Empirical Evaluation of Streamed Online Gaming over WiMAX

Esa Piri, Matti Hirvonen, and Jukka-Pekka Laulajainen

VTT Technical Research Centre of Finland
Kaitoväylä 1, FI-90571, Oulu, Finland
firstname.lastname@vtt.fi

Abstract. Online gaming is one of the main Internet services holding potential for a significant growth. Currently, much attention is paid on studying how high-quality 3D games with varying resource demands can be played over the Internet streamed real-time from a remote game server. This enables end-users to play high-quality games without need for having very powerful game machines as most of the processing is performed already in the game server. This type of gaming is sensitive to network conditions, especially, to end-to-end delays of both uplink and downlink. In this study, we evaluate the quality of streamed online gaming over fixed WiMAX through empirical evidence. We evaluate the gaming quality with various background traffic loads and types in scenarios where a WiMAX link is employed as a backhaul connection. In addition, we assess the importance of various WiMAX QoS scheduling schemes to keep the gaming experience high despite the congestion in the serving base station. We find that the quality of streamed gaming is very sensitive to delays and already the characteristic transmission latencies of WiMAX are near the edge of a smooth gaming experience. With heavy traffic loads, inflicting high delays, the gaming experience faces a radical degradation, which can be clearly mitigated by using scheduling schemes privileging the game traffic.

Keywords: online gaming, game streaming, WiMAX, IEEE 802.16, measurements, QoS.

1 Introduction

Online gaming rapidly became a popular Internet service since the Internet started to become general. With a rough division, there are two types of online gaming: multiplayer mode where players play with peer-players a game installed on their own machines; and player plays a streamed game from a remote gaming server. Moreover, a hybrid version of the precedings is possible. The second option is so far used with relatively simple games, usually played using a web browser and not requiring any installation on end-users' game machines. Games demanding much resources still typically require some sort of installation on the end-user's machine. However, the current research on online gaming is much directed to enable this type of gaming also with more resource-demanding games.

Overall, gaming is purely an on-demand service, which with the latest games require lots of resources such as disk space and processing power from the end-user's game machine when installed on them. When a game is streamed from a remote game server to a player's game machine in real-time, it allows moving the disk space and most of the processing from the player's machine to the remote game server. In addition, other benefits are, for example, easier implementation of pay-per-use mechanism and independence of the gaming machine, that is, player can continue playing the same game with other machine than the one it was started with. For the server side, cloud computing brings several attractive features to improve its scalability. It efficiently allows dynamic resource reservation and establishment for individual players and games.

The capability and characteristics of the players' network connections become more significant with streamed gaming in order to sustain the gaming experience of end-users as high as the game was played purely on the local machine only. In this study, we empirically measure the quality-of-service (QoS) of online gaming from the remote game server over a fixed WiMAX testbed based on the IEEE 802.16d standard [1]. Our measurement scenarios relate to cases where the wireless broadband link is used as a backhaul connection to the Internet. Typical cases are femtocell and connecting of rural areas where wired connections, such as fiber, are too expensive to be deployed. We also expand this scenario with a competing link scenario, where two subscriber stations (SSs) compete for the shared resources of a base station (BS).

The game traffic is bi-directional, where player's commands travel in the uplink (UL) and the game data (graphics commands and sound) in the downlink (DL). Moreover, one TCP control flow to both directions is active. As the game traffic generally employs TCP, also TCP acknowledgment messages employ both uplink and downlink. Overall, the game performance is sensitive to delays and jitters of the network traffic. Moreover, the delay that counts with the game traffic and TCP in overall is the round-trip delay. Thus, both uplink and downlink must perform well with respect to the transmission latencies for a good gaming experience. When approaching the maximum capacity of a WiMAX link, the transmission latencies tend to increase significantly. Thus, in addition to a Best Effort (BE) scheduling scheme, we also evaluate different WiMAX-specific QoS service class schemes to keep the game traffic quality (both uplink and downlink) high despite heavy background traffic.

The rest of this study is organized as follows. Section 2 relates our study to previously published results. Our testbed, streamed online gaming, and the measurement setup adopted in our study are introduced in Section 3. Then, Section 4 presents our measurement results. Finally, Section 5 concludes this paper.

2 Related Work

Plenty of studies have been carried out to investigate the effect of network delay to the performance of online games, such as [2,3,4]. However, most of them are measuring game performance in multiplayer scenarios, where the actual game is installed on the player's game machine. This type of gaming is much more insensitive to delays than the type we are using. For example, delays of 500 milliseconds (ms) considered as acceptable with many games in [2] is far from tolerable with any streamed game. Jurgelionis et al. [5] have evaluated the streaming of games similar to the ones we are using over wireless LAN (WLAN), but the usual uplink and downlink latencies in uncongested WLAN are constantly clearly below 10 ms, which is not the case with fixed WiMAX.

To the best of our knowledge, this is the first empirical evaluation of this type of online gaming over WiMAX. Wang et al. have experimented World of Warcraft gaming over mobile WiMAX [6]. Although the physical layer of mobile WiMAX differs from fixed WiMAX, they use mobility scenarios, and the gaming type is different (network is used for multiplaying), they noted that WiMAX can provide acceptable performance for online multiplaying most of the time. However, they also remark that BE scheduling is not adequately suitable for the game traffic when the serving BS is stressed also by other traffic flows. They recommend using Real-time Polling Service (rtPS) type of QoS service class to prioritize delay-sensitive game traffic, which is evaluated in our experiments.

Wodarz et al. [7] have measured QoS of a fixed WiMAX with time division duplex (TDD) duplexing scheme. They observed uplink to suffer from significant latency growth with the BE QoS scheduling when the link capacity was reached. However, by using service-specific QoS schemes, such as rtPS, they perceived the uplink latency of the prioritized traffic to stay below 10 ms although the link was congested. Downlink latencies remained around 10 ms independently of the traffic load. We also observe similar results. With frequency division duplex (FDD), which is the case in our measurements, Pentikousis et al. [8] have observed that the latencies of downlink grow sharper than those of uplink when the link capacity is reached, constantly staying below 100 ms in uplink. Although we keep the modulation fixed and signal strength strong, J. De Bruyne et al. [9] have measured the effect of Carrier to Interference-plus-Noise Ratio (CINR) to the latencies in FDD fixed WiMAX using low bitrate traffic flows. In addition to an observation that the latency is the lower the higher the CINR value is, they also found higher latencies and significantly bigger latency variation in the uplink than downlink. Nevertheless, we do not observe much variation in jitters between uplink and downlink with the same CINR values ($\sim$28 dB). Also in [10], the jitters in uplink and downlink were minor with moderate traffic loads, however, with strong signal strength. Measurements of [9] are conducted using half-duplex FDD whereas we use full-duplex FDD, which was the case also in [8] and [10].

3 Methodology

Our experimental testbed, located at VTT's Converging Networks Laboratory, is illustrated in Figure 1. It comprises an Airspan MicroMAX-SoC BS operating in the 3.5 GHz frequency band, two Airspan SSs connected to the BS, and several

PC machines symmetrically connected to the BS and the SSs sides with Gigabit Ethernet. The WiMAX equipment is WiMAX Forum certified. The traffic generator machines act as background traffic sources and sinks while the game client and server stream game traffic between them. The system clocks of the game client and game server are synchronized using Global Positioning System (GPS) in order to enable accurate one-way end-to-end delay measurements. Table 1 lists the main parameters of the WiMAX testbed used in our experiments. The measurements were conducted in a laboratory environment with relatively stable and strong signal conditions and short line-of-sight distances. Automatic Repeat-reQuest (ARQ) is not enabled in our measurements.

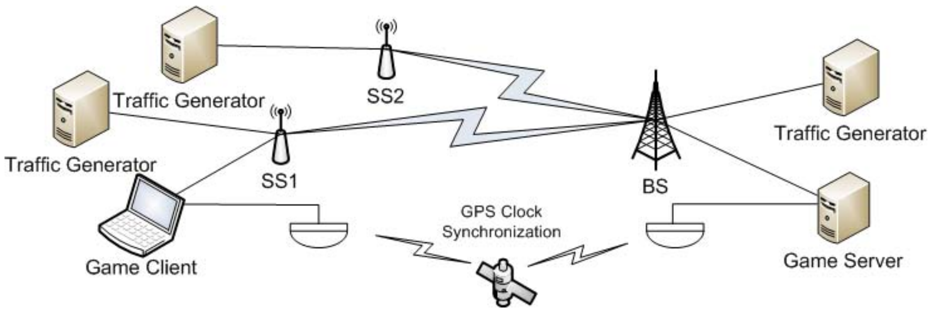

Fig. 1. Schematic of the WiMAX testbed

3.1 Streamed Online Gaming

Network games can be categorized into two groups: multiplayer games using network connection to play together with other players and fully network based games that execute on a distant server and use network connection to stream the game content to the client (Gaming on Demand). The first solution is based on running the game software on the local end device setting high requirements for the computing and processing power of the device. Devices used for playing these games are typically specific video game consoles (e.g. Microsoft Xbox 360, Nintendo Wii, Sony PlayStation 3) or high-end PCs. The latter group targets on decreasing the end device requirements by moving majority of game processing to a separate game server. This way high quality games can be played with cheap low-end devices such as set-top-boxes or mobile devices, or the functionality could be even built in a TV set. Our study focuses on Gaming on Demand solutions based on remote execution and the traditional multiplayer networked games are left out of scope for the rest of the paper.

The most common way to implement a Gaming on Demand system is to use video streaming technology for transmitting the game output to the user. There are several commercial systems based on video streaming, e.g. OnLive, StreamMyGame, Playcast, G-cluster, and Gaikai. All these run the game application, render the game graphics, and encode them as a video stream at the

Table 1. Testbed parameters

BS: Airspan MicroMAX-SoC	
Frequency band	3.5 GHz
Channel bandwidth	3.5 MHz
PHY	IEEE 802.16d, FFT 256, OFDM FDD
Tx power	1.0 dBm
SS1: Airspan ProST	
Modulation (UL and DL)	64 QAM FEC: 3/4
Avg. SNR	33 dB
Distance from BS	10 m
SS2: Airspan EasyST	
Modulation (UL and DL)	64 QAM FEC: 3/4
Avg. SNR	32 dB
Distance from BS	5 m

server and stream that to the client, which only needs to present the video on the screen, capture the user commands and send them back to the game server. The downside of using video streaming is that the requirements for the server-side computing power and the network bandwidth need of a video stream with high-definition resolution are very high. In this paper, we concentrate on an alternative method being developed in a European research project called Games@Large [11].

In Games@Large, a new 3D streaming technology is used instead of video streaming. Unlike in the video streaming based solutions the game graphics are not rendered on the server, but the DirectX or OpenGL commands used for building the graphics are captured and transferred to the client device using a specific pre-rendering streaming protocol. Client device receives this stream, and feeds the graphics commands to the local graphics processing unit for rendering the image to the client display. This method results to an optimal game quality since no lossy compression is involved and the latency of video encoding/decoding is omitted. On the other hand, the client device needs to have a graphics processor capable of rendering the game graphics. This, however, is not a critical limitation since more and more set-top-boxes and mobile devices are equipped with OpenGL ES 2.0 supported by the Games@Large system.

The profile of network traffic in Games@Large system depends heavily on the game being streamed. As the rendering of the graphics happens at the end-device, the screen resolution does not have any effect on the network traffic, but the traffic profile is determined by the game contents and user's position in the game. The data being transmitted includes basic graphics commands and matrix and buffer data that typically are small in size. Depending on the game, the amount of commands needed to render a single frame ranges from some tens to several hundreds and the amount of data to be transmitted is typically in the range of tens of kilobytes per frame.

We used two games in our measurements: Turtix (Figure 2) and Race Cars: Extreme Rally (Figure 3). First one is a platform game while the latter is a racing game. Both of them require low response times from the game to be able to control the play with high precision.

Fig. 2. Turtix

Fig. 3. Race Cars: Extreme Rally

3.2 Traffic Generation and Analysis

We measured the QoS of the gaming in terms of end-to-end delay, jitter, game frame rate, and also subjective playability. Because the game traffic uses TCP, packet loss is not of our major interest but its impact on delays is. The traffic parameters of the game traffic were measured using QoSMeT [12] tool. It allowed us to measure the traffic from multiple points, from the game server and the game client. Multi-point measurements require control messaging between the QoSMeT tools. Although the control traffic amount is negligible compared to the game traffic, we used a separate Ethernet link for the control messaging in order to avoid it to interfere with the game traffic in the wireless link. In addition to QoSMeT, we also followed the frame rate of the games.

The gaming quality was measured by employing two types of background traffic with different loads. In order to progressively fill the link capacity, we injected background traffic into the wireless link. We synthetically emulated VoIP traffic and peer-to-peer (P2P) file downloading. As a VoIP codec, we opted to use ITU-T G.729.1 [13], which is an extension to the narrowband codec ITU-T G.729 [14] providing scalable wideband speech and audio compression. G.729.1 utilizes layered coding, meaning only the core layer is needed for a successful decoding and extension layers improve the quality only at the expense of bitrate grow. Twelve possible bitrates of G.729.1 range from 8kb/s to 32 kb/s. However, we kept the codec operation mode fixed, with four-layer coding producing application-layer bitrate of 16 kb/s (throughput 32 kb/s). The packet size including RTP/UDP/IP protocol headers and with the employed operation mode is 80 bytes. The VoIP traffic was generated with a traffic generator implemented in Perl. It adapts the inter-packet interval errors incurred by the inaccuracy of the *sleep()* function due to lack of real-time operating system in our measurements. As studied in [15], software traffic generators often fail to generate traffic defined. The main reasons are lack of processing resources and the inaccuracy of operating system's process scheduler for *select()* and *sleep()* calls commonly used in the intervals of sequential packets. With the employed traffic generator, the inaccuracy of the VoIP traffic in our measurements was at maximum 1% with the heaviest load (235 parallel VoIP flows).

TCP traffic emulating P2P was generated by using Jugi's Traffic Generator (JTG) [16]. Traffic was sent with full rate and with the maximum transfer unit (MTU) of 1500 bytes. In order to inject several TCP streams into the link, we used shell scripts to progressively add new JTG processes. Like the game traffic also background traffic was measured with the QoSMeT tool, however, only from the traffic sink sides.

We capitalized on the feature of a service flow prioritization of the WiMAX BS in the downlink illustrated in Figure 4. Incoming IP packets are classified and assigned to different service flows with different priorities. Packets can be classified based on their source and destination MAC addresses, source and destination IP addresses, and source and destination UDP and TCP port numbers. In the prioritization measurements, we defined two downlink service flows where the game traffic was stamped as higher priority. The packet scheduler is fair and not all resources are ever given to the higher priority traffic. Service flow definitions are SS-specific and different rules can be assigned to different SSs.

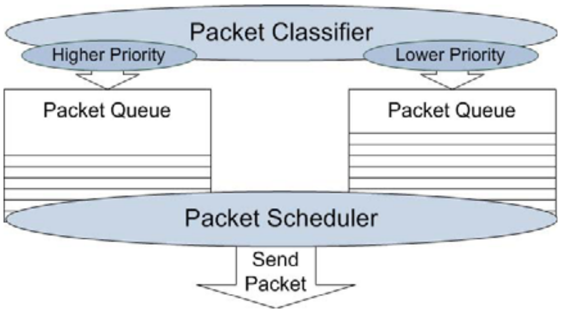

Fig. 4. Service flow classification and scheduling

We opted to evaluate the Real-time Polling Service (rtPS) scheduling service to ensure the required bandwidth of the game traffic in the uplink, that is, user commands and TCP acknowledgments for the graphics commands. The rtPS supports real-time service flows with variable packet sizes and packet intervals to meet the flows' real-time bandwidth needs. Service flow is allowed to request uplink bandwidth at regular intervals. Although rtPS is optimal for higher bitrate real-time traffic flows, such as video streaming, it is suitable also for ensuring variable bitrate streams of small bitrates. Unsolicited Grant

Service (UGS) is designed for constant bitrate traffic, such as VoIP without voice suppression, which makes it less appropriate scheduling service for this usage.

Table 2 lists the QoS parameters for the rtPS service flows. We did not set strict thresholds for the parameters. The obligatory parameters for the rtPS are Minimum Reserved Traffic rate, Maximum Sustained Traffic rate, and Maximum Latency [1]. However, we defined a larger set of parameters.

Table 2. Real-time Polling Service parameters

Parameter	Value
Maximum Sustained Rate	500 kb/s
Maximum Traffic Burst	100 kB
Minimum Reserved Rate	150 kb/s
Minimum Tolerable Rate	100 kb/s
Tolerated Jitter	10 ms
Maximum Latency	30 ms
Scheduling Poll Period	30 ms

4 Results

We used both TCP and UDP traffic as background traffic to emulate P2P and VoIP traffic, respectively. The number of flows was progressively, every 30 s, increased in order to reach the link capacity and also to exceed it. Full rate TCP flow number is started with one flow and increased by 50 flows (1, 50, 100, etc.) until 450 flows is met. In the downlink measurements, the VoIP flow number ranged from 190 to 235, in steps of five flows. In the uplink, the VoIP flow amount is started with 105 and incremented up to 150 flows, also using steps of five flows. Although VoIP call always comprises a bi-directional traffic flow, in this study we separately quantify the effect of the downlink and uplink on the game quality.

In the experiments, we played two different games and the results are assessed with respect to the quality of the game traffic, which is best shown as game frame rate adapted by the game server. Each measurement run lasted 300 seconds and was repeated two times. The average value of the repetitions is shown.

Before we conducted the measurements with the game traffic, we quantified the maximum uplink and downlink capacities for the link between SS1 and BS using UDP traffic with the MTU of 1500 bytes. The attained throughputs of the baseline measurements were 8.560 Mb/s and 11.280 Mb/s with negligible losses (<0.1%) for the uplink and downlink, respectively.

4.1 WiMAX Link as a Backhaul

In this scenario, only SS1 is employed and all traffic goes through one WiMAX link. Figure 5 illustrates the game frame rate for Turtix game. By playing the game over an empty link with the BE scheduling, the frame rate stays on a relatively stable level averaging 17 frames per second (fps). This frame rate corresponds to an average bitrate of 1.3 Mb/s. The playability significantly degrades when background traffic is injected into the link. With TCP, the frame rate slumps to the level near zero fps already with one full rate flow, which means the game is totally unplayable. With VoIP, the game stays playable until the downlink capacity is clearly exceeded with 205 parallel VoIP flows at the point of 90 s. With that number of VoIP flows, the loss rate of VoIP traffic soars to 4%. From 205 VoIP flows on, the VoIP loss rate increases by approximately 2% with the each flow amount increment. The spikes in the frame rates with the BE scheduling and VoIP background traffic are caused by the VoIP traffic generator, which before adding VoIP flows pauses the traffic generation for 0.2 s in order to send the results with the current number of VoIP flows to the sink over an uncongested link. On the other hand, although the breaks in the background traffic are that short, the results indicate that the game adaptation logic rapidly recovers the frame rate to a higher level.

By giving higher priority to the game traffic, the game sustains playable despite the amount of background traffic. With TCP, the average frame rate is only 1.4 fps lower than that with the plain game. With VoIP, the average frame rate drops by 3.8 fps compared to the game over the empty link, by being on the edge of smooth playability. On average, TCP and VoIP background traffic lower the downlink bitrate of the higher priority game traffic by 100 kb/s and 230 kb/s, respectively.

When the prioritization was enabled, nearly eight times more background IP packets were successfully transmitted over the link with the VoIP background traffic than with TCP. Without prioritization, the difference between the received background traffic packets of VoIP and TCP was 15% smaller. This implies the fairness of the scheduling algorithm. It gave proportionally more resources to the VoIP traffic consisting of large number of small IP packets than to the TCP traffic of less packets. This is also one explanation to the lower frame rate of the game with VoIP. Another is the performance decline of the wireless link with large amount of small packets introduced below. We also conducted measurements to gauge the fairness of the downlink scheduling algorithm with two traffic flows (TCP and UDP with MTU of 1500 bytes) of different priorities. With full rate TCP, 5% of the bandwidth was given to the lower priority flow. By sending UDP flows, we observed that 9% of the total downlink capacity was given to the lower priority flow.

The significantly smaller packet size and the lower inter-packet interval with VoIP compared to the TCP, packets of length 1500 bytes, substantially decreases the link capacity, as observed, for example, in [8]. In our measurements, the downlink capacity abated to 9 Mb/s (240-270 s) at maximum with 230 parallel VoIP flows and the game traffic of 300 kb/s when all traffic was equally treated. Interestingly, the downlink capacity increased by 0.3 Mb/s with the VoIP background traffic when the prioritization was enabled. Moreover, the VoIP packet loss rate with 230 VoIP flows was 1.4% with prioritization enabled while a loss rate of 1.7% was perceived already with 215 VoIP flows without prioritization. With TCP, similar behavior was not observed.

Although the frame rate slightly drops from the plain game case, the standard deviation of the frame rates is 0.3 fps and 0.4 fps lower with prioritized TCP and VoIP background traffic than with the plain game, respectively.

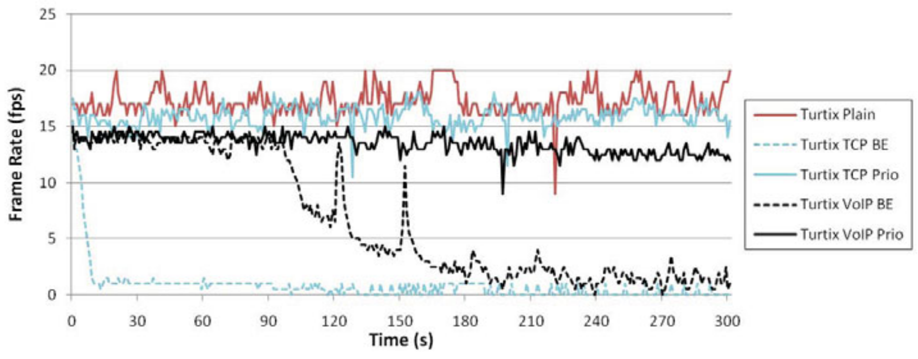

Fig. 5. Turtix frame rates with downlink background traffic

Figure 6 presents the achieved frame rates for the Race Cars game with the same measurement cases as with Turtix. The observed behavior with different traffic loads and scheduling schemes is similar to Turtix. However, with plain Race Cars, the game throughput is on average 300 kb/s lower than with Turtix (1.3 Mb/s), which leads to 15% lower frame rate. This implies stricter demands for the delays. Due to the lower throughput, Race Cars stays playable 50 s longer than Turtix with the VoIP background traffic, slumping with 210 VoIP flows at the point of 140 s.

Although the games are playable with frame rates above 10 fps, as threshold for smooth gaming we observed approximately 13 fps, we also experimented the same games over a short Gigabit Ethernet connection. The frame rates ranged between 120-150 and 90-100 most of the time with Turtix and Race Cars, respectively. The throughputs with these frame rates varied approximately between 5-9 Mb/s. Throughputs of that order of magnitude can be attained also with WiMAX. However, the end-to-end delays have a clear impact on the achieved game frame rate.

Figure 7 shows the one-way delays of Turtix in the downlink. With plain game, the delays average 13.6 ms. When TCP traffic is injected into the link, the delays rapidly face 1 s threshold. For example, in the point of 5 s, about 5500 TCP background traffic packets were already transmitted over the link whereas only 360 game traffic packets were received by the game client so far. As the link capacity left after the game traffic is ten-fold compared to the game bitrate, the proportion of the number of background traffic packets trying to fill the link is substantially bigger than the game traffic packets and eventually diminishes the game traffic flow due to increasing delays and frame rate adaptation in the game server. With VoIP, the delays start to heavily increase after injecting 205 VoIP flows into the link by saturating to 400 ms. Jitters stay on average on 2 ms level with the standard deviation of 0.5 ms with all the background traffic cases, except TCP with the BE scheduling.

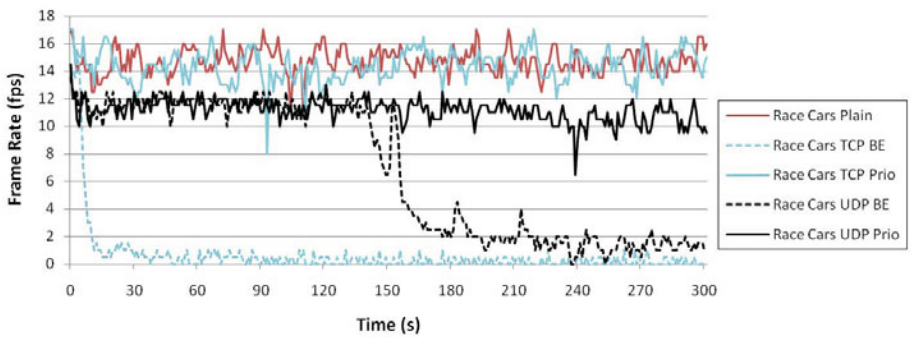

Fig. 6. Race Cars frame rates with downlink background traffic

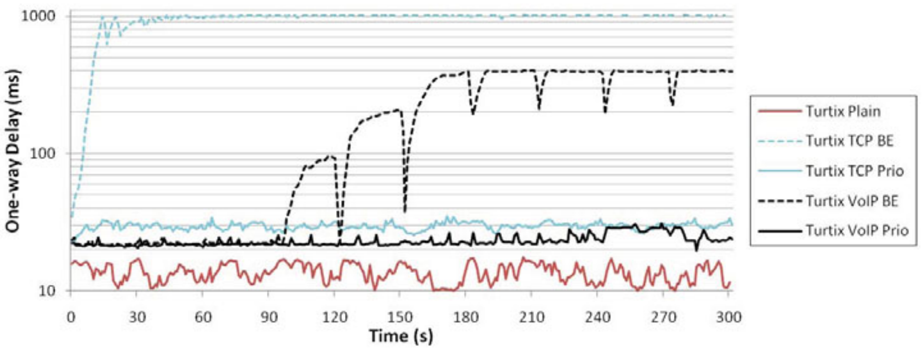

Fig. 7. Turtix one-way downlink delays with downlink background traffic

By prioritizing traffic, the downlink delays of Turtix remain at approximately 23 ms and 29 ms with the VoIP and TCP background traffic, respectively, regardless of the background load. However, already the small delay increase affects the smoothness of gaming and the frame rate as indicated above. Interestingly, although we observe higher delays with the TCP background traffic, the frame rate stayed at higher level than with the VoIP traffic. Thus, the lower frame rate with the VoIP traffic must be due to the capacity decrease attributed to the small packet sizes of VoIP flows.

In the uplink, we noticed interesting behavior in delays, shown in Figure 8. While the plain Turtix game traffic constantly yields on average 34 ms delays in the uplink, by injecting background traffic into the uplink we managed to decrease the one-way delays down to 15 ms when staying below the link capacity. With VoIP background traffic, the delays stay on a 20 ms lower level

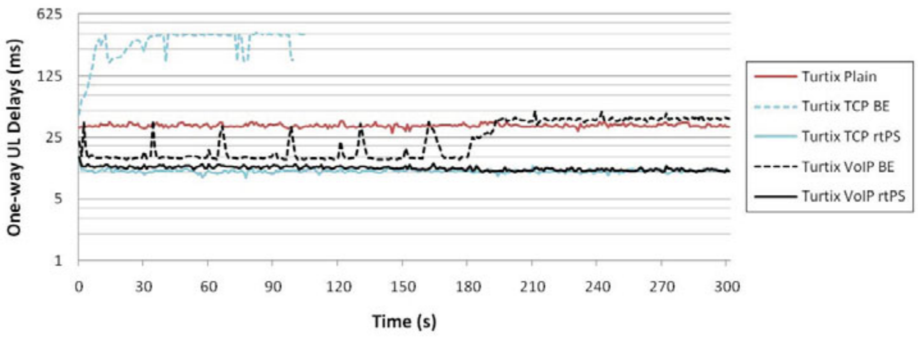

Fig. 8. Turtix one-way uplink delays with uplink background traffic

than with plain game until the capacity is exceeded with 135 VoIP flows (from 180 s on). SS needs to request bandwidth from BS before it is granted to send data. It seems that SS can faster request the bandwidth when a larger number of packets/data needs to be transmitted and this way lower the access latencies. Another explanation could be that BS more easily grants bandwidth when throughput is in orders of megabits. As in the downlink, full rate TCP degrades the game by rapidly raising the delays to an untolerable level, 300 ms, until the game crashed. By ensuring uplink capacity to the game traffic (user controls and TCP acknowledges for graphics) with the rtPS scheduling, we managed to keep the uplink delays constantly at 10 ms. Nevertheless, as found also in [7], rtPS needs trigger traffic, other than rtPS traffic, to meet the defined real-time restrictions in the link. By sending solely the game traffic with the rtPS support, the game lagged and the uplink delays were around 35-40 ms although 30 ms was defined as tolerable.

The lower uplink delays result in notably better frame rates, as can be seen from Figure 9. The low delays in the uplink were also subjectively experienced as faster response to the controls and smoother graphics updates. With the

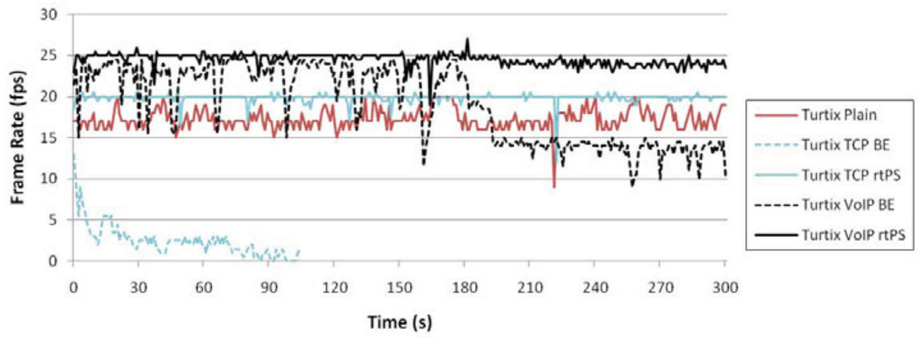

Fig. 9. Turtix frame rates with uplink background traffic

TCP background traffic, the average frame rate is 2.5 fps higher than with plain Turtix. Clearly the best results are attained with UDP background traffic and rtPS when the average frame rate exceeds 24 fps, which is 42% higher than with plain Turtix.

4.2 Competing Links

In addition to the backhaul link, in this scenario BS serves also SS2. The game traffic stresses the link between SS1 and BS and the link between SS2 and BS is employed by the background traffic. For the traffic of both SSs, the BE scheduling scheme is used. Figure 10 illustrates game frame rates for different measurement cases. For clarity, only the results of Turtix without background traffic are shown in the figure. Although clearly the best results with respect to the frame rates are attained with plain games, the background traffic cases do not degrade the playing experience but both games remain playable although the background traffic is progressively increased, also above the BS downlink capacity. As in the one-link scenario, the frame rates of Turtix are higher than the ones attained with Race Cars. With TCP and VoIP background traffic we perceive on average 14 and 13 fps frame rates in Turtix, respectively. The frame rates with Race Cars are 11 fps with the both cases. Although frame rate variation can be seen in Figure 10, the standard deviation of frame rate is around 1 fps with all the cases. Noteworthy is that the frame rate is not constant although link conditions remain similar but varies depending on the user's activity and graphic updates. However, we played same game levels and attempted to keep the playing activity as similar as possible between the measurement runs.

The observed differences in the downlink delays with different background traffic are observed minor. With Turtix, the average delay of 24 ms is achieved with both the TCP and VoIP background traffic. With Race Cars and TCP background traffic, the delays average 25 ms while with the VoIP background traffic an average delay of 4 ms lower than that with TCP is experienced. Thus, due to the lower throughput with Race Cars, the competing link sustains 10 VoIP flows more (215 flows between 150-180 s) with a loss rate of 1% than when Turtix is played. As there is no significant difference in the downlink delays, Race Cars is again detected as more demanding in terms of delays. No background traffic is sent to the uplink and, thus, the uplink delays are of same magnitude in all cases.

The TCP background traffic attains a cumulative throughput of 9.1 Mb/s and 9.3 Mb/s with Turtix and Race Cars, respectively. At the same time, Turtix game achieves a mean throughput of 1 Mb/s and Race Cars 790 kb/s. VoIP with loss rate of 1% translates into cumulative throughput of 7.5 Mb/s with both games saturating to approximately 7.7 Mb/s with higher loss rates. Game traffic throughputs with the VoIP background traffic are close to the ones with the TCP traffic.

Overall, the resource sharing among multiple SSs of same priority seems to be fair and it satisfies the bandwidth requirements of the game traffic despite the substantially heavier load in the competing link. However, the results of the scheduling schemes employed in the measurements presented in Section 4.1 are valid also to this scenario and the traffic prioritization can be exploited to better keep the game frame rate high in spite of traffic load in the BS.

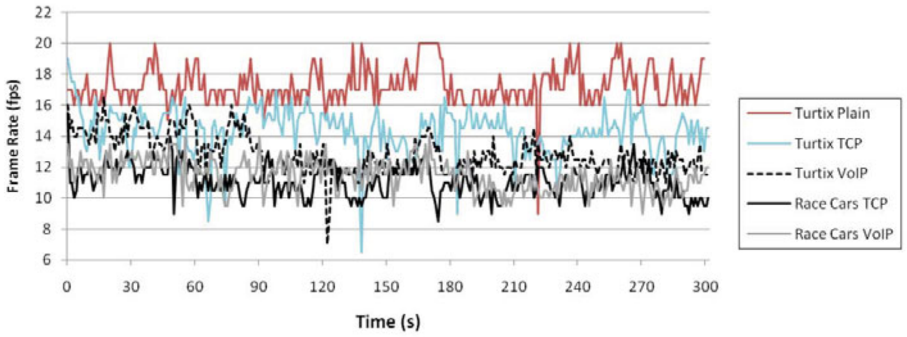

Fig. 10. Turtix frame rates with competing links

5 Conclusion

WiMAX can easily support the bitrates required by streamed online games, but already the characteristic transmission latencies of the fixed WiMAX technology can occasionally lower the gaming experience compared to the wired and WLAN technologies. However, the played games remained playable until the edge of the maximum link capacity, before the congestion started to increase transmission delays, when parallel VoIP flows were injected as background traffic into the uplink and downlink. This is due to the game adaptation mechanism at the game server, which adjusts the game frame rate according to the player's network conditions. Anyway, when the link capacity was exceeded, the playability slumped. Already one full rate TCP flow degraded the playing experience in both uplink and downlink. We attested the clear need for a packet scheduling favoring the traffic of real-time service, which the online game evidently is. In the downlink, we simply raised the priority of the game traffic, which allowed us to play the game despite the background traffic load in the downlink. In the uplink, rtPS scheduling ensuring bandwidth for the user commands was detected as important way to improve the playability. In fact, we noticed that the uplink delay is the main factor in achieving a smooth playing experience, as it commonly yields higher delays than the downlink with the BE scheduling and moderate traffic loads.

To lower the transmission latencies of the WiMAX link is important also due to the fact that we did not consider delays caused by the Internet in our measurements. However, IEEE 802.16's Task Group m is working on an amendment IEEE 802.16m [17], which will introduce an improved air interface capable of supporting higher data rates and lower link latencies, at maximum 10 ms in both the downlink and uplink, than the one currently specified in the IEEE

802.16 standards, indeed, making WiMAX a more suitable technology for the streamed online gaming. It aims to fulfil the requirements of the International Mobile Telecommunications (IMT) -Advanced systems [18], specified by the International Telecommunication Unions Radiocommunication Sector (ITU-R).

Acknowledgment. This work was carried out within the framework of IST FP6 Games@Large project, which is partially funded by European Commission, and as part of the Future Internet program of TIVIT (Finnish Strategic Centre for Science, Technology and Innovation in the field of ICT) funded by TEKES.

References

1. IEEE 802.16 WG. IEEE Standard for Local and Metropolitan Area Networks - Part 16: Air Interface for Fixed Broadband Wireless Access Systems. IEEE Std. 802.16-2004 (October 2004)
2. Claypool, M., Claypool, K.: Latency and Player Actions in Online Games. Communications of the ACM 49(11), 40–45 (2006)
3. Dick, M., Wellnitz, O., Wolf, L.: Analysis of Factors Affecting Players' Performance and Perception in Multiplayer Games. In: Proc. of 4th ACM SIGCOMM Workshop on Network and System Support for Games (NetGames), pp. 1–7. ACM, New York (2005)
4. Henderson, T., Bhatti, S.: Networked games: a QoS-sensitive application for QoS-insensitive users? In: Proc. of the ACM SIGCOMM Workshop on Revisiting IP QoS (RIPQoS), pp. 141–147. ACM, New York (2003)
5. Jurgelionis, A., Bellotti, F., De Gloria, A., Laulajainen, J.-P., Fechteler, P., Eisert, P., David, H.: Testing Cross-Platform Streaming of Video Games over Wired and Wireless LANs, pp. 1053–1058 (April 2010)
6. Wang, X., Kim, H., Vasilakos, A.V., Kwon, T., Choi, Y., Choi, S., Jang, H.: Measurement and Analysis of World of Warcraft in Mobile WiMAX Networks. In: Proc. of 8th Annual Workshop on Network and Systems Support for Games (NetGames), pp. 1–6 (November 2009)
7. Wodarz, M., Taisie, J.P., Schmidt, T.C.: QoS Performance Study of One Way Link Characteristics in an IEEE 802.16d TDD System. In: Proc. of 6th Advanced International Conference on Telecommunications (AICT), pp. 241–246 (May 2010)
8. Pentikousis, K., Piri, E., Pinola, J., Fitzek, F., Nissilä, T., Harjula, I.: Empirical Evaluation of VoIP Aggregation over a Fixed WiMAX Testbed. In: Proc. of 4th International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities (TRIDENTCOM), Article No. 19, Innsbruck, Austria (March 2008)
9. De Bruyne, J., Joseph, W., Verloock, L., Martens, L.: Measurements and Evaluation of the Network Performance of a Fixed WiMAX System in a Suburban Environment. In: Proc. of IEEE International Symposium on Wireless Communication Systems (ISWCS), Reykjavik, Iceland, pp. 98–102 (October 2008)
10. Piri, E., Pinola, J., Fitzek, F., Pentikousis, K.: ROHC and Aggregated VoIP over Fixed WiMAX: An Empirical Evaluation. In: Proc. of 13th IEEE Symposium on Computers and Communications (ISCC), Marrakech, Morocco, pp. 1141–1146 (July 2008)
11. Games@Large project. Website, `http://www.gamesatlarge.eu/`

12. Prokkola, J., Hanski, M., Jurvansuu, M., Immonen, M.: Measuring WCDMA and HSDPA Delay Characteristics with QoSMeT. In: Proc. of IEEE International Conference on Communications (ICC), pp. 492–498 (June 2007)
13. ITU-T. G.729 Based Embedded Variable Bit-rate Coder: An 8-32 kbit/s Scalable Wideband Coder Bitstream Interoperable with G.729. ITU-T Recommendation G.729.1 (2006)
14. ITU-T. Coding of Speech at 8 kbit/s Using Conjugate-structure Algebraic-code-excited Linear Prediction (CS-ACELP). ITU-T Recommendation G.729 (1996)
15. Botta, A., Dainotti, A., Pescapé, A.: Do You Trust Your Software-based Traffic Generator? IEEE Communications Magazine 48(9), 158–165 (2010)
16. Manner, J.: Jugi's Traffic Generator, `http://www.netlab.tkk.fi/~jmanner/jtg.html`
17. IEEE 802.16 WG. IEEE 802.16m System Description Document [Draft], Work in Progress. IEEE 802.16m-09/003r3 (June 2010)
18. ITU-R. Requirements Related to Technical Performance for IMT-Advanced Radio Interface(s). ITU-R Report M.2134 (2008)