Publication P4


© 2007 IEEE

Reprinted with permission.

This material is posted here with permission of the IEEE. Such permission of the IEEE does not in any way imply IEEE endorsement of any of Helsinki University of Technology's products or services. Internal or personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution must be obtained from the IEEE by writing to pubs-permissions@ieee.org.

By choosing to view this document, you agree to all provisions of the copyright laws protecting it.
Active Queue Management for Reducing Downlink Delays in WiMAX

Jani Lakkakorpi
Nokia Technology Platforms
Espoo, Finland
jani.lakkakorpi@nokia.com

Alexander Sayenko, Juha Karhula, Olli Alanen
Telecommunication Laboratory, MIT Department
University of Jyväskylä, Finland
{sayenko, jkarhula, opalanen}@cc.jyu.fi

Jani Moilanen
Nokia Research Center
Helsinki, Finland
jani.moilanen@nokia.com

Abstract—In WiMAX networks, the base station (BS) is a likely bottleneck for downlink (DL) TCP connections due to difference in available bandwidth between the fixed network and the wireless link. This may result in buffer overflows or excessive delays at the BS, as these buffers are connection-specific. In order to avoid buffer overflows, different Active Queue Management (AQM) methods may be applied at the BS. This paper presents an analysis of several AQM mechanisms and proves that they are indeed very useful: AQM reduces considerably DL delays at the WiMAX BS without sacrificing TCP goodput.

Key Words—IEEE 802.16 WiMAX, AQM, QoS, ns-2

I. INTRODUCTION

WiMAX is an IEEE standard for wireless broadband access network [1, 2]. Some of the main advantages of WiMAX over other access network technologies are longer range and more sophisticated support for Quality of Service (QoS) at the MAC level. Several different types of applications and services can be used in WiMAX networks and the MAC layer is designed to support this convergence. The standard defines two basic operational modes: mesh and point-to-multipoint (PMP). In the former mode, subscriber stations (SS) can communicate to each other and to the base stations (BS). In the PMP mode, the SSs are only allowed to communicate through the BS. It is anticipated that providers will use the PMP mode to connect their customers to the Internet. In this case, the SSs do not send data to each other but rather communicate through the BS. Thus, the provider can control the environment to ensure the QoS requirements of its customers.

At the WiMAX BS, all downlink (DL) connections have dedicated buffers\(^1\). However, the effective air interface bandwidth that a connection gets may vary a lot because there are no dedicated radio channels. Since the DL direction is more likely to be a bottleneck than the UL direction, TCP-based flows may experience quite long buffering delays at the BS (see Fig. 1). Thus, techniques that aim at keeping the buffer size small are very useful. At the WiMAX BS (or SS\(^2\)), however, we do not manage traffic aggregates in the same way as in IP routers. Thus, we should not use the well-known Active Queue Management (AQM) techniques from the IP world as such. Queue size averaging, for example, is not feasible with per-connection buffering.

This paper studies the end-to-end performance of TCP-based flows in a WiMAX network with and without AQM at the BS. For these purposes, we run several simulation scenarios and apply various AQM techniques to the BS DL queues. The rest of this paper is organized as follows: section 2 presents the proposed AQM schemes for WiMAX BS, sections 3 and 4 present our simulator and the simulation results, respectively, while section 5 concludes the paper.

II. ACTIVE QUEUE MANAGEMENT

As in the case of other wireless technologies, it is likely that the radio interface [1, 2] is the biggest system bottleneck with WiMAX, too. Thus, we might benefit from keeping the DL per-connection queues at the BS short enough by using AQM algorithms such as Random Early Detection (RED) [3]. This should result in a better TCP performance and thus better goodput, i.e., how many bits per second an application sees. Shorter response time is also an important issue. Earlier experience has shown [4, 5] that AQM is feasible in both 2G-GGSN and RNC (bottlenecks similar to WiMAX BS). In the following subsections, we describe the applied AQM algorithms.

---

1 There can be multiple separate connections per SS. For uplink (UL) connections, however, the BS grants slots per SS and not per connection (bandwidth request based virtual per-SS queues are used). It is the SS that decides how these slots are used.

2 From the SS perspective, the UL direction is the bottleneck. We could apply the mechanisms described in this paper to the queues at the SS, too. However, we do not study this issue in this paper.
A. Random Early Detection

Probably the most well-known method for TCP performance enhancement, RED (and Explicit Congestion Notification, ECN [6]) aims at preventing TCP synchronization\(^3\) by dropping random packets when the averaged queue size exceeds the minimum threshold. The bigger the averaged queue size, the bigger the probability of a packet drop. However, RED does not usually work well if there is only a single flow or a couple of flows sharing the buffer. In such a case, the buffer occupancy will vary a lot, and we are forced to use the instantaneous queue length instead of a slowly averaged one.

B. Packet Discard Prevention Counter

In [7], Sågfors, Ludwig, Meyer and Peisa have presented a technique called Packet Discard Prevention Counter (PDPC). They propose a deterministic packet dropping mechanism with a counter: after each packet drop, \(N\) packets have to be accepted before another packet can be dropped (unless maximum threshold is exceeded). This technique assumes that adaptive minTh and maxTh are used. However, it is not described in [7] how to tune these thresholds. Thus, we shall simply use static thresholds. Moreover, drop from tail shall be used instead of drop from front.

C. TTLRED

In time-to-live based RED, i.e., TTLRED [4, 5], we provide the DL TCP packets with timestamps: each packet is given a random dropping time when it enters the BS. Packet marking algorithm is illustrated in Fig. 2 and it is somewhat analogous to gentle RED [8].

\[
\text{if } (U(0, 1) < \text{maxDP}) \\
\quad \text{lifetime} := U(\text{lowTh}, 0.5 \times \text{highTh}) \\
\text{else} \\
\quad \text{lifetime} := U(0.5 \times \text{highTh}, \text{highTh})
\]

Fig. 2. TTLRED packet marking algorithm.

Moreover, higher lifetime is always assigned when a packet enters an empty buffer. Our scheme is a bit similar to assigning

\(^{\text{A phenomenon that occurs when the bottleneck queue gets full and several packets, belonging to different TCP flows, are dropped at the same time. Thus, all affected TCP flows will have to decrease their sending rate.}}\)

### III. WiMAX Network Simulation Model

The basic implementation of our WiMAX MAC level model is described in [11]. It includes features such as transport and management connections, fragmentation and packing, ranging and bandwidth request contention periods, support for all the important MAC level signaling messages, and the ARQ mechanism. With fragmentation, a sender can split a large packet between several MAC PDUs, while packing allows putting several packets and packet fragments into a single PDU. The implementation chooses automatically the suitable mode. ARQ allows retransmitting dropped PDUs.

Support for the contention period allows simulating accurately the extended real-time polling service (ertPS), non-real-time polling service (nrtPS) and best effort service (BE) classes: connections that belong to these classes use the bandwidth requests to inform the BS about their current requirements. The ranging transmission opportunities are used by the SSs (usually in the beginning of a simulation run) when they enter the network.

Additionally, the implementation provides the BS scheduler as well as OFDM and OFDMa PHY levels. On top of that, link adaptation and dynamic MAC overhead are modeled.

A. MCS, Link Adaptation and Errors

Modulation and coding scheme (MCS) defines how many bits a connection can send during one slot. The BS can dynamically change the MCSs by the means of uplink channel descriptor (UCD) and downlink channel descriptor (DCD) messages. Link adaptation is modeled by a Markov chain (see Table I), where the states represent different MCSs. Transition from one state to another is possible in each frame. We have obtained the state transition probabilities from system simulations. Naturally, these results always correspond to a certain traffic load, environment, link adaptation parameters etc.

### TABLE I

<table>
<thead>
<tr>
<th>Direction</th>
<th>MCS</th>
<th>Bytes/slot</th>
<th>Transition probability to</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>64QAM-3/4</td>
</tr>
<tr>
<td>DL</td>
<td>64QAM-3/4</td>
<td>27</td>
<td>0.79486090</td>
</tr>
<tr>
<td></td>
<td>64QAM-2/3</td>
<td>24</td>
<td>0.28293998</td>
</tr>
<tr>
<td></td>
<td>16QAM-3/4</td>
<td>18</td>
<td>0.01279176</td>
</tr>
<tr>
<td></td>
<td>16QAM-1/2</td>
<td>12</td>
<td>0.00000000</td>
</tr>
<tr>
<td></td>
<td>QPSK-3/4</td>
<td>9</td>
<td>0.00000000</td>
</tr>
<tr>
<td></td>
<td>QPSK-1/2</td>
<td>6</td>
<td>0.00033259</td>
</tr>
<tr>
<td>UL</td>
<td>64QAM-3/4</td>
<td>27</td>
<td>0.68184019</td>
</tr>
<tr>
<td></td>
<td>64QAM-2/3</td>
<td>24</td>
<td>0.36491229</td>
</tr>
<tr>
<td></td>
<td>16QAM-3/4</td>
<td>18</td>
<td>0.02981995</td>
</tr>
<tr>
<td></td>
<td>16QAM-1/2</td>
<td>12</td>
<td>0.00374267</td>
</tr>
<tr>
<td></td>
<td>QPSK-3/4</td>
<td>9</td>
<td>0.00049722</td>
</tr>
<tr>
<td></td>
<td>QPSK-1/2</td>
<td>6</td>
<td>0.00026013</td>
</tr>
</tbody>
</table>

\(^{3}\) A phenomenon that occurs when the bottleneck queue gets full and several packets, belonging to different TCP flows, are dropped at the same time. Thus, all affected TCP flows will have to decrease their sending rate.
The aforementioned Markov model also stores the MAC PDU error probabilities (i.e., the probability that a MAC PDU transmission fails due to channel errors) in its states. Error rate for a 100-byte MAC PDU is 10% with each MCS when ARQ is applied and 1% when ARQ is not applied.

B. Scheduler

The scheduler at the BS grants slots for the SSs according to their QoS requirements and the bandwidth request sizes\(^4\). For DL connections, we use the BS queue sizes and the QoS requirements. Our scheduler is somewhat simpler than the one presented in [12]. Slots are assigned in DRR [13] fashion. Quantum size is a configuration parameter (default is 17 slots for all connections); a bigger quantum size decreases the MAP overhead as we then serve less connections per frame. If ARQ is enabled for a connection, we apply the following connection-interval scheduling order: 1) ARQ feedback messages, 2) retransmissions and 3) all other PDUs.

We have implemented support for two WiMAX QoS classes: real-time polling service (rtPS) and BE. rtPS connections are naturally served first; they are assigned slots until the rtPS (UL: virtual, DL: real) queues are empty or until there are no slots left. However, 5% of available UL and DL slots are reserved for the BE connections in order to avoid their starvation. UL rtPS connections are polled regularly (frequency depends on connection parameters), because they cannot participate in contention resolution like the UL BE connections. BE connections are served after rtPS connections; they are served until the BE (UL: virtual, DL: real) queues are empty or until there are no slots left. If there are still available slots, they are assigned for contention resolution. Admission control should be executed for the rtPS connections. Otherwise, their QoS requirements cannot be guaranteed.

UL-MAP and DL-MAP sizes are calculated based on the number of served connections in the previous frame (of course, this is a simplification and it cannot be done in real life). As the QoS solution at the SS is vendor-specific, we simulate only cases with a single connection per SS.

IV. PERFORMANCE EVALUATION

We use a modified version of the ns-2 simulator [14]. The WiMAX related modifications have been described in the preceding section. Six simulations are run in each case in order to obtain the 95% confidence intervals. Simulation time is 200 seconds. One-way core network delay between a server and the BS is set to 31 ms. The only bottleneck in our system is the WiMAX air interface (see Fig. 3). There are no packet drops or variable delays elsewhere in the system. The most important WiMAX network parameters are listed in Table II. Fig. 4–12 illustrate the main results of our simulations.

Each AQM algorithm is modified so that we only drop TCP packets. This may be beneficial for UDP-based variable bit rate streaming traffic – the jitter buffer at the receiver will handle the occasional bursts and drop packets if they are delayed too much. Per-connection queue size is limited to 50 packets. Assuming that all packets are of MTU size, this translates into 75 kB buffers. For RED and PDPC, we apply the following parameter values (where applicable): \(\text{minTh} = 15\, \text{kB}, \text{maxTh} = 45\, \text{kB}, \text{maxDP} = 1.0\), \(w_{\text{QoS}} = 1.0\) and \(N = 20\). For TTLRED, we apply the following parameter values: \(\text{lowTh} = 0.3\, \text{s}, \text{highTh} = 0.9\, \text{s}\) and \(\text{maxDP} = 1.0\).

### Table II
\begin{tabular}{|c|c|}
\hline
Parameter & Value \\
\hline
\text{PHY} & OFDMA \\
\hline
\text{Duplexing mode} & TDD \\
\hline
\text{Frame length} & 5 ms \\
\hline
\text{Bandwidth} & 10 MHz \\
\hline
\text{FFT size} & 1024 \\
\hline
\text{Guard period} & 1/8 \\
\hline
\text{DL/UL permutation zone} & FUSC/PUSC \\
\hline
\text{DL/UL antenna technique} & SISO/SISO \\
\hline
\text{DL/UL ratio} & 35/12 \\
\hline
\text{DL-MAP/UL-MAP fixed overhead} & 13 bytes / 8 bytes \\
\hline
\text{DL-MAP/UL-MAP IE size} & 8 bytes / 4 bytes \\
\hline
\text{DL frame preamble size} & 1 symbol \\
\hline
\text{Number of ranging opportunities} & 1 \\
\hline
\text{Number of ranging opportunities} & 9/15 \\
\hline
\text{Number of request opportunities} & 3 \\
\hline
\text{Request backoff start/end} & 3/15 \\
\hline
\text{Maximum MAC PDU size} & 100 bytes \\
\hline
\text{Fragmentation} & Yes \\
\hline
\text{Packing} & Yes \\
\hline
\text{ARQ} & For all but VoIP connections \\
\hline
\text{ARQ feedback types} & All \\
\hline
\text{ARQ block size / window size} & 16 bytes / 1024 \\
\hline
\text{ARQ block rearrangement} & No \\
\hline
\end{tabular}

\(^4\) Virtual SS queue sizes are updated whenever slots are granted and refreshed every time a bandwidth request (that includes the real SS queue size) arrives.
The variable bit rate video traffic source is simulated according to [15]. The following parameters are used: mean rate of 125 kbps, maximum rate of 250 kbps; MTU (1500 bytes) sized packets are sent; UDP is used as transport protocol; IPv4 adds 20 bytes of overhead, which results in a payload size of 1500-8-20 = 1472 bytes.

Our web browsing traffic source is simplified from [16]; it is necessary to limit the variability of page sizes as the TCP goodput depends on that, too. We have a main page and 30 inline items per page, all items have a size of 4.91 kB, which results in a total page size of 152 kB (this is based on our measurements). Page reading time is uniformly distributed between 1 and 5 seconds and four NewReno TCP [17] connections are utilized. The same HTTP traffic model is used for modeling file downloading, but this time only a single 250 kB file is downloaded. Time between two downloads is uniformly distributed between 1 and 5 seconds. A single NewReno TCP connection is utilized.

Fig. 4–7 illustrate the effect of RED, PDPC and TTLRED algorithms on DL packet delay and TCP goodput (with two different TCP advertised window values and 5 VoIP, 5 video streaming, 10 web browsing and 5 FTP connections) for short-lived web browsing sessions and long-lived file downloading sessions. It can be seen that the AQM algorithms decrease the delay most, when the advertised window is relatively big and when the TCP connections are long-lasting. Fig. 6 and Fig. 7 show that AQM does not increase the TCP goodput this time. However, non-optimal AQM parameters can lead into decreased goodput.

Fig. 8–11 illustrate the effect of the most promising AQM algorithms, PDPC and TTLRED, on DL packet delay and TCP goodput for short-lived web browsing sessions and long-lived file downloading sessions. This time with 60-packet TCP advertised window and 5 VoIP, 5 video streaming, 10–30 web browsing and 5–15 FTP connections. As the load grows, the improvement in delay is more and more obvious with both algorithms. Fig. 10 and Fig. 11 show that TCP goodput is not really affected.

As a side note, we can observe from Fig. 12 that VoIP needs QoS support: as the traffic load grows and VoIP is given BE treatment, UL VoIP packets are delayed due to contention resolution procedure. Moreover, when there is enough traffic, congestion will happen in DL direction, too. Naturally, this depends on the DL/UL ratio and traffic patterns.

It should be noted that rtPS may not be the best choice for VoIP QoS class; erTPS or unsolicited grant service (UGS), depending on VoIP traffic characteristics, could actually be a more suitable choice. This is for further study. Either way, our weighted scheduler, with separate DRR instances for VoIP and BE, is able to guarantee QoS for VoIP and it is fair towards BE connections, too, as there is a minimum weight for them.
packet drops. It is not reasonable to apply AQM to traffic that does not react to the packet header and drop TCP (or TCP-friendly) packets only.

(scheduling should guarantee low enough DL queuing delays for class. AQM could be applied to BE and nrtPS connections only for a given connection or not, should depend on the selected QoS negotiate it with the customer. Thus, whether AQM is applied is not a part of the WiMAX QoS profile, there is no way to parameter, just like ARQ or QoS class. Indeed, a provider can the BS, it makes sense to treat this feature as a connection well in all tested conditions.

TTLRED, on the other hand, is simple to configure, it does not changing all the time. Thus, we find these methods not feasible. Parameters according to available bandwidth, which keeps static parameters is not enough but we should adjust the can even decrease the goodput due to excessive packet drops.

AQM helps to reduce DL queuing delays of TCP connections, provided enough buffer space will maximize TCP goodput. However, as AQM helps to reduce DL queuing delays of TCP connections, we shall have better user-experience in web surfing and in other semi-real time activities. Moreover, less buffer memory is needed when queues are kept short. TCP goodput is not increased – given that the DL buffer size and TCP advertised window are sufficient.

It is worth mentioning that badly configured AQM algorithms can even decrease the goodput due to excessive packet drops. Thus, RED, PDCP, or most probably any AQM scheme with static parameters is not enough but we should adjust the parameters according to available bandwidth, which keeps changing all the time. Thus, we find these methods not feasible. TTLRED, on the other hand, is simple to configure, it does not require parameter adjustments (when the queuing delays are short enough, the algorithm is effectively disabled) and it performs well in all tested conditions.

Since we apply AQM to individual per-connection queues at the BS, it makes sense to treat this feature as a connection parameter, just like ARQ or QoS class. Indeed, a provider can turn on and off AQM for a particular connection without compromising the WiMAX specification. However, since AQM is not a part of the WiMAX QoS profile, there is no way to negotiate it with the customer. Thus, whether AQM is applied for a given connection or not, should depend on the selected QoS class. AQM could be applied to BE and rtPS connections only (scheduling should guarantee low enough DL queuing delays for the more real-time connections). However, we should still check the packet header and drop TCP (or TCP-friendly) packets only. It is not reasonable to apply AQM to traffic that does not react to packet drops.

V. CONCLUSIONS

In this paper, we have studied the end-to-end performance of various applications in a WiMAX network with and without AQM. We have shown that AQM is quite useful in reducing the queuing delays caused by the BS DL buffering. Providing enough buffer space will maximize TCP goodput. However, as AQM helps to reduce DL queuing delays of TCP connections, we shall have better user-experience in web surfing and in other semi-real time activities. Moreover, less buffer memory is needed when queues are kept short. TCP goodput is not increased – given that the DL buffer size and TCP advertised window are sufficient.

It is worth mentioning that badly configured AQM algorithms can even decrease the goodput due to excessive packet drops. Thus, RED, PDCP, or most probably any AQM scheme with static parameters is not enough but we should adjust the parameters according to available bandwidth, which keeps changing all the time. Thus, we find these methods not feasible. TTLRED, on the other hand, is simple to configure, it does not require parameter adjustments (when the queuing delays are short enough, the algorithm is effectively disabled) and it performs well in all tested conditions.

Since we apply AQM to individual per-connection queues at the BS, it makes sense to treat this feature as a connection parameter, just like ARQ or QoS class. Indeed, a provider can turn on and off AQM for a particular connection without compromising the WiMAX specification. However, since AQM is not a part of the WiMAX QoS profile, there is no way to negotiate it with the customer. Thus, whether AQM is applied for a given connection or not, should depend on the selected QoS class. AQM could be applied to BE and rtPS connections only (scheduling should guarantee low enough DL queuing delays for the more real-time connections). However, we should still check the packet header and drop TCP (or TCP-friendly) packets only. It is not reasonable to apply AQM to traffic that does not react to packet drops.

REFERENCES