The Uniformly-Fair Deficit Round-Robin (UF-DRR) Scheduler for Improved QoS Guarantees in IEEE 802.16 WiMax Networks

Vishal Sharma
Metanoia, Inc., 888 Villa St., Suite 500, Mountain View, CA 94041
Namita Vamaney
Department of Electrical Engineering
Indian Institute of Technology Bombay, Powai, Mumbai 400076, India

ABSTRACT

We propose a fair and efficient scheduling algorithm for broadband wireless networks, which we refer to as Uniformly-Fair Deficit Round Robin (UF-DRR). Our algorithm, which is designed for the IEEE 802.16 MAC, gives bandwidth and delay guarantees to QoS sensitive applications while maintaining fairness among flows and achieving high system utilization. Our scheme is a refined version of the Wireless Deficit Round Robin (WDRR) technique proposed by Fattah and Leung. We modify the redistribution strategy of WDRR to achieve better performance in terms of delay, throughput, and fairness, and analyze the performance of our scheme via exhaustive packet-level simulations, under realistic wireless scenarios. This contrasts with most of the previous work, where the analysis and simulations were confined to only an error-free system. The effects of system parameters like channel quality are also studied in our work. We demonstrate that under all channel fading characteristics and for the range of applicable loads, UF-DRR is capable of providing bounded delay, fairness, and high throughput. We show that UF-DRR outperforms WDRR and DRR in the wireless environment, giving, on average, throughput that is 30% higher and fairness that is 20-30% better, even when the wireless channel is good only 50% of the time. [1] [2]

I. INTRODUCTION

The IEEE 802.16 standard [1], [2] (also known as "WiMax": Worldwide Interoperability for Wireless Access) for broadband wireless access (BWA) has been developed keeping in view the stringent QoS requirements of emerging wireless applications. The standard provides a large set of parameters to allow users to describe their traffic profile and service needs, but does not prescribe a specific scheduling algorithm to achieve these QoS requirements, leaving that as an implementation differentiator. Thus, developing an efficient traffic scheduler is one of the main design challenges in a WiMax network. Since wireless channel quality and capacity varies with time, frequency, and user location due to fading, shadowing, interference, and noise, scheduling techniques designed for wired networks are not suitable as is for wireless networks.

In this paper, we present a new fair and efficient scheduling algorithm, for wireless networks, named Uniformly-Fair Deficit Round Robin (UF-DRR), which is a refined version of the Wireless Deficit Round Robin (WDRR) algorithm proposed in [3]. The main design objectives for our algorithm are: to provide bandwidth and delay guarantees to QoS-sensitive applications, to maintain fairness among flows, and achieve high bandwidth utilization. The effectiveness of our scheme is demonstrated through a series of extensive simulations on wireless channels, and discussed later in the paper.

The remainder of our paper is organized as follows. In Section II, we provide a brief overview of the IEEE 802.16 standards and of scheduling in wireless networks. In Section III, we discuss the WDRR algorithm and point out some of its drawbacks, while in Section IV, we present the uniformly-fair deficit round-robin algorithm that we have developed, and explain its operation and key features. In Section V, we present our simulation model and the parameters that we examined, and in Section VI, we present our results and evaluation of the performance of our scheme. Finally, Section VII concludes the paper.

II. IEEE 802.16 STANDARD & SYSTEM MODEL

The WiMax standard, in its simplest form, uses time-division duplex (TDD), and provides access to each

[1]Funded by the Industrial Research and Consultancy Center (IRCC), IIT Bombay, via Grant 04IR018.

[2]Vishal Sharma was an Associate Professor (Contract) and Namita Vamaney was a Master's student at the Dept. of Electrical Engineering IIT Bombay. She is currently with Qualcomm, India.

1-4244-1513-06/07/$25.00 ©2007 IEEE

subscriber station SS or user using demand assignment multiple-access time-division multiple access (DAMA-TDMA). In WiMax, time is divided into frames [1], [2], [4], which are divided into physical slots. Each frame consists of an *uplink subframe* and a *downlink subframe* both of which have a logical channel comprised of control slots, the uplink (UL) channel and downlink (DL) channel, respectively. Requests by subscriber stations (SSs) are made in the uplink channel, while grants from the base station (BS) are communicated to SSs in the downlink channel.

The downlink channel is a broadcast channel, used by the BS for transmitting control information and downlink data to the various SSs. The uplink channel is time-shared among all SSs with the BS granting bandwidth to individual SSs in the uplink direction by assigning a variable number of physical slots and *burst profile* (a specific combination of modulation, coding rate, and FEC) to each SS according to its bandwidth demand. This information is sent to all SSs through an uplink control message.

A. Wireless System Model

All scheduling algorithms discussed in this paper use the following network and wireless link model.

Network Model: We assume a cell-structured wireless network with a BS in every cell, and the SSs in the cell communicating only via the BS. The BSs are interconnected via a wireline network. The BS is responsible for scheduling both downlink and uplink packet transmissions. For the uplink, the SSs must send their transmission requests (and queue status, if necessary) to the BS, for it to compute a schedule.

Wireless Link Model: The wireless links between mobile hosts and the base station are subject to bursty errors and considered independent. A two-state Markov channel model [5] characterizes the state of a wireless link, with the link being either in a *"good"* (*error-free*) state or in a *"bad"* (*errored*) state, in which the transmitted packets are corrupted with high probability, and no transmission is possible. Transitions from the *good* to *bad* state and vice-versa occur with probabilities P_g and P_d, respectively.

III. DRR and Wireless DRR

For scheduling algorithms to work well in a wireless environment, they should possess the following properties [6], [7] (a) short-term fairness among sessions that perceive a good channel, (b) long-term fairness among all sessions, (c) ability to guarantee QoS requirements, (d) ability to ensure maximum utilization of the wireless channel, (e) a simple implementation, and (f) low processing complexity.

The Deficit Round-Robin (DRR) algorithm [8] is an approximation of fair queueing algorithms (such as WFQ, SFQ [9], or STFQ [10]) that provides good throughput fairness at a work complexity of only $O(1)$, and a simple hardware implementation. The DRR algorithm operates in rounds (a round being one iteration over the backlogged queues). Each flow i is allocated a quantum Q_i worth of bits at the start of each round. To avoid unfairness, a state variable called deficit counter DC_i is maintained for each flow, which keeps a running count of the portion of the quantum that could not be sent from the previous round. In each round, the deficit counter of each non-empty flow is incremented by the quantum Q_i, and the scheduler sequentially visits each non-empty queue and transmits up to DC_i bits from that queue. Any balance remaining in the deficit counter is carried over to the next round. If L_{max} is the maximum packet size over all sessions, and $Q_i > L_{max}$ for all i, then the fairness measure for any interval (t_1, t_2) in DRR is bounded by a small constant and is given by [8]

$$FM_i(t_1, t_2) \leq 2 \times L_{max} + Q. \quad (1)$$

The DRR algorithm is not suitable as is for a wireless environment because it does not account for channel state. A modified version of DRR, called *Wireless Deficit Round Robin* (WDRR) proposed in [3] takes channel state into consideration. In WDRR, the channel state and the packet queue status for all sessions are assumed known at the BS. The network and wireless link model are the same as described in Sec. II-A. A session is said to be backlogged if its queue is non-empty else it is unbacklogged. The session is *clean* if its channel state is good else it is *dirty*.

WDRR consists of the following key components: an error-free service model, which provides wireless service to sessions with error-free channels; a lead and lag model, which determines which sessions are leading, lagging, or in-sync with their error-free service model and by how much; and a compensation model, which requires leading sessions to compensate the lagging sessions for service received while the channels of the lagging sessions were in the bad state.

In WDRR, a set N of sessions shares the outgoing channel, with each session $i \in N$ having a normalized reserved rate r_i. The sum of the reserved rates assigned to sessions is less than the outgoing channel capacity C, i.e., $\sum_{i \in N} r_i \leq 1$. In each round, at most Q bits

Unbacklogged	Assign Q_i to session i	No quantum assigned
Backlogged	Lagging or insync session i, give Q_i to session, else	Clean, backlogged lagging session j, give it Q_j
	Leading session -- clean backlogged lagging session j exists, give fraction of Q_i; increment lag_i & lag_j, else	else clean, backlogged leading session j, give it Q_j
	Assign Q_i to session i	else clean, backlogged insync. session j, give it Q_j
	Clean	Dirty

Fig. 1. Summary of WDRR Algorithm Operation

are nominally transmitted from all the clean backlogged sessions, where Q is the system quantum parameter, and a session i's quantum Q_i is given by $Q_i = r_i * Q$. The system quantum Q is chosen such that $Q_i \geq L_{max}$ for a session i. Each session i also maintains a deficit counter DC_i.

In WDRR, the channel changes state only after each round. Each session i is associated with a parameter lag_i that represents the difference between the service that session i would have received in an error-free system and the service that it has actually received in the error-prone system. A session i is lagging if lag_i is positive, leading if lag_i is negative, and in-sync if lag_i is zero, with $\sum_{i \in N} lag_i = 0$, holding at all times.

The working of WDRR is summarized in Fig. 1, which shows the four cases that arise when allocating/re-allocating the quantum Q_i of a session i at the start of a new round, and how the algorithm proceeds in each case.

To the best of our knowledge, none of recent the wireless scheduling algorithms has been exhaustively simulated for a wireless scenario. Also, most of their analysis has also been limited to only an error-free environment, which is equivalent to analyzing a wireline network. The performance results for WDRR have also only been provided for an error-free scenario, when it has an implementation complexity of $O(1)$. When a session perceives a bad channel, however, the algorithm complexity is $O(N)$ as the scheduler must search for a clean session from among the N sessions that share the outgoing channel.

IV. Uniformly Fair-DRR Scheduler

In UF-DRR, we modify the quantum redistribution strategy of WDRR in two cases: for leading sessions and when the channel condition of a session/flow is bad, resulting in the following three modifications to WDRR:

1) At the start of a new round, the sessions in WDRR are examined serially from 1 to N and are allocated their quantum based on their channel state and lag value. If a session, say i, is leading and there exist some clean backlogged lagging sessions, then session i's quantum is relinquished and redistributed among these sessions in proportion to their lag values. Since, in each round, the sessions are picked in order from 1 to N, the above redistribution strategy leads to a bias against sessions later in the list (as our simulation results in Sec. VI show). This is because, in every round, leading sessions earlier in the list always get an opportunity to relinquish their quantum while those further down in the list may not, as there may be no lagging sessions left to absorb their quantum. Thus, the later leading sessions continue to be leading, acquiring more bandwidth than they need.

In UF-DRR, at the beginning of each round, the quantum of all clean backlogged leading sessions is first collected (summed) and then distributed among all clean backlogged lagging sessions in proportion to their lag values. Thus, each leading session gets an *equal opportunity* to relinquish its quantum, with the resulting distribution being more uniform, leading to better fairness for all flows.

2) The quantum of a dirty session, say i, in WDRR is relinquished only to one clean backlogged lagging session, namely the one that is immediately adjacent to it in the round-robin order. In our scheme, instead of giving the entire quantum of session i to only one session, it is redistributed among all clean backlogged lagging sessions in proportion to their lag values, resulting in better short-term fairness.

3) In WDRR, in the absence of any clean backlogged lagging session, the quantum of a dirty session i is redistributed among leading sessions even if there exist clean backlogged in sync. sessions. The authors [3] argue that this is the case because the scheduler wishes to maintain the sessions insync. Due to the random and time varying nature of the wireless channel, however, preserving the insync nature of the sessions throughtout is not really possible. In UF-DRR, therefore, we redistribute the excess quantum of a dirty session to insync sessions before considering the leading sessions, which gives better fairness.

Observe that neither one of the above modifications increases the complexity of UF-DRR beyond $O(N)$, which is of the same order as that of WDRR in a wireless environment, and is inevitable when the scheduler has to search for clean sessions to schedule from among the set of N sessions.

A. Algorithm for UF-DRR

Consider the model in Sec. III with a set N of sessions sharing output capacity in proportion to their normalized reserved rates r_i. Let N_{lag}, N_{lead} and N_{insync} be the sets of clean backlogged lagging, clean backlogged leading and insync sessions, respectively. Also, let N_{dirty} be the set of dirty backlogged sessions. The algorithm works as follows. At the start of each round:

– Allocate quantum Q_i to all clean backlogged lagging and insync sessions, i.e., for all sessions $i \in \{N_{lag}, N_{insync}\}$, add Q_i worth of quantum to their Deficit Counter, DC_i.

– Divide the clean backlogged leading sessions into those with $lag_i > Q_i$, and those with $lag_i \leq Q_i$. From each flow in the former set take quantum Q_i, while from each flow in the latter set take quantum lag_i, leaving $Q_i - lag_i$ with the flow. Sum these quanta to compute *Excess Quantum* and increment the lag values of all affected sessions by the amount of quantum relinquished. Redistribute *Excess Quantum* among all clean backlogged lagging sessions (i.e., to sessions in the set N_{lag}) in proportion to their lags and increment the lag of each by the share of Excess quantum received. If there do not exist clean backlogged lagging sessions (i.e., the set N_{lag} is empty), allocate quantum Q_i to each session $i \in N_{lead}$.

– Sum the quantums of all dirty backlogged sessions (i.e., from each session $i \in N_{dirty}$), into the variable *Excess Quantum*, and increment their lag values by Q_i.

- if there exists a clean backlogged lagging session (i.e., the set N_{lag} is non-empty), distribute *Excess Quantum* among all sessions of N_{lag} in proportion to their lags and decrement their lag values by the amount allocated.
- if there is no clean backlogged lagging sessions (i.e., the set N_{lag} is empty), and if there exists a clean backlogged insync session (i.e., if set N_{insync} is non-empty), distribute the remainder of *Excess Quantum* equally among all sessions of the set N_{insync}.
- if set N_{insync} is also empty (i.e., there are no clean backlogged insync sessions) and the set N_{lead} is non-empty (i.e., there exist clean backlogged leading sessions), redistribute *Excess Quantum* to sessions in that set in *inverse proportion* to their lag values so that sessions that lead more get a lesser share of *Excess Quantum*.

V. Simulation Model and Parameters

A. Simulation Model

Our simulations are based on the system model of Sec. II-A, with the SS's equidistant from the BS (to eliminate location-dependent effects). All our simulations were conducted using Matlab. We consider the TDD mode of 802.16 and the polling mode of bandwidth request, where the BS polls each SS at the beginning of every frame to determine its rate requirements. Thus, the queue status for all the flows is available at the BS at the beginning of each frame.

The wireless channel between the BS and SS is considered to be a Rayleigh fading channel with coherence time greater than a frame duration. Thus, channel state is available to the BS at the start of each frame and does not change for a frame.

B. Simulation Parameters

Our simulations were done for the IEEE 802.16 PMP architecture with one BS and 25 SSs, with one flow per SS. The frame length was 10 ms, with the downlink and uplink subframes being equal. The parameters used in our simulations are detailed in Table I.

At the start of each frame, the BS compares the measured Signal to Interference plus Noise Ratio (SINR) for each channel to a predefined SINR threshold, to determine whether the channel is good or bad. We observed performance under 3 different wireless environments: "Good", "Moderate", and "Poor" environments, where the channel state is good 70%, 50%, and 30% of the time, respectively, obtained by setting $P_G = 0.7, 0.5, 0.3$, respectively, in the 2-state channel model of Sec. II-A.

Traffic at each SS is Poisson with mean λ packets per frame and all packet arrivals occurring at the beginning of a frame. Each flow has specific QoS parameters associated with it, specifying bandwidth (converted into physical-slot equivalents) and delay (in frames). Each SS has an infinite queue so that packets are never dropped due to queue overflow, but rather only upon a violation of the specified delay bound, allowing us to measure how well our scheme provides bounded delay. The maximum achieved load for each scheme is determined by varying the average input rate, assuming no delay bound and infinite queues, and observing the average queue length. The maximum load is the highest possible input rate for which the average queue length remains bounded, ensuring that our measurements are all made on a stable system.

Parameter	Value
No. of Users (N)	25
Simulation Interval	5000 Frames
Frame Length	10 ms
Channel Bandwidth	25 MHz
Symbol Rate	20 Mbaud
Output Link Capacity	40 Mbps
Nyquist roll off factor	0.25
Physical Slots per frame	50000
PSs per UL-subframe	25000
Symbols per Slot (n)	4
Modulation Scheme	QPSK
Bits per Symbol	2
Channel Model	2-State ON-OFF
SNR Threshold (Good)	20 dB
SNR Threshold (Moderate)	23 dB
SNR Threshold (Bad)	26 dB
Packet Arrivals	Poisson (Mean rate λ: 6-20 packets per frame)
Packet Size	Uniform: 1-100 bytes
Max. Packet Size	100
Max. Delay	50 ms
Queue Size	∞ (Packets dropped only on delay bound violation)

TABLE I
SIMULATION PARAMETER VALUES

VI. SIMULATION RESULTS AND ANALYSIS

In this section, we compare the performance of UF-DRR vis-a-vis DRR and WDRR.

We first show, in Fig. 2, the user throughput (or the total bits dequeued) for each flow for WDRR and UF-DRR, under moderate channel conditions ($P_G = 0.5$) and 50% load. The plots clearly demonstrate that WDRR is biased against the lower numbered flows, as argued in Section IV. By contrast, the bits dequeued (served) from each flow in UF-DRR are within a small bound of each other, illustrating the fair behavior of our scheduling algorithm. Our extensive simulations demonstrate that a similar bias is present in WDRR in the average per-packet delay experienced by different flows and in the total bits dropped from different flows (which is consistent with what one would expect based on the plots in Fig. 2). Indeed, the spread of the throughputs of different flows in UF-DRR, compared to that for flows in WDRR, is almost 30-40% less, making UF-DRR that much fairer in its flow bandwidth allocation.

We next compare the throughput of DRR, WDRR, and UF-DRR, under different channel conditions and varying loads (cf. Figs. 3, 4, 5). We observe that for all loads and channel conditions, the performance of DRR is worst, because it blindly allows a user to transmit a quantum Q_i worth of bits per round irrespective of that user's channel condition, leading to excessive packet loss for poor channels. Further, we see that the maximum throughput achieved by UF-DRR is consistently more than that for WDRR for the same input load. Also, the performance difference becomes more pronounced as the channel worsens, with UF-DRR performing, on average, about 10-15% better than WDRR at moderate to high loads, demonstrating that UF-DRR also handles channel degradation more gracefully than WDRR. (Note that in this comparison, the delays and drop probabilities experienced by flows under WDRR and UF-DRR are not equal. If we consider their performance under the same delay bound, as we do next, the performance of UF-DRR is even better.)

Finally, we contrast the average delay and packet drop probability versus load for DRR, WDRR, and UF-DRR in Figs. 6, 7, 8 (for delay), and in Figs. 9 and 10 (for drop probability). We point out that since DRR transmits packets without regard to the channel state, packets in DRR are never dropped due to a violation of their delay bound. They are, however, lost with very high probability, when they are transmitted in a frame in which the output channel condition is bad, making the effective throughput of DRR very poor in a wireless environment (as one would expect).

We see that even under different channel conditions, UF-DRR regularly outperforms WDRR, because it removes the bias against earlier flows that exists in WDRR, resulting in a fair and efficient distribution of the quantum. This prevents packets in a queue from waiting, and results both in lower delay and in lower packet drop probability. Further, we observe from Figs. 7 and 8 that for the same average load, UF-DRR achieves uniformly lower delay than WDRR, under all channel conditions, with the delay of UF-DRR being lower by between 25-35% at moderate loads to almost 50-60% at high loads. Note also, that for a given delay bound (say, 30 ms under moderate channel conditions, where the channel is good only 50% of the time), UF-DRR achieves throughput that is between 30-40% better than that achieved by WDRR. This additional throughput translates directly to more supportable traffic (and, hence, higher revenue) for the operator, making UF-DRR an attractive choice in emerging WiMax networks.

VII. CONCLUSION

In this paper, we presented the UF-DRR scheduling algorithm for providing QoS guarantees and fairness in broadband wireless networks, and compared its performance to that of DRR and WDRR in an IEEE 802.16 environment, under different channel conditions and over a range of loads. Our results demonstrate that under

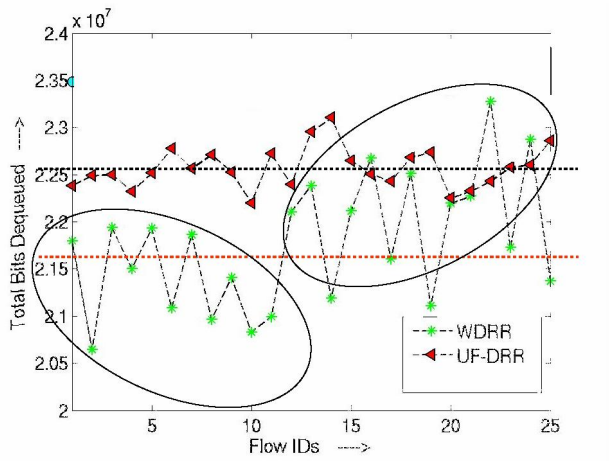

Fig. 2. User Throughput under Moderate Channel and 50% Load

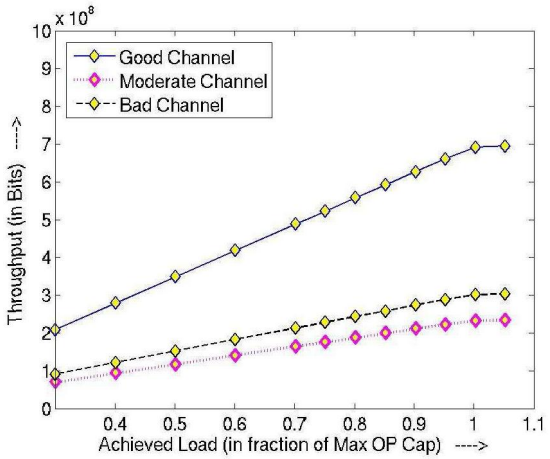

Fig. 3. Throughput vs Achieved Load for DRR

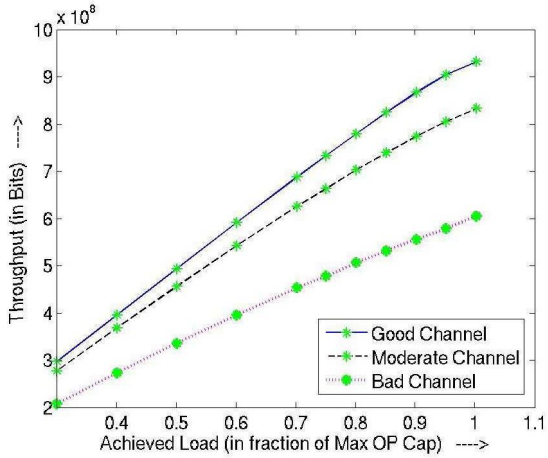

Fig. 4. Throughput vs Achieved Load for WDRR

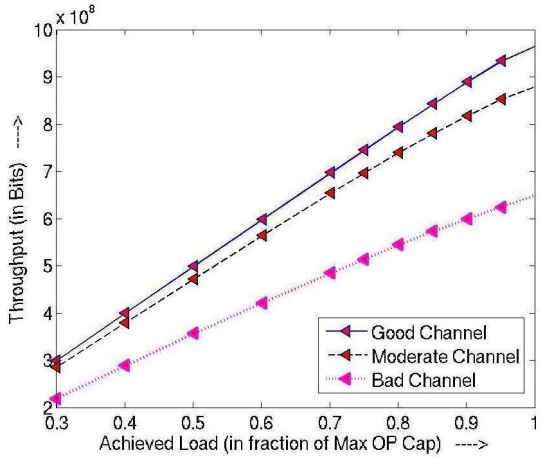

Fig. 5. Throughput vs Achieved Load for UF-DRR

all fading characteristics and under all loads, UF-DRR is more efficient at allocating bandwidth and ensuring fairness than DRR or WDRR. By updating the status of user queues only once every M frames, our scheme is also usable in broadband wireless networks where the queue status is not frequently available.

Several interesting directions of work are are possible from here. One is to consider the performance of these schemes when users with a diverse set of requirements are to be scheduled. Another is to carry out simulations for not one (as has been done here), but multiple classes of traffic. Finally, the IEEE 802.16 standard supports extensive adaptive modulation and coding (AMC) schemes, which enable the system to optimize the throughput based on propagation characteristics. An interesting extension of our work would be to evaluate the benefit of combining AMC with our proposed algorithm.

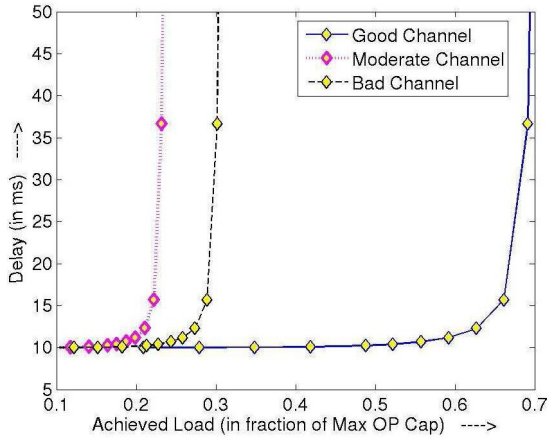

Fig. 6. Delay vs Achieved Load for DRR

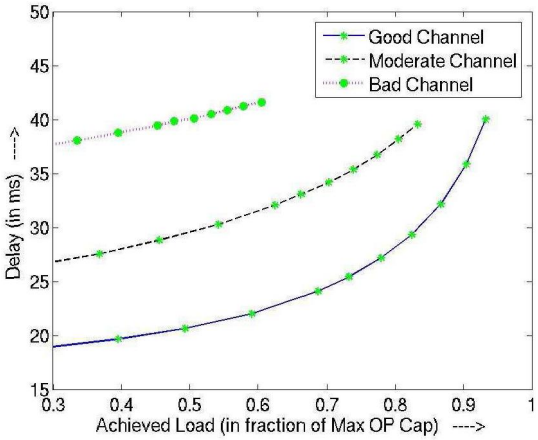

Fig. 7. Delay vs Achieved Load for WDRR

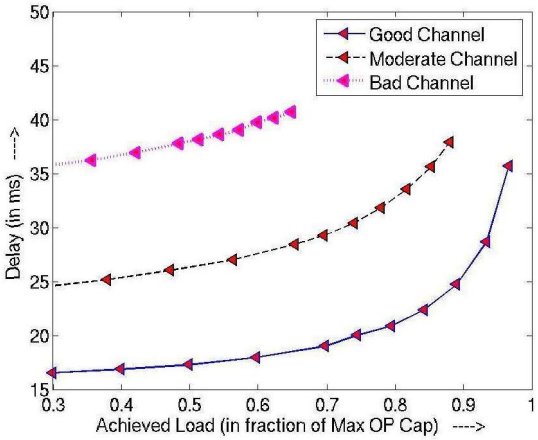

Fig. 8. Delay vs Achieved Load for UF-DRR

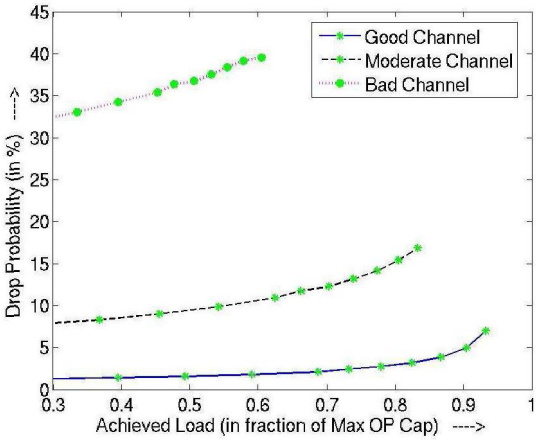

Fig. 9. Drop Probability vs Achieved Load for WDRR

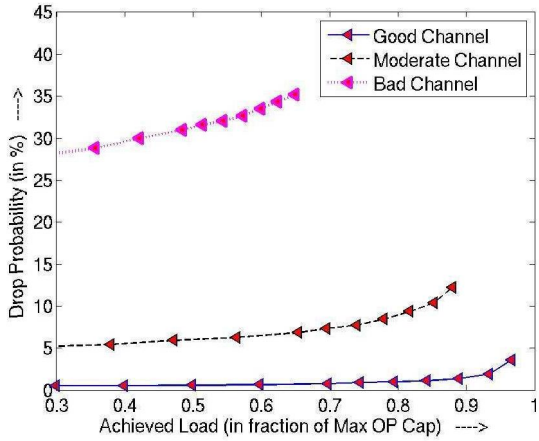

Fig. 10. Drop Probability vs Achieved Load for UF-DRR

REFERENCES

[1] C. Eklund, R. B. Marks, K. L. Stanwood, and S. Wang, "IEEE standard 802.16: A technical overview of the WirelessMAN air interface for broadband wireless access," *IEEE Communications Magazine*, vol. 40, pp. 98–107, June 2002.

[2] A. Ghosh, D. R. Wolter, J. G. Andrews, and R. Chen, "Broadband wireless access with WiMax/802.16: Current performance benchmarks and future potential," *IEEE Communications Magazine*, vol. 43, pp. 129 – 136, February 2005.

[3] H. Fattah and C. Leung, "An efficient scheduling algorithm for packet cellular networks," in *Proc. VTC*, vol. 4, pp. 2419–2423, September 2002.

[4] S. Redana, M. Lott, and A. Capone, "Performance evaluation of point-to-multi-point (PMP) and mesh air-interface in IEEE Standard 802.16a," in *IEEE VTC2004 Fall*, September 2004.

[5] H. Wang and N. Moayen, "Finite state Markov channel-A useful model for radio communication channels," *IEEE Trans. on Vehicular Technology*, vol. 44, no. 1, pp. 163–171, February 1995.

[6] S. Lu, V. Bharghavan, and T. Nandagopal, "Fair queuing in wireless networks: Issues and approaches," *IEEE Personal Communication Magazine*, vol. 6, no. 1, pp. 44–53, February 1999.

[7] V. Bharghavan and T. Nandagopal, "Design and analysis of an algorithm for fair service in error-prone wireless channels," *ACM/Baltzer Wireless Neworks Journal*, vol. 6, no. 4, pp. 323–343, September 2000.

[8] M. Shreedhar and G. Varghese, "Efficient fair queuing using deficit round robin," *IEEE/ACM Trans. on Networking*, vol. 4, no. 3, pp. 375–385, June 1996.

[9] P. McKenney, "Stochastic fairness queuing," in *Proc. INFOCOM*, vol. 2, pp. 733–740, June 1990.

[10] P. Goyal, H. M. Vim, and H. Chen, "Start-time fair queuing: A scheduling algorithm for integrated services packet switching networks," in *ACM SIGCOMM*, vol. 26, pp. 157–168, October 1996.