Scheduling Delay-Constrained Data in Wireless Data Networks

Xiaoqiao Meng*, Thyaga Nandagopal◇, Starsky H.Y. Wong*, Hao Yang‡, Songwu Lu*
*NEC Laboratories America, Princeton, NJ 08540
◇Bell Laboratories, Lucent Technologies, Holmdel, NJ 07733
*UCLA Computer Science, Los Angeles, CA 90095
‡IBM T.J. Watson Research Center, Hawthorne, NY 10562
E-mails:*xqmeng@nec-labs.com, *{hywong1, slu}@cs.ucla.edu, ◇thyaga@lucent.com, ‡haoyang@us.ibm.com

***Abstract*— In modern cellular networks, the channel quality is dynamic among users and also over time. The time-granularity for such dynamics is significantly diverse - either slow or fast compared to packet transmission time. Because of these issues most existing scheduling policies can not work consistently well. In this work, we propose a scheduling policy with performance relatively insensitive to the time-granularity of the dynamics of channel quality. Our policy is self-adaptive to the scale of channel variations by using an ensemble of proposed algorithms. The proposed scheduling policy is proved to have a worst-case performance bound in the existence of both slow and fast time-varying channels. Simulation results confirm that the policy better tolerates channel variations than other popular schemes such as EDF and the Greedy algorithm.**

I. Introduction

Next-generation wireless data networks such as 3G and WiMAX are expected to support a wide variety of real-time, interactive applications (e.g., VoIP, mobile games, mobile TV). Due to the nature of these applications, designing a delay-sensitive scheduling policy, which can provide stringent delay requirements, is important and becomes the focus of this paper.

Although delay-sensitive scheduling issue has been the topic for a large number of work ([1][2][3][4][5], etc.), most of them assume the underlying transmission rate is constant, i.e., all data packets are transmitted on the wireless channel at the same rate. Unfortunately this assumption does not comply with the real wireless channel characteristics. In fact the wireless channel quality exhibits significant dynamics. Many wireless communication systems even offer multi-rate capability by adapting the coding, modulation and error-correction schemes to the varying channel quality. For example, the current deployment of 1xEV-DO (also known as High-Data-Rate (HDR)) [6] supports 11 discrete data rates, ranging from 38.4 Kbps to 2.4 Mbps. In addition to the fact that data rate is varying over time, the variation itself exhibits vastly different time-granularity due to the Doppler effect, that is, the data rate may keep relatively constant (e.g., change once for hundreds of time slots) when the Doppler effect is moderate. It can change much faster (at a granularity of a few slots) when the Doppler effects are strong.

The above channel characteristics have important implications for packet scheduling design. We can illustrate this by comparing the time-granularity of channel variation with delay requirement of typical delay-sensitive applications. The end-to-end delay for VoIP must be less than 300-400 ms (following ITU-T recommendation G.114 [7]), the typical end-to-end delay for video traffic is about 200 ms [8]. Given such end-to-end delay requirements, the maximum delay at the base-station should be much smaller, say, tens of million seconds. Within this time period, the wireless channel quality may fall into two kinds of situations: almost constant (i.e., moderate Doppler effects), or rapidly changing in every timeslot (i.e., strong Doppler effects). We call them as *slow time-varying channel* and *fast time-varying channel*, respectively. Unfortunately, none of the existing scheduling designs can work well in both situations.

In this paper, we propose a new scheduling policy which is *self-adaptive* to the time-granularity of channel variation and performs consistently well in both slow and fast scenarios. Our contributions are two-fold:

- We design a scheduling policy adapt to the time-granularity of channel variation. The proposed policy consists of an ensemble of scheduling algorithms. Two algorithms developed in this work, OPT_UNDERLOAD and ED-EDF, perform well in the slow time-varying situation. On the other hand, the Greedy algorithm [9] performs well in fast time-varying situations. When the proposed policy is employed, the scheduler at the base station differentiates a slow time-varying period from the fast one, and uses the best algorithm accordingly.
- The performance of the proposed policy is also insensitive to traffic load. For the case that the scheduler is under-loaded[1], we design an algorithm OPT_UNDERLOAD to achieve optimality. When the scheduler is over-loaded and dropping a few packets becomes inevitable, we develop another algorithm *Early-Dropping EDF (ED-EDF)* which has an intelligent packet dropping mechanism to reduce the amount of dropped throughput whenever over-load occurs. We prove that the ratio of dropped throughput between ED-EDF and the optimal solution is

[1]Here, *Underloaded (overloaded)* means that an optimal scheduling algorithm can (cannot) transmit all the packets in the buffer.

bounded by a constant. From this perspective, ED-EDF outperforms both the EDF and Greedy since the latter two have unbounded ratios.

The proposed scheduling policy is analytically studied and further evaluated by simulations. Through analysis, we show when both slow and fast time-varying periods exist, the proposed policy achieves a higher worst-case performance bound than the popular EDF and the Greedy algorithm. Simulation results further confirm the consistency of the proposed policy.

The rest of paper is organized as follows. Section II introduces background knowledge and related work. Section III gives an overview of our proposed scheduling policy. The proposed policy consists of several components and is discussed from Section IV to VI. Section VII presents the results of our evaluation through simulations and Section VIII concludes the paper.

II. Background and Related Work

A. System model

We consider wireless cellular networks in which channel access is scheduled in slotted time intervals. Each time slot has a fixed duration (e.g., 1.67 ms in 1xEV-DO system). Data transmission rate might vary among time slots. A packet scheduler implemented at the base station follows a polling-cycle model ([10]). More specifically, the base-station periodically polls each mobile host and collects information (e.g., deadline, packet size) about all their buffered packets. The scheduler then arbitrates the transmission order for these buffered data during the current polling cycle before moving to the next cycle. We assume that channel prediction is supported by the system. By channel prediction, the scheduler can forcast the channel condition and further data rate within a certain time range. We note that channel prediction is already feasible in several real systems such as 1xEV-DO [6].

In this work, we focus on scheduling downlink delay-constrained data flows. Multiple flows are served by the scheduler. Each flow has continuously buffered packets each of which has known deadline. A packet not transmitted before its deadline is automatically discarded. We assume that packet length is variable, and fragmentation is allowed if packet length is too large for one time slot. In such a case, the packet is fragmented into multiple fragments. If any fragment does not transmit before its deadline, the entire packet is considered lost. In every slot, only one flow is allowed to access the channel yet it can transmit multiple packets whenver possible.

B. Notations

We use t to denote the current time slot. The slow time-varying period is denoted by $[t, t+T], T \geq 1$. Among all buffered packets in current time slot, P_T denotes those packets with deadline within $[t, t+T]$, and $P_{\overline{T}}$ denotes the rest of buffered packets. The amount of data contained by P_T and $P_{\overline{T}}$ are $|P_T|$ and $|P_{\overline{T}}|$ respectively. Both $|P_T|$ and $|P_{\overline{T}}|$ are measured by bits or bytes. Within $[t, t+T]$, the data rate received by flow i is constant and denoted by r^i. The maximum and minimum r^i within $[t, t+T]$ are r_{max},r_{min} respectively. Occasionally, we use $r(i)$ to denote the data rate received by packet i.

C. Existing approach

Perhaps the most commonly used delay-constrained scheduling approach is the Earliest-Deadline-First (EDF), in which the packet with the earliest deadline has the highest priority. EDF is an optimal solution to a so called BASIC network model studied by [1][2][3][4]. In BASIC model, each flow has equal and constant data rate. Packets in the same flow have equal length and can be transmitted in exactly one time slot.

Nevertheless, the BASIC model does not reflect the reality because of the aforementioned time-varying characteristics of wireless channel. Specifically, the channel quality is varying over time due to multipath fading and Doppler effect. In the slow time-varying situation, data rates are constant over time, but can be different between flows. In the fast time-varying situation, not only data rates differ among flows but they also vary over time. Therefore, the BASIC model is violated and accordingly EDF is no longer optimal, Even worse, since EDF ignores the data rate issue, in the fast time-varying situation EDF is not able to exploit the instantaneous high data rate of certain flows, thus leading to low system utilization.

The authors of [9] show that it is NP-hard to find the optimal scheduling solution to maximize throughput for time-varying channel quality and deadline-constrained packets. They propose the Greedy algorithm, which is to *always pick the flow with the highest instantaneous data rate*, and show its worst-case throughput is $\frac{1}{2}$ of the optimal one. However, their analysis is only applicable to our context when packet size is equal.

III. Overview of proposed scheduling policy

The proposed scheduling policy consists of an ensemble of algorithms. The policy differentiates slow channel-varying periods, where every individual flow's rate is constant, from fast channel-varying periods, where flow rates are varying. The proposed policy is self-adaptive to the variation of both channel quality and workload. In the following, we describe the proposed policy step by step.

(I) In every time slot t, the base station uses channel prediction to determine whether a slow channel-varying period (denoted by the interval $[t, t+T]$) exists. if it exists, go to Step II, otherwise, go to Step III.

(II) All buffered packets with deadline falling into $[t, t+T]$ are denoted by P_T. If P_T are schedulable, optimal scheduling will be determined in linear computational time; Otherwise, since finding the optimal algorithm is a NP-complete problem [9][11], the following procedure will be employed instead.

- The scheduler first needs to decide whether all buffered packets P_T are schedulable. To this end, it tries the algorithm OPT_UNDERLOAD on scheduling P_T. If no packets are left after applying OPT_UNDERLOAD, the scheduler knows that P_T are really schedulable. Therefore the schedule will perform OPT_UNDERLOAD. On the other hand, if some packets still left after trying OPT_UNDERLOAD, the scheduler knows that P_T is not schedulable. Consequently, it has to go to the next step which corresponds to unschedulable cases.

- In unschedulable cases, the scheduler uses the algorithm CHECK_OVERLOAD to judge whether it is an overloaded situation.
  - If the scheduler is slightly overloaded, it uses the proposed ED-EDF algorithm to schedule packets.
  - If heavily overloaded, the well-known Greedy algorithm [9] is used.

(III) When scheduling packets in the slow channel-varying period is done, the scheduler further uses the Greedy algorithm to schedule remaining packets within the fast channel-varying period.

It is evident that in our scheduling procedure, the exact algorithm (i.e., strategy) used in a given scenario is subject to the time-granularity of channel variation and traffic load. The rationale behind such algorithm switching is the different properties of various algorithms. We emphasize that these algorithms are not isolated. In fact, they interact with each other in a coherent framework, and work in concert to achieve high performance.

In the following sections, we describe each component of the proposed scheduling policy.

IV. Optimal Scheduling in Under-loaded Scenario

As shown in Section II, a few existing solutions such as EDF, S-OPT [3] and bipartite matching [1] are optimal for the BASIC model in which all packets have the same length and perceive the same time-constant transmission rate. We want to modify the existing optimal solutions to make it still optimal for the slow time-varying scenario: packets have different length. Each flow perceive time-constant transmission rate yet the rate varies among flows.

The idea is to transform the problem to a *dual* problem which follows the BASIC model. The transformation ensures that the solutions to the two problems have a one-to-one unique mapping. Then we can readily apply EDF (or S-OPT, bipartite matching) to solve the dual problem and map the resulting solution back to the original problem.

Let r^i denotes the throughput a flow i can transmit in one time slot. The main idea behind the transformation is that if flow i can send r_i units of data (starting with data in packet j) in one time slot, then we can represent these r_i units of data by one unit of data in the corresponding dual model and tag it with the same deadline of packet j.

We illustrate the transformation by an example scenario shown in Figure 1. The scenario contains two flows, with data rates of 2 and 1 units/slot respectively. Flow 1 has four buffered packets, and flow 2 has two. In the figure, packet deadline is represented by the head of the arcs and packet length is represented by the value near the arcs.

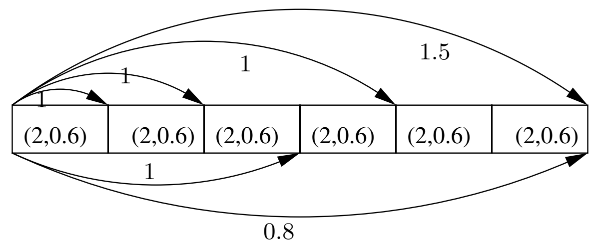

Fig. 1. Original example scenario

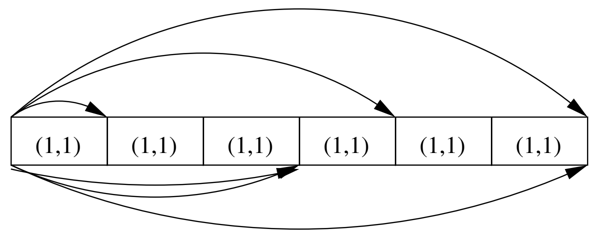

Fig. 2. Dual scheduling problem transformed from Figure 1 (now it follows the BASIC model).

We can transform this scenario into a dual one shown in Figure 2. To explain the mapping between the two scenarios, we first look at flow 1. In the original scenario, the first two packets in flow 1 can be transmitted in the first slot. Correspondingly, in the dual scenario flow 1 has the first packet with the same deadline. Next, the third packet and $\frac{2}{3}$ of the fourth packet (obtained via fragmentation) of flow 1 can be transmitted together in a single slot. Correspondingly, in the dual scenario the third packet of flow has the same deadline. The other $1/3$ of the fourth packet of flow 1 is still left. It can be mapped to another packet with the same deadline in the dual scenario. We further look at flow 2. Flow 2 can only transmit 0.6 units in one slot, which leaves 0.4 units of data from the first packet. We represent the first 0.6 units by the first packet of flow 2 in the dual scenario. For the remaining 0.4 units, we extract 0.2 units from the second packet of flow 2, then map such combined 0.6 units to one packet with the deadline of the first packet in the dual scenario. We continue the above procedure until all the packets in the original scenario are mapped to certain packets in the dual scenario.

Since the dual scenario complies with the BASIC model, EDF can be applied to find the optimal schedule for it. Nevertheless, the transformation is valid only when the scheduler is underloaded, that is, all buffered packets are schedulable. The following proposition gives the condition on whether these packets are schedulable.

Proposition 1: Buffered packets are schedulable if and only if $\sum_{i \in \mathcal{F}} N^i(t) \leq t, \forall t$, where $N^i(t)$ is the number of packets of flow i with deadlines at most t.

We refer the proof (and all the other proofs in this paper) to [12]. According to the above proposition, as long as those buffered packets are schedulable, we can always use EDF to solve the dual scheduling problem. The resulting schedule is then transformed back to the optimal schedule for the original one. We show that such a transormation is optimal as in the

Flow index	1			2	
Transmission rate (Mbps)	2			1	
Packet length l_i (bits)	400	800	2400	2400	$b \times 10^3$
Packet deadline d_i (ms)	0.2	0.8	1.8	4.4	$b + 4$
CTD s_i (ms)	0.2	0.6	1.8	4.2	$b + 4.2$

TABLE I

AN OVERLOAD SCENARIO (WHEN USING EDF, THE LAST PACKET IN FLOW 2 CANNOT BE SCHEDULED BECAUSE ITS CTD VALUE EXCEEDS THE DEADLINE)

following proposition.

Proposition 2: Given any scheduling problem in a slow time-varying and underloaded scenario, the transformation described above ensures that an optimal schedule exists if and only if there exists an optimal solution for the dual problem. Moreover, the optimal schedule for the dual problem can be uniquely mapped to an optimal one for the original problem.

V. SCHEDULING IN OVERLOADED SCENARIO

We now discuss how to schedule packets in slow time-varying, overloaded scenario. An overload scenario arises due to transient bursty traffic or lack of admission control. In such a case, the scheduler has to decide which packets to drop and which packets to schedule. The ultimate goal is to maximize the total throughput, or minimize throughput that needs to be dropped.

We thereby propose a new scheduling algorithm *Early-Dropping EDF* (ED-EDF). The basic idea of ED-EDF is to intentionally drop some unexpired packets to ensure that the overall schedulable throughput is increased. For the purpose of illustration, in the following we first discuss why the Greedy and EDF do not work well in the slow time-varying, overload scenario, then we describe ED-EDF and analyze its performance.

A. Limits of EDF and Greedy in slow time-varying, overloaded scenario

For slow time-varying, overloaded scenarios, the authors of [11] have proved that it is NP-hard to find a schedule that maximizes the throughput. Therefore, the most widely used scheduling algorithms in this case are those simple, heuristic ones such as EDF and the Greedy. Nevertheless, the Greedy and EDF do not work consistently well. We can illustrate this point by an example. Suppose all buffered packets are sorted in the ascending order of their deadline. In the sorted sequence, packet i's deadline is d_i and its length is l_i. In the example of Table I, there are 5 buffered packets belonging to two flows. It is obvious that EDF and the Greedy will schedule these packets from left to right, which results an arbitrarily bad performance. could be arbitrarily bad.

Now let us define s_i, the *cumulative transmission duration (CTD)* for each packet i. s_i is the total time required for transmitting all the packets no later than packet i (including packet i itself). Clearly if s_i is larger than d_i, not all the packets $1, 2, \ldots, i$ are schedulable. We thus refer to the first packet for which $s_i > d_i$ holds as *critical packet*. In the above example, packet 5 is a critical packet since it is the only one with $s_i > d_i$. Both EDF and the Greedy will drop packet 5. Nevertheless, it is easy to see that the optimal scheduling policy here is to drop packet 1 because when packet 1 is dropped, all the other four packets become schedulable. To further explain this observation, we introduce *competitive ratio*: the competitive ratio for a scheduling algorithm is defined as the infimum ratio of the throughput between the algorithm and the optimal one. We can figure out that the competitive ratio for both EDF and the Greedy is $6/(b+5.6)$ (b is the length of the last packet), which tends to be zero as $b \to \infty$. As a result, both EDF and the Greedy have zero competitive ratio.

The above example shows that EDF and the Greedy algorithm may unnecessarily drop too much data in certain overloaded scenarios. Fundamentally, this is because when either algorithm always drop choose the critical packets to drop without considering any other packet dropping schemes. In fact, other packet dropping schemes do exist - if appropriately chosen, the total amount of data need to be dropped can be significantly reduced. Based on this motivation, we describe a scheduling algorithm ED-EDF which adopts a more intelligent dropping scheme.

B. ED-EDF

Suppose there are N buffered packets. $r(i)$ is the data rate used for transmitting packet i. The procedure of ED-EDF is as following:

(I) Sort the N packets in the ascending order of their deadline.
(II) By comparing s_i with d_i, the scheduler look for the critical packet i. If no critical packets exist, go to (IV).
(III) Given that packet i is the critical one, the scheduler searches for a packet j with the minimum transmission time $\frac{l_j}{r(j)}$ in the packet set $\{1, 2, \ldots, i\}$. Packet j will be dropped immediately. If more than one such packet exists, the one with the earliest deadline is dropped. Then the scheduler returns to (II).
(IV) When no more critical packets are found, the scheduler uses EDF to transmit packets.

The core idea of ED-EDF is: ED-EDF searches all the packets ahead of (and including) the critical packet. ED-EDF drops the packet consuming the minimum service time and expects that such a *frugal* dropping will make all the other packets schedulable. Now if we apply ED-EDF to the example scenario in Table I, we can easily see that ED-EDF gives the optimal throughput.

C. Performance analysis of ED-EDF

ED-EDF does not guarantee optimal scheduling as the optimal solution is NP-hard, yet the amount of dropped data by ED-EDF is bounded when compared to the optimal scheme. Such a result is formally stated below.

Theorem 1: In a slow time-varying, overloaded scenario, let r_{max}, r_{min} be the highest and lowest data rates perceived by flows respectively. The amount of data dropped by ED-EDF is no more than $2\frac{r_{max}}{r_{min}}$ times the amount of data dropped by the optimal scheduling algorithm that maximizes throughput. The performance bound is tight. □

If the disparity between r_{max} and r_{min} is large, the bound in Theorem 1 may seem to be loose. However, this is the worst-case bound and it does not necessarily mean that the average performance of ED-EDF is much worse than the optimal one. Moreover, in many cases, especially when the data in flows having the high or low rates is much smaller when compared to the total amount of packets dropped, the worst-case bound can be much closer to the optimal. We use an example to illustrate this point.

Assume D_e and D_{opt} are the amount of data dropped by ED-EDF and the optimal strategy respectively. There are three flows in the buffer, and their data rates are 5, 1.2 and 1. The first flow, which receives the data rate of 5 units/second, contains only n units of data and $n \ll D_{opt}$. According to Theorem 1, there is $\frac{D_e}{D_{opt}} \leq 2.\frac{5}{1} = 10$. Now imagine the first flow does not exist and we compare ED-EDF and the optimal strategy again. The amount of data dropped by ED-EDF and the optimal strategy are denoted by D'_e,D'_{opt} respectively. Since, r_{max} is now decreased to 1.2 units/second, we have $\frac{D'_e}{D'_{opt}} \leq 2.\frac{1.2}{1} = 2.4$ by using Theorem 1. By simple reasoning we know that $D_{opt} - n \leq D'_{opt} \leq D_{opt}$ and $D_e \leq D'_e + n$. Hence, we have $\frac{D_e}{D_{opt}} \leq \frac{D'_e+n}{D'_{opt}} = \frac{D'_e}{D'_{opt}} + \frac{n}{D'_{opt}} \leq 2.4 + \frac{n}{D_{opt}-n}$. If $n \ll D_{opt}$, the ratio $\frac{D_e}{D_{opt}}$ can have a much tighter bound than the original bound 10.

VI. Overload Estimation

In the previous section, we gave a scheduling algorithm when the data rates do not vary over time. We also showed that while ED-EDF has a bounded performance under any circumstance, when the packet length is equal, the Greedy algorithm performs better once the load exceeds a certain threshold. This threshold was shown to depend on the network throughput capacity C_{opt}, a quantity that cannot be directly measured. In this section, we describe a heuristic to estimate C_{opt}.

Through further analysis in our technical report [12], we can know that when channel rates are constant over time, if $|P_T| < C_{opt}(1+\gamma)$ where γ is a constant, ED-EDF performs better than the Greedy. Now we denote the throughput of the Greedy algorithm in this period by C_g, we know that $C_g \leq C_{opt} \leq 2C_g$. Let the throughput of the algorithm OPT_UNDERLOAD (presented in Section IV) be denoted by C_u which satisfies $C_u \leq C_{opt}$. Next, we use $\max(C_u, C_g)$ as an estimate for C_{opt}, which guarantees that the worst-case performance bound of the proposed algorithm is much better than either ED-EDF or Greedy.

It is always better to underestimate C_{opt} than to over-estimate it. This is because we do not want our worst-case performance to be below that of the Greedy algorithm. Underestimating C_{opt} will cause us to switch to the Greedy scheduling strategy at a lighter load, while over-estimating it will lead us to switch to Greedy algorithm at a higher load and lead to a worst-case bound less than that of $1/2$.

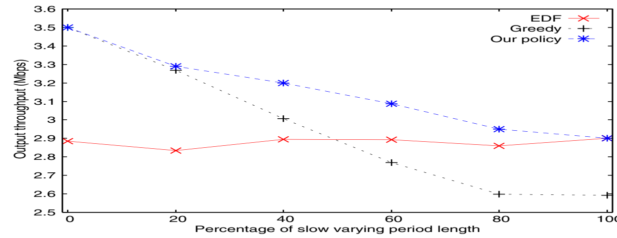

Fig. 3. Throughput comparison when slow varying period becomes longer

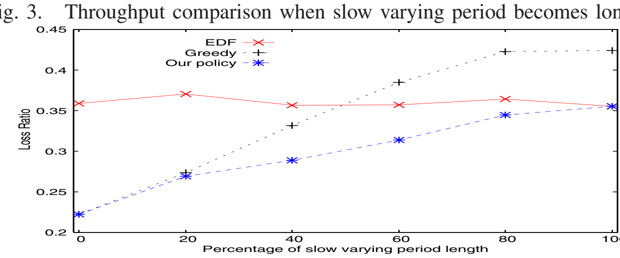

Fig. 4. Data loss comparison when slow varying period becomes longer

VII. Simulations

In this section, we use numerical simulations to evaluate the proposed scheduling policy. Throughput and data loss ratio are used as performance criteria. The data loss ratio is defined as the percentage of data not transmitted before deadlines. Due to space constraints, we only show the results for scenarios where both slow and fast time-varying periods exist. For comparison purposes, we simulate EDF and the Greedy algorithm as well.

We simulate a time-slotted system taking similar parameters to the HDR network. Each time slot is 1.67 ms. The polling-cycle model is simulated by periodically injecting traffic into the system. The polling-cycle period is set to be $T_c = 500$ ms. T_c always starts with a slow time-varying period T followed by a fast time-varying period. We generate 20 flows in all scenarios. Within the slow varying period T, the data rates for the 20 flows are randomly chosen from the range $[1, 5]$ Mbps. In the fast time-varying period, data rates are generated from a realistic HDR channel model. Each flow has L bits of data. These L bits of data are converted into packets by randomly assigning packet length from [100 bytes,2500 bytes]. The deadline for each packet is randomly chosen from [0,T_c ms]. In this way, we can control the traffic load by tuning the values of L and t. Each data point in all our simulation results are averaged over 10 runs.

In the first scenario, we evaluate how the proposed policy performs when the time-granularity of channel variation changes. To simulate different time-granularities of the variation, we fix the polling-cycle period T_c to be 500 ms, then we vary T, the slow time-varying period length. The total traffic arrival rate is 6 Mbps, which indicates an overloaded scenario.

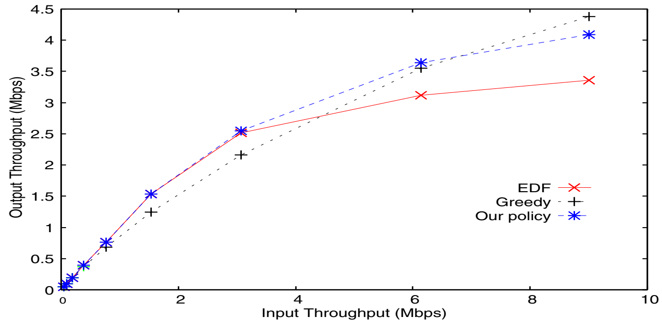

Fig. 5. Comparison of throughput when network load is increasing

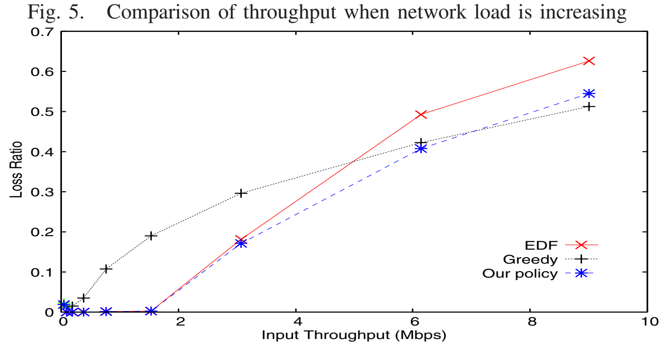

Fig. 6. Comparison of data loss ratio when network load is increasing

Now we gradually increase T. Meanwhile we measure the performance for the three scheduling policies: the proposed one, EDF, and the Greedy. The results, in terms of throughput and data loss ratio, are shown in Figures 3 and 4. The two figures show that when most of T_c is fast time-varying, namely, the slow varying period length is close to 0, the throughput for EDF is 2.9 Mbps, much worse than that of the Greedy (3.5 Mbps). On the other hand, when T_c is mainly consisted of slow time-varying periods, EDF gives a throughput of 2.9 Mbps, better than the 2.6 Mbps throughput achieved by the Greedy. In contrast to the inconsistent performance of both EDF and Greedy, our scheduling policy does not lose to either of them. Moreover, when the slow and fast varying periods have comparable length, which corresponds to the middle areas in Figures 3 and 4, our policy outperforms both EDF and the Greedy. Specifically, when the slow and fast time-varying periods have equal length, the proposed policy's data loss ratio is 17% lower than that of EDF and the Greedy.

Next, we evaluate how our policy reacts to network load variation. The first half of T_c is set to be a slow time-varying period while the rest is fast time-varying. In this scenario, the total traffic arrival rate increases from 0 to 7.5 Mbps. During the increase of the traffic rate within this range, the system starts in an underloaded situation and then gradually becomes overloaded. The throughput and data loss ratio are shown in Figures 5 and 6 respectively. The two figures show that the proposed scheduling policy is more robust to the imposed traffic load compared with EDF and the Greedy algorithm. It is worthy of mentioning that although the Greedy guarantees a $1/2$ performance bound, it drops a lot more data than other algorithms. Our proposal policy drops packets only when they are not schedulable. Moreover, it achieves better aggregate throughput performance with a smaller amount of packet dropping even in overloaded situations.

VIII. Conclusion

In this paper, we propose a scheduling policy to schedule delay-sensitive data over time-varying wireless channels. The advantages of our proposed scheduling policy are that: it is insensitive to the time-granularity of channel variation; it is optimal for underloaded situations, and its performance is consistently better than other existing solutions for overloaded scenarios. The proposed policy consists of different algorithms whose performance is dependent on time-scale of the channel variation as well as workload. Among these algorithms, OPT_UNDERLOAD has been proposed as the optimal algorithm for slow time-varying and underloaded scenarios. ED-EDF algorithm is further proposed as the scheduling algorithm for slow time-varying and lightly overloaded scenarios. ED-EDF has a novel intelligent packet dropping mechanism which reduces unnecessarily expired packets. When using dropped throughput as the performance criterion, ED-EDF has a provable competitive ratio, a missing feature for EDF and the Greedy. By adaptively switching between different algorithms, the overall performance of the proposed scheduling policy is much more consistent than EDF and the Greedy algorithm.

References

[1] E.L.Lawler, *Combinatorial Optimization: Networks and Matroids*, Rinehart and Winston, New York, 1976.

[2] J.R.Jackson, "Scheduling a production line to minimize maximum tardiness," Research Report 43, 1955.

[3] T.Ling and N.Shroff, "Scheduling real-time traffic in atm networks," in *IEEE INFOCOM*, 1996.

[4] B.Hajek and P.Seri, "Lex-optimal on-line multi-class scheduling with hard deadlines," Preprint, 2000, `http://www.comm.csl.uiuc/hajek/`.

[5] Q.Zheng and K.G.Shin, "On the ability of establishing real-time channels in point-to-point packet-switched networks," *IEEE Transactions on Communications*, vol. 42, no. 2/3/4, pp. 1096–1105, 1994.

[6] P.Bender, P.Black, M.Grod, R.Padovani, N.Sindhushayana, and A.Viterbi, "CDMA/HDR: a bandwidth-efficient high-speed wireless data service for nomadic users," *IEEE Communications Magazine*, vol. 38, pp. 70–77, July 2000.

[7] ITU-T Recommendation G.114, "General characteristics of international telephone connections and international telephone circuits: One way transmission time," Feb. 1996.

[8] M.Karam and F.Tobagi, "On the traffic and service classes in the internet," in *IEEE Globecom*, San Francisco, 2000.

[9] M.Agarwal and A.Puri, "Base station scheduling of requests with fixed deadlines," in *IEEE INFOCOM*, 2002.

[10] S.Shakkottai and R.Srikant, "Scheduling real-time traffic with deadlines over a wireless channel," *ACM/Baltzer Wireless Networks Journals*, vol. 8, no. 1, pp. 13–26, January 2002.

[11] M.R.Garey and D.S.Johnson, *Computers and Intractability: A Guide to The Theory of NP-completeness*, W.H.Freeman and Company, New York, 1979.

[12] X.Meng, T.Nandagopal, H.Y.Wong, H.Yang, and S.Lu, "Exploiting wireless channel variation for scheduling delay-constrained data," Technical Report, CS Department, UCLA, October 2006, `http://www.cs.ucla.edu/~xqmeng/tr06.ps`.