SMART: A Synchronization Scheme for Providing Multimedia Quality in Emerging Wireless Internet

Hua Zhu, Imrich Chlamtac, Jorge A. Cobb, Guoping Zeng
The University of Texas at Dallas, Richardson, TX 75083-0688, USA
{zhuhua, chlamtac, cobb, gzeng}@utdallas.edu

Abstract—**In this paper, we present a novel synchronization scheme for multimedia applications that can provide high quality playback performance in emerging wireless environments. Because of the increasing size of global networks and the integration of wireless services, end-to-end delays are gradually increasing. Under these new conditions the existing synchronization schemes based on feedback, can no longer provide the necessary synchronization and thus quality needed for multimedia applications. We propose a novel synchronization scheme that requires no feedback, and which at the same time minimizes the required buffer space. We show that the proposed scheme performs very well, while significantly reducing communication overhead due to the control scheme by removing the feedback loop. Moreover, as its minimal buffer requirements are independent of network delays, the proposed scheme is inherently scalable for the evolving wireless environments.**

1 INTRODUCTION

Several control schemes have been proposed to solve the problem of intra- and inter-stream synchronization for the smooth playback of multimedia presentations in wired networks. Ramanathan and Rangan [4] introduced the feedback techniques in the multimedia information retrieval; Hać and Xue [5, 6] proposed a feedback controller at the client buffer; Biersack and Geyer [7] proposed feedback control scheme including both distributed media servers and the client; Boukerche et al. [8,9] designed MoSync system for cellular wireless and mobile networks.

These control schemes use the idea of feedback to maintain intra- and inter-stream synchronization. Except Hać and Xue in [5,6], all schemes chose feedback loops between the client and media servers. Information of the asynchrony is collected at the client, and then is sent to the server as feedback messages. Based on the feedback, media servers take appropriate actions to resynchronize the playback at the client. End-to-end delays from the client to media servers will cause feedback delays, called the deadtime of control schemes. In [5,6] a feedback loop excluding media servers is chosen to avoid system deadtime. However, this solution cannot provide a satisfactory solution due to the uncontrollability of the server side.

Because of the increasing size of global networks and the integration of wireless services, end-to-end delays are gradually increasing. For streaming media services commonly used by the video-on-demand applications, the end-to-end delay may range from 5 to 10 sec [13]. If we consider the handoff problem that occurs in mobile networks, the end-to-end delay will be even larger. Naturally, all feedback control schemes assume finite buffers size. In this configuration assumption, if end-to-end delays in the system are very large, feedback messages will experience delays longer than the duration of playback of the full buffer, hence there is no enough grace period for feedback control schemes to work properly. This problem is becoming critical in the emerging wireless Internet.

In this paper, we propose a novel synchronization scheme – we term Synchronization scheme for Multimedia Applications for the wireless Internet (SMART), which requires no feedback. Using this scheme, the application playback performance is directly related to the buffer configuration (i.e. the pre-buffer threshold N_{th} and buffer size B), while the buffer configuration is determined by the network jitters, not end-to-end delays. The proposed scheme shows following advantages:

1) It exhibits a significantly reduced control scheme overhead as it eliminates the need for feedback messages;
2) It requires minimum buffer space for each media stream;
3) The minimum buffer requirements and the playback performance are independent of end-to-end delays, and, therefore;
4) It is inherently scalable for the evolving wireless environments.

This paper is organized as follows. Section 2 gives the synchronization scheme algorithm. Section 3 shows the simulation and results. This paper is concluded in Section 4.

2 A SYNCHRONIZATION SCHEME WITHOUT FEEDBACK

Given the feasibility analysis in [3], we know that due to the constraints of the distributed multimedia system model, especially when the average end-to-end network delay d_{avg} is large (compared to the duration of the playback of the full buffer), within a single stream (at least the master stream), the playback performance of any kind of feedback control scheme cannot be better than the uncontrolled system with same buffer size. However we still need to design a control scheme to minimize the required buffer size in order to solve the contradiction between the extremely large size of multimedia data and the limitation of client buffer space. This limitation is important to account for and maybe significant also because of the hardware limitation of mobile clients in mobile wireless systems. Therefore we propose a novel synchronization scheme without feedback messages.

2.1 *Algorithm*

The algorithm of the synchronization scheme can be describe as following:

At media servers:

1) The client sends a request to all distributed media servers. Media servers start to send MMUs to the client once it receives the request.
2) Assuming the server i starts to send the 1st MMU at local time t_{i0}, the scheduled sending time of the k^{th} MMU is $t_{i0}+(k-1)T_P$.

This work was supported in part by Texas ATP/ARP under Grant No.003594-0092-2001

3) Each media server adjusts (behind or ahead) the scheduled sending time of one specific MMU every certain period that is long enough to cause clock drift deviation larger than T_P.

At the client:

1) The 1st MMU of the master stream is scheduled to playback at time t_s that all first N_{th} MMUs of the master stream have arrived at the client buffer, where N_{th} is the pre-buffer threshold. Among the first N_{th} MMUs, if any MMU gets lost, the client restarts sending the request.
2) The scheduled playback time of k^{th} MMU in the master stream is $t_s+(k-1)T_P$.
3) Slave streams play corresponding MMUs with the same scheduled playback time as the master stream.
4) If the buffer underflow occurs in a specific stream, the buffer will duplicate the last media unit of the corresponding stream, the user application will play this duplicated media unit when it cannot obtain the correct media unit from the buffer.
5) If the buffer overflow occurs, the buffer of the corresponding stream will discard the media units arrive later, the user application will continue playing until the time it cannot play those discarded media units, at that time, it will simply play the media unit right before the discarded media unit.
6) If the buffer receives media units with the sequence number that already missed the corresponding scheduled playback deadline, the buffer of the corresponding stream will simply discard those media units.
7) When client handover occurs in mobile networks, several implementation resolutions can be chosen. For example, the client sends a message with current base station and timestamps right before and after handover to media servers. Media servers resend all MMUs with sending time in that period. Handover will introduce the problem of very large end-to-end delays for certain MMUs, which will be considered in Section 2.3.2.

T_P is the playback period of a single multimedia unit (MMU), which is usually determined by the measurements of human perception, which is also implemented as the concept of frame by standards of multimedia data, such as MPEG. For example, the playback period of a video MMU is 33 msec, while the playback period of an audio MMU is 26.1 msec.

2.2 *Properties of the Proposed Scheme*

In above algorithm, we introduced the concept of N_{th}, *pre-buffer threshold number*, which will be explained in the following section of buffer determination. The client buffer size and the pre-buffer threshold number are properly configured based on the information of maximum jitter of the corresponding network environments if there is sufficient buffer available to be allocated; if not, simply using the setting of maximum available buffer. Media servers can obtain measurements of clock drift rate to the client by any means. One of methods can be found in [14]. Usually the order of the clock drift is very small (between 10^{-5} and 10^{-11}) [12]. For T_P=33 msec, the single adjustment at media servers is only needed for every period longer than 3300 sec. Therefore there is enough time for servers getting measurements of clock drift and taking actions.

Moreover, in our scheme, the playback rate will be constant for each media type, i.e. same T_P for all media streams. However, it is very easy to extent it to the case that media streams with different T_P.

In most schemes proposed in the literature, all slave streams need to maintain the inter-stream synchronization with the reference, i.e. the master stream, by using the feedback messages. Because of the similar reason of the large system deadtime due to end-to-end delays, it may not work properly. Therefore, in our scheme the inter-stream synchronization is obtained by setting scheduled playback times of each stream at the beginning of the playback. Without the occurrence of asynchrony, all media streams will be played at the scheduled playback time. Therefore there is not any skew occurs between the master stream and the slave streams. Any time asynchrony occurs in one media stream, the corresponding media unit cannot be played at the correct time. It is substituted with the duplication of last available media unit, i.e. the skew occurs. Hence, the measurements of the skew are directly related to the occurrences of asynchrony. For the particular case of human speech, skew between the related audio and video stream can be tolerated within the region from -80 (audio behind video) msec to +80 msec (audio ahead of video)[11], which is more than two playback periods. As long as we limit the occurrences of asynchrony of the single stream within the acceptable range, the problem of skew is solved automatically.

In summary, the proposed synchronization scheme has following advantages:

1) It exhibits a significantly reduced control scheme overhead as it eliminates the need for feedback messages;
2) It requires minimum buffer space for each media stream;
3) The minimum buffer requirements and the playback performance are independent of end-to-end delays, and, therefore;
4) It is inherently scalable for the evolving wireless environments.

2.3 *Client Buffer Determination*

By removing the feedback control, the critical issue of our scheme is the buffer configuration. The function of the client buffer is to compensate for network delays and jitters incurred during the transmission and processing of multiple streams, thus achieving the smooth playback to the application users.

With sufficient buffer size to compensate for buffer overflow, sufficient pre-buffered MMUs to compensate for buffer underflow, and a constant server sending rate $r_S=r_P=1/T_P$, applications can achieve constant playback rate without any interruption due to the underflow and the overflow, i.e. the smooth playback. The minimum buffer size is related to the occurrence of buffer overflow while the pre-buffer threshold number is related to the occurrence of buffer underflow.

Subsection 2.3.1 derives the per-buffered threshold number and the minimum required buffer size of any single stream with the assumption of bounded jitter. Subsection 2.3.2 looses the bounded jitter assumption.

2.3.1 *Buffer Determination for Single Stream with Bounded Jitter Assumption*

We denote the average network delay $E[d_i]$ by d_{avg}, where d_i is the network delay of the i^{th} media unit, then the network jitters are bounded by (d_{avg}+Δ^+) and (d_{avg}-Δ^-), where $\Delta^+ = max\{d_i\}-d_{avg}$, and $\Delta^- = d_{avg}-min\{d_i\}$.

Let us first derive the parameter *pre-buffer threshold* N_{th}, where N_{th} is the number of MMUs required to pre-buffer in order to avoid the occurrence of the underflow. The application will start the playback of the first MMU once the number of MMUs in the buffer reaches the threshold value N_{th}. Based on the knowledge of N_{th}, we will derive the minimum buffer size B_{min} in order to avoid the occurrence of the overflow.

To proceed, we first consider the restriction that no MMU can arrive earlier than any previous MMUs. In this case, all MMUs must arrive in order, which can be achieved by a connection-oriented protocol like TCP. However, due to the large overhead of the TCP protocol, the extremely large amount of multimedia data and the delay restriction of live video streams, the light-weighted connectionless real time protocols, such as RTP, are preferred. Therefore, it is necessary to remove the assumption that no MMU can arrive earlier than any previous MMUs.

We now remove the restriction that no MMU can arrive earlier than any previous MMUs. Let us consider the worst case that the underflow can occur in Fig.1. The Server started sending the first MMU at t_0. All pre-buffered MMUs arrived at the client buffer with the minimum delays, i.e. d_{avg}-Δ^-. The application starts playback of the first MMU at the time the N_{th}th MMU arrived, i.e. $t_1 = t_0+(N_{th}-1)T_P+d_{avg}-\Delta^-$. Then the next MMU experiences the maximum delay. Thereby, it arrives the client buffer at the time $t_2 = t_0+N_{th}T_P+d_{avg}+\Delta^+ = t_1+T_P+(\Delta^-+\Delta^+)$. If the application finishes the playback of all N_{th} MMUs before t_2, the underflow will occur. That is the worst case that the underflow may occur. Hence the pre-buffer threshold number can be calculated as

$$N_{th}T_P \geq T_P + (\Delta^- + \Delta^+);$$
$$\Rightarrow N_{th} = \left\lceil \frac{\Delta^- + \Delta^+}{T_P} \right\rceil + 1; \tag{1}$$

where $\lceil\ \rceil$ is the ceiling operator.

Next, let us consider the worst case that the overflow can occur with the pre-buffered N_{th} MMUs in Fig. 2. All pre-buffered MMUs arrived at the client buffer with the maximum delays, i.e. (d_{avg}+Δ^+). The application starts playback of the first MMU at the time the N_{th}th MMU arrived, i.e. $t_1 = t_0+(N_{th}-1)T_P+(d_{avg}+\Delta^+)$. Then all successive MMUs experience the minimum delays. Thereby, before the application finishes the playback of the first MMUs at time $t_2 = t_1+T_P$, the maximum number of MMUs that can arrive is calculated as

$$t_0 + (k + N_{th} - 1)T_P + d_{avg} - \Delta^- \leq t_2 = t_1 + T_P = t_0 + (N_{th} - 1)T_P + d_{avg} + \Delta^+ + T_P;$$
$$\Rightarrow k = \left\lfloor \frac{\Delta^- + \Delta^+}{T_P} \right\rfloor + 1;$$

where $\lfloor\ \rfloor$ is the flooring operator.

Therefore, excluding the MMU being played, the minimum required buffer size B_{min} to avoid overflow is

$$B_{\min} = N_{th} - 1 + k = \left\lceil \frac{\Delta^- + \Delta^+}{T_P} \right\rceil + \left\lfloor \frac{\Delta^- + \Delta^+}{T_P} \right\rfloor + 1. \tag{2}$$

We notice that removing the restriction that no MMU can arrive earlier than any previous MMUs does not affect the derivation of N_{th}. The only difference locates at the derivation of minimum buffer size to avoid the overflow. We again consider the worst case. The N_{th}th MMU experiences the maximum delay, which arrives at the client buffer after $(N_{th}+1)$th to $(N_{th}+k)$th MMUs. The maximum value of k attains when $(N_{th}+k)$th experiences the minimum delay. Thereby,

$$kT_P + d_{avg} - \Delta^- \leq d_{avg} + \Delta^+;$$
$$\Rightarrow k = \left\lfloor \frac{\Delta^- + \Delta^+}{T_P} \right\rfloor.$$

Therefore the required buffer size must be larger than $(N_{th}+k)$, which has the same value as that with the restriction.

In conclusion, if the network jitters are bounded by Δ^- and Δ^+, there is no need for any feedback control mechanism. If the number of pre-buffered media units is larger than N_{th}, no underflow occurs; if the buffer size is larger than B_{min}, no overflow occurs. A constant playback rate can be achieved by simply setting a constant server sending rate to the same value as the client playback rate, setting a buffer size larger than the calculated minimum required buffer size B_{min}, and starting the playback at the time the client buffer receives the N_{th}th MMU. Since neither underflow nor overflow occurs in this system, there is no information loss (skipping or duplicating MMUs) either. We can interpret this as follows. When the server sending rate is constant, the only factor causing the variation of the buffer levels is the network jitter. If the network jitters are bounded, the variation of the buffer levels is bounded too, and so the minimum buffer size can be determined.

2.3.2 Buffer Determination for Single Stream without Bounded Jitter Assumption

With the bounded jitter assumption and appropriate buffer setting, the occurrence of buffer underflow and overflow can be completely eliminated. However, in the real cellular network environments, the minimum network delay can be usually determined, while the maximum delay of transmission a particular media unit may not be bounded, which can occurs when the handover occurs. In that case, the jitter bounds are determined by 95% confidence level. We will take the network jitters as Gaussian distribution random variables.

As described in [10], the network delays $d_1, d_2, \ldots, d_n, \ldots$ are a sequence of independent, identically distributed random variables, each with mean d_{avg} and variance σ^2. Let

$$X_n = \frac{\sum_{i=1}^{n} d_i - E(\sum_{i=1}^{n} d_i)}{\sqrt{D(\sum_{i=1}^{n} d_i)}} = \frac{\sum_{i=1}^{n} d_i - nd_{avg}}{\sqrt{n}\sigma}; \quad n = 1,2,3\cdots \tag{3}$$

be the normalized sequence. Then for all real delay d, applying the central limit theorem, we obtain

$$\lim_{n\to\infty} P\{X_n \leq x\} = \lim_{n\to\infty} P\left\{ \frac{\sum_{i=1}^{n} d_i - nd_{avg}}{\sqrt{n}\sigma} \leq x \right\} = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}} dt. \tag{4}$$

In other words, X_n must converge to the standard Gaussian as $n \rightarrow \infty$. Therefore we can take the delay sequence as a Gaussian distributed random variable with mean d_{avg} and variance σ^2. Thereby the relationship between the 95% jitter bounds and the variance can be easily obtained as

$$\frac{1}{2}\left[erfc(\frac{\Delta^-}{\sqrt{2}\sigma}) + erfc(\frac{\Delta^+}{\sqrt{2}\sigma}) \right] \leq 5\%. \tag{5}$$

3 Simulation Results

In order to give a comparative performance of the proposed buffer configuration and control schemes, we have simulated a sample network with distributed media servers. End-to-end delays between the client and media servers are Gaussian distributed. We measured the number of *incorrect playbacks*, i.e. the number of playbacks that are not the corresponding media units, including the MMUs that arrive later than the playback deadline (*missing the playback deadline*) and the MMUs being *discarded*. The reason that we do not choose the number of occurrences of asynchrony (the buffer underflow and overflow) as the measurement is that one asynchrony may last for multiple playback periods. Besides, the occurrence of overflow may further cause underflow later at the playback time of the discarded MMUs. We do not want to count those MMUs as incorrect playback twice. Therefore the occurrence of underflow and overflow may not directly reflect the amount of information loss, i.e. the quality of service.

In our simulation, we use the delay parameters of UMTS (the Universal Mobile Telecommunications System), because it is the developing third generation system to provide global mobile seamless personalized multimedia communications and information services [2], and it is the system that converges the wireless services into the traditional wired networks, thereby the end-to-end delays in this system may be very large. In UMTS, the end-to-end transfer delay consists of three parts: the transfer delay of local bearer service, external bearer service, and UMTS bearer service, as shown in Fig. 3. The first two parts are simply the transfer delays in wired Internet, while the third part is the additional transfer delays experience in cellular mobile networks. The delay parameters of UMTS bearer service are provided in 3GPP (3rd Generation Partnership Project) standards TS 23.107 [1], which are shown in Table 1. The total end-to-end delay is the sum of UMTS bearer delay and traditional wired Internet delay including local bearer service and external bearer service.

The applications and maximum total end-to-end delay requirement of each type of QoS class are shown in Table 2. The maximum end-to-end delay can be used to decide the variance of delays of the experiments. We consider the first two QoS types, i.e. conversational and streaming, which represent live and stream multimedia presentation respectively.

We simulated 340,000 MMUs. For the playback period T_P = 33 ms, the duration of playback is longer than 3 hours, which is typically enough for video presentations. The jitter variance σ^2 is determined that 95% jitters located in the region between upper and lower bounds. The order of clock drift is 10^{-7} in our simulation.

From Table 3 and 4, we can see that the mean value of buffer levels is directly related to the value of pre-buffer threshold, and that given the constant server sending rate of media units, the variance of buffer levels is directly related to the network jitter variance. Therefore, given the network environments and the variance of the buffer level, one can achieve certain quality of playback service with certain setting of buffer size and the pre-buffer threshold.

The simulation results also verify that the buffer size is the major issue to avoid the occurrence of asynchrony. When the available buffer is sufficient, neither the buffer underflow nor the buffer overflow will occur. A smooth playback without any information loss is achieved. When the available buffer size is only half of the minimum buffer requirement, the information loss caused by buffer underflow and overflow is still in the acceptable range. However, if the buffer available buffer size is smaller than the half of the minimum buffer requirement, the information loss will increase exponentially. Consequently one cannot obtain a satisfied playback performance.

We also simulate two scenarios in which jitter bounds, variance, and buffer configurations remain constant. The only changing parameter is the average end-to-end delay. As the results shown in Fig. 4, the mean and variance of master stream buffer level are independent of the average end-to-end delay.

Finally, Fig. 5 shows that SMART scheme performs better than feedback control schemes. In this test, every scenario has a different value of the average delay d_{avg}, ranged from 275 ms to 2275 ms, with fixed jitter bounds $\Delta^+=\Delta^-=255$ ms. For each scenario, we randomly generate 100 arrival sequences with 20000 MMU arrivals each. We use the same arrival sequence to test different synchronization schemes, such as SMART, switch mode and 2nd order control schemes in [5,6], and the server side control scheme in [7]. In order to obtain comparative results, all schemes using the same buffer size B = 14 MMUs. Table 9 shows the average number of incorrect playbacks per sequence, and Fig. 10 shows the percentage of incorrect playbacks for different synchronization schemes.

4 Conclusion

In this paper we presented a novel synchronization scheme for multimedia applications that can provide high quality playback performance in emerging wireless Internet. The simulation results showed that the proposed scheme performs well, while significantly reducing communication overheads due to the control scheme by removing the feedback loop. Moreover, as its minimal buffer requirements are independent of the network delays, the proposed scheme is inherently scalable for the evolving wireless environments. From the simulation result, we also noticed that the quality of playback performance (the percentage of total incorrect playback) can be determined based on the buffer configurations.

References

[1] 3GPP standards TS. 23.107.

[2] J.F. Huber, D. Weiler, and H. Brand, “UMTS, the mobile multimedia vision for IMT 2000: a focus on standardization,” IEEE Communications Magazine, Vol. 38 Issue 9, pp. 129-136, Sept. 2000.

[3] H. Zhu, G.P. Zeng, I. Chlamtac, “Control scheme analysis for multimedia inter- and intra-stream synchronization,” IEEE International Conference on Communications 2003 (ICC’03), Vol. 1, pp. 7-11, Anchorage, Alaska, May 10-16, 2003.

[4] S. Ramanathan and P. V. Rangan, “Adaptive feedback techniques for synchronized multimedia retrieval over integrated networks,” IEEE/ACM Trans. Networking 1, 2, pp. 246-260, 1993.

[5] A. Hać and C.X. Xue, “Synchronization in multimedia data retrieval,” International Journal of Network Management, Vol. 7, pp. 33-62, 1997.

[6] A. Hać and C.X. Xue, “Buffer control scheme in multimedia synchronization,” Performance, Computing, and Communications Conference, 1997(IPCCC 1997), IEEE International, pp. 516-522, 1997.

[7] E. Biersack and W. Geyer, "Synchronization delivery and play-out of distributed stored multimedia streams,” ACM/Springer-Verlag: Multimedia Systems, Vol. 7, pp. 70-90, 1999.

[8] A. Boukerche, S. Hong and T. Jacob, “MoSync: a synchronization scheme for cellular wireless and mobile multimedia systems,” Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 2001. Proceedings. Ninth International Symposium on, pp. 89-96, 2001.

[9] A. Boukerche, S. Hong and T. Jacob, “A distributed synchronization scheme for multimedia streams in mobile systems: proof and correctness,” Local Computer Networks, 2001. Proceedings. LCN 2001. 26th Annual IEEE Conference, pp. 638-645, 2001.

[10] Y. Xu, Y. L. Chang, Z.J. Liu, "Buffer design for p_qos in multimedia synchronization systems," Info-tech and Info-net, 2001. Proceedings. ICII 2001 - Beijing. 2001 International Conferences, Vol. 2, pp. 486-491, 2001.

[11] Y. Xu, Y. L. Chang, Z.J. Liu, "Calculation and analysis of compensation buffer size in multimedia systems," IEEE Communications Letters, Vol. 5, No. 8, pp. 355-357, August 2001.

[12] S.T.S. Chia and W.C.Y Lee, "A synchronized radio system without stable clock sources," IEEE Personal Communications, Vol. 8, Issue 2, pp. 45-50, April 2001.

[13] S.N. Fabri and A.M. Kondoz, "Provision of streaming media services over mobile networks," 3G Mobile Communication Technologies, 2001. Second International Conference on (Conf. Publ. No. 477), pp. 104-108, 2001.

[14] L. Zhang, Z. Liu and C. H. Xia, "Clock Synchronization Algorithms for Network Measurements," INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, Volume: 1, pp. 160-169, 2002.

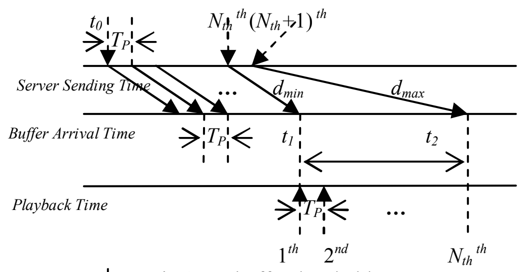

Fig.1 Pre-buffer threshold N_{th}

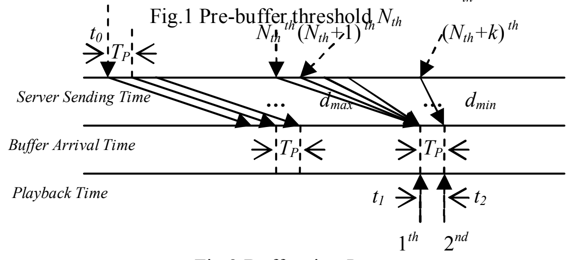

Fig.2 Buffer size *B*

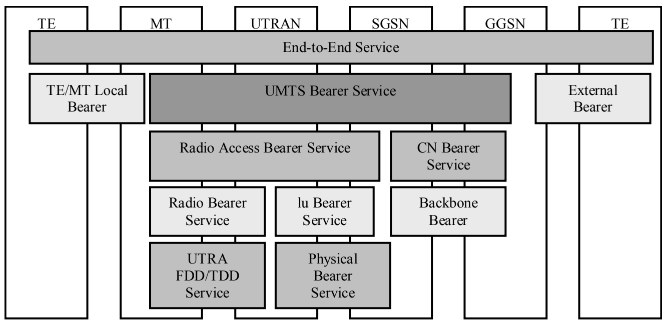

Fig. 3. UMTS QoS architecture

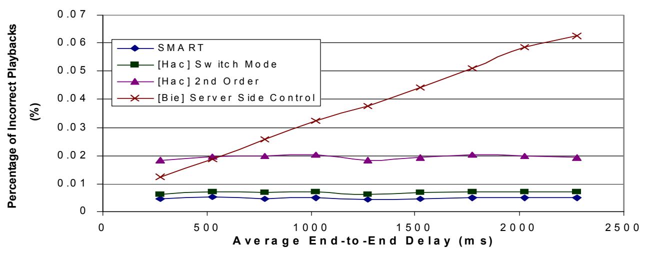

Fig. 5. Percentage of incorrect playbacks vs. average end-to-end delay (Fixed jitter bounds $\Delta^+=\Delta^-=225$ ms, B=14 mmu, N_{th}=7 mmu).

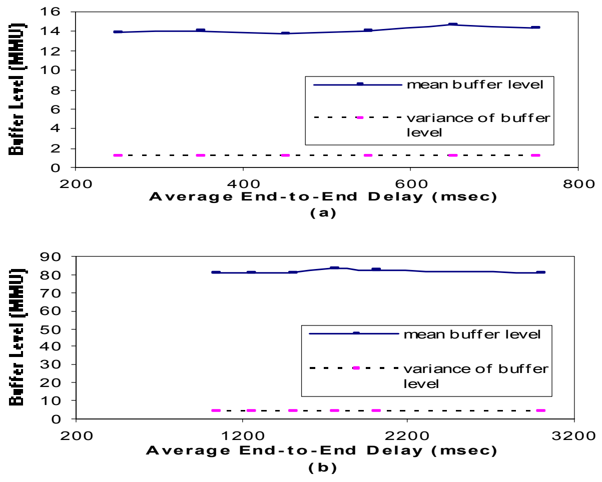

Fig. 4. Buffer level (master stream) vs. average end-to-end delay
(a) Jitter bounds $\Delta^+=\Delta^-=196$ ms, jitter variance $\sigma=90$ ms, B=24 mmu, N_{th}=13 mmu.
(b) Jitter bounds $\Delta^+=\Delta^-=980$ ms, jitter variance $\sigma=500$ ms, B=120 mmu, N_{th}=61 mmu.

TABLE 1. UMTS DELAY PARAMETERS

Bearer\QoS Class	**Conversational**	**Streaming**	**Interactive**	**Background**
UMTS Bearer (RAB+CN Bearer)	150 ms	250 ms	400 ms	1000 ms (guess)
Radio Access Bearer (RAB) (UTRAN+lu)	120 ms	200ms(80%of UMT bearer)	320ms(80%of UMT bearer)	800ms(80%of UMT bearer)
Core Network Bearer (CN) from MSC or SGSN to GGSN	30 ms	50 ms	80 ms	200 ms
lu Bearer	24 ms (20% of RAB)	40 ms (20% of RAB)	64 ms (20% of RAB)	160 ms (20% RAB)
Radio Bearer	96 ms (80% of RAB)	160 ms (80% of RAB)	256 ms (80% of RAB)	640 ms (80% of RAB)

TABLE 2. APPLICATIONS AND QOS REQUIREMENTS

	Conversational	**Streaming**	**Interactive**	**Background**
Error tolerant Applications	Conversational voice and video	Streaming audio and video	Voice messaging	Fax
Error intolerant Applications	Telnet, interactive games	FTP, still image, paging	E-commerce, web browsing	Email arrival notification
End-to-end Delay	<< 1 sec	< 10 sec	Approx. 1 sec	> 10 sec

TABLE 3. CASE 1. CONVERSATIONAL CLASS

Average delay d_{avg} = 200 ms, Minimum delay (d_{avg}-Δ^-) = 50 ms, Maximum delay (d_{avg}+Δ^+) = 500 ms, Jitter variance σ= 90 ms.	Sufficient buffer space = minimum buffer requiremnt	Insufficient buffer space = 50% of min. requirement	Insufficient buffer space = 40% of min. requirement
Available Buffer Size *B (MMU)*	28	14	12
Pre-buffer Threshold N_{th}*(MMU)*	15	8	6
Buffer Underflow (#)	0	0	1
Buffer Overflow (#)	0	72	37
Miss Deadline Incorrect *(MMU)*	0	2095	4291
Discard Incorrect *(MMU)*	0	72	39
Total Incorrect Playback (%)	0	0.6374%	1.2735%
Mean Buffer Level *(MMU)*	16.61319	8.815618	7.046079
Buffer Level Variance *(MMU)*	1.326044	1.221226	1.224636

TABLE 4. CASE 2. STREAMING CLASS

Average delay d_{avg} = 600 ms, Minimum delay (d_{avg}-Δ^-) = 50 ms, Maximum delay (d_{avg}+Δ^+) = 1800 ms, Jitter variance σ= 334 ms.	Sufficient buffer space = minimum buffer requiremnt	Insufficient buffer space = 50% of min. requirement	Insufficient buffer space = 40% of min. requirement
Available Buffer Size *B (MMU)*	108	54	44
Pre-buffer Threshold N_{th}*(MMU)*	55	23	22
Buffer Underflow (#)	0	0	19
Buffer Overflow (#)	0	0	164
Miss Deadline Incorrect *(MMU)*	0	533	900
Discard Incorrect *(MMU)*	0	0	171
Total Incorrect Playback (%)	0	0.1568%	0.315%
Mean Buffer Level *(MMU)*	68.84419	32.80899	31.63529
Buffer Level Variance *(MMU)*	3.222303	2.473644	2.502061