE HSDPA ACCESS

Recently service providers have introduced several new access modes and among them HSDPA [15] tends to become the most popular one because of reasonable price and relatively high data speed. As a result, HSDPA can also become the access technology for mobile gaming. This prompts to launch investigation into its delay characteristics. From the point of delays HSDPA can be modeled as the finite state machine indicated by Figure 1.

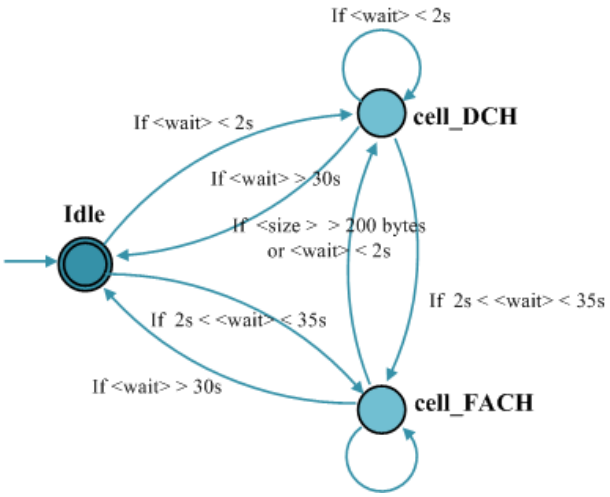

Figure 1. The finite state machine of HSDPA.

The HSDPA access uses three user states as far as the delays are concerned [14]:

Idle: The user is in this state after the connection has been built up.

cell_DCH: In this state the average packet delay is 60–80ms – it must be noted that this is the relevant state for gaming scenario.

cell_FACH: In this state the average packet delay is 240 – 300 ms

After successful connection (it takes about 1.6-1.9 sec) the user gets into the *idle state*. If emitting packet starts within 2 sec after the connection built-up, then the user gets into the *cell_DCH* state which ensures relatively small delay. It will stay in this state as long as the inter-arrival time of emitted packets is smaller than 2 sec. If packet emission starts later than 2 sec but earlier than 30-35 sec then the user gets into the *cell_FACH* state. Furthermore, the user will visit *cell_FACH* state any time when the inter-arrival time of emitted packets exceeds 2 sec but still smaller than 30-35 sec. From the *cell_FACH* state the user can get back to the *cell_DCH* state if the size of the emitted packets is larger than 200 bytes or if the inter-arrival time gets smaller than 2 sec. To return from *cell_FACH* to *cell_DCH* takes about 600-700 msec. If the inter-arrival time exceeds 30-35 sec than the user returns to the *Idle* state.

IV. MEASUREMENT SETUP AND ENVIROMENT

The measurement setup is shown by *Figure 2*.

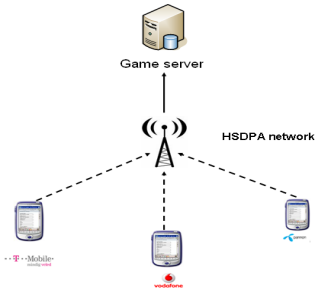

Figure 2. Measurement setup.

The measurements were performed in three type of locations: downtown Budapest, outskirt of Budapest, countryside. The access guaranteed by three different service providers. The measurement tools were laptops, USB/PCMCIA HSDPA modems (3.6Mbps, 7.2Mbps).

We performed

- continuous measurements throughout the day (pinging the server continuously);
- continuous measurements in different periods of the day (busy and non-busy hours)
- suspended measurements (measurement are suspended for 100ms – 30s periods of time)

V. STATISTICAL ANALYSIS

Some of the most typical average values computed from several measurements are indicated by the next table:

TABLE II. AVERAGE HSDPA DELAYS IN DIFFERENT NETWORKS

	$\mathbf{avg}\ \max \tau^{HSDPA}$	$\mathbf{avg}\ \min \tau^{HSDPA}$
Service Provider 1	328 ms	92 ms
Service Provider 2	364 ms	88 ms
Service Provider 3	273 ms	87 ms

Based on the measurements, the underlying probability density functions of the delays were estimated by histograms shown in the Figure 3 and Figure 4.

The delays in the two HSDPA states (FACH, DCH) were also determined by the following measurement setup: a client machine was set up to ping a server in a random time between 100ms – 45sec, and random packet size between 100-500 byte. The result is depicted by *Figure 5*, where the two states can easily be seen.

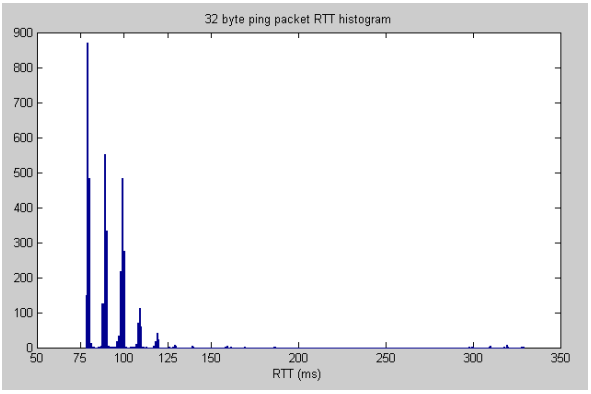

Figure 3. Histogram of 32 byte long ping RTT (downtown at noon).

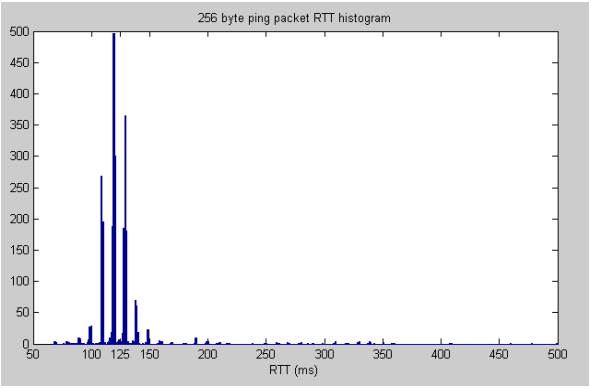

Figure 4. Histogram of 256 byte long ping RTT (outskirt at night)

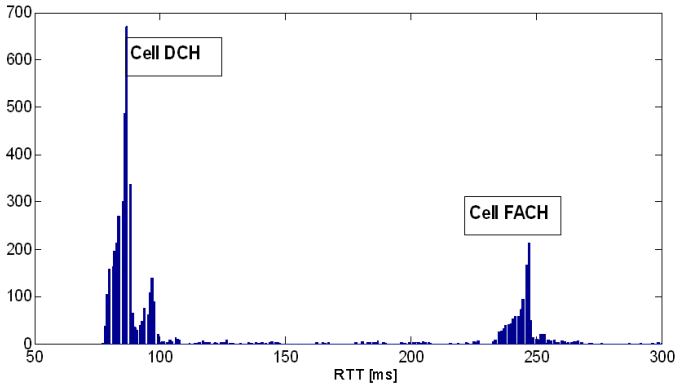

Figure 5. Two different states of the HSDPA network.

As it can be seen from *Figures 3* and *4,* a packet is retransmitted in the system up to five times, and the packet retransmission interval is approximately 12 msec.
These measurements were performed in two different place, with 32-500 by packet size.

Figure 5 shows the delay histogram of a user behavior from which the two states of the HSDPA can be well identified. The statistical fluctuation of the delays is well centered on two delay values corresponding to CellFACH and CellDCH with a relatively low variance. As it turns out in the low delay CellDCH state the guaranteed 60-100msec delay is just low enough to provide a satisfactory gaming quality [2].

The appropriate batch size (i.e. the minimal number of samples) for evaluating the delay p.d.f.-s has been identified by the Kolmogorv-Smirnoff and the Wilcocxon test. Wilcoxon test [18] performs a two-sided rank sum test of the hypothesis that two independent samples, in two different vectors, come from distributions with equal medians. Kolmogorov-Smirnoff test [19] performs a two-sample test to compare the distributions of values in the two data vectors.

Table III shows the result of the hypothesis test, where the null-hypothesis is if the dataset is stationary. The different columns respects to the result with different batches, where e.g 5-batch means that the stationary analysis was tested on the dataset using 5 measured RTT sets as one unit. It can be seen that using 2 sec long sets (approx. 20 pings) the delay set can be regarded stationary.

TABLE III. OPTIMAL BATCH SIZES OBTAINED BY KOLMOGOROV-SMIRNOFF AND WILCOXON TESTS

Accept the nul-hypothesis	5-batch 0.5s	10-batch 1s	20-batch 2s	30-batch 3s	40-batch 4s	50-batch 5s	100-batch 10s
Kolmogorov-Smirnov test		NO	YES	YES	YES	YES	YES
Wilcoxon test	YES/NO	YES/NO	YES	YES	YES	YES	YES

VI. ASYNCHRONOUS SERVER UPDATE OPTIMIZATION

In this section we set on optimizing the server update period in order to improve the gaming quality. Gaming quality is measured as the probability of missed gaming action which is due to the late arrival of gaming action, when the server update period has already expired. To evaluate this measure we will use the p.d.f.-s of delay process τ_i of the users i obtained from the measurements and denoted by $F_i(t) = P(\tau_i < t),\ \ i = 1,\ldots,N$ from which we will develop the extremal statistics.

Furthermore, we assume independent delays. In the following discussion, let $\tau_{\min}$ and $\tau_{\max}$ denote the minimum and maximum access time, respectively (both of them are random variables). One can assume, that server update takes place with some waiting period A, after the client with the fastest access $\tau_{\min}$ has reached the server. This implies that the time interval available for receiving actions from the clients is $[\tau_{\min}, \tau_{\min} + A]$. One can see that this scheme indeed yields an asynchronous server update since there can be an arbitrary long period between two consecutive updates denoted by *T(k)*, as depicted by the next figure:

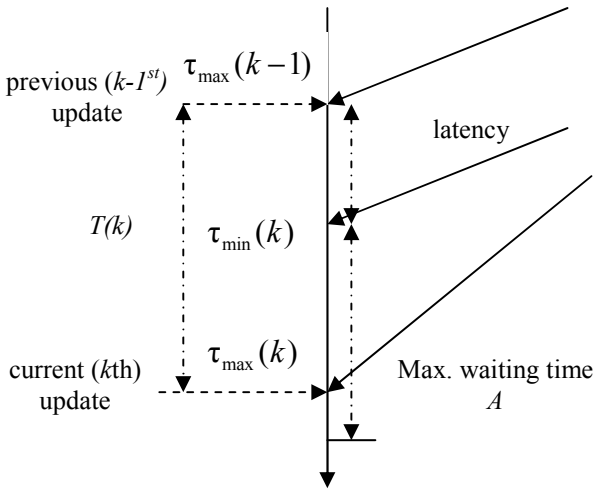

Figure 7. Server update process in the case of asynchronous operation.

From the figure, it is easy to deduce that $T(k) = \tau_{\min} + R;\quad R := \min\{A, \tau_{\max}\}$. In the asynchronous case, the main concern is to evaluate QoS parameter, being defined as the probability that the client with the worst access $\tau_{\max}$ is also accommodated in the update interval (i.e. its action is not lost). This probability is given as follows:

$$P_{no\text{-}loss} := P(\tau_{\max} < \tau_{\min} + A) = \int_0^\infty P(\tau_{\max} < t + A) f_{\min}(t)\, dt$$

$$= \int_0^\infty F_{\max}(t+A) f_{\min}(t)\, d \tag{1}$$

Where $F_{\max}(u) := P(\tau_{\max} < u)$ and

$$f_{\min}(u) = \frac{dF_{\min}(u)}{du};\quad F_{\min}(u) := P(\tau_{\min} < u) \tag{2}$$

Thus to calculate expression (1) we need to evaluate $F_{\max}(u)$ and $F_{\min}(u)$, respectively. This can be done as follows, where *J* is the number of players:

$$P(\tau_{\min} > u) = P\left(\bigcap_{i=1}^{J}(\tau_i > u)\right) = \prod_{i=1}^{J} P(\tau_i > u) = \prod_{i=1}^{J}(1 - F_i(u)) \tag{3}$$

thus

$$F_{\min}(u) = P(\tau_{\min} < u) = 1 - P(\tau_{\min} > u) = 1 - \prod_{i=1}^{J} P(\tau_i > u) = \tag{4}$$

$$= 1 - \prod_{i=1}^{J}(1 - F_i(u))$$

And

$$F_{\max}(u) = P(\tau_{\max} < u) = P\left(\bigcap_{i=1}^{J}(\tau_i < u)\right) = \tag{5}$$
$$= \prod_{i=1}^{J} P(\tau_i < u) = \prod_{i=1}^{J} F_i(u)$$

Furthermore

$$f_{\min}(u) = \frac{dF_{\min}(u)}{du} = \frac{d}{du}\left(1 - \prod_{i=1}^{J}(1 - F_i(u))\right) = \tag{6}$$
$$= \sum_{i=1}^{J} f_i(u) \prod_{l=1, l\neq i}^{J} (1 - F_l(u))$$

This yields

$$P_{no-loss} := P(\tau_{\max} < \tau_{\min} + A) =$$
$$= \int_0^{\infty} F_{\max}(t + A) f_{\min}(t) dt = \tag{7}$$
$$= \int_0^{\infty} \prod_{i=1}^{J} F_i(t + A) \left(\sum_{i=1}^{J} f_i(t) \prod_{l=1, l\neq i}^{J} (1 - F_l(t)) \right) dt$$

Based on the formula above, one can evaluate the probability of no loss if the maximum waiting time A before update is given. Or, conversely, one can set the parameter A in order to enforce that $P_{no-loss} > 1 - \varepsilon$ for a pre-defined ε.

The discussion above gives rise to the following computational model in *Figure 8*:

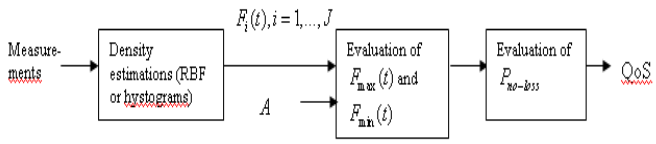

Figure 8. Computational model of server update time optimization.

VII. ADAPTIVE SERVER UPDATE TIME OPTIMIZATION

In the previous section, server update time optimization has been dealt with as an off-line task, carried out prior to the game. In this case, the user's delay p.d.f.-s $f_i(t), i = 1,...,N$ has been obtained by fitting a model p.d.f. (RBF) to the measured data. In the present section, we investigate an on-line approach when delay measurements during the game are taken into account. Thus, the aim is to update the delay densities based on the current measurements by implementing a recursive estimation $f_i(t, k+1) = \Psi\left(f_i(t,k), t_k^{(i)}\right), i = 1,...,N$ where k refers to the fact that the p.d.f. is estimated after observing the first k measurements and $t_k^{(i)}$ denotes the kth observation of the delay of client i. Now we use a histogram estimation given as follows:

$$f_i(t,k) := \sum_{l=1}^{L} n_l(k) I_l(t) \tag{8}$$

where $n_l(k)$ is the relative frequency of the samples falling into the interval $\Delta t_l := t_l - t_{l-1}$ and

$$I_l(t) := \begin{cases} 1 & \text{if } t \in \Delta t_l \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

When a new measurement is taken about the delay of client i in the course of the game, the corresponding density is updated as

$$f_i(t, k+1) = \sum_{l=1}^{L} n_i(k+1) I_l(x) \tag{10}$$

Where

$$n_l(k+1) = \frac{n_l(k) N(k) + s_l(x)}{N(k) + 1} \tag{11}$$

and $s_l(x) = \begin{cases} 1 & \text{if } t_k^{(i)} \in \Delta t_l \\ 0 & \text{otherwise} \end{cases}$ (12)

Performing recursion (11) in each new measurement, the server update time is optimized recursively by plugging the updated p.d.f.-s into expression (3) and (4), respectively. In this way, the server can optimize the update time based on the newly obtained delay information in the course of the game.

VIII. PERFORMANCE ANALYSIS

The result of the performance analysis is shown by *Figure 9*. It can be seen that the randomly chosen sever update time or the standard *40*ms (used by well known games by default) can result a notable QoS degradation. However, with the novel asynchronous server update time optimization method the best server update time can be achieved for the game. From *Figure 9* one can see that if *123*ms is set to be the server update period time – instead of *40*ms - , then the packet loss probability is low enough, and the probability that the server idle period is shorter than a predefined value is minimal, while any other value result in high loss probability or undefined server idle time.

Furthermore games characteristics in mobile networks are changing due to new user connections or to new mobile environment which can be seen from *Figure 10*. It also shows the result of the adaptive server update time optimization process. It can be seen, that in the beginning of the game the optimal server update time was 22ms more than in the next period. In *Figure 10* a tick represents approximately 4 sec, which value was given by the stationary analysis.

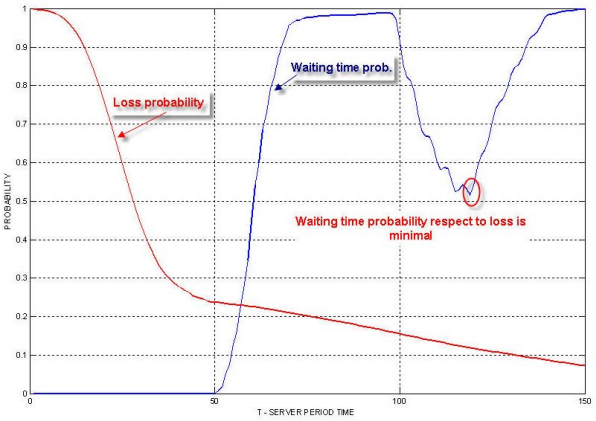

Figure 9. Result of server update time optimization.

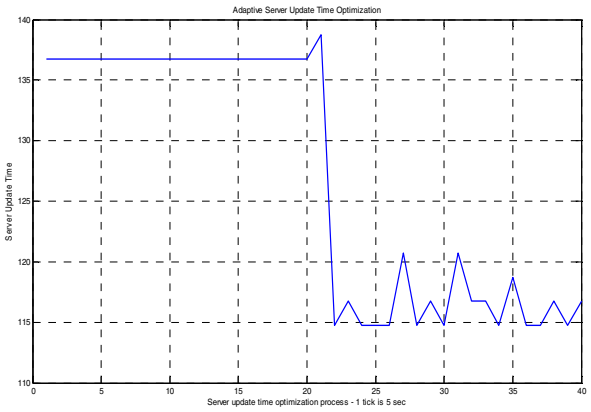

Figure 10. Optimal server update time given by adaptive server update time optimization.

IX. CONCLUSIONS

In this paper, measurements have been reported and a corresponding statistical analysis has been carried out to discover the delay characteristics of HSDPA mobile networks. We found that HSDPA networks have a potential of running FPS games, but further improvement is necessary in order to provide good quality for wide range of gamers and games. The analysis also demonstrated to us, that traditional protocols are not efficient in mobile environment, where latency and delay is typically higher than in high-speed fixed networks.

We viewed game protocols as arrival and update processes running on a server and found that the loss, average idle time and the probability that the idle time is higher than a specific value greatly depend on the choice of server update period *T*. Thus, server update time optimization proved to be an efficient tool to compensate latencies.

We have developed a statistical model to express the tail probability of maximal idle time as function of the server update period. Based on this model and the result of the stationary analysis the server update time can be adaptively optimized which also provide better performance for game in more varying mobile environment. Our new solution works on the server side and has a fast convergence speed.

By decreasing the maximum idle time by the new method, the game provider can support much more clients from mobile network environment. Therefore, our methods can contribute to achieving higher revenues from network games. Moreover, finding the optimal server update time can further increase the perceived game quality and satisfaction.

ACKNOWLEDGEMENT

The authors gratefully acknowledge the kind help of András Veres, Zsolt Kenesi and József Barta from Ericsson Traffic Lab. The research was supported by NKTH-OTKA grant CNK77802 and S. Molnár was supported by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences.

REFERENCES

[1] W. Holden, Juniper research, Mobile Games Subscription & download 2009.

[2] M. Busse, B. Lamparter, M. Mauve, W. Effelsberg: Lightweight QoS-Support for Networked Mobile Gaming, SIGCOMM Workshops, Aug. 30+Sept. 3, 2004, Portland, Oregon, USA.

[3] A. Chandler, J. Finney: On the Effects of Loose Causal Consistency in Mobile Multiplayer Games, NetGames'05, October 10–11, 2005, Hawthorne, New York, USA.

[4] C. Schaefer, T. Enderes, H. Ritter, and M. Zitterbart, "Subjective Quality Assessment for Multiplayer Real-Time Games," *In Proc. of the 1st Workshop on Network and System Support for Games (NetGames), Braunschweig, Germany*, pp. 74–78, April 2002.

[5] T. Henderson, "Latency and user behaviour on a multiplayer games server," *In Proc. of the 3rd International Workshop on Networked Group Communication (NGC), London, UK*, pp. 1–13, November 2001.

[6] J. Nichols and M. Claypool, "The Effects of Latency on Online Madden NFL Football," 14th International Workshop on Network and Operating Systems Support for Digital Audio and Video, Cork, Ireland, 2004.

[7] J. Korhonen, O. Aalto, A. Gurtov, and H. Laamanen, "Measured Performance of GSM HSCSD and GPRS," IEEE International Conference on Communications (ICC), January 2001.

[8] "3GPP TS 22.060 GPRS, Service description Stage 1."

[9] "3GPP TS 23.107 Quality of Service concept and architecture."

[10] M. Busse, B. Lamparter, M. Mauve, and W. Effelsberg, "Lightweight QoS-Support for Networked Mobile Gaming," *In Proc. of the 3rd Workshop on Network and System Support for Games, NETGAMES 2004, Portland, Oregon, USA*, pp. 85–92, August 2004.

[11] A.Gurtov, M. Passoja, O. Aalto, and M. Raitola, "Multi-Layer Protocol Tracing in a GPRS Network," IEEE Vehicular Technology Conference (Fall VTC 2002), Vancouver, Canada, September 2002.

[12] R.Chakravorty and I. Pratt, "Performance Issues with General Packet Radio Service," *Journal of Communication and Networks (JCN)*, 2002.

[13] M. Terrano and P. Bettner, "1500 archers on a 28.8: Network programming in Age of Empires and beyond", 15th Games Developers Conference, San Jose, CA, USA, March 2001.

[14] 3GTS 25.331 - 3rd Generation Partnership Project, RRC protocol spec.

[15] 3GPP TS 25.855 - 3rd Generation Partnership Project, Overal description, stage 2, Release 6, 2004.

[16] Prokkola, J.; Hanski, M.; Jurvansuu, M.; Immonen, M. "Measuring WCDMA and HSDPA Delay Characteristics with QoSMeT", Communications, 2007. ICC apos

[17] M. Jurvansuu, J. Prokkola, M. Hanski and P. Perälä „HSDPA Performance in Live Networks" ICC 2007

[18] http://www.mathworks.com/access/helpdesk/help/toolbox/stats/ranksum.html

[19] http://www.mathworks.com/access/helpdesk/help/toolbox/stats/kstest.html