Application-oriented Multimedia Scheduling Over Lossy Wireless Networks

Xiaoqiao Meng, Hao Yang and Songwu Lu
UCLA Computer Science Department, Los Angeles, CA 90095
E-mails: {xqmeng, hyang, slu}@cs.ucla.edu

***Abstract*— This work seeks a better understanding of the relations between the better network service provided by QoS-oriented wireless packet scheduling and the actual benefits perceived by the multimedia applications that use them. Through extensive simulations driven by real (multimedia and wireless channel error) traces, we observe that in general, there is a performance gap between application perceived QoS and network QoS provided by the wireless fair packet scheduler. This gap tends to increase further as the channel error rate aggravates. The exact distribution of channel errors greatly affects the multimedia application performance, but its impact on network QoS is much smaller. We then present the solution, which is based on Idealized Weighted Fair Queuing (IWFQ) to further improve three popular multimedia applications' performance.**

I. Introduction

The emerging wireless networking technologies such as 3G, 4G, and IEEE 802.11a offer orders-of-magnitude higher throughput, ranging from a few hundred kilobits per second to about 54Mbps, thus making it feasible to support real-time video and audio multimedia applications. To function properly, real-time multimedia applications require effective network quality of service (QoS) support in terms of throughput, delay, jitter and loss ratio. To provide network QoS, researchers have proposed various packet scheduling algorithms, notably through wireless fair queuing (see [6][7][8][9][10] for a few examples). These wireless fair scheduling proposals seek to provide packet-level throughput and delay bounds over the error-prone wireless channel.

While the performance benefit in terms of network QoS has been convincingly demonstrated by each of these earlier proposals [6][7][8][9][10], what is missing is how well such network performance gain can be translated into the application domain. Therefore, in this work, we seek a better understanding of the relation that exists between the better network service provided by wireless packet scheduling and the actual benefits as perceived by the applications that use them. Since this is obviously a broad topic with possibly as many different answers as there are applications, we narrow our study to three specific multimedia applications, i.e., MPEG-4 streaming video, H.263 streaming video, and real-time audio. We are interested in evaluating the performance of wireless fair scheduling algorithms, under various realistic wireless channel conditions, from the multimedia application's perspective.

We make two contributions in this work:

- Using three popular multimedia applications for case studies, we evaluate the QoS-oriented wireless fair scheduler from the application QoS perspective. The evaluation is based on extensive simulations driven by real (multimedia and wireless channel error) traces. The impact of wireless channel errors upon application QoS and network QoS is carefully studied. We find out that, (a) channel error will incur a performance gap between network-level QoS and application perceived QoS, and (b) channel error distribution hardly affects network-level QoS but may greatly affect application-layer QoS. Each message (such as I, P, B frames) has different application semantic, thus carrying different weight in application QoS.
- With simple modifications to the current packet scheduler by considering application message semantics, we are able to significantly improve application perceived QoS as demonstrated in the simulations. Fundamentally, the proposed design is to shift the paradigm from the current packet scheduling to message scheduling. The underlying principle is similar to *application-level framing*, which has been proven as a very effective technique to improve protocol performance in modern computer networks.

The rest of this paper is organized as follows. Background introduction of the three multimedia applications and the wireless packet scheduler are provided in Section II. The proposed solution approach is discussed in Section III. An evaluation of the proposed solution, with detailed discussions, is presented in Section IV. Section V discusses the related work and Section VI concludes this paper.

II. Background

A. Network Model and Channel Error

We consider a packet cellular network with a high-speed wired backbone and multiple wireless cells. In each cell there is a base station performing scheduling of packets transmissions for the mobile hosts in the cell.

High channel error rate is an inherent characteristic of the wireless networks [1]. For the purpose of this paper, we have conducted real measurements to collect channel error traces in our departmental WaveLAN. The mobile host used is a notebook equipped with a Lucent 802.11b compatible wireless card. During the measurement, we collect the channel error patterns in three different situations, i.e., *Static Mild* error trace is collected by placing the notebook stationary inside the departmental building. *Static Heavy* trace is collected by placing the notebook near the gate of the building. *Mobile Heavy* error trace is collected by frequently roaming outside and inside the building.

	Static Mild	Static Heavy	Mobile Heavy
Error Ratio	0.46%	4.90%	4.68%

TABLE I
MEASURED CHANNEL ERROR RATIOS

The measurement results of Table I show that, the channel error ratio in wireless networks is indeed much higher than their wired counterparts. High channel error ratio will reduce the effective bandwidth available to applications, thus negatively affecting application performance. This problem is even more severe for multimedia applications which typically require bandwidth and delay bounds for effective operations.

B. Multimedia Applications

In this paper, we focus on three multimedia applications, i.e., MPEG-4 encoded streaming video, H.263 encoded streaming video and real-time audio. MPEG-4 and H.263 are two popular video compression standards to support streaming video service in low bit-rate environments. Audio has been the killer application for cellular phone networks.

MPEG-4 MPEG is developed by the Moving Picture Experts Group (MPEG) of ISO/IEC as the standard method for compression, processing and representation of moving pictures, audio and their combination. MPEG-4 is currently the most stable and widely used version of MPEG standard.

H.263 H.263 is another widely used video compression standard recommended by the International Telecommunications Union (ITU). Its design is specially aimed at video applications with low bit rates, such as video-conferencing and video-telephony.

Real-time Audio The third multimedia application considered here is real-time audio. Such kind of applications has been employed by some popular Internet audio services such as [18]. Real-time audio typically has a low bandwidth requirement, ranging from 3.0Kbps to 7.6Kbps, and its packet size is comparatively small (several hundred bytes).

C. Idealized Wireless Fair Queueing Algorithm

The Idealized Wireless Fair Queueing Algorithm (IWFQ) is among the first wireless scheduling algorithms that are able to accommodate bounded channel errors and provide performance assurances in terms of delay, throughput and loss ratio. This work is based on IWFQ, but the results also hold true for other scheduling proposals. The operation of IWFQ is summarized as follows [6]:

- The arriving packet n of flow i is assigned two tags, the same as in WFQ [5]: a start tag $s_{i,n}$ and a finish tag $f_{i,n}$.

$$s_{i,n} = \max\{v(A(t_{i,n})), f_{i,n-1}\} \quad (1)$$
$$f_{i,n} = s_{i,n} + L_P/r_i \quad (2)$$

  where L_P is the packet size and virtual time $v(t)$ is defined as $\frac{dv(t)}{dt} = \frac{C}{\sum_{i\in B(t)} r_i}$. C is the link capacity, r_i is the flow weight of flow i and $B(t)$ denotes the set of backlogged flows at time t.
  Each flow has both a packet queue and a slot queue. The incoming packet is stored in the packet queue, and the tag is recorded in the slot queue. The slot queue and packet queue mechanisms provide a means to answer two questions independently: which flow to schedule at the moment, or which packet from the scheduled flow to transmit at this time.
- For each flow, set its service tag equal to the finish tag of its head of line (HOL) packet. Among the flows perceiving a *clean* wireless channel, select the flow with the smallest service tag in the slot queues and transmit its HOL packet from its packet queue.
- Readjusting tags for lagging flows and leading flows.
  – For a lagging flow i, if there are more than B packets with finish tags less than the virtual time $v(t)$, then only the first B_i packets are retained and the left packets are removed from the packet queue. The slot queue remains intact and maintains the precedence of the lagging flows.
  – For leading flow i, if the start tag of its HOL packet $s_{i,HOL}$ is greater than $v(t)$ by more than $\frac{l_i}{r_i}$, then set $s_{i,HOL} = v(t) + \frac{l_i}{r_i}$ and $f_{i,HOL} = s_{i,HOL} + \frac{L_P}{r_i}$.

In essence, IWFQ seeks to swap the short-term channel allocation between error-prone flows and error-free flows. This way, all flows will receive a fair share of the wireless channel over a larger time interval while the effective channel utilization is also maximized since only error-free flows are scheduled at any time.

III. SCHEDULING DISCIPLINES AND APPLICATION-LEVEL QOS FOR MULTIMEDIA APPLICATIONS

IWFQ and other fair scheduling algorithms [6]-[10] seek to provide flow isolation among different flows while supporting network-level QoS such as delay and throughput in the presence of location-dependent channel errors. However, for multimedia applications, the performance of these scheduling algorithms should be evaluated in terms of application-level QoS, which is the users' perceived satisfaction. In this section, we use MPEG-4, H.263 and real-time audio to demonstrate that these fair scheduling algorithms should be modified to further improve the application-level QoS, especially in the presence of channel errors.

The application-level QoS is heavily dependent on the message semantics of the data packets and the encoding scheme being employed by the multimedia applications. Consequently, our proposed scheduling algorithms rely on the specific encoding scheme. In the following, for each multimedia application, we first introduce its coding scheme, then present the corresponding modified scheduling algorithm.

A. MPEG-4

MPEG-4 scheme encodes the information of a scene into multiple *Group of Pictures* (GOP). Each GOP is further divided into three types of frames, i.e., I, P and B frames. The I frame (Intra-coded) is simply coded from a picture frame. The

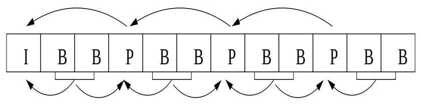

Fig. 1. MPEG-4 Frame Sequence (arrow means *reference to*)

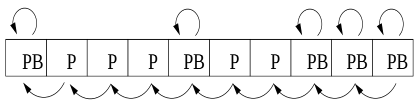

Fig. 2. H.263 Frame Sequence (arrow means *reference to*)

P frame (Predicted-coded) is coded by referring to the most recent I or P frame. Thus at the decoder side, to reconstruct a P-frame, the most recent I or P frame has to be used. The third type of frame is B-frame (Bidirectional-coded). It is coded by referring to the two closest I or P frames, one in the past and one in the future. Such a reference relationship is illustrated in Figure 1.

In the simplest case (as in our simulation), a MPEG-4 stream is constructed by periodic frame sequences which take the form *IBBPBBPBBPBB*. Such a frame sequence is called GOP. Each GOP is responsible for showing about 0.5-second movie. When reconstructing pictures at the decoder side, *I* and *P* frames are generally considered more important than *B* frame. Once an *I* or *P* frame is lost, the entire GOP is useless, which means 0.5-second degraded movie quality from the user perspective. Another critical factor affecting the performance of MPEG-4 is the GOP delay. GOPs with long delay will disturb users' perception. Based on these two observations, we define the application-level QoS of MPEG-4 video as the percentage of successfully received GOPs with delay less than a given threshold.

When evaluating the scheduling performance through this metric, we can show that IWFQ does not optimize the performance. First, once an I or P frame in a GOP is corrupted by channel errors[1], the whole GOP is useless. Continuing to transmit the remaining frames of this GOP, as IWFQ does, is not able to improve the percentage of successfully received GOPs. In addition, this will increase the delay of the following GOPs. Second, it is also meaningless to schedule frames in a GOP which have a long queuing delay.

To address these issues, we propose two simple modifications to IWFQ as follows:

- When an I or P frame in a MPEG-4 flow is lost, the scheduler should discard all the remaining frames in the *packet queue* until another I frame, which starts a new GOP. Note that we only remove the frame packets from the *packet queue* while keeping the *slot queue* intact. This way, the scheduling precedence for this MPEG-4 flow and its long-term fairness is still preserved.
- Before transmitting the I frame of each GOP, the scheduler examines the queuing delay of this GOP. If it already exceeds the given threshold, the scheduler removes the whole GOP from the *packet queue* while keeping the *slot queue* intact. To record the arrival time of each GOP, we create a *time-tag queue* in addition to the *packet queue* and *slot queue*. When a new I frame arrives, a new time tag is created and assigned the arrival time of the new I frame, then this time tag is added to the *time-tag queue*.

[1]One frame may be fragmented into multiple packets during transmission. In this case, when one packet is corrupted, the total frame is considered to be corrupted.

An extreme case is that, if the first I frame of the GOP is corrupted by channel errors, IWFQ will continue to send the left 11 frames. However, the modified IWFQ will directly discard all of them, thus the delay of the next GOP is decreased by $\frac{11*L_P}{r}$, where L_P is the maximum frame size and r is the flow weight. Such an improvement is significant when the channel error is heavy, as shown in Section IV.

B. H.263

In contrast to the object-based MPEG-4, H.263 is a frame-based coding scheme. H.263 employs inter-picture prediction to reduce temporal redundancy and also provides several other coding options to improve coding efficiency. In the H.263 traces used in our simulation (provided by [17]), there only exist I frame and PB frame. The PB frame is generated by an optimization option of H.263. This option compresses a P frame and a B frame into one PB frame, thus decreasing the bit rate without seriously deteriorating the picture quality. Similar to MPEG-4, decoding a P frame relies on the previous P frame. Decoding PB frame is more complicated, for the PB frame is actually consisting of a P frame and a B frame. Decoding this P frame partly depends on previous P frame, while decoding the B frame partly depends on its preceding P frame and the P frame in the same PB frame. One example of this decoding relationship is depicted in Figure 2. Note that the sequence of P and PB frames in H.263 streams is random, largely depending on the video context.

Performance measures for H.263 are defined the same as MPEG-4. To improve the application performance of H.263, we need to modify IWFQ. The first modification we have proposed to better support MPEG-4, is no longer applicable simply because H.263 coding scheme does not possess any periodic structure like GOP in MPEG-4. However, we can still apply the second modification to improve the QoS of H.263. The scheduler discards any backlogged frames with queuing delay larger than a given threshold. Due to the special coding format of H.263, it is more subtle to compute the frame delay here. For example, consider a sequence *P PB* in the *packet queue* of a H.263 flow. In this sequence, the second entity, a PB frame, is responsible for delivering two movie frames. The first movie frame is reconstructed from the previous P frame and the PB frame itself. Thus the queuing delay for this movie frame should be computed as the time interval from the time the first P frame enters the *packet queue* until the time the PB frame is scheduled. However, computation of the queuing delay for the second movie frame is slightly different. Since it is solely reconstructed from the PB frame, its queuing delay should be computed as the time interval from the time when the PB frame arrives until the time when it is scheduled.

C. *Real-time Audio*

There are multiple compression schemes for real-time audio, such as RPE-LPC, CELP, MP3 and etc. Because real-time audio carries a 2D signal which has fewer temporal redundancy, all these compression schemes have small frame size (e.g., 245 bytes) and less content interdependence between different packets. For simplicity, we consider each packet as an independent frame. The frame speed of real-time audio varies greatly, due to the burstiness of human speech.

Determined by the characteristics of real-time audio applications, users are able to tolerate mild frame loss, but they are extremely sensitive to frame delay or jitter. Researchers from human perception fields have reported that 20 ms is the minimum perceptible delay for human ears [20]. When the frame jitter (inter frame delay) exceeds this threshold, the user is able to perceive voice distortion. Therefore, we define the percentage of frames with delay less than a predefined threshold and jitter less than 20 ms as the application-level QoS of real-time audio streams.

To meet such service requirements of real-time audio and consider the low bandwidth consumption of real-time audio, we decouple the delay allocation from the bandwidth allocation in IWFQ. Specifically, for real-time audio flows, the computation of each packet's finish tag is modified as

$$f_{i,n} = L_P/r_i + \max\{v(A(t_{i,n})) - \delta, f_{i,n-1}\}$$

where δ is a positive parameter.

Another modification is to disable IWFQ's one-step prediction for channel error in the mild error case. When using the standard IWFQ, if the current transmission of audio packets is corrupted by channel error, the audio flow will be forbidden to be scheduled in the next time slot even though it has the least service tag. However, since the performance of real-time audio puts most emphasis on promptness, we consider the negative impact brought by a lost frame as severe as a frame with large jitter. Therefore, we disable the error prediction mechanism of IWFQ when handling real-time audio streams.

D. *Discussions*

In our previous study, we use application-level measurements *frame loss* and *frame delay* to replace the network-level measurements *packet loss* and *packet delay*. Such application-level measurements are directly linked to the real semantics of the underlying data, and thus they are more meaningful in evaluating network design in the presence of multimedia applications where the data packets have rich semantics. The application-level QoS we have used are coarse and simplified. In fact, there has been a surge in the literature addressing the measured quality of multimedia applications [11]. We believe that more accurate measurements can be applied here in a similar manner.

We do not consider packet loss concealment techniques employed by some multimedia applications. Such kinds of techniques are usually deployed at the video source to alleviate the negative impact of packet loss by adding redundant information. We also do not consider various methods of repairing packet loss in the real-time audio stream [15]. We believe that these techniques can be combined with the modified scheduling algorithm to further improve the performance.

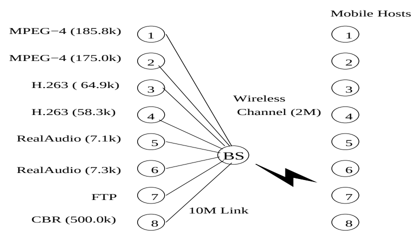

Fig. 3. Simulation Topology

IV. Simulation and Evaluation

In this section, we first use simulations to measure the performance of the modified IWFQ and compare it to traditional scheduling algorithms. Secondly, we use real channel error traces to simulate the negative impact on multimedia application performance brought by channel errors. All the multimedia applications in the simulation are driven by real traces.

A. *Experimental Settings*

The network topology is shown in Figure 3. Each mobile host runs a client application and communicates with a wired multimedia server via the base station. We simulate a heavy traffic scenario with nine flows, including two MPEG-4 video streams, two H.263 video streams, two real-time audio streams, one FTP flow and one CBR (constant-bit-ratio) flow. The simulator we used is *ns-2* [16] with CMU wireless extensions.

All the MPEG-4 streams and H.263 streams are driven by real traces downloaded from the public domain [17] (these traces are captured and encoded from four different movies. The methodology on creating these video traces are described in [17]). Real-time audio traces used in our simulation are based on the empirical model of [19]. To evaluate the impact of wireless channel errors, we also use the aforementioned real channel error traces.

We implement four scheduling algorithms in *ns-2*: First-In-first-Out (FIFO), Weighted Fair Queueing (WFQ), Idealized Wireless Fair Queueing (IWFQ) and the Modified IWFQ (MIWFQ).

B. *Experimental Results and Evaluation*

We now compare the four schedulers under the different channel error traces from both the network and the application perspectives. For each scheduler and each channel error trace, the simulation run lasts for 15 minutes. Due to space limitation, we only present the results for one MPEG-4 flow, one H.263 flow and one audio flow. We first present the observations on the application-level QoS.

1) *Application-level QoS:*

a) MIWFQ Increases Multimedia Application Performance: As can be seen from Table IV, compared to FIFO, WFQ and IWFQ, MIWFQ provides better service to multimedia applications when measuring the application performance in terms of the application-level QoS. This improvement tends to be more significant when the channel error is heavy.

		Zero	Mild	Stationary	Mobile
FIFO	MPEG-4	0.99%	1.63%	5.68%	5.89%
	H.263	1.15%	1.73%	5.77%	5.89%
	Audio	1.12%	1.82%	4.97%	6.09%
WFQ	MPEG-4	0.98%	1.57%	5.69%	6.03%
	H.263	0.92%	1.67%	5.68%	5.75%
	Audio	0.19%	0.46%	4.06%	4.72%
IWFQ	MPEG-4	0.98%	1.48%	4.75%	6.00%
	H.263	0.93%	1.33%	4.67%	6.39%
	Audio	0.19%	1.20%	3.62%	4.57%
MIWFQ	MPEG-4	0.98%	4.03%	20.34%	28.9%
	H.263	0.93%	1.35%	5.00%	5.83%
	Audio	0.20%	0.67%	3.39%	4.78%

TABLE II
NETWORK-LEVEL QOS: PACKET LOSS

		Zero	Mild	Stationary	Mobile
FIFO	MPEG-4	2.54	2.54	2.54	2.54
	H.263	2.51	2.50	2.50	2.50
	Audio	2.54	2.54	2.55	2.55
WFQ	MPEG-4	5.60	5.60	5.60	5.60
	H.263	5.58	5.58	5.58	5.58
	Audio	0.29	0.29	0.29	0.29
IWFQ	MPEG-4	5.60	5.58	5.59	5.66
	H.263	5.58	5.56	5.57	5.66
	Audio	0.29	0.44	0.42	0.43
MIWFQ	MPEG-4	5.60	5.56	4.89	4.68
	H.263	5.58	5.53	4.88	4.68
	Audio	0.29	0.33	0.20	0.36

TABLE III
NETWORK-LEVEL QOS: PACKET DELAY (UNIT: SECOND)

		Zero	Mild	Stationary	Mobile
FIFO	MPEG-4	98.9%	93.8%	70.4%	65.2%
	H.263	98.7%	97.1%	88.3%	86.7%
	Audio	5.9%	6.0%	6.1%	5.9%
WFQ	MPEG-4	98.9%	94.2%	70.9%	61.5%
	H.263	99.1%	96.9%	87.3%	85.9%
	Audio	82.9%	82.9%	83.4%	83.2%
IWFQ	MPEG-4	98.9%	95.3%	74.3%	61.8%
	H.263	99.1%	97.9%	89.7%	86.8%
	Audio	82.9%	74.9%	77.8%	77.4%
MIWFQ	MPEG-4	98.9%	95.7%	82.3%	72.4%
	H.263	99.1%	98.1%	93.5%	91.2%
	Audio	82.9%	77.9%	88.8%	82.7%

TABLE IV
APPLICATION-LEVEL QOS (FOR MPEG-4 AND H.263, IT IS THE PERCENTAGE OF SUCCESSFULLY RECEIVED GOPS WITH QUEUING DELAY LESS THAN 4S; FOR REAL-TIME AUDIO, IT IS THE PERCENTAGE OF FRAMES WITH DELAY LESS THAN 0.5S AND JITTER LESS THAN 20MS)

b) Performance Gap Between Network-level QoS and Application-level QoS: For real-time audio, WFQ, IWFQ and MIWFQ can achieve an application-level QoS around 80% when exposed to any kind of channel error patterns. This indicates that channel error brings little impact to delay/jitter-sensitive applications. For MPEG-4, though MIWFQ increases the packet loss by a factor of 5 (from 4.75% to 20.34%) compared to IWFQ in the stationary case, its application QoS increases from 74.3% to 82.3%. This counter-intuitive result is explained by the capability of MIWFQ to remove useless packets as described in Section III. Therefore, the network-level QoS may not always be faithfully mapped to the application level. A design that achieves high network-level QoS does not ensure high application performance.

c) Channel Error Distribution Affects Application-level QoS: As we can see from Tables II and III, no matter which scheduler is used, the packet loss and delay are almost the same for the mobile and stationary cases. The reason is that, in the mobile and stationary cases, the error ratios are almost equal, i.e., 4.90% and 4.68% respectively. However, in these two cases, the application-level QoS of MPEG-4 is quite different. For example, when using MIWFQ, the application QoS of MPEG-4 is only 72.4% in mobile case, much less than 82.3%, which is for the stationary case. Such a large discrepancy is actually caused by the different error distributions in the mobile and stationary cases. In the stationary case, channel errors are more likely to be clustered. While for the mobile case, channel errors are more likely to be sporadically distributed. Even though the error ratios are similar, these different distributions can have quite different impact on the application QoS of MPEG-4 applications. We illustrate this issue by the following back-of-the-envelope computation.

Assume there are totally N time slots and at most e slots are in error. As we know, the MPEG-4 encoded video stream consists of periodic frame sequence *IBBPBBPBBPBB*. For simplicity, we assume each frame in this GOP needs exactly one slot to be transmitted. Thus, a successfully received GOP means that all its I and P frames are transmitted in clean slots. Now we consider two different error distribution models. The first model is an extreme case of the stationary error pattern, i.e., all of the e slots in error are clustered continuously, thus the maximum number of corrupted GOP is $e/12$. The second model is an extreme case of the mobile error pattern, in which the e slots in error are uniformly distributed among the whole N slots. Therefore, in average, there will be one frame in error for every $12e/N$ GOP. Since this corrupted frame is randomly located among these $12e/N$ GOPs, and we also know in each GOP, the ratio of I and P frames is $1/3$, the probability of having one corrupted GOP among these $12e/N$ GOPs should be $1/3$. Thus the expected total number of corrupted GOP is $1/3 * e$, which is higher than $e/12$. From the above two extreme cases we can see that the less bursty the channel error distribution is, the more damage it brings to the application-level QoS of MPEG-4 applications.

Then, we may ask why the channel error distribution plays such an important role in the application-level QoS while it rarely affects the network-level QoS? The fundamental reason is the different value contained by individual data packet when measured from the application perspective. For example, if we measure the QoS in terms of packet loss, which is a network-level QoS, then every packet, if corrupted by the channel error, brings an equal degradation to the QoS. In this case the distribution of channel errors makes no difference. However, if we measure the QoS from application perspective, due to the different semantics carried by packets, the value of each packet varies.

d) Video Application Performance Is More Sensitive To Channel Error Than Network-Layer Packet Loss: For GOP loss sensitive MPEG-4, if we use IWFQ, the packet loss is 1.63% with mild error, and its application QoS is 93.8%. How-

	10%	20%	30%
FIFO	7.5%	18.9%	29.8%
WFQ	9.2%	20.0%	30.6%
IWFQ	3.3%	8.3%	13.5%

TABLE V
PACKET LOSS OF MPEG-4 FLOW AT DIFFERENT CHANNEL ERROR RATIOS (ERROR TRACES ARE GENERATED BY TWO-STATE MARKOV CHAIN MODEL)

ever, when its packet loss increases to 5.89%, the application QoS drops to 65.2% (in the mobile channel error case). This shows that a small percentage of packet loss may lead to significant application performance degradation. Therefore, in order to provide users high-quality MPEG-4 video services, techniques such as Forward Error Correction may have to be used to further decrease packet loss ratio.

In addition to above observations, we also observe some interesting results on the network-level QoS (see Tables II and III).

2) Network-level QoS:

a) Different Packet Delay Caused By Different Schedulers: From Table III, we observe that the packet delay of real-time audio when using WFQ, IWFQ or MIWFQ is much smaller than that in FIFO. This feature is caused by the mechanism of decoupling bandwidth and delay in WFQ, IWFQ and MIWFQ. For this reason, FIFO is obvious not appropriate when bandwidth-sensitive or delay-sensitive applications are employed . We also note that IWFQ will increase packet delay when channel error ratio is high. This is easy to understand because IWFQ adopts the one-step channel error prediction which may defer scheduling some flows.

b) IWFQ Shows More Advantages For Stationary Hosts: Table II shows that the packet loss using IWFQ is smaller than WFQ, but the performance gain is marginal. It is expected that when channel error is even heavier, this gain of IWFQ will be even more significant. To demonstrate this, we use a two-state Markov chain model to generate heavy error patterns. The simulation results for this error pattern are shown in Table V. We observe that IWFQ results reduce packet loss of WFQ and FIFO by more than 50%. From this result, we can see that IWFQ is more appropriate for stationary cases where the channel error is easier to predict.

c) Performance Gain of IWFQ Is Questionable For Mobile Users: Though IWFQ prevails over WFQ under stationary scenarios, its benefit is questionable for mobile error cases. As we observe from Table II, when mobile error pattern is employed, the packet loss of MPEG-4 is 6.03% when using WFQ, while it is 6.00% for IWFQ. The performance gain of IWFQ is not so significant as the stationary channel error case. The explanation is that, in the mobile case, channel error is less predictable as the user roams, and the one-step prediction adopted by IWFQ is more likely to fail.

V. RELATED WORK

Several recent and ongoing projects are directed at integrating application-oriented QoS in the network design in the wired networking domain [13][12]. [13] considered providing application-level requirements, i.e., application-level delay, loss ratio and fairness, in network-layer protocol designs. The difference between their work and this paper is that our focus is on providing application-level QoS through wireless scheduling, and we also study the impact of wireless channel errors. [12] also considered supporting application-level QoS for complex network services in wired networks. Their approach is built on a concept of service brokers.

[14] assigns a priority to each speech packet based on the application structure and coding scheme. The paper uses prioritized packet discarding to improve performance of the speech applications. Our proposals in improving MPEG-4 application performance bear similarity with this prioritized discarding approach but in the wireless domain.

VI. CONCLUSIONS

In this paper, we evaluate the traditional network QoS-oriented wireless fair scheduler from the application QoS perspective. We find out that there is a gap between these two performance measurements. We also study the impact of channel errors upon the network and application QoS. Finally, to improve the application QoS, we propose modifications to existing wireless scheduling algorithms upon three multimedia applications.

REFERENCES

[1] D. Eckhardt and P. Steenkiste, A trace-based evaluation of adaptive error correction for a wireless local area network, *Mobile Networks and Applications (MONET)*, Special Issue on Adaptive Mobile Networking and Computing, 1998.
[2] D.D.Clark, S.Shenker and L.Zhang, Supporting real-time applications in an integrated services packet network: architecture and mechanism, *SIGCOMM'92*, 1992.
[3] Y.Bernet et. al., A framework for differentiated services, November 1998. *Internet Draft*, draft-ietf-diffserv-framework-01.txt.
[4] R.Braden, D.Clark and S.Shenker, Integrated services in the Internet architecture: an overview, June 1994, Internet RFC 1633.
[5] A.Demers, S.Keshav and S.Shenker, Analysis and simulation of a fair queueing algorithm, *SIGCOMM'89*, 1989.
[6] S.Lu, V.Bharghavan and R.Srikant, Fair scheduling in wireless packet networks, *SIGCOMM'97*, 1997.
[7] S.Lu, T.Nandagopal and V.Bharghavan, Fair scheduling in wireless packet networks, *MOBICOM'98*, 1998.
[8] P.Ramanathan and P.Agrawal, Adapting packet fair queueing algorithms to wireless networks, *MOBICOM'98*, 1998
[9] M.Srivastava, C.Fragouli and V.Sivaraman, Controlled multimedia wireless link sharing via enhanced class-based queueing with channel-state-dependent packet scheduling, *INFOCOM'98*, March 1998.
[10] T.S.Ng, I.Stoica and H.Zhang, Packet fair queueing algorithms for wireless networks with location-dependent errors, *INFOCOM'98*, March 1998.
[11] ITU-T P.800, Methods for subjective determination of trnsmission quality.
[12] P.Chandra, A.L.Fisher, C.Kosak and P.Steenkiste, Network support for application-oriented QoS, *IWQoS'98*, 1998.
[13] G.G.Xie, S.S.Lam, An efficient network architecture motivated by application-level QoS, *Journal of High-Speed Networking*, 6(3), pp.165-179, 1998.
[14] D.W.Petr, L.A.Dasilva and U.S.Frost, Priority discarding of speech in integrated packet networks, *Journal of Selected Areas in Communication*, 7(5), pp.644-656, 1989.
[15] M.Podolsky, C.Romer and S.McCanne, Simulation of FEC-based error control for packet audio on the Internet, *INFOCOM'98*, March 1998.
[16] CMU Monarch extensions to ns. http://www.monarch.cs.cmu.edu/
[17] http://www-tkn.ee.tu-berlin.de/ fitzek/TRACE/trace.html
[18] http://www.broadcast.com
[19] Kun-chan Lan, John Heidemann, Multi-scale Validation of Structural Models of Audio Traffic, *ISI-TR-544*, USC Information Sciences Institute.
[20] http://www.ee.columbia.edu/ dpwe/e6820/slides/E6820-L04-percep.pdf