Call Admission Control for Real-time Applications in Wireless Network

S. Agrawal, P. Chaporkar, R. Udwani

Abstract—Supporting real-time applications is paramount to sustaining the growth of wireless networks. Recent standards like LTE and WiMax do provide a framework for supporting real-time applications. These standards, however, are limited in the sense that the resource allocation and call admission control procedures are left to designers’ discretion. In fact, to the best of our knowledge, no satisfactory solution exists. In this paper, our aim is to develop a scalable admission control algorithm for the real-time applications.

Real-time applications require strict delay guarantee, i.e., a packet delayed beyond certain predefined value is dropped. Fortunately, depending on the codec used, real-time applications can sustain some loss gracefully. Aim of an admission control algorithm is to make sure that when a new flow is admitted, its and other existing flows’ packet loss on account of deadline violation is below their respective acceptable limit. The problem of admission control has been studied extensively for wiredline networks. However, this analysis does not extend to wireless case on account of fading. Here, we consider a wireless network with TDMA based MAC, and for this network obtain a scalable admission control algorithm.

I. INTRODUCTION

In the past decade there has been a steady rise in the importance of wireless networks as a preferred access network. To sustain the growth, wireless networks have to efficiently support real-time services such as voice over IP, multimedia downloads, real-time gaming. The common feature of all such real-time applications is the need for transmission of data with low time lags. One common way to address this issue is through the concept of delay. Here, all packets have to be transmitted within a fixed time delay and if it is not possible, then the packets are dropped. This guarantees that whichever packets have arrived can not have a lag more than a particular constant. So, we are able to send data in “real time”. However, this introduces the unavoidable issue of packet drops as the wireless network is not static in nature, i.e., the channel conditions are location dependent and time varying. Thus, if the channel is in deep fade for a long enough time (which can happen), then some of the packets would have to be dropped on account of deadline violations. Hence, we may never achieve the 100% packet delivery ratio. Fortunately, depending on the codec used, the real-time applications can sustain some loss gracefully. Thus, quality of service (QoS) guarantee required for real-time applications is that the packet drops on account of deadline violations have to be below certain predefined value.

Due to the limited availability of resources, primarily bandwidth, only a limited number of real-time flows can be accommodated in the system so that the respective QoS requirement of each is met. Thus, in a dynamic scenario, where flows arrive and depart upon completion of service, one has to decide whether the arriving flow should be admitted in the system. Clearly, the criteria for admission has to be that the admitted flow gets its required QoS without violating the QoS guarantees of the existing flows. A procedure that arbitrates the admission of an arriving flow in the system is referred to as admission control algorithm. Apart from making correct admission control decision, admission control algorithm has to be scalable, i.e., it should remain computationally efficient even as the number of flows increases. In this paper, our aim is to propose a scalable admission control algorithm for down-link of TDMA based wireless system. Key challenge here is to account for the location dependent and time varying channel conditions. The main reason for considering TDMA is that it has low operational complexity, which makes it an attractive option for implementation. Moreover, many deployed wireless technologies such as GSM based cellular networks and WiMax use TDMA. Note that WiMax uses TDMA for Unsolicited Grant Service (UGS) [12] [13].

The paper is arranged as follows: In Section II we describe the system model that we consider. In Section III we solve the model and find conditions for the system to be feasible. In Section IV we describe in detail the proposed admission control algorithm. In Section V we discuss the algorithm and provide generalizations. In Section VI we evaluate the proposed algorithm using simulations. In Section VII we discuss the related work, and in Section VIII we conclude.

II. SYSTEM MODEL

As a first step towards obtaining an admission control algorithm, we first study the down-link of a wireless system with an access point and N real-time flows. Here, the system is static, i.e., flows do not arrive or depart. The dynamic scenario is considered latter in Section IV. Let \( \mathcal{N} = \{1, 2, \ldots, N\} \) denote the set of flows. Time is divided into slots. Access point employs Time Division Multiple Access (TDMA) for distributing resources among flows. Thus, flow \( n \) can only be served in slots \( t \) such that \( t \mod N = n \). Now, fix any user \( n \), and partition time into frames \([n + kN, n + (k + 1)N) \) for \( k = 0, 1, 2, \ldots \). Here, \( k \)th frame \([n + kN, n + (k + 1)N) \) has slots \( t \) such that \( n + kN \leq t < n + (k + 1)N \). Note that in each frame, flow \( n \) is served only in the first slot of the frame. Next, we describe queuing model we consider for flow \( n \).

Let \( X_n(k) \) denote the number of packets arriving for flow \( n \) in \( k \)th frame. The arrival process \( X_n = \{X_n(k)\}_{k \geq 0} \) is a stochastic process comprising of independent and identically
distributed (i.i.d.) random variables with known distribution function given by \( p_{n,a} = \mathbb{P}\{X_n(k) = a\} \) for \( a = 0, 1, \ldots \). Let 
\[ \alpha_n = \mathbb{E}[X_n(k)] \quad \text{and} \quad \mathcal{A}_n = \{0, 1, \ldots, A\}, \]
where \( A = \sup(a : p_{n,a} > 0) \) be the maximum element in the support of \( X_n(k) \).

We assume that the arrival processes for various flows are independent. Now, let \( Y_n(k) \) denote the number of packets that can be served for flow \( n \) in \( k \)th frame. To capture the location dependent and time varying channel conditions, the service process \( Y_n = \{Y_n(k)\}_{k \geq 0} \) is also assumed to be i.i.d. with distribution \( q_{n,b} = \mathbb{P}\{Y_n(k) = b\} \). This is analogous to the commonly used block fading model. Moreover, service processes for various flows are assumed to be independent. This is justified if the distance between the receivers corresponding to the flows is much larger than the carrier wavelength. Let the delay requirement for flow \( n \) be \( D_n \) frames, i.e., the packets arriving in frame \( k \) must depart before \( (k + D_n + 1) \)th frame, otherwise they will be dropped. The arriving packets are queued in the buffer until they are either served or dropped.

Separate queue is maintained for each flow. We assume that the packets arriving in frame \( k \) are queued in the buffer until they are either served or dropped. Because of the random nature of packet arrivals, there is no systematic queueing order. This is a property of the queueing system and not an assumption. Thus, the queue state at the beginning of frame \( k \) reflects the state of the queue processes for various flows and is independent.

A SYSTEM WITH FLOWS IN SET \( \mathcal{N} \)

We note that to determine system feasibility we need to consider the packet drop rate for each flow, i.e., the flow \( \hat{n} \) th packet. With this we put forth the following definition.

**Definition 1 (Feasible System):** A system with flows in set \( \mathcal{N} \) is said to be feasible if the fraction of packets dropped for every flow \( n \) is almost surely less than \( \tau_n \), i.e., \( L_n \leq \tau_n \alpha_n \) w.p. 1 for every \( n \in \mathcal{N} \).

It should be clear that the problem of admission control is the same as determining feasibility of the system that comes about on arrival of a new flow. So, in the next section, we discuss how feasibility of the system can be determined.

**III. SYSTEM FEASIBILITY**

We note that to determine system feasibility we need to consider \( L_n \) for all \( n \in \mathcal{N} \). Quantifying \( L_n \)’s involves exploration of queue processes \( \{S_n(k)\}_{k \geq 1} \) for every \( n \). Fortunately, once \( \mathcal{N} \) is specified queue processes for various flows are independent and can be studied in isolation. Hence, in this section, we fix arbitrary flow \( n \), and determine its loss \( L_n \). Since, there is no ambiguity regarding which flow is under consideration, we drop subscript “\( n \)” from notation for brevity and better readability. With this note lets begin.

**A. Modeling Queue Process as a Markov Chain**

To see that the queue process \( \{S(k)\}_{k \geq 1} \) is indeed a Markov chain, let us understand how this process evolves with time. Fix any \( k \) and let \( S(k) = s = [s_0 \cdots s_{D-1}]' \). Here, capital letters indicate random variables and small letters indicate their value. Recall that \( S(k) \) is the queue state at the beginning of frame \( k \). Now, in frame \( k \), some of the packets from the queue are served and some new packets arrive. Thus, the queue state at the beginning of frame \( k + 1 \) is given as follows: Let \( \hat{a} \) be the largest index such that \( \sum_{d=\hat{a}}^{D-1} s_d \geq Y_k \); set \( \hat{d} = -1 \) if no such index exists. Then,

\[
S_0(k+1) = X(k), \quad S_d(k+1) = s_{\hat{a}+1} \quad \text{for} \quad 0 < d \leq \hat{a}, \quad S_{\hat{d}+1}(k+1) = \sum_{d=\hat{d}}^{D-1} s_d - Y(k) \quad \text{if} \quad \hat{d} < D - 1, \quad S_d(k+1) = 0 \quad \text{for} \quad d > \hat{d} + 1 \quad \text{if} \quad \hat{d} < D - 1.
\]

Equation (2) holds because new arrivals will occupy \( 0^{th} \) place in vector \( S(k+1) \) as these are the only packets with \( D \) more frames till expiry. Now, note that since service is FIFO, only packets that can depart in the current frame are the ones in positions \( d \geq \hat{d} \). Alternatively, packets in position \( d < \hat{d} \) remain in the queue. But, for these packets, the number of frames till expiry is reduced by one. Thus, (3) follows. Note that (3) is not required when \( \hat{d} = 0, -1 \). Relations (4) and (5) reflect for the state change on account of packet transmissions. We observe that all the packets from positions \( d > \hat{d} \) are transmitted. Hence, (5) follows. Now, some packets in position \( \hat{d} \) may still remain in the system, and for these the number of frames till expiry is reduced by one. Thus, (4) follows. We note that when \( \hat{d} = -1 \), there are no packets at position \( \hat{d} \) and hence for this case (4) is not valid. Also equations (4) and (5) are not required when \( \hat{d} = D - 1 \). Note that in this case packets in position \( D - 1 \) are the only ones that are transmitted. Even when all packets in this position are not transmitted, remaining will be dropped. In fact, it is worthwhile to note that

\[
L(k) = [S_{D-1}(k) - Y(k)]^+, \quad \text{where} \quad [u]^+ = \max\{u, 0\},
\]

Now, from (2) to (5), it is clear that \( S(k+1) \) is a function of \( S(k) \), \( X(k) \) and \( Y(k) \). Since \( X(k) \) and \( Y(k) \) are independent of the observed queue process \( \{S(l)\}_{1 \leq l \leq k} \), we conclude that the queue process is indeed a time-homogeneous Markov chain. Next, we state the following result (proof is omitted due to space constraints):

**Theorem 1:** Queue process is a Markov chain with a single non-empty aperiodic closed communicating class which is positive recurrent.

Required can be established by showing the existence of a state which is reachable from every state. Thus, any closed
communicating class must contain this state, and the result follows.

Note that since the Markov chain is stationary ergodic, we can write the loss \( L \) in terms of the stationary distribution \( \pi \) of the Markov chain. Indeed from [19], we conclude the following by using the Renewal Reward Theorem (RRT):

\[
L = \sum_{s \in A^D} \pi(s) \mathbb{E}[s_{D-1}(k) - Y(k)]^+ \text{ w.p. 1.}
\]

Thus, the problem of finding packets dropped per frame for a flow is the same as finding the stationary distribution of the Markov chain. Unfortunately, finding \( \pi \) for general distributions of \( X(k) \) and \( Y(k) \) is not easy. Brute force approaches that solve the balance equation \( \pi = \pi P \) and the normalization equation \( \sum_{s \in A^D} \pi(s) = 1 \), require for us to solve \( (A + 1)^D \) equations, which can turn out to be computationally prohibitive even for the modest values of \( D \).

In the remaining part of this section, our aim is to devise computationally simple ways of quantifying \( L \).

**B. Reversing the System**

As we have seen in the previous section, the computational complexity of determining the system feasibility depends on the support \( A \) of the arrival process while support of the service process has little bearing. Note that in the TDMA based wireless system, the support of the arrival process may depend on the number of flows in the system as addition of a flow increases frame length. Since arrival process counts the number of packet arrivals in a frame, value of \( A \) may increase with \( N \). In contrast, the support of the service process is mostly fixed. This is because the support depends on the modulation and coding schemes available at the transmitter as these determine the possible transmission rates. Once the transmission rates are fixed, the values of number of packets that can be transmitted in a slot are also fixed. Thus, a scalable procedure should have complexity that depends on the support of the service process rather than that of the arrival process.

In this section, our aim is to show that it is indeed possible. Key idea for achieving the required is to look at the system with roles of arrival and service processes reversed, i.e., in the reversed system \( \mathcal{Y} \) is the arrival process, while \( \mathcal{X} \) is the service process.

Let \( L^{(O)} \) and \( L^{(R)} \) denote the losses under the original and the reversed systems respectively. Then, our key result states that \( L^{(O)} - L^{(R)} = \mathbb{E}[X(1)] - \mathbb{E}[Y(1)] \). Thus, computing the loss for the reversed system is sufficient to compute the loss for the original system. Now, we describe our approach.

In this section, since we are studying relation between two processes, we distinguish between them using a superscript “O” and “R” for original and reversed respectively. Next we precisely define the two systems:

**Original System:** Arrival process \( X^{(O)}(k) = X(k) \) and the service process \( Y^{(O)}(k) = Y(k) \) for every \( k \geq 0 \). The initial queue state \( S^{(O)}(1) \) is assumed to be such that \( S^{(O)}_d(1) = 0 \) for every \( d > 0 \) and \( S^{(O)}_0(1) = X^{(O)}(0) \).

**Reversed System:** Arrival process \( X^{(R)}(k) = Y(k + D) \) for every \( k \geq 0 \); and \( Y^{(R)}(k) = Y(k - 1) \) for every \( k \geq 1 \). The initial queue state \( S^{(R)}_d(1) \) is assumed to be such that \( S^{(R)}_d(1) = Y(D - d) \) for every \( d \geq 0 \).

**Remark:** We note that the final performance does not depend on the initial state, and hence these states are chosen without loss of generality. Moreover, in the reversed system, shifted version of the service process acts as the arrival process. This, however has no impact as original service process and the shifted service process are stochastically identical. We need these specifications as we invoke sample path arguments to prove the required.

Before proving the main result of the section, we prove some supporting results. To this end, let's first define \( W^{(u)}_d(k) \) for \( u \in \{O, R\} \) which we refer to as waste function. The waste function captures the capacity lost on account of not having enough queued packets in \( k \)th frame. Mathematically, the function is defined as follows:

\[
W^{(u)}_d(k) = \left[ Y^{(u)}(k) - \sum_{j=d}^{D-1} S^{(u)}_j(k) \right]^+. \tag{7}
\]

Note the \( W^{(u)}_0(k) \) actually denotes the unutilized capacity in frame \( k \). It is worthwhile to note that the queue states can be updated directly using waste function as shown by the following relation.

\[
S^{(u)}_d(k + 1) = \left[ S^{(u)}_d(k) - W^{(u)}_d(k) \right]^+. \tag{8}
\]

Now let us define

\[
W^{(u)}(k) = \lim_{K \to \infty} \frac{1}{K} \sum_{k=1}^{K} W^{(u)}_0(k) \tag{9}
\]

to be average capacity wastage in the system. Since, original and reversed queue processes are positive recurrent (see Theorem 1), \( W^{(u)} \) can be expressed in terms of the stationary distribution. Specifically,

\[
W^{(u)} = \sum_{s \in (A^{(u)})^D} \pi(s) \mathbb{E} \left[ Y^{(u)}(k) - \sum_{j=0}^{D-1} s_j \right]^+ \text{ w.p. 1.}
\]

Now, we state our first supporting result.

**Lemma 1:** The loss and capacity waste functions satisfy \( L^{(u)} - W^{(u)} = \mathbb{E}[X^{(u)}(1)] - \mathbb{E}[Y^{(u)}(1)] \) w.p. 1 for \( u \in \{O, R\} \).

The above result states that the difference between the average loss and the average capacity waste is exactly equal to the difference between arrival rate and the service rate. Note that for a system in which \( \mathbb{E}[X^{(u)}(1)] \) packets arrive and \( \mathbb{E}[Y^{(u)}(1)] \) packets depart in every frame, \( \mathbb{E}[X^{(u)}(1)] - \mathbb{E}[Y^{(u)}(1)] \) is the difference between loss and capacity waste. Thus, in the long run, randomness (higher moments to be specific) in the arrival and service processes does not play any role in determining the difference between loss and capacity.
waste. Note, however, that the higher moments of arrival and service processes do impact loss and waste. The result says that the impact is the same on both.

In the following result, we establish the relation between the original and the reversed systems.

Lemma 2: Following relations hold w.p. 1: (1) \( L^{(O)} = W^{(R)} \) and (2) \( L^{(R)} = W^{(O)} \).

The result says that the average loss in the original (reversed, resp.) system is equal to the average capacity waste in the reversed (original, resp.) system. This result couples the performance of the original and reversed system. This result along with Lemma 1 yields the following main result of the section.

Theorem 2: Relation \( L^{(O)} - L^{(R)} = \mathbb{E}[X(1)] - \mathbb{E}[Y(1)] \) holds w.p. 1.

With this result, we have shown that the complexity of finding the loss is independent of the number of flows in the system. However, the brute force method still remains computationally infeasible.

C. Reduction of States

As we have seen in the previous section, computing \( L \) has complexity that is exponential in \( D \) for the brute force approach. In this section, we show that with careful modeling the complexity of computing \( L \) can be made polynomial in \( D \). Broadly, the central idea is not to deal with the entire \( D \)-dimensional queue state vector, rather only the last non-zero dimensional queue state. However, the brute force method still remains computationally infeasible.

Transition to states of type (1) will occur if only \( m \) arrivals in frame \( k \) depart, i.e., \( Y(k) = m \). Transitions of type (2) and (3) are similar to those of type (2) and (3) to \( \phi_0 \). Transitions of type (2) and (3) are similar to those before. Hence, we only discuss the transition of type (1). Note that the transitions of type (1) can occur in two cases: (1a) \( Y(k) \leq m \) and \( X(k - \delta) = m \), and \( Y(k) > m \) and \( Y(k) = m + X(k - \delta) = m \). Finally, let us consider another boundary condition \( \tilde{S}(k) = \phi_0 \). In this case, there are only two types of transitions possible: (1) to \( \phi_0 \) and (2) to \( \phi_1 \). Type (1) transition occur if \( Y(k) \geq X(k - 1) \) and type (2) transitions occur when \( Y(k) = X(k - 1) - m \).

Since \( X_k \)'s and \( Y_k \)'s are i.i.d. and \( \tilde{S}(k) \) does not depend on \( Y(j) \) for \( j > k \) and \( X(j) \)'s those are not revealed in the abridged state yet, we conclude that the abridged queue process is a time homogeneous Markov chain.

Theorem 3: The abridged queue state process has a single non-empty aperiodic closed communicating class. Moreover, this process has a unique stationary distribution \( \tilde{\pi} \) such that \( \tilde{\pi}(\delta) > 0 \) if \( \delta \) belongs to the closed communicating class, and \( \tilde{\pi}(\delta) \) is zero otherwise.

The proof of this theorem is similar to that of Theorem 1. We omit the proof on account of space constraints.

Now, notice that the loss can be computed as follows:

\[
L = \sum_{\delta \in \{(D-1,m)\} : m \in A, m > 0} \tilde{\pi}(\delta) \mathbb{E}[m - Y(k)]^{+}.
\]

Above relation holds because the loss can occur only in states of type \( (D-1,m) \). Thus, it suffices to study the abridged queue process in order to determine loss. Abridged queue process provides significant state space reduction, but as they say “there are no free lunches”. Thus, question we need to address is: what penalty we incur on account of reduction
in the size of state space, and does this penalty cancel the impact of state space reduction? Notice that the computation of an entry in the transition probability matrix (TPM) for the abridged process is computationally more challenging than that in the TPM of the original process. Here the main bottleneck is computation of the distribution of the sum of packet arrival random variable. Fortunately, using dynamic programming approach, these computations can be done with \(O(A^2D)\) complexity. Thus, the overall complexity of finding TPM of the abridged process is \(O(A^2D^2)\). Moreover, to obtain the stationary distribution of the abridged process, we need to solve \(O(AD)\) linear equations. Thus, the overall complexity is \(O(A^3D^3)\) that is polynomial in \(A\) and \(D\).

IV. ADMISSION CONTROL ALGORITHM

In this section, our aim is to describe how the feasibility analysis translates into admission control decisions. Unlike previous sections, here we consider the dynamic scenario where flows arrive and depart. On arrival of a new flow, the admission control procedure is invoked. The aim of admission control procedure is to check whether QoS requirement of the new flow can be met without violating the QoS requirements of the existing flows. If the check is successful, i.e., the new flow can be given its required QoS without compromising QoS guarantees of the existing flows, then the new flow is admitted in the system. Next, we explain how the check is performed.

We assume that each new flow seeking admission specifies four things: (1) Delay requirement (in slots), (2) loss tolerance, (3) arrival process at slot level, and (4) service process. First two parameters are the QoS parameters of the flow. Arrival process can be specified in the parametrized form. For example, the flow can specify that the arrival process is geometric (Poisson, alternatively) with rate \(u\) per slot. Another way of specifying the arrival process is by specifying the application corresponding to the flow. For example, a flow can specify that the application is telephony with/without voice suppression; or it can specify that the application is real-time video with resolution \(x\) and codec \(y\). For most of the common real-time applications arrival process is well understood. Arrival process is specified at the slot level and not at the frame level as considered in the previous section. This is because “frame” is an artifact of TDMA scheduling. Thus, arriving flow may not have any information about the frame duration. Moreover, the frame duration varies with arrival and departure of the flows and as a result arrival process at a frame level also changes with flow arrival and departure. In addition, if the arrival process at the slot level is known, then the arrival process at the frame level can be obtained. Note that the first three parameters are application related and hence known to the flow. On the contrary, the last parameter (the service process) is not known. For this we assume that before seeking admission, the receiver corresponding to the flow listens to the channel for predefined time. During this period, its aim is to exploit broadcast property of the wireless channel by overhearing the transmissions. Overheard transmissions are used to learn the channel statistics. The flow can then use these learned statistics to describe the service process. For example, it the fading process is found to be Raleigh with average SNR \(\beta\), then the receiver can find the service process depending on the choice of modulation and coding schemes available to it. We note that estimation in relatively short finite time is not guaranteed to be exact, but depending on the confidence required, channel estimation period can be chosen. Without knowing channel process, arbitrating admission control to guarantee the required QoS is infeasible. Admission control algorithm uses these parameters to decide whether the new flow should be admitted.

Pseudo code for the admission control algorithm is described in Algorithm 1. We briefly describe the algorithm. The key idea is to consider a modified system assuming that the flow seeking admission is actually admitted. When a new flow is admitted, the frame length has to increase by one (line 4). But when the frame length changes, delay requirements measured in number of frames and the arrival process at the frame level also change for existing flows (lines 6 and 7). Now, only thing remains is to check the feasibility of the updated system (line 8). If the system is feasible, then the new flow can be guaranteed its required QoS without violating QoS guarantees of the existing flows. Thus, in this case the new flow can be admitted in the system (line 10). The newly admitted flow is assigned a slot, and the system state is updated to reflect the change (lines 11 and 12). Finally, if the system is not feasible, then the flow is not admitted (line 14).

Note that the analysis done in Section III is employed to check the feasibility of the system.

Algorithm 1 Admission Control Algorithm

1: Let \(\mathcal{N} = \{1, \ldots, N\}\) be the set of existing flows in the system
2: Let new flow, say \(N + 1\), seeks admission
3: Let \((\tilde{D}_n, \tau_n, \tilde{X}_n, \tilde{Y}_n)\) be the parameters specified by flow \(n \in \mathcal{N} \cup \{N + 1\}\)
4: Update frame size to \(N + 1\) from \(N\)
5: for each \(n \in \mathcal{N} \cup \{N + 1\}\) do
6: Update delay \(D_n = \lfloor \tilde{D}_n/(N + 1) \rfloor\)
7: Update frame level arrival process \(\tilde{X}_n\) using slot level arrival process \(\tilde{X}_n\)
8: Check feasibility of the system \(\mathcal{N} \cup \{N + 1\}\) with updated parameter
9: if System is feasible then
10: Admit flow \(N + 1\)
11: Update slot allocation
12: \(\mathcal{N} \leftarrow \mathcal{N} \cup \{N + 1\}\)
13: else
14: Do not admit flow \(N + 1\)

V. DISCUSSION AND GENERALIZATIONS

In this section, we discuss advantages and limitations of our approach. We also elaborate on some possible generalizations. Note that for TDMA, time is divided into frames, and the network operates at the frame level. However, the packet arrivals and channel conditions evolve at slot level. So, we
In the analysis, we have made some simplifying assumptions. Our aim here is to relax them and verify whether the algorithm retains its desirable traits. Some important assumptions were: (1) packet delay are multiple of the frame length, (2) sessions last for ever (recall that the admission control is decided based on the steady state behavior), and (3) the channel is known perfectly.

In Fig. 1, we relax assumption (1) and evaluate the difference between the calculated packet drops and the actual packet drops for a typical user. Our simulation setup is as follows: We consider a static scenario with $N$ identical flows in the system. For each flow, the delay requirement is 10 slots, arrival process is Bernoulli with rate 0.5 packets per slot and the service process is also Bernoulli with mean 0.8. We plot the expected packet drop that is calculated with our algorithm and the average packet drop that is observed in the simulations for user 1 as $N$ varies. To calculate expected packet drop in analysis, we take the delay requirement for each user to be $\max\{1, \lceil 10/N \rceil \}$ frames. Fig. 1 shows that the two values are close, but begin to diverge as $N$ becomes large.

Now, we relax assumptions (1) and (2), and study the system performance. Specifically, we consider a dynamic heterogeneous scenario where users arrive and depart. The user arrival process is considered to be Poisson process with the expected inter-arrival time $1/\lambda$ slots. The admitted flows stay in the system for exponential duration with mean $1/\mu$. For each arriving user $n$: (i) the packet arrival process is assumed to Bernoulli with rate $\alpha_n$ chosen uniformly from interval $[0.4, 0.8]$, (ii) the channel process is Bernoulli with the expected service rate chosen uniformly from interval $[0.5, 1]$, (iii) the delay $D_n$ is chosen uniformly in the interval $[5, 20]$ slots, and (iv) the tolerance $\tau_n$ is chosen uniformly from the interval $[0.1, 0.4]$. The simulation is performed until 2000 users depart from the system. Fig. 2(a) shows that the blocking probability increases sharply with the user arrival rate $\lambda$ as expected. Here, we keep $\mu$ fixed. In Fig. 2(b) we study the impact of the finite holding time for each user. Here, we fix $\lambda$ and vary $\mu$. Specifically, we find the fraction of users for which tolerance requirement is not met by the time they depart. Recall that the decision on admitting a user is based on its performance as time goes to infinity. We observe from Fig. 2(b) that the fraction of unsatisfied users is below 5% for all values of $\mu$ that we consider.

Finally, we evaluate the system performance when the channel state is not correctly estimated. To simulate, say $x\%$, estimation error, we consider that the estimate $\hat{\alpha}_n$ given to the admission control algorithm is uniformly drawn from the interval $\left[\alpha_n - \frac{x}{100}, \alpha_n + \frac{x}{100}\right]$. Admission control is performed with respect to values $\hat{\alpha}_n$ of the existing and the incoming flow. Fig. 2(c) shows the percentage increment in the fraction of unsatisfied flows on account of the erroneous channel estimation. Here, the percentage increment is computed as $\frac{\mu(x) - \mu(0)}{\mu(0)} \times 100$, where $\mu(x)$ is the fraction of unsatisfied users when there is $x\%$ error in channel estimation.

The simulation results, though not prove, increase our confidence that the proposed admission control algorithm performs reasonably well even when the analytical assumptions are inherently have mismatch between the time scale at which network operates and that at which system evolves. Handling this mismatch is the key challenge for the analysis. In our approach, we made approximations that allowed us to model system evolution at the frame level. In particular, we assumed that the exact packet arrival times do not matter, i.e., we assumed that no matter where packet has arrived inside the frame, it will be considered to have arrived at the frame boundary. Next, we converted delay in slots to delay in frames. Both these approximations reduce the precision at which system can potentially operate. Hence, our admission control algorithm is a little conservative. Nonetheless, we believe that without these simplifications, analysis will remain intractable.

Also, we have assumed that the packet arrival and service processes to be i.i.d. across slots. This is unrealistic for many real-time applications and channel models. However, our analysis can be easily extended to Markovian arrivals and Markovian channel models. For this, the state space, which currently contains queue state alone, has to be augmented to contain states of arrival and channel processes. The computational complexity of admission control algorithm increases on account of increase in the size of state space, but analysis holds. We have not discussed this case as the additional insight we obtain is not worth ensuing notational clutter.

Some generalizations are also possible regarding the medium access used. For example, our analysis holds for the randomized flow scheduling, i.e., a slot is given to a flow with certain probability. This flow scheduling probability can depend on the channel states, but has to be queue state oblivious. Note that this scheduling allows us to consider slot level dynamics and also accounts for opportunism, but it also increases the variance of service process as compared to TDMA. Thus, its delay performance need not be better always. We consider TDMA as many deployed systems use it for access. Next, we discuss the related work.

VI. SIMULATION RESULTS

In this section, our aim is to evaluate the performance of the proposed admission control algorithm using simulations. In the analysis, we have made some simplifying assumptions.
relaxed.

VII. RELATED WORK

The problem of admission control for the delay constrained real-time traffic is explored extensively in case of wire-line networks. Unfortunately, this analysis does not extend to wireless networks on account of location dependent and time varying fluctuations in channel capacity. Initial effort in optimizing the performance of wireless networks was focused towards designing scheduling policies that guarantee finite expected delay to each flow, whenever feasible. Such policies are called “throughput optimal”. To this end, Tassiulas et al. have shown that max-weight policies, aka back-pressure policies, are throughput optimal \[5, 6\]. This policy does not provide any guarantees on the delay performance of the system, except that the expected delay is finite. Recently, Neely has shown that the expected packet delay under max-weight policy remains unchanged irrespective of the network size \[9\]. Efforts have been made to design alternate throughput optimal policies that provide some sort of guarantees on the delay performance as well. In \[1, 17\], authors have proposed throughput optimal scheduling scheme that minimizes the maximum expected delay. In \[7\], authors have proposed a throughput optimal scheduling scheme that provides delay differentiation to the users. In \[8, 10\], authors have proposed Markov decision process based approaches to minimize the expected packet delay in the network. Note that all this work provides guarantees on the expected packet delay, and they do not consider hard delay requirement of real-time applications.

There has also been a considerable amount of effort in generalizing approaches proposed in the case of wire-line networks to the case of wireless networks. For example, in \[16\], the authors have shown that the Earliest Due Date (EDD) scheduling scheme, which is shown to be optimal for wire-line networks, does not perform well in wireless networks. Here, authors have also proposed some heuristic scheduling scheme that performs better than EDD. In \[14, 15\], authors have adapted fair scheduling schemes to wireless case. But, issue of admission control has not been answered. In \[11\], authors have explored the problem of admission control for general scheduling scheme in wireless environment. However the quantification of the resource requirement for providing the required delay is done under the worst case assumptions and hence results in very conservative estimates leading to a large blocking probability. In \[18\], authors have developed heuristic admission control criteria for static priority scheduling with the assumption that the channel is the same for all users.

Recently, Hue et al. have made significant progress in developing an admission control algorithm along with an optimal scheduling scheme to provide hard delay guarantees to the real-time users \[2\]–\[4\]. This work is the closest to ours. Here, authors assume that time is divided into frames. In each frame the channel for all users remains constant, but it may change at frame boundaries. Also, at most one packet can arrive at the beginning of a frame and has deadline in the same frame. Thus, the system can have at most one backlogged packet at any point in time. This assumption provides considerable simplification in a sense that the queues need not be kept. Authors use this simplification to propose optimal scheduling scheme. In contrast, we analyze the fixed TDMA scheme. The problem of designing optimal scheduling and admission control in a system that allows for packet delays to be more than channel coherence time remains open. In fact, in such cases, to the best of our knowledge, we are the first ones to provide scalable admission control algorithm.

VIII. CONCLUSION

We have explored the problem of designing admission control algorithm for TDMA based wireless system. We have shown that the key component of admission control is to simply find the stationary distribution of the queue state process. Unfortunately, the brute force techniques for finding the stationary distribution are computationally infeasible on account of having exponential complexity. Hence we have devised approaches like reversing the queue state process and state space compression that made computational complexity \(O(A^3D^3)\) which is polynomial in A & D. Using the stationary distribution computation, we can determine whether the system is feasible. Using feasibility analysis as a key stone, we developed admission control algorithm. Simulation results show that the admission control algorithm performs reasonably even when the assumptions made for analysis are relaxed.
Appendix

of Lemma [7] We first prove the following relation for every $k \geq 0$.

$$
D-1 \sum_{j=0}^{D-1} S_j^{(u)}(k+1) + L^{(u)}(k) = \sum_{j=0}^{D-1} S_j^{(u)}(k) + X^{(u)}(k) + W^{(u)}(k) - Y^{(u)}(k). \quad (12)
$$

The above relation holds on every sample path. Fix $k$. To see [12] for the given $k$, let us consider two cases: (1) $Y^{(u)}(k) \geq \sum_{j=0}^{K-1} S_j^{(u)}(k)$ and (2) $Y^{(u)}(k) < \sum_{j=0}^{D-1} S_j^{(u)}(k)$.

Consider case (1). Here, all the packets present in frame $k$ are transmitted. Thus, in $(k+1)th$ frame, only packets in the queue are those arrive afresh. This implies that

$$
\sum_{j=0}^{D-1} S_j^{(u)}(k+1) = X^{(u)}(k),
$$

$$
L^{(u)}(k) = 0,
$$

$$
W^{(u)}(k) = Y(k) - \sum_{j=0}^{D-1} S_j^{(u)}(k).
$$

Putting everything together, we get [12].

Now, consider case (2). Here, we have two sub-cases: (2a) $Y^{(u)}(k) > S_{D-1}^{(u)}(k)$ and (2b) $Y^{(u)}(k) \leq S_{D-1}^{(u)}(k)$.

Now, in both these cases, capacity is fully utilized, i.e., $W^{(u)}(k) = 0$. Now, consider case (2a). Here, note that all packets at the position $D-1$ of the queue state in frame $k$ are served, i.e., in frame $k$ there is no packet loss. Also, the total number of packets in the queue at the beginning of frame $k+1$ are the ones left unserved in slot $k$ plus the new arrivals.

Thus, we get the following:

$$
\sum_{j=0}^{D-1} S_j^{(u)}(k+1) = \sum_{j=0}^{D-1} S_j^{(u)}(k) - Y^{(u)}(k) + X^{(u)}(k),
$$

$$
W^{(u)}(k) = L^{(u)}(k) = 0.
$$

Again, putting everything together yields [12].

Now, lets consider case (2b). Here, not all packets in the position $D-1$ of the queue state in frame $k$ are served. Nonetheless, all packets in this position leave the system, some on account of transmission and other on account of drop. Thus, we get the following:

$$
\sum_{j=0}^{D-1} S_j^{(u)}(k+1) = \sum_{j=0}^{D-2} S_j^{(u)}(k) + X^{(u)}(k),
$$

$$
L^{(u)}(k) = S_{D-1}^{(u)}(k) - Y^{(u)}(k),
$$

$$
W^{(u)}(k) = 0.
$$

Again, putting everything together yields [12].

Since $k$ is arbitrary [12] follows for every $k \geq 0$. Now, starting from $k = K$ for some $K$ and using [12] recursively till $k = 0$, we get

$$
\sum_{j=0}^{K} S_j^{(u)}(K+1) - \sum_{j=0}^{K} S_j^{(u)}(K) = \sum_{k=0}^{K} \left(X^{(u)}(k) + W^{(u)}(k) - Y^{(u)}(k) - L^{(u)}(k)\right). \quad (13)
$$

Dividing both sides of (13) by $(K+1)$ and then taking limit as $K \to \infty$ we get:

$$
L^{(u)} - W^{(u)} = E[X^{(u)}(1)] - E[Y^{(u)}(1)] \text{ w.p. 1.}
$$
Note that the result follows as queue state process is a positive recurrent Markov chain, and arrival and service processes are i.i.d.

of Lemma 2 For notational convenience, let us define $S_D^{(u)}(k) = L^{(u)}(k-1)$ for $k \geq 1$. Now, we prove the following two statements for every $k$.

\begin{align*}
W_{D-d}^{(R)}(k) = S_D^{(R)}(k + d) & \quad \text{for } 0 < d \leq D, \quad (14) \\
W_{D-d}^{(O)}(k + D - d) = S_D^{(R)}(k + 1) & \quad \text{for } 0 < d \leq D. \quad (15)
\end{align*}

We proceed by induction. We first show that (14) and (15) hold for $k = 1$. Consider (14) and note that

$$
W_{D-d}^{(R)}(1) = \left[ Y^{(R)}(1) - \sum_{j=D-d}^{D-1} S_j^{(R)}(1) \right]^{+} \quad \text{(see (7))} \\
= \left[ X(0) - \sum_{j=D-d}^{D-1} Y(D - j) \right]^{+} \quad .
$$

Now, consider $S_D^{(O)}(1 + d)$. Note that $S_D^{(O)}(1) = X(0)$. Now, in slot $(1 + d)$ the $d^{th}$ position of the queue state vector under original system will be occupied by the unserved packets among $X(0)$. Thus, $S_D^{(O)}(1 + d) = [X(0) - \sum_{j=D-d}^{D-1} Y(D - j)]^{+}$. Hence, (14) follows for $k = 1$.

Now, we prove (15) for $k = 1$. Note that

$$
W_{D-d}^{(O)}(1 + D - d) = \left[ Y(1 + D - d) - \sum_{j=D-d}^{D-1} S_j^{(O)}(1 + D - d) \right]^{+} \quad \text{(see (7))} \\
= \left[ Y(1 + D - d) - S_D^{(O)}(1 + D - d) \right]^{+} \quad \text{(initial state ensures $S_D^{(O)}(1 + D - d) = 0$ for $j > D - d$)} \\
= \left[ Y(1 + D - d) - W_{D-d}^{(R)}(1) \right]^{+} \quad \text{(see (14))} \\
= \left[ S_D^{(R)}(1) - W_{D-d}^{(R)}(1) \right]^{+} \quad \text{(see initial state of (R) process)} \\
= S_D^{(O)}(2) \quad \text{(see (8)).}
$$

Hence, (15) follows for $k = 1$.

For induction hypothesis, let assume that (14) and (15) be true for $k < k_0$. We now prove these statements for $k = k_0$.

**Part 1:** To prove: $W_{D-d}^{(R)}(k_0) = S_D^{(O)}(k_0 + d)$ for $0 < d \leq D$; for $d = 1$ we get

$$
S_D^{(O)}(k_0 + 1) = [S_0^{(O)}(k_0) - W_1^{(O)}(k_0)]^{+} \quad \text{(see (8))} \\
= [X^{(O)}(k_0 + 1) - S_D^{(R)}(k_0 + 1)]^{+} \quad \text{(using induction hypothesis on (15) for $k = k_0 - 1$)} \\
= [Y^{(R)}(k_0) - \sum_{j=D-1}^{D-1} S_j^{(R)}(k_0)]^{+} \\
= W_{D-1}^{(R)}(k_0) \quad \text{(see (7)).}
$$

Thus, the statement holds for $d = 1$. Now, let it be true for all $d < d_0$. We consider the case $d = d_0$.

$$
S_D^{(O)}(k_0 + d_0) = [S_D^{(O)}(k_0 + d_0 - 1) - W_{d_0}^{(O)}(k_0 + d_0 - 1)]^{+} \quad \text{(see (8))} \\
= [W_{D-d_0}^{(R)}(k_0) - S_D^{(R)}(k_0)]^{+} \quad \text{(using induction hypothesis on $d$ and on (15) for $k = k_0 - 1$)} \\
= [Y^{(R)}(k_0) - \sum_{j=D-d_0}^{D-1} S_j^{(R)}(k_0)]^{+} \\
= W_{D-d_0}^{(R)}(k_0).
$$

Hence Part 1 proved.

**Part 2:** To prove $W_{D-d}^{(O)}(k_0 + D - d) = S_D^{(R)}(k_0 + 1)$ for $0 < d \leq D$. Consider $d = 1$. Here, we get

$$
S_1^{(R)}(k_0 + 1) = [S_0^{(R)}(k_0) - W_1^{(R)}(k_0)]^{+} \quad \text{(see (8))} \\
= [X^{(R)}(k_0 - 1) - S_D^{(O)}(k_0 + D - 1)]^{+} \quad \text{(see (14) for $k = k_0$)} \\
= [Y^{(R)}(k_0 + D - 1) - \sum_{j=D-1}^{D-1} S_j^{(O)}(k_0 + D - 1)]^{+} \\
= W_{D-1}^{(O)}(k_0 + D - 1)
$$

Now let it be true for all $d < d_0$. Consider $d = d_0$. Let

$$
\alpha = [Y^{(R)}(k_0 + D - d_0) - \sum_{j=D-d_0}^{D-1} S_j^{(O)}(k_0 + D - d_0)]^{+}.
$$

$$
S_1^{(R)}(k_0 + 1) = [S_0^{(R)}(k_0) - W_1^{(R)}(k_0)]^{+} \\
= [W_{D-d_0}^{(O)}(k_0 + D - d_0) - S_D^{(O)}(k_0 + D - d_0)]^{+} \\
= [\alpha - S_0^{(O)}(k_0 + D - d_0)]^{+} \\
= [Y^{(R)}(k_0 + D - d_0) - \sum_{j=D-d_0}^{D-1} S_j^{(O)}(k_0 + D - d_0)]^{+} \\
= W_{D-d_0}^{(O)}(k_0 + D - d_0).
$$

This proves part 2. Now, the result follows by summing both sides of (14) and (15) for $k = 1$ to $K$, dividing them by $K$, and then taking limit $K \to \infty$.

\[ \square \]