Bandwidth Allocation in WiMAX Networks Using Reinforcement Learning

Saeid M. Jafari and M.R. Meybodi

Department of Computer and IT Engineering, Islamic Azad University of Qazvin, Qazvin, Iran
Department of Electrical and Computer Engineering, Amirkabir University of Technology, Tehran, Iran

Abstract - An important problem for the WiMAX networks is how to provide a guaranteed quality of service for applications. A key aspect of this problem is how BSs should share bandwidth capacity between different classes of traffic. This decision needs to be made for each incoming packet and is known as the packet scheduling problem. A major challenge in packet scheduling is that the behavior of each traffic class may not be known in advance and can vary dynamically. In this paper, we describe how we have modeled the packet scheduling problem as an application for reinforcement learning (RL). We demonstrate how our RL approach can learn scheduling policies that satisfy the quality of service requirements of multiple traffic classes under a variety of conditions. The proposed solution has been designed to have an ability to accommodate integrated traffic in the networks with effective scheduling schemes. A series of simulation experiments have been carried out to evaluate the performance of the proposed scheduling algorithm. The results reveal that the proposed solution performs effectively to the integrated traffic composed of messages with or without time constraints and achieves proportional fairness among different types of traffic.

Key words: 802.16 %WiMax %Scheduling Algorithms %Channel assignment

INTRODUCTION

WiMAX technology based on the IEEE 802.16 standard [1, 2] has a very rich set of features. Indeed, it is a very promising Broadband Wireless Access (BWA) technology. The major attractions of WiMAX systems come from their ability to provide broadband wireless access and potential ability to compete with existing wired systems such as fiber optic links, coaxial systems using cable modems and digital subscriber line (DSL) links with much scalability. The major attractions of WiMAX systems come from their ability to provide broadband wireless access and potential ability to compete with existing wired systems such as fiber optic links, coaxial systems using cable modems and digital subscriber line (DSL) links with much scalability. The WiMAX networks have the capacity to provide flexibility and efficiency to allow coexistence of different types of traffic, such as real-time and multimedia traffic. One important issue in the WiMAX networks design is to support QoS services to different types of traffic. IEEE802.16d standard [1, 2], ratified in June 2004, has specified all the techniques of the WiMAX systems to deliver broadband service in the fixed point-to-point (PTP) or point-to-multipoint (PMP) topologies. And it has proposed a framework for the QoS services for four types of traffic. Unsolicited Grant Service (UGS), real-time Polling Service (rtPS), non real-time Polling Service (nrtPS) and Best Effort (BE) QoS classes. UGS supports real-time service flows that have fixed-size data packets on a periodic basis. rtPS supports real-time service flows that generate variable data packets size on a periodic basis. The BS provides unicast grants in an unsolicited manner like UGS. Where as the UGS allocations are fixed in size. nrtPS is designed to support non real-time service flows that require variable size bursts on a regular basis. BE is used for best effort traffic where no throughput or delay guarantees are provided. Those service classes are defined in order to satisfy different types of Quality of Service (QoS) requirements. However, the IEEE 802.16 standard does not specify the scheduling algorithm to be used. Vendors and operators have to choose the scheduling algorithm(s) to be used. Three types of schedulers must be defined; an uplink and a downlink scheduler both in the Base Station (BS) and just an uplink scheduler for the Subscriber Station (SS) between the different simultaneous connections of the SS.

In this paper we present a system for packet scheduling that is based on Learning Automaton. In our approach,
Learning Automaton is used to learn a scheduling policy in response to feedback from the network about the delay experienced by each traffic class. Key advantages of our approach are that our system does not require prior knowledge of the statistics of each traffic flow and can adapt to changing traffic requirements and loads. In practice, this helps network providers to deliver a guaranteed QoS to customers, while maximizing network utilization and minimizing the need for manual intervention.

We make three key contributions in this paper: (1) we present a model for using RL to address the problem of packet scheduling in Base Station with QoS requirements; (2) we demonstrate the advantages of RL in terms of convergence time in comparison to other scheduling schemes; and (3) we provide an insight into the relative merits of two alternative RL algorithms in the context of this application. We begin by describing the application of packet scheduling. We then describe our solution based on LA and demonstrate its effectiveness in a range of simulated traffic conditions.

**Previous Work:** In this section, we present some dynamically depending on the wireless link state of subscribers. At the beginning of each scheduling epoch, the BS resets the eligible and scheduled sets and repeats the above mentioned process.

The Opportunistic Deficit Round Robin (O-DRR) scheduler [7] is used in the uplink direction. The BS polls subscribers periodically. After each period, the BS determines the set of subscribers that are eligible to transmit and their bandwidth requirements. This set is defined as the eligible set. A number of conditions must be verified by an SS to be in this set: ² The queue is not empty. ² The received SIR is above a minimum threshold denoted \(SIR_{th}\). Once these conditions are satisfied, the subscriber is eligible to transmit during a given frame of the current scheduling epoch. The scheduled set changes dynamically depending on the wireless link state of subscribers. At the beginning of each scheduling epoch, the BS resets the eligible and scheduled sets and repeats the above mentioned process.

Previous Work: In this section, we present some dynamically depending on the wireless link state of subscribers. At the beginning of each scheduling epoch, the BS resets the eligible and scheduled sets and repeats the above mentioned process.

**Problem Definition:** Our problem definition is based on the model of packet scheduling described by Hall and Mars [14]. We have used their model so that we can compare the performance of their SLA with our own proposal. Our aim is to schedule \(N\) classes of traffic, where each traffic class has its own queue \(q_i; i = 1: N\). Let \(q_N\) denote the queue for best-effort traffic, which has no predefined delay requirements. For each remaining queue \(q_i; i = 1: N - 1\), there is a mean delay requirement \(R_i\), which is the maximum acceptable mean queuing delay per packet for the traffic class assigned to \(q_i\). Let \(M_i\) denote the measured mean queuing delay of packets in \(q_i\) over
In the next section, we describe our method for using Learning Automatons to learn scheduling policies for queue management. First, we require a representation of the states of our system, which reflects the state of the traffic in our queues. Second, we require a suitable reward function \( r: S \times A \rightarrow \mathbb{R} \), which reflects the immediate value of our scheduling actions. Finally, we require a learning algorithm to refine our policy function \( A(s) \) based on the feedback provided by our reward function. Let us now describe our solution for each of these components of our system.

**State Representation:** The reason for introducing the system state into the policy function is so that the scheduler can learn how to act in different situations. This is in contrast to the approach of using a SLA, which uses a single state in its policy function, i.e., the scheduling policy does not depend on the state of the queues. By introducing a more sophisticated state representation we can potentially gain greater control, albeit at the risk of greater complexity. However, we need to ensure that the state representation is not too complex; otherwise there may be too many parameters to be tuned, which may slow the convergence rate of the algorithm.

Our aim is to use different scheduling policies depending on which queues are not meeting their delay requirements. We represent the state of the system by a set of \( N \) binary variables \( \{s_1: s_{N-1}\} \), where each variable \( s_i \) indicates whether traffic in the corresponding queue \( q_i \) is meeting its mean delay requirement \( R_i \).

Note that there is no variable corresponding to the best-effort queue \( q_N \), since there is no mean delay requirement for that queue. For example, the state \( \{0; 0; ::; 0\} \) represents that all queues have satisfied their mean delay constraint, while \( \{1; 0; ::; 0\} \) represents that the mean delay requirements are being satisfied for all queues except \( q_1 \). Thus, if there are \( N \) queues in the system including one best-effort queue, then there are \( 2^N \) possible states. In practice, the number of traffic classes is normally small, e.g., four classes in Cisco routers with priority queuing, in which case the number of states is acceptable.

**Reward Function:** The role of the reward function is to provide feedback to the Learning Automaton algorithm about the effect of a scheduling action.
Based on this feedback, the learning algorithm can decide how to update the current scheduling policy. Our aim is to provide a positive reward when packets are serviced within their delay requirement and a negative reward when they are late. We also want to provide a positive reward when the system moves to a better state, i.e., when the measured mean delay for a queue falls below the required mean delay. Thus our reward function $r$ comprises a time reward component $r_{\text{time}}$ and a state reward component $r_{\text{state}}$, where $r = r_{\text{time}} + r_{\text{state}}$.

Every time a scheduling action is executed, the time reward $r_{\text{time},i}$ for each queue $q_i$ is calculated in terms of the mean delay requirement $R_i$ and the measured mean delay $M_i$.

$$r_{\text{time},i} = \begin{cases} C_1 \frac{M_i}{R_i} & \text{if } M_i < R_i \\ C_1 & \text{if } M_i = R_i \\ -C_2 & \text{if } M_i > R_i \end{cases}$$

The time reward is positive when $M_i = R_i$ and negative when $M_i > R_i$. It is maximized when the mean delay requirement is just satisfied. There is a diminishing reward as $M_i$ approaches zero, since any reduction in $M_i$ below $R_i$ is wasting bandwidth that could be allocated to other queues. In general, it is possible to change the form of the reward function depending on the type of QoS requirements that need to be satisfied. The total time reward is a weighted sum of the rewards for each queue.

$$r_{\text{time}} = \sum_{i=1}^{N-1} w_i r_{\text{time},i}$$

Where the weights $w_i$ depend on which queue was serviced by the last scheduling action. In practice, we have found that suitable weights are $w_i = 0.3$ if $q_i$ was the queue serviced by the last action, otherwise $w_i = 1.0$.

These weights discourage the scheduler from servicing queues with satisfactory performance if there are other queues experiencing unsatisfactory performance. Although the choice of weight values is not critical, we found that we can significantly improve the convergence rate of our system by using a non-zero weight for queues that were not serviced by the last action.

The state reward is positive if a scheduling action causes the system to move to a better state $S'$ compared to the previous state $s$. State $S'$ is considered to be better than $s$ if it has more queues whose mean delay requirements are being met, e.g., $S' = \{0; 0; \cdots; 0\}$ is better than $s = \{1; 0; \cdots; 0\}$. Thus.

$$r_{\text{state}} = \begin{cases} C_3 & \text{if } S' \text{ is better than } S \\ 0 & \text{Else} \end{cases}$$

**Learning Algorithm:** Our approach to learning a scheduling policy $A(s)$ is based on the standard Learning Automaton. Let us begin by describing the general goal of learning in this context. Consider the system at time $t$ in state $s$. The scheduler selects an action at and in turn receives a reward $r_{t+1}$. As a result of this action and the arrival of new packets, the system moves to a new state $s_{t+1}$. From this new state the scheduler then selects another action at $t+1$ according to its current policy $A$ and consequently another reward $r_{t+2}$ is received. This process continues and we can think of a trial resulting in a particular sequence of future rewards $(r_{t+1}; r_{t+2}; \cdots)$ as having been generated by the scheduler in following its policy at time $t$.

**Evaluation:** Simulation performance solution for testing new technologies is. Network simulator (NS2) widely for wireless network simulation is used. The simulation parameters are being in following tables:

<table>
<thead>
<tr>
<th>Table 1: Traffic model distribution for different class’s services</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Service Classes (Application)</strong></td>
</tr>
<tr>
<td>-----------------------------------</td>
</tr>
<tr>
<td>UGS (Voice with silence)</td>
</tr>
<tr>
<td>Rtps (Video conferencing)</td>
</tr>
<tr>
<td>Nrtps (FTP-Web Browsing)</td>
</tr>
<tr>
<td>BE (Email)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 2: The scenario used in our simulation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Topology</strong></td>
</tr>
<tr>
<td>-------------</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 3: IEEE 802.16-Transmission Modes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Modulation type</strong></td>
</tr>
<tr>
<td>Inner code rate</td>
</tr>
<tr>
<td>Bits / symbol</td>
</tr>
<tr>
<td>Raw data rate (Rb/Mbps)</td>
</tr>
<tr>
<td>Bytes / mini slot</td>
</tr>
</tbody>
</table>
Fig. 3: Latency versus traffic

Fig. 4: Throughput versus traffic

Fig. 5: Latency versus traffic

Fig. 6: Latency versus traffic

The Simulation Results Are Following Figures: In this section, we compare five scheduling algorithms: the RR, mSIR, WRR, TRS+RR and TRS+mSIR schedulers.

CONCLUSION

In this article, the behavior of some scheduling algorithms and the proposed algorithm based on the delay parameters and simulation were compared. Algorithm proposed algorithms and the maximum noise and interference elimination of combined noise and interference has the best results. RL schemes are able to satisfy QoS requirements. For multiple traffic classes, without starving resources from best-effort traffic. Furthermore, our RL schemes can adapt to changing traffic statistics and QoS requirements. Simulation results show that the proposed Scheduler Find the best solution for nrtPS and rtPS traffic and bandwidth are the fair is divided between different applications.

REFERENCES


