Real-Time Communication over Unreliable Wireless Links: A Theory and Its Applications

I-Hong Hou

Computer Engineering and Systems Group

Department of ECE

Texas A&M University

College Station, TX 77843, USA

ihong.hou@gmail.com

Abstract

Wireless networks are increasingly used to serve real-time flows. We provide an overview of an emerging theory on real-time wireless communications and some of its main results. This theory is based on a model that jointly considers the delay bounds and throughput requirements of clients, as well as the unreliable and heterogeneous nature of wireless links. This model can be further extended in several aspects. It provides solutions for three important problems, namely, admission control, packet scheduling, and utility maximization. The theory can also be extended to consider broadcast of real-time flows, and to incorporate network coding.

Index Terms

Wireless networks, real-time communication, deadlines, timely-throughput.

I. Introduction

With the advance of broadband wireless transmission, the increasing popularity of mobile devices, and the deployment of wireless sensor networks, wireless networks are increasingly used to serve real-time flows that require strict per-packet delay bounds. Such applications include VoIP, video streaming, real-time surveillance, and networked control. For example, a study by Cisco [1]

This material is based upon work partially supported by USARO under Contract Nos. W911NF-08-1-0238 and W-911-NF-0710287, NSF under Contracts CNS-1035378, CNS-0905397, CNS-1035340, and CCF-0939370, and AFOSR under Contract FA9550-09-0121. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the above agencies.

has predicted that wireless traffic will grow exponentially, and mobile video will dominate wireless traffic in the near future, accounting for 62% of wireless traffic by the year 2015. Serving real-time flows is also a key component of many cyber-physical systems. In a one example of a centralized cyber-physical system, a server may gather surveillance data from wireless sensors, based on which it makes control decisions and disseminates them to actuator units. Both surveillance data and control decisions need to be delivered in a timely manner, or the performance of the system may be degraded. In addition to delay bounds, such applications also require some guarantees on their *timely-throughput*, defined as the throughput of packets that are delivered on time.

Serving real-time flows in wireless networks is particularly challenging since wireless transmissions are subject to shadowing, fading, and interference, and thus usually unreliable. Further, the channel qualities may differ from link to link. A desirable solution for serving real-time flows thus needs to explicitly take into account the stochastic and heterogeneous behavior of packet losses due to failed transmissions.

In this paper, we introduce a theory for serving real-time flows over wireless links. The core of this theory is a model that takes into account all the aforementioned challenges, the delay bounds and timely-throughput requirements of applications, and the unreliable and heterogeneous nature of wireless transmission. We address three important problems concerning the service of real-time flows: admission control, packet scheduling, and utility maximization when timely-throughput requirements are elastic. We derive a sharp condition that characterizes when it is feasible to fulfill the demands of all real-time flows, as well as a polynomial-time algorithm for admission control. We also introduce two on-line scheduling policies that are proven to fulfill every feasible system. For the scenario where timely-throughput requirements are elastic, we formulate an utility maximization problem that aims to maximize the total utility over all flows. We introduce a bidding procedure and an on-line scheduling algorithm which together achieve the maximum total utility.

We extend the model to incorporate various real world complexities. We consider the scenario where different applications may generate traffic with different patters and require different delay bounds. We also consider the situation where channel quality may vary over time, and where rate adaptation may be employed to enhance transmission reliability. We introduce a framework for designing scheduling policies and derive tractable feasibility optimal policies for two different scenarios involving unreliable fading channels and rate adaptation. We also address the problem of maximizing the total utility when the contending flows are elastic.

In addition to unicast flows, we also address the problem of scheduling real-time flows that are

broadcasts. The major challenge for broadcasting is that due to the lack of ACK the sender may not have immediate feedback information on whether a transmission is successful. We extend our model to address this challenge. The extended model also allows the optional usage of network coding. We introduce a framework for designing scheduling policies for any arbitrary network coding scheme, and derive policies for various coding schemes.

The rest of the paper is organized as follows. Section II describes the basic model for real-time wireless communication. Section III establishes a necessary and sufficient condition for feasibility. Section IV introduces two on-line scheduling policies that are feasibility optimal. Section V addresses the problem of serving real-time flows when their timely-throughput requirements are elastic. Section VI extends the basic model to incorporate various real world complexities. Section VII develops a framework for designing scheduling policies, which is then used to derive policies for two different scenarios. Section VIII addresses the utility maximization problem when rate adaptation is incorporated. Section IX studies the problem of broadcasting real-time flows. Section X concludes the paper.

II. Basic Model for Real-Time Wireless Communications

Consider a system with one access point (AP) and N wireless clients. Figure 1 illustrates an example of such a system. Each client is associated with one real-time flow that requires service from the AP. We assume that time is slotted and slots are numbered by $t \in \{1, 2, 3, \dots\}$. The length of a time slot is set to be the time needed for the AP to transmit one packet to a client, including all possible overheads. Time slots are further divided into *intervals*, where an interval consists of T consecutive time slots in $(kT, (k+1)T]$, for some k. The AP is in charge of scheduling transmissions in each time slot. This model naturally applies to downlink traffic. When both downlink traffic and uplink traffic are present, this model can still be applied by incorporating server-centric access mechanisms, such as 802.11 PCF, WiMax, and Bluetooth.

At the beginning of each interval, that is, at times $1, T+1, 2T+1, \dots$, each real-time flow generates one packet. We assume that packets in all real-time flows need to be delivered within a delay bound of T time slots if they are to be useful. In other words, packets that are generated at the beginning of an interval are only useful if they are delivered no later than the end of the interval. If a packet is not delivered before its delay bound, we say that the packet expires and it is dropped from the system. By dropping expired packets, it is guaranteed that the delay of every delivered packet is at most T time slots.

We consider unreliable and heterogeneous wireless links. When the AP makes a transmission for client n, the transmission is successful with probability p_n. The values of p_n may differ from client to client, as the channel reliabilities for different clients may be different. The AP has instant feedback information on whether a transmission is successful, e.g., by enforcing acknowledgements.[1]

We measure the performance of a client by its *timely throughput*, which is defined as the long-term average number of successfully delivered packets for the client per interval. Specifically, the timely-throughput of client n is

$$\liminf_{K\to\infty} \frac{\sum_{k=1}^{K} 1(\text{there is a packet delivery for client } n \text{ in the } k^{th} \text{ interval})}{K},$$

where $1(\cdot)$ is the indicator function. We assume that each client has an inelastic timely-throughput *requirement*. We denote the timely-throughput requirement of client n by q_n.

In the following sections, we will discuss how to characterize whether it is *feasible* to *fulfill* the requirements of all clients in the system, and how to design a *feasibility optimal* policy that fulfills the requirements of any set of clients as long as the set is feasible. These terms are formally defined as follows.

Definition 1: A system is said to be *fulfilled* by some scheduling policy η, if, under η, the timely-throughput provided to each client n is at least q_n.

Definition 2: A system is *feasible* if there exists some scheduling policy that fulfills it.

Definition 3: A scheduling policy is *feasibility optimal* if it fulfills every feasible system.

III. A Necessary and Sufficient Condition for Feasibility

In this section, we characterize when a system is feasible, given the channel reliabilities, p_n, and timely throughput requirements, q_n, of clients. This solves the problem of admission control.

We first observe that the actual timely-throughput of a client n is determined by the long-term average number of time slots that the AP spends on transmitting packets for client n per interval. This observation is formally captured by the following lemma.

Lemma 1 ([4], Lemma 1): Let $u_n(k)$ be the number of time slots that the AP spends transmitting the packet for client n in the k^{th} interval. The timely-throughput of a client n is at least q_n if and only if

$$\liminf_{K\to\infty} \frac{\sum_{k=1}^{K} u_n(k)}{K} \geq \frac{q_n}{p_n}. \tag{1}$$

[1]In this case, by a slot is meant the time to deliver the packet as well as receive the ACK. The quantity p_n can then be regarded as the probability that the transmission from the AP to client n is successful, as well as the ACK from client n to the AP.

□

We hereafter define $w_n := \frac{q_n}{p_n}$ and call it the *load* imposed by client n. Verifying whether a system is fulfilled is now equivalent to verifying whether the average number of time slots that the AP spends on client n is at least w_n, for all n.

Since the AP can make at most one transmission in each time slot, we can immediately obtain a necessary condition for feasibility:

Lemma 2: A system is feasible only if $\sum_{n=1}^{N} w_n \leq T$.□

In other words, the total loads imposed by all clients cannot exceed the number of time slots available in an interval, if the set of clients' requirements is to be feasible. However, while this condition is necessary for feasibility, it is not sufficient, because it may be impossible for the AP to utilize all the T time slots in every interval. For example, consider a system with two clients and $T = 3$. Suppose in a certain interval, the AP transmits the two packets for the two clients in the first two time slots, respectively, and both transmissions are successful. Now, as the AP has delivered all the packets in the system, there are no more packets to be transmitted in the third time slot. Thus, the AP is forced to be idle in the third time slot. This example suggests that we need to take the amount of unavoidable idle time into account when evaluating feasibility.

While the amount of idle time slots may differ from policy to policy,, we show that it is the same for all *work-conserving* policies [4, Lemma 3].

Definition 4: A scheduling policy is a *work-conserving* policy if it schedules some packet for transmission as long as there are any undelivered packets, and only idles when all unexpired packets have been delivered.

It can be shown that, for every feasible system, there exists a work-conserving policy that fulfills it [4, Lemma 4]. Thus, we can limit our attention to work-conserving policies.

Let I be the long-term average number of idle time slots in an interval under any work-conserving policy. Since, on average, the AP can only make $T - I$ transmissions in an interval, we obtain a more stringent necessary condition for feasibility.

Lemma 3: A system is feasible only if $\sum_{n=1}^{N} w_n \leq T - I$. □

It turns out that this condition too is only necessary and not sufficient for feasibility. It can be further refined. Observe that, if we remove some clients from a feasible system, resulting in a system that only consists of a subset of clients of the original one, the resulting system must also be feasible. Thus, by letting I_S denote the long-term average number of idle time slots in an interval when only a subset $S \subseteq \{1, 2, \ldots, N\}$ of clients is present, we obtain an even more

stringent necessary condition.

Lemma 4: A system is feasible only if $\sum_{n\in S} w_n \leq T - I_S$, for all $S \subseteq \{1, 2, \ldots, N\}$. □

We note that the conditions in Lemma 3 and Lemma 4 are not equivalent. The following example demonstrates that the condition in Lemma 4 is indeed a stronger one.

Example 1: Consider a system with two clients and interval length $T = 3$. The reliabilities for both clients are $p_1 = p_2 = 0.5$. Client 1 requires a timely-throughput of $q_1 = 0.876$, while the timely-throughput requirement of client 2 is $q_2 = 0.45$.

Now, we have:

$$\begin{aligned} w_1 &= \frac{q_1}{p_1} = 1.76, \\ w_2 &= \frac{q_2}{p_2} = 0.9, \\ I_{\{1\}} = I_{\{2\}} &= 2p_1 + (1 - p_1)p_1 = 1.25, \\ I \equiv I_{\{1,2\}} &= p_1 p_2 = 0.25. \end{aligned}$$

If we evaluate the condition for the set of all clients $\{1, 2\}$, we have $w_1 + w_2 = 2.66 < 2.75 = T - I_{\{1,2\}}$. However, if we evaluate the condition for the subset of $S = \{1\}$, we find $w_1 = 1.76 > 1.75 = T - I_{\{1\}}$. This indicates that the set of clients is not feasible. Thus, merely evaluating the condition for the set of all clients is not enough. □

It turns out that the more stringent necessary condition in Lemma 4 is actually both necessary and sufficient.

Theorem 1 ([4], Theorem 4): A system is feasible if and only if $\sum_{n\in S} w_n \leq T - I_S$, for all $S \subseteq \{1, 2, \ldots, N\}$. □

While the above theorem provides a sharp characterization of feasibility, it involves testing the condition $\sum_{n\in S} w_n \leq T - I_S$ for all subsets $S \subseteq 1, 2, \ldots, N$. Sine there are 2^N such subsets, this appears to require exponentially many tests. However, it turns out that this condition can be greatly simplified.

Theorem 2 ([4], Theorem 5): Sort all clients so that $q_1 \geq q_2 \geq \cdots \geq q_N$. Let $S_m := \{1, 2, \ldots, m\}$ be the subset that consists of clients 1 through m. A system is then feasible if and only if $\sum_{n=1}^{m} w_n \leq T - I_{S_m}$, for all $m \in \{1, 2, \ldots, N\}$. □

This theorem reduces the number of tests for checking feasibility to N. Each such test can be performed efficiently by using the Fast Fourier Transform, using which [4] has developed a polynomial time algorithm based on this theorem.

IV. Feasibility Optimal Scheduling Policies

In this section, we introduce two scheduling policies that are feasibility optimal, that is, they fulfill every feasible systems. Both policies are what we call the *largest debt first scheduling policies*. When employing a largest debt first scheduling policy, the AP calculates the *debt* it owes to each client at the beginning of each interval. The AP then prioritizes all clients according to the debts owed to them, such that clients with larger debts get higher priority. In each time slot of the interval, the AP transmits the packet for the client with the largest debt among those whose packets are not delivered yet.

Figure 2 shows an example of scheduling using the largest debt first scheduling policy. In this example, we assume that, at the beginning of some interval, client 1 has the largest debt, client 3 has a medium debt, and client 2 has the smallest debt. The AP first transmits the packet for client 1, and keeps retransmitting the packet in case the previous transmission fails. The packet for client 3 is only transmitted after the packet for client 1 is successfully delivered. Finally, the packet for client 2 is only transmitted after both packets for client 1 and client 3 are successfully delivered.

The two scheduling policies differ in their definitions of debt. The first kind of debt, the *time-based debt*, is derived from the concept of load w_n defined in Section III.

Definition 5: Let $u_n(k)$ be the number of time slots that the AP spends transmitting the packet for client n in the k^{th} interval. The *time-based debt* of client n at the beginning of the $(k+1)^{th}$ interval is defined as $r_n^{(1)}(k+1) := kw_n - \sum_{j=1}^{k} u_n(j)$. The largest debt first policy that applies the time-based debt is called the *largest time-based debt first* scheduling policy.

The second kind of debt, the *weighted-delivery debt*, is derived directly from the timely-throughput requirement q_n of a client.

Definition 6: Let $d_n(k)$ be the indicator function of the event that the AP delivers a packet for client n in the k^{th} interval. The *weighted-delivery debt* of client n at the beginning of the $(k+1)^{th}$ interval is defined as $r_n^{(2)}(k+1) := \frac{kq_n - \sum_{j=1}^{k} d_n(j)}{p_n}$. The largest debt first policy that prioritizes according to the weighted-delivery debt is called the *largest weighted-delivery debt first* scheduling policy.

It has been shown that both largest debt first policies fulfill every feasible systems, and are hence feasibility optimal.

Theorem 3 ([4], Theorem 2 and Theorem 3): Both the largest time-based debt first policy and the largest weighted-delivery debt first policy are feasibility optimal. □

A. Simulation Results

We now present simulation results when applying the two largest debt first policies for VoIP traffic. We follow the G.711 codec, which is an ITU-T standard for audio compression, and the IEEE 802.11b to decide parameters for the simulation. Important parameters are summarized in Table I.

We assume that there are two groups, A and B, of clients. The timely-throughput requirement of each client in group A is 0.99 packets per interval, while that of each client in group B is 0.8 packets per interval. The channel reliability of the n^{th} client in each group is $(60+n)\%$. From the admission control result developed in Section III, it follows that a system with 6 group A clients and 5 group B clients is feasible, while a system with 6 group A clients and 6 group B clients is not.

We compare the two largest debt first policies against two standard techniques. One is the IEEE 802.11 DCF mechanism, and the other is a centralized policy where the AP assigns priorities to clients randomly at the beginning of each interval. The two policies are referred to as *DCF* and *Random*, respectively. We use the *total timely-throughput deficit*, defined as $\sum_n (q_n -$ actual timely-throughput of client $n)^+$, to evaluate the performance of a policy. A system is fulfilled by a policy if, under that policy, the total timely-throughput deficit converges to zero.

The simulation results for the feasible set is shown in Figure 3. It can be seen that both largest debt first policies fulfill the feasible set, as the total timely-throughput deficits converge to zero over time under these two policies. On the other hand, both the DCF policy and the Random policy fail to fulfill this feasible set. This result shows these two policies are not adequate for serving real-time flows.

The simulation result for the infeasible set is shown in Figure 4. In this setting, the total timely-throughput deficits remain strictly positive and bounded away from zero under all four policies. This illustrates that it is indeed infeasible to fulfill this system, and that the conditions for feasibility are sharp. Moreover, the two largest debt first policies achieve smaller total timely-throughput deficits than the other two policies. This suggests that the largest debt first policies have better performance even for infeasible systems.

V. Utility Maximization for Elastic Traffic

In previous sections, we have supposed that the timely-throughput requirement of each client, q_n, is specified by the client and inelastic. In this section, we discuss scenarios where the timely

throughput requirements are elastic.

A. Problem Formulation

Suppose now that the value of q_n is tunable by the AP, for each n. Suppose also that each client n receives some *utility* based on the timely-throughput that is provided. The relation between the utility derived and the timely-throughput provided is characterized by each client's *utility function*, $U_n(q_n)$. The utility function, $U_n(\cdot)$, is assumed to be concave and infinitely differentiable. Different clients may have different utility functions. In the sequel, we use $[x_n]$ to denote the vector containing $(x_1, x_2, \ldots, x_N)$.

The goal of the AP is to choose a feasible vector $[q_n]$ so as to maximize the total utility of all clients. Since Theorem 1 already has established a necessary and sufficient condition for feasibility, we can formulate the problem of serving elastic traffic as the following SYSTEM optimization problem:

SYSTEM:

$$\text{Max } \sum_{n=1}^{N} U_n(q_n), \tag{2}$$

$$\text{s.t. } \sum_{n \in S} \frac{q_n}{p_n} \leq T - I_S, \forall S \subseteq \{1, 2, \ldots, N\}, \tag{3}$$

$$\text{over } 0 \leq q_n \leq 1, \forall n. \tag{4}$$

B. Problem Decomposition and a Bidding Procedure

Since the utility functions of clients are all concave, the above optimization problem is indeed a convex one, and standard techniques for solving convex optimization problems can be employed to determine the optimal solution that maximizes the total utility of the system. However, this problem involves 2^N feasibility constraints, as in (3), resulting in a huge computation overhead for standard techniques. Further, in practice, the AP may not even know the utility functions of all clients. To alleviate these challenges, we can employ a decomposition technique introduced in [10], and decompose the SYSTEM problem into two subproblems. The total utility in the system can be maximized by jointly solving these two subproblems. In the sequel, we will establish online algorithms that solve these two subproblems.

The decomposition is best described by a bidding procedure. Suppose that, at the beginning of each interval, each client n pays the AP a certain amount of money, ρ_n. The AP then chooses a feasible vector $[q_n]$ to achieve weighted proportional fairness among clients, where the weight

of client n is ρ_n. It is known that weighted proportional fairness can be achieved by maximizing $\sum_{n=1}^{N} \rho_n \log q_n$. Thus, the AP aims to solve the following problem:

ACCESS-POINT:

$$\text{Max } \sum_{n=1}^{N} \rho_n \log q_n, \tag{5}$$

$$\text{s.t. } \sum_{n \in S} \frac{q_n}{p_n} \leq T - I_S, \forall S \subseteq \{1, 2, \ldots, N\}, \tag{6}$$

$$\text{over } 0 \leq q_n \leq 1, \forall n. \tag{7}$$

On the other hand, after receiving a timely-throughput of q_n, each client n estimates its price per unit of timely-throughput as $\psi_n := \rho_n / q_n$. Client n assumes a linear relation between payment and timely-throughput, and aims to find a new ρ_n so as to maximize its own net utility, that is, the difference between its utility and the amount of its payment. More formally, client n aims to solve the following problem:

CLIENT$_n$:

$$\text{Max } U_n(\frac{\rho_n}{\psi_n}) - \rho_n, \tag{8}$$

$$\text{over } 0 \leq \rho_n \leq \psi_n. \tag{9}$$

Under this bidding procedure, the AP aims to achieve weighted proportional fairness among clients, and does not need to know the exact utility functions of clients. On the other hand, each client n only needs to selfishly solve its own CLIENT$_n$ problem, and does not need to consider the behavior of other clients. The total utility in the system is maximized when the solutions to the two subproblems converge:

Theorem 4 ([6], Theorem 2): There exists non-negative vectors $[q_n]$, $[\psi_n]$, and $[\rho_n]$, with $[\rho_n] = [q_n \psi_n]$, such that:

1) Given that each client n pays ρ_n per interval, $[q_n]$ solves the ACCESS-POINT problem.
2) For all n such that $\psi_n > 0$, ρ_n solves the CLIENT$_n$ problem.

In addition, if $[q_n]$, $[\psi_n]$, and $[\rho_n]$ are all vectors with positive entries, $[q_n]$ solves the SYSTEM problem. □

C. An On-Line Scheduling Policy for ACCESS-POINT

In our bidding procedure, we require each client n and the AP to solve the CLIENT$_n$ problem and the ACCESS-POINT problem, respectively. Solving CLIENT$_n$ is easy, as it only involves the

utility function of client n. On the other hand, while, in contrast to solving SYSTEM, solving ACCESS-POINT does not require knowledge of utility functions, it still involves all the 2^N feasibility constraints. However, we can show the interesting results that there exists an on-line scheduling policy that not only incurs negligible computation overhead, but also requires no knowledge of channel reliabilities, $[p_n]$.

The scheduling policy is called the *weighted-transmission* (WT) policy. Let $u_n(k)$ be the number of time slots that the AP spends transmitting the packet for client n in the k^{th} interval. At the beginning of the $(k+1)^{th}$ interval, the AP sorts all clients in increasing order of $\frac{\sum_{j=1}^{k} u_n(j)}{\rho_n}$. In each time slot within the $(k+1)^{th}$ interval, the AP transmits the packet for the client that has the smallest $\frac{\sum_{j=1}^{k} u_n(j)}{\rho_n}$ among those whose packets are yet to be delivered.

It can be shown that this simple on-line scheduling policy solves the ACCESS-POINT problem.

Theorem 5 ([6], Theorem 6): Given $[\rho_n]$, the timely-throughput $[q_n]$ resulting from the WT policy is a solution to the ACCESS-POINT problem. □

D. Simulation Results

We now present some simulation results. As in Section IV-A, we follow the G.711 codec and IEEE 802.11b standard for setting parameters of the simulation. We assume that the utility function of each client n is $U_n(q_n) = \gamma_n \frac{q_n^{\alpha_n}-1}{\alpha_n}$, where γ_n is a positive integer and $\alpha_n \in (0,1)$. We consider four different policies: (i) a policy that applies the WT scheduling policy and the bidding procedure, which we call *WT-Bid*; (ii) a policy that only applies the WT scheduling policy and does not apply the bidding procedure, that is, clients do not update their bids, which we call *WT-NoBid*; (iii) a random policy (*Rand*) that assigns priorities to clients randomly at the beginning of each interval; and (iv) a policy that assigns higher priorities to clients with larger γ_n, and breaks ties randomly at the beginning of each interval, which we call *P-Rand*. The performance of each policy is evaluated by its resulting total utility in the system, $\sum_n U_n(q_n)$.

We consider two different settings, both with 30 clients. In the first setting, we set $p_n = (50+n)\%$, $\gamma_n = (n \mod 3)+1$, and $\alpha_n = 0.3+0.1(n \mod 5)$, for each client n. In the second setting, we set $p_n = (20+2n)\%$, $\gamma_n = 1$, and $\alpha_n = 0.3+0.1(n \mod 5)$, for each client n.

Simulation results for the two settings are shown in Figure 5 and Figure 6, respectively. In both settings, the WT-Bid policy outperforms all the other three policies.

VI. A Generalized Model for Unicasting Real-Time Flows

The model introduced in Section VI has assumed that each client generates one packet at the beginning of each interval and requires the same delay bound, the channel reliability p_n does not change over time, and all transmissions take exactly one time slot. In practice, however, different applications may generate traffic at different patterns, may require different delay bounds, channel qualities may vary over time due to fading, and the amount of time needed for a transmission can depend on the transmission rate which can differ from client to client and change over time. In this section, we generalize the model in Section II to incorporate all these concerns.

We still consider a system with one AP and N clients, and assume that time is slotted and grouped into intervals, where one interval consists of T time slots. We suppose that different clients may generate packets at different patterns. At the beginning of each interval, each client may or may not generate one packet. The process of packet generation is assumed to be a stationary, irreducible Markov chain with finite states. The long-term average probability that only a subset S of clients generates a packet in an interval is denoted by $R(S)$. Further, different clients may require different delay bounds. We denote the delay bound of client n by τ_n, which is assumed to be smaller or equal to T. If client n generates a packet at the beginning of an interval, this packet needs to be delivered no later than the τ_n^{th} time slot of the interval, or the packet expires and is dropped from the system.

We consider fading channels. We assume that channel conditions evolve as a stationary, irreducible Markov chain with finite states. While channel conditions can vary from interval to interval, we assume that they are static during an interval. We use $c \in C$ to denote the channel state in an interval, which includes the channel conditions of all clients.

We consider both systems with and without rate adaptation. When rate adaptation is not employed, the AP uses the same data rate to transmit packets for all clients. Therefore, we assume that all transmissions take one time slot. In this case, channel conditions determine channel reliabilities. To be more specific, we suppose that under channel state c a transmission for client n will be successful with probability $p_{c,n}$.

On the other hand, when rate adaptation is employed, the AP chooses a suitable data rate for each client, so as to prevent transmissions from failing. Different data rates effectively result in different times needed for a transmission. We model this by supposing that under channel state c a transmission for client n takes $s_{c,n}$ time slots. In this case, all transmissions are successfully delivered. Since their durations can vary according to channel condition, reliability is bought at

the expense of data rate.

VII. A Framework for Scheduling Policies and Its Applications

In this section, we study the problem of designing feasibility optimal scheduling policies under the above generalized model. We introduce a framework for designing such policies. We also demonstrate the utility of this framework by deriving feasibility optimal policies for two specific scenarios.

This framework starts by extending the concept of *debt* introduced in Section IV.

Definition 7: A variable $r_n(k)$, whose value is determined by the past history of the client n up to the k^{th} period, or time slot kT, is called a *pseudo-debt* if:

1) $r_n(0) = 0$, for all n.
2) At the beginning of each period, $r_n(k)$ increases by a constant strictly positive number $z_n = z_n(q_n)$, which is an increasing linear function of q_n.
3) $r_n(k+1) = r_n(k) + z_n(q_n) - \mu_n(k)$, where $\mu_n(k)$ is a non-negative and bounded random variable whose value is determined by the behavior of client n. Further, $\mu_n(k) = 0$ if the AP does not transmit any packet for client n.
4) The system is fulfilled if and only if $Prob\{\frac{r_n(k)}{k} < \varepsilon\} \to 1$, as $k \to \infty$, for all n and all $\varepsilon > 0$.

It can be shown that for the basic model in Section II, both the time-based debt, $r_n^{(1)}$, and the weighted-delivery debt, $r_n^{(2)}$, are also pseudo-debts.

Example 2: The time-based debt can be expressed as $r_n^{(1)}(k+1) = r_n^{(1)}(k) + w_n - u_n(k)$, where $w_n = \frac{q_n}{p_n}$ and $u_n(k)$ is the number of time slots that the AP spends transmitting packets for client n in the k^{th} interval.

On the other hand, the weighted-delivery debt can be expressed as $r_n^{(2)}(k+1) = r_n^{(2)}(k) + \frac{q_n}{p_n} - \frac{d_n(k)}{p_n}$, where $d_n(k)$ is the indicator function that the AP delivers a packet for client n in the k^{th} interval.

Furthermore, when the generalized model is considered, we can define a *delivery debt*, $r_n^{(3)}(k+1) := kq_n - \sum_{j=1}^{k} d_n(j)$. We then have $r_n^{(3)}(k+1) = r_n^{(3)}(k) + q_n - d_n(k)$, and thus delivery debt is also a pseudo-debt. □

We can establish the following sufficient condition for a scheduling policy to be feasibility optimal for the considered scenario.

Theorem 6 ([5], Theorem 3): Let $r_n(k)$ be a pseudo-debt, c_k be the channel state of the k^{th} interval, and S_k be the set of clients that has generated packets at the beginning of the k^{th} interval. A policy that maximizes $\sum_{n=1}^{N} E\{r_n(k)^+ \mu_n(k) | [r_n(k)], c_k, S_k\}$ for each interval k is feasibility optimal, where $x^+ := \max\{x, 0\}$.

The above theorem provides a framework for designing scheduling policies: For each scenario, one should first choose a suitable pseudo-debt, and then aim to maximize

$$\sum_{n=1}^{N} E\{r_n(k)^+ \mu_n(k) | [r_n(k)], c_k, S_k\}$$

for each interval k. In the following sections, we demonstrate the utility of this framework by deriving tractable feasibility optimal scheduling policies for two different scenarios.

A. A Scheduling Policy for Unreliable Fading Channels

We first consider the scenario where rate adaptation is not employed. Thus, all transmissions take one time slot long, and a transmission for client n is successful with probability $p_{c,n}$, under channel state c. Further, we assume that all clients require the same delay bound, that is, $\tau_n \equiv T$.

We use the delivery debt, $r_n^{(3)}(k)$. The proposed policy is called the *joint debt-channel policy*. In the joint debt-channel policy, the AP sorts all clients in descending order of $p_{c_k,n} r^{(3)}(k)$, at the beginning of each interval, where c_k is the channel state of the k^{th} interval. In each time slot within the interval, the AP schedules a transmission for the client whose $p_{c_k,n} r^{(3)}(k)$ is the largest among those whose packets are not delivered yet. This policy is similar to the Longest Connected Queue (LCQ) policy proposed by Tassiulas and Ephremides [13], although they consider a much simpler channel model and do not provide delay guarantees. On one hand, this policy gives higher priority to clients with larger $r_n^{(3)}(k)$, so as to prevent a client from being starved. On the other hand, the policy also takes advantage of current channel state by opportunistically assigning higher priority to clients with larger $p_{c_k,n}$. We can prove that this simple on-line policy is feasibility optimal.

Theorem 7 ([5], Theorem 4, and [9], Theorem 3): The joint debt-channel policy is feasibility optimal.

B. A Scheduling Policy for Rate Adaptation

We next consider the scenario where rate adaptation is employed. A transmission for client n takes $s_{c,n}$ time slots under channel state c, and all transmissions are error-free. Also, each client n specifies its individual delay bound, τ_n, with each $\tau_n \leq T$..

We use the delivery debt, $r_n^{(3)}(k)$. In each interval, the AP schedules transmissions for some of its clients. The schedule of transmissions can be expressed as an ordered list $L = \{l_1, l_2, \dots\}$, with the interpretation that the AP first transmits the packet for client l_1, and then the packet for client l_2, and so on. The packet of client l_m is then delivered at time $\sum_{n=1}^{m} s_{c,l_n}$. Thus, to meet the delay bound of each client, we require that $\sum_{n=1}^{m} s_{c,l_n} \leq \tau_{l_m}$, for all scheduled clients.

Suppose that, in some interval k, the AP schedules transmissions for a list L of clients, and delivers all their packets on time. We then have $\mu_n(k) = 1$, for all $n \in L$, and $\mu_n(k) = 0$, otherwise. Thus, $\sum_{n=1}^{N} E\{r_n^{(3)}(k)^+ \mu_n(k) | [r_n^{(3)}(k)], c_k, S_k\} = \sum_{n \in L} r_n^{(3)}(k)^+$. Maximizing $\sum_{n \in L} r_n^{(3)}(k)^+$ while meeting all delay bounds for all scheduled clients is actually a variation of the knapsack problem. The policy shown in Algorithm 1, the *modified knapsack policy*, can be proved to be feasibility optimal.

Algorithm 1 Modified Knapsack Policy [5]

Sort clients such that $\tau_1 \leq \tau_2 \leq \cdots \leq \tau_N$
$L[0,0] = \phi$
$M[0,0] = 0$
for $n = 1$ to N **do**
  for $t = 1$ to T **do**
    if $t > \tau_n$ **then**
      $M[n,t] = M[n,t-1]$
      $L[n,t] = L[n,t-1]$
    else if client n has a packet AND $r_n^{(3)}(k) + M[n-1, t-s_{c,n}] > M[n-1,t]$ **then**
      $M[n,t] = r_n^{(3)}(k) + M[n-1, t-s_{c,n}]$
      $L[n,t] = L[n-1, t-s_{c,n}] + \{n\}$
    else
      $M[n,t] = M[n-1,t]$
      $L[n,t] = L[n-1,t]$
schedule according to $L[N,T]$

VIII. Utility Maximization for Systems with Rate Adaptation

In this section, we address the problem of utility maximization for systems that employ rate adaptation. we will address two complexities. The first is that each client requires a certain minimum timely-throughput. The second is that clients can engage in strategic behavior. That is, they can lie about their utility functions to the AP, in the hope that they can thereby secure a higher throughput.

Similar to Section V, we assume that the timely-throughput requirements of clients are elastic. Each client n receives a utility of $U_n(q_n)$ when the timely-throughput it receives is q_n. The utility

functions, $U_n(\cdot)$, are assumed to be concave and infinitely differentiable. We will further consider the additional complexity that, although the timely throughput requirements are elastic, each client n still requires a guaranteed minimum timely-throughput, which is denoted $\underline{q}_n$, i.e., the AP needs to guarantee that $q_n \geq \underline{q}_n$.

The goal of the AP is to maximize the total utility of the system, while providing minimum timely-throughput guarantees for all the clients. This problem can be formally stated as follows:

$$\text{Max} \sum_{n=1}^{N} U_n(q_n) \tag{10}$$

$$\text{s.t. Network dynamics and feasibility constraints,} \tag{11}$$

$$\text{and } q_n \geq \underline{q}_n, \forall n. \tag{12}$$

In addition, we allow for the selfish behavior by clients. In practice, as the AP does not have exact information of utility functions, a strategic client may lie about its utility function so as to gain more service from the AP. Thus, a mechanism that can both prevent clients from benefiting by lying while still achieving the maximum total utility in the system is needed.

A. A Truthful Auction Design and Its Optimality

Before we introduce our proposed mechanism, we first introduce the concept of *truthful auction*. We denote the timely-throughput of client n up to the k^{th} interval by $q_n(k)$, i.e., $q_n(k) := \frac{\sum_{j=1}^{k} d_n(j)}{k}$, where $d_n(j)$ is the indicator function that the AP has delivered a packet for client n in the j^{th} interval. We assume that, if the system terminates, or client n leaves the system, at the end of the k^{th} interval, then client n receives a *reward* in the amount of $kU_n(q_n(k))$, and hence its time-average reward is $U_n(q_n(k))$ per interval. Let $q_n^+(k)$ and $q_n^-(k)$ denote the values of $q_n(k)$ if client n is scheduled in the k^{th} interval, and if it is not, respectively. To be more specific, we have $q_n^+(k) = \frac{k-1}{k} q_n(k) + \frac{1}{k}$, and $q_n^-(k) = \frac{k-1}{k} q_n(k)$. If the system terminates at the end of the k^{th} interval, the amount of reward that client n receives is either $kU_n(q_n^+(k))$ or $kU_n(q_n^-(k))$, depending on whether client n is scheduled in the k^{th} interval or not. We hereafter define the *marginal reward* of client n at the k^{th} interval to be $kU_n(q_n^+(k)) - kU_n(q_n^-(k))$.

In an auction, each client n makes a *bid*, $b_n(k)$, to the AP at the beginning of each interval k. Based on these bids, the AP schedules some clients for transmissions and charges each client n an amount of $\rho_n(k)$. The scheduling and charging decisions of the AP are determined by the employed auction mechanism. The *net reward* of client n, if the system terminates at the end of the k^{th} interval, can hence be expressed as $kU_n(q_n(k)) - \sum_{j=1}^{k} \rho_n(j)$. At the beginning of each interval

k, client n aims to greedily maximize its net reward based on past system history, assuming that the system will terminate at the end of this interval. Thus, client n aims to maximize $kU_n(q_n(k)) - \sum_{j=1}^{k} \rho_n(j)$, given $q_n(k-1)$ and $\rho_n(1), \rho_n(2), \ldots, \rho_n(k-1)$. We say that an auction mechanism is *truthful* if a strategic client may bid according to its real marginal reward.

Definition 8: An auction mechanism is *truthful* if choosing

$$b_n(k) = kU_n(q_n^+(k)) - kU_n(q_n^-(k)) \tag{13}$$

yields the highest net reward for each client n, if the system terminates at the end of the k^{th} interval.[2]

We now present our proposed auction mechanism. This mechanism is based on the Vickrey-Clarke-Grove auction [2], [3], [14]. At the beginning of each interval k, the AP announces a non-negative *discount*, $\delta_n(k)$, to each client n. Each client n then offers its bid, $b_n(k)$. The AP chooses a sorted list $L = \{l_1, l_2, \ldots\}$ and schedules clients accordingly, such that $\sum_{n\in L}(b_n(k) + \delta_n(k))$ is maximized among all sorted lists under which the packet of each scheduled client is delivered before its delay bound. Let L^{-n} be the schedule that the AP would have chosen if client n were not present in the system. The AP charges each scheduled client n an amount of $\rho_n(k) = \sum_{m\in L^{-n}}(b_m(k) + \delta_m(k)) - \sum_{m\in L, m\neq n}(b_m(k) + \delta_m(k)) - \delta_n(k)$. The AP does not charge anything to clients that are not scheduled. It can be shown that this auction mechanism is truthful [7].

In addition, it can be shown that [7], using a dual-decomposition technique proposed by Lin and Shroff [11], the total utility in the system can be maximized by employing this truthful auction mechanism and adapting $\delta_n(k)$ according to

$$\delta_n(k+1) = \{\delta_n(k) + \frac{1}{k}[\underline{q}_n - q_n(k)]\}^+. \tag{14}$$

Finally, this auction mechanism requires the AP to find the list L that maximizes $\sum_{n\in L}(b_n(k) + \delta_n(k))$. This is actually a similar problem to the one discussed in Section VII-B. By substituting $r_n^{(3)}(k)$ with $\sum_{n\in L}(b_n(k)+\delta_n(k))$, Algorithm 1 finds the desired list L. In sum, the proposed auction mechanism is both truthful and optimal, while being easy to implement.

IX. Broadcasting Real-Time Flows with Network Coding

In this section, we discuss the problem of broadcasting real-time flows. The major difference between wireless broadcasting and wireless unicasting is that, since ACKs are not implemented for

[2]A broader and more detailed discussion of truthful auction mechanisms can be found in [12].

broadcasting, and it is costly to obtain feedback information from all clients, the AP usually does not have immediate feedback information on whether a client receives a transmission correctly. It turns out that network coding can be fruitfully employed in such situations. In this section, we discuss a model for broadcasting such real-time flows. We then introduce a framework for designing scheduling policies. The utility of this framework is demonstrated by deriving policies for several various scenarios, including those that employ different network coding techniques.

A. System Model

We consider a system where the AP broadcasts I real-time flows to N clients. Time is divided into slots, and further grouped into intervals, where an interval consists of T time slots. At the beginning of each interval, each flow i may or may not generate a packet. The packet generations of flows are assumed to constitute a stationary irreducible Markov chain with finite states. All packets require a delay bound of T time slots. Each client n has an inelastic timely-throughput requirement for each flow i, which is denoted by $q_{i,n}$.

For wireless broadcasting without ACKs, it is helpful for the AP to use network coding. In each time slot, the AP broadcasts a packet, either a raw packet from a real-time flow, or a coded packet that involves packets from multiple flows. We do not make any assumptions on the actual coding scheme that the AP can employ. When the AP broadcasts a packet, each client n receives the packet with probability p_n. However, the AP has no knowledge on whether a particular client receives the packet of a particular transmission. To enhance the probability of packet delivery, the AP may transmit the same packet multiple times in an interval, which, in turn, may result in some clients receiving several duplicate packets. In this case, redundant duplicate packets are dropped by the client, as they do not carry any new useful information.

B. A Framework for Scheduling Real-Time Broadcast Flows

We now introduce a framework for designing scheduling policies. This framework is similar to the one for unicast flows in Section VII. Although the AP does not have per-transmission feedback, it can still estimate the number of packet deliveries. For example, suppose in some interval, the AP broadcasts the packet from flow 1 twice, and the packet from flow 2 once, then the probabilities that a client n receives the packet from flow 1 and that from flow 2 are $1-(1-p_n)^2$, and p_n, respectively. We hereafter use $\xi_{i,n}(k)$ to denote the probability that client n obtains the packet from flow i in the k^{th} interval. The law of large numbers shows that the long-term timely-throughput of client n on flow i is $\liminf_{K\to\infty}\frac{\sum_{k=1}^{K}\xi_{i,n}(k)}{K}$. We can then define the *expected delivery debt* as follows:

Definition 9: The *expected delivery debt*, $\hat{r}_{i,n}(k+1)$, of client n on flow i at the beginning of the $(k+1)^{th}$ interval is defined as

$$\hat{r}_{i,n}(k+1) := kq_{i,n} - \sum_{j=1}^{k} \xi_{i,n}(k). \tag{15}$$

We can obtain a sufficient condition for a scheduling policy to be feasibility optimal with respect to all policies using some specified network coding scheme:

Theorem 8 ([8], Theorem1): Given past system history and the set of flows that generate packets at the beginning of the k^{th} interval, a policy that maximizes

$$\sum_{i=1}^{I}\sum_{n=1}^{N} \hat{r}_{i,n}(k)^{+}\xi_{i,n}(k+1),$$

for all k, among all policies using a specified network coding scheme, is feasibility optimal under this network coding scheme.

In the following sections, we demonstrate the utility of this framework by deriving policies for three scenarios, one that does not employ network coding, one that employs XOR coding, and one that employs linear coding.

C. Broadcasting without Network Coding

We first discuss the scenario when the system does not employ network coding, that is, the AP can only broadcast a raw packet from a single flow in each time slot. If the AP broadcasts the packet from flow i a total of σ_i times in an interval, a client n will successfully receive this packet with probability $1-(1-p_n)^{\sigma_i}$. We define the *marginal delivery probability* of the ς_i^{th} transmission for flow i and client n as $\pi_{i,n,\varsigma_i} := [1-(1-p_n)^{\varsigma_i}] - [1-(1-p_n)^{\varsigma_i-1}] = p_n(1-p_n)^{\varsigma_i-1}$, which is the *additional* delivery probability that flow i and client n can benefit from the ς_i^{th} transmission.

We proposed a greedy policy for broadcasting without network coding. Suppose that each flow i has been transmitted ς_i times before a time slot in the $(k+1)^{th}$ interval. The AP then schedules a transmission for the flow that has generated a packet in the interval and maximizes $\sum_{n=1}^{N} \hat{r}_{i,n}(k)^{+}\pi_{i,n,\varsigma_i}$ for this time slot. The following example illustrates this scheduling policy.

Example 3: Consider a system with one client, whose channel reliability is $p_1 = 0.6$, two flows, and $T = 3$. Suppose that, at the beginning of some interval k, the expected delivery debts of the two flows are $\hat{r}_{1,1} = 1$ and $\hat{r}_{2,1} = 0.8$. In the first time slot of the interval, the marginal delivery probabilities for both flows are 0.6. The AP thus broadcasts the packet from the first flow. In the second time slot of the interval, as the packet for flow 1 has been already broadcast once, the

marginal delivery probabilities for the two flows become $0.4 \times 0.6 = 0.24$, and 0.6, respectively. As $\hat{r}_{1,1} \times 0.24 < \hat{r}_{2,1} \times 0.6$, the AP broadcasts the packet from flow 2. In the last time slot of the interval, the marginal delivery probabilities for both flows are 0.24, and the AP broadcasts the packet from flow 1. □

Theorem 2 in [8] shows that this policy is feasibility optimal among the class of all policies which do not employ network coding.

D. Broadcasting with XOR Coding

In this section, we discuss the scenario where the AP may employ XOR coding as a specific network coding mechanism. In this scenario, in addition to broadcasting raw packets of flows, the AP can also broadcast a packet that contains the XOR of packets from two flows. We use $i \oplus h$ to denote the packet that contains the XOR of packets from flow i and flow h. A client can obtain the packet from flow i by either receiving a raw packet from flow i, or receiving both a raw packet from flow h and a coded packet $i \oplus h$. The following example shows that we can indeed improve system performance by incorporating XOR coding.

Example 4: Consider a system with two flows that generate one packet in each interval, and only one client whose channel reliability is $p_1 = 0.5$. Assume that there are six time slots in an interval. Suppose that the AP transmits each packet three times in an interval. Then we have $\xi_{1,1}(k) = \xi_{2,1}(k) = 0.875$. Thus, a system with timely-throughput requirements $q_{1,1} = q_{2,1} > 0.875$ is not feasible when network coding is not employed. On the other hand, a system that employs XOR coding can transmit each of the three different types of packets, the raw packet from each flow and the encoded packet $1 \oplus 2$, twice in each interval, which achieves $q_{1,1} = q_{2,1} = 0.890625$. □

Designing a feasibility optimal policy for XOR coding may be computationally complicated. Instead, we aim to design a policy that is both tractable and has better performance than policies that do not employ network coding. Let σ_i be the number of times that the packet from flow i is transmitted under the greedy policy in some interval, given the expected delivery debts of flows and packet generations at the beginning of this interval. We first sort all flows so that $\sigma_1 \geq \sigma_2 \geq \ldots$. We enforce two mild restrictions for our policy with XOR coding:

1) In addition to raw packets, we only allow encoded packets of the form $(2i-1) \oplus (2i)$. The intuition behind this restriction is that we only combine two packets which have each been transmitted a similar number of times under the greedy policy, which implies that they have similar importance.

2) The total number of transmissions scheduled for the raw packets from flow $2i-1$ and flow $2i$, as well as the encoded packet $(2i-1)\oplus(2i)$, equals $\sigma_{2i-1}+\sigma_{2i}$. The intuition behind this restriction is that we aim to enhance the performance of flows $2i-1$ and $2i$ by XOR coding without hurting other flows.

These two restrictions are called the *pairwise combination restriction* and the *transmission conservation restriction for XOR coding*, respectively. By enforcing these restrictions, designing a policy for XOR coding breaks down to determining the number of transmissions for each of the three packets, raw packet from flow $(2i-1)$, raw packet from flow $2i$, and coded packet $(2i-1)\oplus(2i)$, respectively, for all i, so that the resulting $\sum_{i=1}^{I}\sum_{n=1}^{N}\hat{r}_{i,n}(k)^{+}\xi_{i,n}(k+1)$ is maximized. This problem can be optimally solved in polynomial time. In [8] it is proved that this policy is feasibility optimal among all policies that employ XOR coding and follow the above two restrictions. Further, it should be noted that this policy fulfills all systems that can be fulfilled without network coding.

E. Broadcasting with Linear Coding

We now discuss the scenario where network coding with linear coding is employed. When linear coding is employed, the AP divides all flows into several groups at the beginning of each interval. In each time slot within the interval, the AP chooses a group of flows and broadcasts a packet that contains a linear combination of packets from flows in this group. For each group G, if the number of coded packets that a particular client has received for the group G is at least the size of G, then this client can obtain all packets from flows in G through Gaussian elimination; otherwise, the client cannot obtain any packets from flows in this group. The following example suggests that linear coding may improve performance.

Example 5: Consider a system with one client, whose channel reliability is $p_1=0.5$, three flows that generate one packet in each interval, and nine time slots in an interval. A similar argument as that in Example 4 shows that $q_{1,1}=q_{2,1}=q_{3,1}>0.875$ is not feasible when network coding is not employed. On the other hand, if the AP employs linear coding and broadcasts a linear combination of the three flows in each time slot, the client can decode all packets from the three flows if it receives at least three packets out of the nine transmissions in an interval, which has probability 0.91015625. □

To reduce the computational overhead of linear coding, we also enforce two restrictions similar to those used for XOR coding. Let σ_i be the number of times that the packet from flow i is transmitted under the greedy policy in some interval, given the expected delivery debts of flows

and packet generations at the beginning of this interval. We sort all flows so that $\sigma_1 \geq \sigma_2 \geq \ldots$. The two restrictions are then:

1) Flows are grouped into subsets as $G_1 = \{1, 2, \ldots, g_1\}, G_2 = \{g_1 + 1, \ldots, g_2\}, \ldots$. In each time slot, the AP broadcasts a linear combination of packets from flows in one of the subsets $G_1, G_2, \ldots$. The intuition behind this restriction is that we only combine packets that have been scheduled similar numbers of times.
2) The AP broadcasts linear combinations of packets from the subset $G_h = \{g_{h-1} + 1, g_{h-1} + 2, \ldots, g_h\}$ a total number of $\sum_{i=g_{h-1}+1}^{g_h} \sigma_i$ times, where we set $g_0 = 0$. The intuition behind this restriction is that we aim to enhance the performance of flows within G_h without hurting other flows.

The two restrictions above are called the *adjacent combination restriction* and the *transmission conservation restriction for linear coding*, respectively. Under these two restrictions, the problem of finding the optimal schedule for linear coding can be solved by dynamic programming. In [8], it is proved that the resulting policy is feasibility optimal among policies that employ linear coding and follow the above two restrictions. Further, the resulting policy fulfills every system that can be fulfilled without network coding.

X. Concluding Remarks

We have introduced a theory for real-time communication over unreliable wireless links and have surveyed some of its important results. The core of this theory is a model that incorporates three important aspects of wireless real-time communications: a per-packet delay bound, a timely-throughput requirement of each flow, and the presence of unreliable and heterogeneous wireless channels. This model can be extended to different scenarios, including those where flows generate packets under different patterns and have different delay bounds, those where channel qualities vary over time, and those where rate adaptation is employed. We have addressed three important problems vis-a-vis serving real-time flows: admission control, packet scheduling, and utility maximization. Further, we have also considered the problem of broadcasting real-time flows. We have incorporated the optional usage of network coding and derived policies for broadcasting.

There remain several open problems concerning serving real-time flows through wireless networks. Section IV has proposed two scheduling policies that provide long-term timely-throughput guarantees for clients. However, the short-term performance of these policies, that is, whether the timely-throughput a client receives may fluctuate greatly in a small time scale, is not addressed.

Section V has proposed a bidding procedure and shown that the total utility in the system is maximized when the bidding procedure converges. However, whether this procedure is guaranteed to converge is not understood. Moreover, the models discussed in this paper all focus on the scenario where there is one AP that schedules transmissions for all clients. In many deployments, there may be multiple APs that contend for wireless resources. Also, in wireless ad-hoc networks, packet deliveries may require multi-hop relaying. How to provide per-packet delay guarantees for these networks is a challenging open problem.

REFERENCES

[1] Cisco. Cisco visual networking index: Forecast and methodology, 2010-2015.

[2] E. Clarke. Multipart pricing of public goods. *Public Choice*, 11(1):17–33, Sep. 1971.

[3] T. Groves. Incentives in teams. *Econometrica*, 41(4):617–631, Jul. 1973.

[4] I-H. Hou, V. Borkar, and P.R. Kumar. A theory of QoS in wireless. In *Proc. of IEEE INFOCOM*, pages 486–494, 2009.

[5] I-H. Hou and P.R. Kumar. Scheduling heterogeneous real-time traffic over fading wireless channels. In *Proc. of IEEE INFOCOM*, 2010.

[6] I-H. Hou and P.R. Kumar. Utility maximization for delay constrained QoS in wireless. In *Proc. of IEEE INFOCOM*, 2010.

[7] I-H. Hou and P.R. Kumar. Utility-optimal scheduling in time-varying wireless networks with delay constraints. In *Proc. of ACM MobiHoc*, pages 31–40, 2010.

[8] I-H. Hou and P.R. Kumar. Broadcasting delay-constrained traffic over unreliable wireless links with network coding. In *Proc. of ACM MobiHoc*, pages 33–42, 2011.

[9] I-H. Hou, A. Truong, S. Chakraborty, and P.R. Kumar. Optimality of periodwise static priority policies in real-time communications. To appear in *Proc. of CDC*, 2011.

[10] F.P. Kelly, A.K. Maulloo, and D.K.H. Tan. Rate control in communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, 49:237–252, 1998.

[11] X. Lin and N. Shroff. Joint rate control and scheduling in multihop wireless networks. In *Proc. of IEEE CDC*, pages 1484–1489, 2004.

[12] N. Nisan, T. Roughgarden, E. Tardos, and V.V. Vazirani. *Algorithmic game theory*. Cambridge University Press, 2007.

[13] L. Tassiulas and A. Ephremides. Dynamic server allocation to parallel queues with randomly varying connectivity. *IEEE Trans. on Information Theory*, 39(2):89–103, 1993.

[14] W. Vickrey. Counterspeculation, auctions and competitive sealed tenders. *J. of Finance*, 16:8–37, 1961.

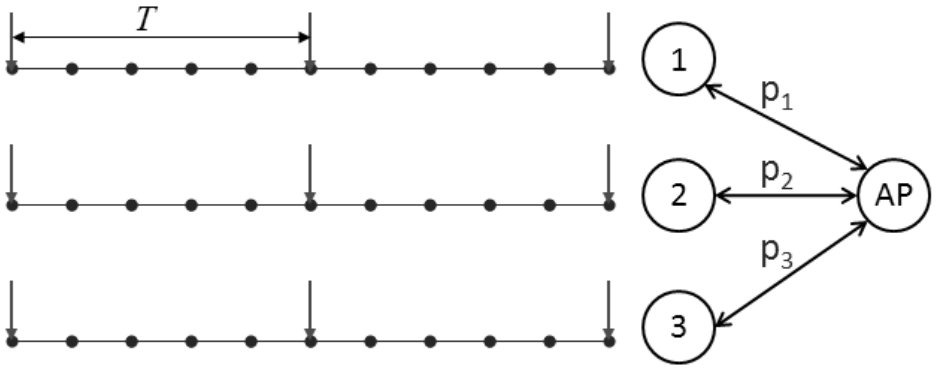

Fig. 1: An example that illustrates the system model. The right half of the figure shows the topology that includes one AP and three clients. The left half of the figure shows the timeline of each client. We use an arrow to indicate the beginning of an interval.

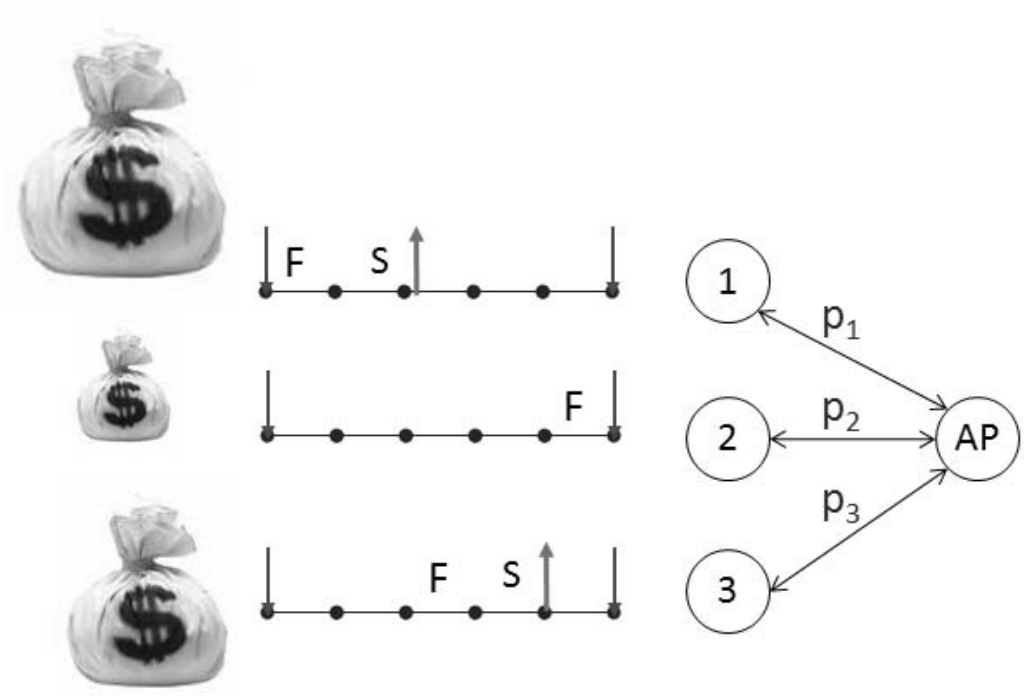

Fig. 2: An example illustrates the largest debt first scheduling policy. The size of the money bag indicates the amount of debt that the AP owes to the client. We use 'F' to denote a failed transmission, and 'S' to denote a successful one.

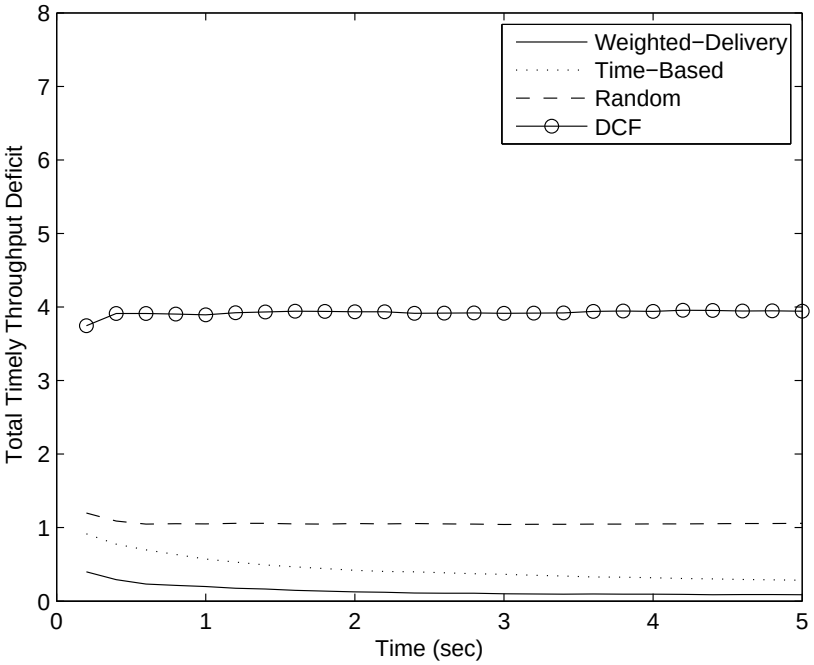

Fig. 3: Performance of a feasible set

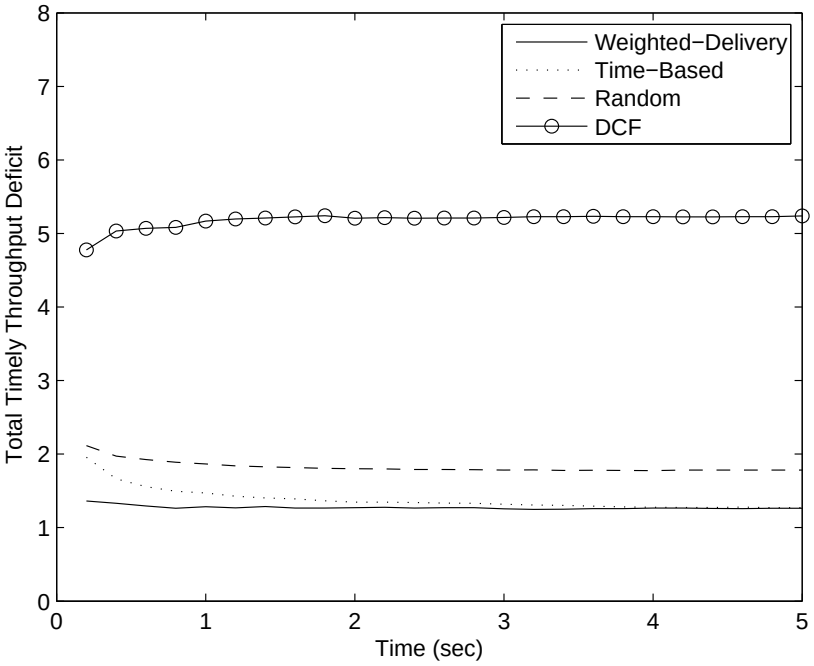

Fig. 4: Performance of an infeasible set

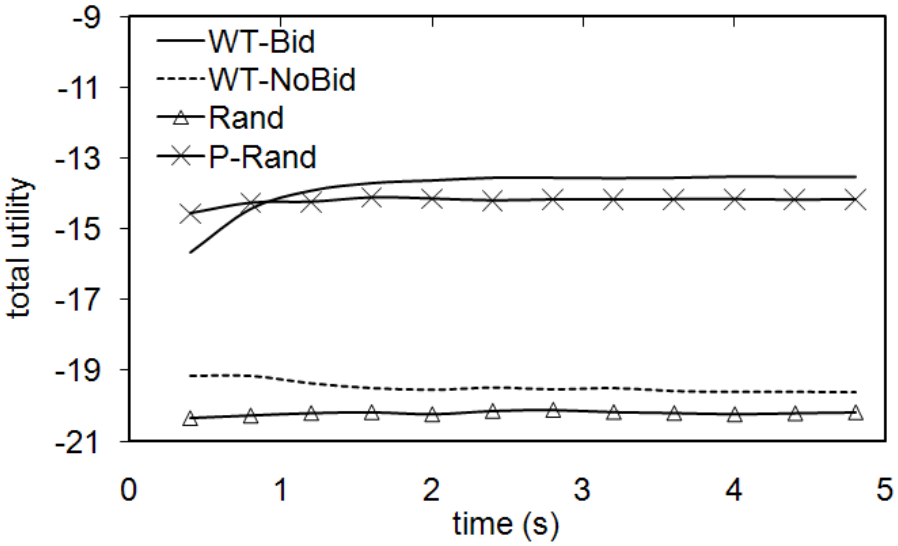

Fig. 5: Performance of the first setting

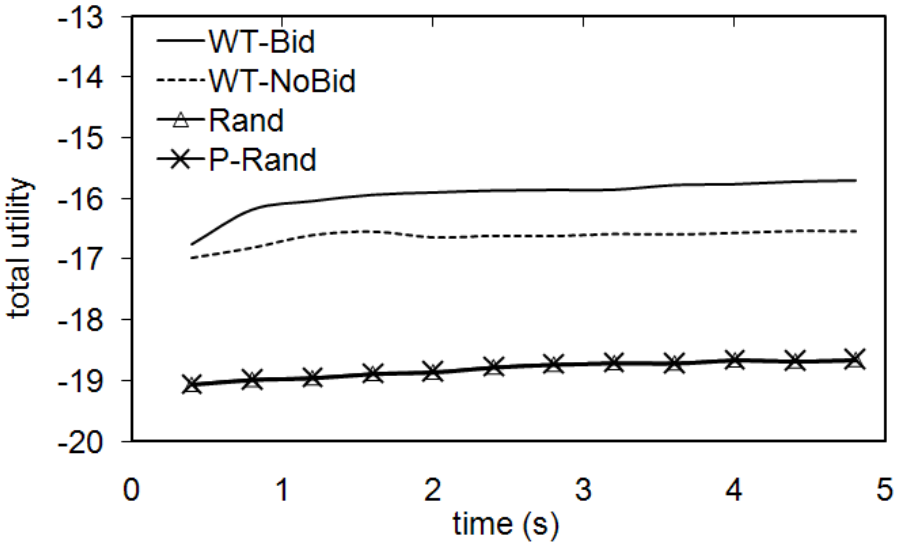

Fig. 6: Performance of the second setting

TABLE I: Simulation Setup

Bit rate	64 kbp
Packetization interval	20 ms
Payload size per packet	160 Bytes
Time slot length	610 μs
Transmission data rate	11 Mb/s