Characterization and Optimization of Delay Guarantees for Real-time Multimedia Traffic Flows in IEEE 802.11 WLANs

Yan Gao, Member, IEEE, Chee Wei Tan, Senior Member, IEEE, Shan Lin, Ying Huang, Member, IEEE, Zheng Zeng, Member, IEEE, and P. R. Kumar, Fellow, IEEE

Abstract—Due to the rapid growth of real-time applications and the ubiquity of IEEE 802.11 MAC as a layer-2 protocol for wireless local area networks (WLANs), it becomes increasingly important to support delay-based quality of service (QoS) in such WLANs. In this paper, we develop a simple but accurate analytical model for predicting the queuing delay of real-time multimedia traffic flows in non-homogeneous random access based WLANs. This leads to tractable analysis for meeting queuing delay specifications of a number of flows. In particular, we address the feasibility problem of whether the mean delays required by a set of User Datagram Protocol (UDP) flows supporting real-time multimedia traffic can be guaranteed in WLANs. Based on the model and feasibility analysis, we further develop an optimization technique to minimize the delays for the traffic flows. Moreover, we present a decentralized algorithm and report its implementation and present extensive simulation and experimental trace-based results to demonstrate the accuracy of our model and the performance of the algorithms.

Index Terms—Delays in 802.11 WiFi, Optimization, Delay Guarantees, Quality of Service, WLAN.

I. INTRODUCTION

The recent rapid growth of real-time applications has led to a strong need to provide delay-based quality of service (QoS) for mobile computers and portable devices in wireless local area networks (WLANs). This has to be supported over the IEEE 802.11 since it has gained widespread popularity and become the de facto WLAN standard. However, the mechanisms employed in the IEEE 802.11 MAC, namely random access and the distributed coordination function (DCF), render it substantially more difficult to ensure delay guarantees because of channel contention and the random back-off mechanism used. Therefore, as the first task confronting researchers in this field, it is necessary to characterize the delays in such networks. Second, it is important to devise solutions that provide the required delay performance. We address both issues in this paper.

Existing studies on the performance analysis of the IEEE 802.11 MAC have focused on its throughput capacity in networks with saturated traffic, and not delay; see Bianchi [1], Cali, Conti, and Gregori [2]. In [3], a M/G/1 queue is analyzed under network saturation. Models for unsaturated homogeneous networks have also been reported in the literature. For example, Medepalli and Tobagi [4] present a unified model for multi-hop networks that approximates each queue by an independent M/M/1 queue. However, this approximation may not be adequate for a delay analysis in WLANs. Tikoo and Sikdar [5] present a G/G/1 queueing model for delay analysis in homogeneous networks. Their focus is on the performance analysis of the standard IEEE 802.11 DCF. There has also been various studies using M/G/1 queue to analyze the performance in WLANs, see for example [6], [7]. Also, most work consider centralized polling techniques based on the point coordination function (PCF). For example, Coutras, Gupta and Shroff [8] analyze the performance of PCF in support of voice services. However, they do not address the fact that both best-effort traffic and real-time traffic can coexist in WLANs, and IEEE 802.11 DCF is the de facto setting used in most WLANs.

Providing delay-based QoS requires WLAN networks to support service differentiation under non-homogeneous traffic dynamics. The networks should also reallocate limited resources from the over-provisioned flows to the under-provisioned flows. IEEE 802.11e has been proposed to enhance the original standard to support QoS. However, IEEE 802.11e classifies flows only by their applications (e.g., voice, video, etc.) and provides the same service to flows that fall in the same class. Moreover, it only differentiates priority among flows, but does not actually provide delay guarantees, i.e., it is only best effort service. A non-homogeneous and adaptive WLAN is preferred over one that operates in a fixed homogeneous manner. However, an accurate model of non-homogeneous flows in random access WLANs, especially with respect to their delay characterization, is still elusive.

In this paper, we develop a simple but sufficiently accurate analytical model based on an M/G/1 queueing model for non-homogeneous unsaturated IEEE 802.11 networks. We characterize the mean service time with respect to the contention window and the probability that the queue is nonempty. Both the probability of being non-empty and the access delay can...
be jointly obtained by solving a coupled system of nonlinear equations through a fixed point iteration with a carefully chosen initial point so that it converges. Moreover, we show that, in random access networks, the second moment of the access delay is determined only by its first moment if the packet size is sufficiently large. This approximation simplifies the formula of the queueing delay. Using this analysis, we determine whether the network can provide the mean delay guarantees required by the QoS flows.

The contributions of the paper are summarized as follows:

1) We derive a simple but accurate model for mean queueing delay in non-homogeneous IEEE 802.11 MAC based networks. We use it to determine the feasibility of using a random-access based WLAN to serve a set of real-time flows with mean delay requirements.
2) We provide a characterization of the average delay and access rate, and propose a fixed point algorithm to compute them. A linear system approximation is derived to complement the analysis.
3) We provide an algorithm to minimize the mean delays for a set of User Datagram Protocol (UDP) traffic flows while meeting mean delay requirements.
4) We design a distributed algorithm based on the proposed queueing model and show that the distributed algorithm performs as well as the centralized algorithm.
5) We validate our algorithm to provide performance guarantees through extensive NS-2 simulations and trace-based experiments.

We motivate the non-homogeneous IEEE 802.11 flows problem in Section 2. In Section 3, we characterize the mean service time and the mean queueing delay. In Section 4, we study fixed point iterations related to the queueing model. We show how to optimize the mean delay performance in Section 5. Simulation results and numerical results are compared in Section 6. Furthermore, we derive a distributed algorithm based on the analytical model and address several design issues in Section 7. Results of trace-based experiments are given in Section 8. Finally, we conclude the paper in Section 9.

II. PROBLEM STATEMENT

A. Non-homogeneous IEEE 802.11 networks

In IEEE 802.11 DCF networks, each node with a packet to transmit selects randomly a back-off timer counter $BC$ from $[1, CW - 1]$, where $CW$ denotes the contention window. If the channel is sensed idle, these nodes decrement their timers until one of them expires. Then that node attempts to access the channel while the remaining nodes pause their timers. The decrementing mechanism resumes when the channel is idle once again. If more than one node attempts to transmit in the same slot, a collision occurs. A collided transmission is tried again until a retransmission limit is reached before it is discarded. In the standard IEEE 802.11 network, the contention window of each node is set to be the same. This homogeneous or uniform setting works well for best-effort traffic where fairness is taken into account. However, the increasing need to support the QoS requirements of different flows requires networks to have the ability to provide service differentiation especially delay guarantees to real-time flows. The standard IEEE 802.11e has been proposed to enhance the original standard to support QoS. However, IEEE 802.11e categorizes flows only by their applications (e.g., voice, video, etc.) and provides the same service to all the flows that fall in the same class. It is important to note that IEEE 802.11e only differentiates priorities to flows but does not actually provide delay guarantees. Our goal is to provide mean delay guarantees in random access WLANs. Towards this end, a non-homogeneous and adaptive wireless network is preferable.

We consider wireless networks with nodes that are capable of changing their backoff parameters. In other words, we tune only $CW$. Thus, our scheme is compatible with the IEEE 802.11 standard. In other words, we analytically show that $CW$ alone can effectively be used for resource allocation and performance differentiation.

B. Soft Deadline Guarantees

In this paper, we focus on the soft deadline related to the mean delay of a flow. Soft-deadline guarantees are important for several real-time applications such as voice over Internet Protocol (VoIP), online games and Internet protocol television (IPTV), since they often require a fixed bit rate but are sensitive to average delays. Consider a WLAN where $N$ nodes are active and each has a QoS flow to the access point (AP). These flows differ in rate and mean delay requirements. Assume that for each node $i$ the packet arrival is a Poisson process and the inter-arrival time is exponentially distributed with mean $1/\lambda_i$. Let $\lambda = [\lambda_1, \lambda_2, ..., \lambda_N]$ be the arrival rate vector. Additionally, each flow has a soft deadline $D_i$ to meet. The average queueing delay of the packet for flow $i$ is required to be less than $D_i$. Let $D = [D_1, D_2, ..., D_N]$ denote the target delay vector.

Now, suppose both $\lambda$ and $D$ are given. Does there exist an assignment of contention windows $CW = [CW_1, CW_2, ..., CW_N]$ for the $N$ flows such that all the deadlines of all the nodes are met? This question is important to admission control that decides whether it is feasible to accommodate a flow in the network without hurting the performance of existing higher priority flows. Furthermore, if it is feasible, a natural question is how to achieve all these mean delays, i.e., how to assign $CW_i$ to each node $i$. We answer these two key questions in this paper.

III. ANALYTICAL MODEL OF NON-HOMOGENEOUS IEEE 802.11 NETWORK

A. Media Access Delay

We begin by first analytically addressing the access delays in a non-homogeneous WLAN. We do not consider the exponentially increasing back-off mechanism implemented in the IEEE 802.11 protocol because our scheme explicitly determines the contention window for each flow so as to meet the delay requirements for all the flows. Imposing an additional $CW$ adjustment mechanism, e.g., exponential back-off algorithm, may complicate the analysis and is left for future work. In the literature, the schemes proposed in [7], [7] disable the exponential backoff mechanism, and directly adjust
the contention window. However, their goal is to maximize throughput, while ours is to provide mean delay guarantees for nonhomogeneous flows.

We will consider an “access rate” for node $i$ that is equal to $2/CW_i$. This corresponds to IEEE 802.11 DCF with BC chosen randomly from $[1, CW-1]$ [7], [8], [9], [10]. Since our flows are not “saturated”, the queues may be empty, and in this case they do not transmit. Let $NE_i$ denote the event that “queue is not empty” and $E$ denote the event that “queue is empty.” Then, the unconditional channel access (CA) probability of node $i$ is


It is obvious that $P[CA|E]$ is equal to zero because a node has no packet to transmit when the queue is empty. In particular, we approximate $P[CA|NE]$ by $\frac{2}{CW_i}$. Note that this is an approximation only when the backoff mechanism is enabled to choose uniformly within $[1, CW-1]$, but otherwise this is exact (and not an approximation) if transmission is attempted after an exponentially distributed interval. Denoting $p_i = 2/CW_i$, and $\rho_i$ as the probability that the queue is not empty, we have

$$P[CA] = \frac{2\rho_i}{CW_i} = \rho_i p_i.$$ (1)

Let $p$ be the vector $[p_1, p_2, ..., p_N]$, noting that $0 \leq p_i \leq 1$. Likewise, let $\rho := [\rho_1, \rho_2, ..., \rho_N]$. Next, we need to compute the probability $P_i$ that the channel is idle when node $i$ has a packet to send, the probability $P_{S_i}$ that the channel is successfully carrying a packet of node $i$, and the probability $P_{I_i}$ that node $i$ sees the channel as busy though $i$ itself is not transmitting a successful packet. Note that $P_i + P_{S_i} + P_{I_i} = 1$ for all $i$. All these quantities are in fact functions of the vector $p$.

We now focus on the dependence of $\rho$ on $p$. Note that node $i$ competes for the channel access only when it has a packet to transmit. Thus, node $i$ finds the channel idle in a time slot if it itself does not attempt and no other node attempts at the beginning of this slot. Hence,

$$P_i^i = (1 - p_i) \prod_{j \neq i} (1 - \rho_j p_j).$$ (2)

Node $i$ successfully transmits a packet if only it attempts and no other node attempts in the same slot. This probability is

$$P_i^S = p_i \prod_{j \neq i} (1 - \rho_j p_j).$$ (3)

Otherwise, node $i$ sees the channel occupied by other activities, consisting of either successful transmissions of other nodes or collided transmissions. Since the collided transmissions consist of both the transmissions involving node $i$ as well as those not involving node $i$, we have

$$P_i^O = 1 - P_i^i - P_i^S = 1 - \prod_{j \neq i} (1 - \rho_j p_j).$$ (4)

Let us define the service time $x$ of a packet as the time from the instant the packet reaches the head of the queue in the node till the instant it successfully departs from the queue. This service time includes two parts, the channel contention delay and the packet transmission time. For simplicity, we assume that all the packets are of the same size and all nodes employ the same bit rate for transmissions. Thus, they have the same packet transmission airtime, denoted by $T$. More formally, in the IEEE 802.11 network, the packet transmission airtime is

$$T := DIFS + PACKET + SIFS + ACK,$$ (5)

where DIFS denotes the duration of the distributed interframe space, PACKET denotes the transmission time of a data packet, SIFS denotes the duration of the short interframe space, and ACK denotes the transmission time of an acknowledgement. There are two access modes used in IEEE 802.11 DCF, namely the basic access mode and the request to send/clear to send (RTS/CTS) access mode. The RTS/CTS access mode is usually disabled in practice due to its large overhead. Thus, in this paper, we focus only on the basic access mode. In the basic access mode, a collision is detected when a node does not receive an ACK within an ACK-timeout. This ACK-timeout is defined to be the time to transmit an ACK frame plus SIFS. Thus, we assume that the airtime spent on a collided transmission is the same in duration as that of a successful transmission.

We denote a slot-time duration by $\tau$. Let $t_k$ denote the time instant when the $k$-th idle slot begins, i.e., the instant that the channel is idle at the beginning of the corresponding slot. There are two possible events following this instant: a) the channel continues to be idle for a duration of $\tau$ until the next idle slot begins; b) at least one of the nodes attempts to transmit in this slot that results in a $T$ time units channel-busy period. We assume the intervals $S_i(k) = t_{k+1} - t_k$ are independent and identically distributed random variables and refer to these intervals as virtual slots.

Assume that the time interval from the time the packet reaches the head of the queue $i$ to the time it starts to depart from the queue consists of $K_i$ virtual slots, where $K_i$ is a random variable independent of $S_i$. Its distribution is geometric and thus is given by

$$P[K_i = n] = (1 - P_S^i)^n P_S^i,$$ (6)

for $n = 0, 1, 2, ...$

It follows that

$$E[K_i] = \frac{1 - P_S^i}{P_S^i}.$$ (7)

For node $i$, its service time is therefore

$$x_i = \sum_{k=1}^{K_i} S_i(k) + T,$$ (8)

where the $S_i(k)$ are Bernoulli random variables that are either equal to $\tau$ if the channel is idle or equal to $T$ if a transmission of a node other than $i$ occurs:

$$S_i(k) = \begin{cases} \tau & \text{with probability } \frac{P_S^i}{1 - P_S^i} \\ T & \text{with probability } \frac{P_i^O}{1 - P_S^i} \end{cases}.$$ (9)

$^1$In the sequel we will employ a fixed point analysis since $p$ itself depends on $\rho$. 


Then, we have
\[ E[S_i] = \frac{P_i^1 \tau + P_i^0 T}{1 - P_S^i}. \]  
(10)

It is easy to see that both \( E[S_i] < \infty \) and \( E[K_i] < \infty \). From the independence of \( S_i \) and \( K_i \), we can apply Wald’s equation \(^7\) to obtain
\[ X_i := E[x_i] = E[K_i] E[S_i] + T. \]  
(11)

Substituting (10) into (11) yields
\[ X_i = \frac{P_i^1 \tau + P_i^0 T}{P_S^i} + T. \]  
(12)

Hence, (12) expresses the interesting relationship between the expected service time and the access rate in carrier sense multiple access based (CSMA-based) random access wireless networks.

Since the network is unsaturated, we need to determine the probability \( p_i \) that the queue is non-empty. Since each node is an M/G/1 queue, we have
\[ p_i = \lambda_i X_i. \]  
(13)

Substituting (12) into (13), we thus obtain \( N \) equations with \( N \) unknowns \( [x_1, x_2, \cdots, x_N] \). Solving this \( N \) dimensional vector fixed point problem gives us the service times for non-homogeneous flows in the WLAN.

In summary, we have obtained the fundamental relationship that allows us to compute the mean service times for non-homogeneous flows in random access WLAN: Given the contention windows \( CW_i \), the mean service times are given by (12), where \( P_i^1 \), \( P_S^i \) and \( P_O^i \) are given by (13), with \( p_i \) defined by (13). In addition, the quantities \( p_i \)'s in (12) satisfy (13).

B. Queueing Delay

In the previous section, we have derived an analytical model that can be used to compute the service time if the access rates of all nodes are given via their contention windows. Since many real-time applications such as online games, VoIP and IPTV impose requirements on jitter and delay, we next study how the non-homogeneous contention window settings and the non-homogeneous throughput requirements jointly affect the average queueing delay.

Define the queueing delay of a packet to be the time from the instant that the packet arrives at the queue to the instant that the packet successfully departs from the queue. The average queue size of the \( M/G/1 \) queue is given by (7):
\[ E[Q_i] = \lambda_i X_i + \frac{\lambda_i^2 E[x_i^2]}{2(1 - \lambda_i X_i)}. \]  
(14)

where \( Q_i \) denotes the queue size and \( E[x_i^2] \) is the second moment of the service time. Using Little’s law, the average queueing delay \( Y_i \) is
\[ Y_i = \frac{E[Q_i]}{\lambda_i} = X_i + \frac{\lambda_i E[x_i^2]}{2(1 - \lambda_i X_i)}. \]  
(15)

To determine the average queueing delay (14), we need to also determine the second moment of the service times. In (10), we have characterized the service time \( x \) as a sequence of the virtual slots \( S \) plus a transmission airtime \( T \). Next, taking squares on both sides of (10), we have
\[ x_i^2 = \left( \sum_{k=1}^{K_i} S_i[k] + T \right)^2 \]
\[ = \sum_{k=1}^{K_i} S_i^2[k] + 2 \sum_{k=2}^{K_i} \sum_{l=1}^{K_i} S_i[k]S_i[l] + 2T \sum_{k=1}^{K_i} S_i[k] + T^2. \]  
(16)

Applying Wald’s equation, we get
(17)

Using the distribution of \( S_i \), in (17), we compute
\[ E[S_i^2] = \frac{\tau^2 P^1_i + T^2 P_O^i}{1 - P_S^i}. \]  
(18)

To determine \( E[K_i^2 - K_i] \), we first obtain the moment generating function of \( K_i \) from (17) as follows:
\[ M_{K_i}(B) = \sum_{n=0}^{\infty} B^n (1 - P_S^i)^n P_S^i = \frac{P_S^i}{1 - (1 - P_S^i)B}. \]  
(19)

It is easy to verify that
\[ \frac{d^2 M_{K_i}(B)}{dB^2} |_{B=1} = \sum_{n=0}^{\infty} n(n-1)B^{n-2} (1 - P_S^i)^n P_S^i |_{B=1} \]
\[ = \sum_{n=0}^{\infty} n(n-1)(1 - P_S^i)^n P_S^i = E[K_i^2 - K_i]. \]  
(20)

Hence, from both (17) and (18), we get
\[ E[K_i^2 - K_i] = \frac{2(1 - P_S^i)^2}{(P_S^i)^2}. \]  
(21)

Finally, substituting (17), (18), (19) and (20) into (17), we get the second moment of the service time for node \( i \) as follows:
\[ E[x_i^2] = \frac{\tau^2 P^1_i + T^2 P_O^i}{P_S^i} + 2\left( \frac{\tau P^1_i + T P_O^i}{P_S^i} \right)^2 \]
\[ + 2T \frac{\tau^2 P^1_i + T P_O^i}{P_S^i} + T^2. \]  
(22)

After substituting (14) and (15) into (15), we obtain the average queueing delay with respect to vector \( p \).

Queueing delays as a function of contention windows \( Y_i(p) \): In summary, for a non-homogeneous flows in the random access WLAN with contention windows \( CW_i \) and packet transmission time \( T \), the mean queueing delay is given by (14), where \( E[x_i] \) is given by (14), \( E[x_i^2] \) is given by (15), \( P_i^1 \), \( P_S^i \) and \( P_O^i \) are given by (13), and \( \rho = [\rho_1, \rho_2, \cdots, \rho_N] \) is a fixed point of (13).
C. Queueing Delay and Service Time for Small Slot Times

We simplify our above analysis by using some practical assumptions. Substituting (27) into (28), we have
\[
E[x_i^2] = 2(X_i - T)^2 + 2T(X_i - T) + T^2 + 2P_iT^2 + P_iT^2(X_i - T).
\] (23)

Since \( \lim_{\tau \to 0} E[x_i^2] = (2X_i - T)^2 + 2T(X_i - T) + T^2 + T(X_i - T) \), if we assume that the packet transmission airtime \( T \) is sufficiently large compared to the slot-time \( \tau \), then we get a simplified formula for \( E[x_i^2] \) given by
\[
E[x_i^2] = (2X_i - T)X_i.
\] (24)

Note that (24) implies that the second moment of \( x \) can be determined only by its first moment. We believe that this intriguing property is an inherent characteristic of the random access mechanism in WLAN. Therefore, the average delay reduces to
\[
Y_i = \frac{(2 - \lambda_i T)X_i}{2(1 - \lambda_i X_i)}.
\] (25)

Now, (25) is equivalent to
\[
X_i = \frac{2Y_i}{2 - \lambda_i T + 2\lambda_i Y_i}.
\] (26)

Interestingly, (26) illustrates an important characteristic under the small slot-time assumption: The queueing delay in a random access network is determined only by the service time.

IV. ANALYSIS OF FIXED-POINT PROBLEMS

A. Nonlinear Characterization of Delay and Access Rate

We have determined that when the transmission airtime \( T \) is sufficiently large compared to the slot time \( \tau \), the queueing delay \( Y \) can in fact be determined by \( X \). Recall that the delay \( X_i \) can be characterized by (26). Based on (26), we derive a set of fixed point equations given by
\[
p_iX_i + (1 - p_i)(T - \tau) = \frac{T}{\prod_{j \neq i}(1 - \lambda_j X_j p_j)} \quad \forall i.
\] (27)

There are two perspectives to viewing (27): analysis or design. The analysis part (or the performance analysis problem analysis below) consists of determining the delay, given the access rates. The design part (or the access rate assignment problem below) consists of determining the access rates for the flows so as to meet all the delay constraints. Both parts can be solved using the fixed point equation in (27).

Performance Evaluation (PE): We fix the access rate \( p \), and evaluate the service time \( X \). For node \( i \), its delay can be written as
\[
X_i = I^{PE}_i(X) := \frac{T}{p_i \prod_{j \neq i}(1 - \lambda_j X_j p_j)} - (1 - p_i)(T - \tau).
\] (28)

We denote by \( X^* \) as a fixed point of (27), assuming that it exists. Thus, we consider the following fixed point iteration to solve (27):
\[
X(k + 1) = I^{PE}(X(k)).
\] (29)

Access Rate Assignment (ARA): From a protocol designer’s viewpoint, it can be interesting to compute the access rate assignment such that all the flows meet their required delays. In other word, we adapt the access rate \( p \) such that all the delays \( X \) are fulfilled. For node \( i \), the access rate is given by
\[
p_i = I^{ARA}_i(p) := \frac{T}{(X_i - T + \tau) \prod_{j \neq i}(1 - \lambda_j X_j p_j)} - \frac{T - \tau}{X_i - T + \tau}.
\] (30)

We denote by \( p^* \) a fixed point of (27), assuming that it exists. Thus, we consider the following fixed point iteration to solve (27):
\[
p(k + 1) = I^{ARA}(p(k)).
\] (31)

B. Linear System Approximation

Note that \( \lambda_i X_i p_i < 1 \) for all \( i \) if the queueing system is stable. Now, using the fact that \( 1/(1 - z) \geq 1 + z \) for nonnegative \( z < 1 \), we can lower bound the RHS of (27) by an affine expression. Thus, we have
\[
p_iX_i + (1 - p_i)(T - \tau) \geq (1 + \sum_{j \neq i}\lambda_j X_j p_j) \quad \forall i.
\] (32)

Furthermore, we can approximate the inequality in (27) by an equality if we assume small\(^2 \) \( p_i X_i \) for all \( i \), and apply a Taylor expansion for the RHS of (27). This leads us to consider the following fixed point equation:
\[
p_iX_i - \sum_{j \neq i}\lambda_j T p_j X_j = p_i T + (1 - p_i)\tau \quad \forall i.
\] (33)

Now, we can consider two different linear fixed point equations in the form of (27): One in terms of \( X \) for the performance evaluation assuming a fixed \( p \), and the other in terms of \( p \) for the access rate assignment assuming a fixed \( X \).

The following result shows that each of these two linear fixed point iterations has a unique solution.

Theorem 4.1: Suppose that \( p_1, p_2, \ldots, \) and \( p_N \) are given, (27) has a unique solution for \( [X_1, X_2, \ldots, X_N] \).

Proof: Let \( y \) denote the vector \([p_1X_1, p_2X_2, \ldots, p_NX_N]^T\) and \( b \) denote the vector \([p_1T + (1 - p_1)\tau, p_2T + (1 - p_2)\tau, \ldots, p_N T + (1 - p_N)\tau]^T\). Then, we represent (27) by
\[
y = Fy + b,
\] (34)

where \( F \) is an irreducible nonnegative matrix with entries:
\[
F_{ij} = \begin{cases} 0, & \text{if } l = j \vspace{1mm} \\
\lambda_i T, & \text{if } l \neq j. \end{cases}
\] (35)

We now apply nonnegative matrix theory to characterize the solution to (27). Let \( \Lambda_A \) denotes the spectral radius of a nonnegative matrix \( A \). By the Collatz-Wielandt theorem (see, e.g., [1]),
\[
\Lambda_F \leq \max_i \sum_{j \neq i} \lambda_j T < \sum_{i} \lambda_i T < 1,
\] (36)

\(^2\)A sufficiently small condition on \( X_i p_i \) in order for (27) to hold is \( \lambda_i X_i p_i < \frac{1}{n - 1} \forall i. \) We omit the detail here for the limit of space.
where the last inequality follows from the necessary condition that the $M/G/1$ system is stable only if the workload is strictly less than 1, i.e., $\sum_i \lambda_i T < 1$. Next, we state the following result from \cite{myref}.

**Lemma 4.2:** A necessary and sufficient condition for a solution $z \geq 0$, $z \neq 0$ to exist to the equations $(I - A)z = c$, for any $c \geq 0$, $c \neq 0$ is that $\Lambda z < 1$. In this case there is only one solution $z$, which is strictly positive, i.e., $z \neq 0$ and $z \geq 0$, and given by $z = (I - A)^{-1}c$. Applying Lemma \ref{lem:stability} to \eqref{eq:stability}, this implies that $(I - F)^{-1}b$ has a unique positive solution. This proves the theorem.

**Lemma 4.3:** Assume that $X$ is given. If $\tilde{p}$ is the fixed point of \eqref{eq:fixed_point}, and $p^*$ is the fixed point of \eqref{eq:fixed_point}, then we have, component wise, $\tilde{p} \leq p^*$ and $\tilde{p} \neq p^*$.

**Proof:** Suppose the following holds:

$$\tilde{p}_i X_i + (1 - \tilde{p}_i)(T - \tau) = T(1 + \sum_{j \neq i} \lambda_j X_j \tilde{p}_j) \forall i,$$

$$p^*_i X_i + (1 - p^*_i)(T - \tau) = \prod_{j \neq i}(1 - \lambda_j X_j p^*_j) \forall i.$$

For each $i$, we subtract \eqref{eq:stability} from \eqref{eq:stability} to obtain

$$(X_i - T + \tau)(p^*_i - \tilde{p}_i) = T \prod_{j \neq i}(1 - \lambda_j X_j p^*_j) - T(1 + \sum_{j \neq i} \lambda_j X_j \tilde{p}_j) > T(1 + \sum_{j \neq i} \lambda_j X_j p^*_j) - T(1 + \sum_{j \neq i} \lambda_j X_j \tilde{p}_j) = \sum_{j \neq i} \lambda_j T X_j (p^*_j - \tilde{p}_j).$$

Let $u$ denote the vector $[X_i(p^*_i - \tilde{p}_i)]$. Now, \eqref{eq:stability} for all $i$ can be written in a compact form as

$$(I - C)u = \nu > 0,$$

where $\nu$ denotes some positive vector (with the positive slack of the inequality \eqref{eq:stability} as its $i$th entry), and $C$ is a positive matrix with entries

$$C_{ij} = \begin{cases} (T - \tau)/X_i, & \text{if } i = j, \\ \lambda_i T, & \text{if } i \neq j. \end{cases}$$

Since $C$ is a positive matrix, using the Perron-Frobenius theorem, $\Lambda C$ is strictly positive. Now, $\Lambda C$ satisfies

$$\Lambda C \leq \max_{i}(\sum_{j \neq i} \lambda_j T + \frac{T}{X_i}),$$

where inequality (a) is due to the Collatz-Wielandt theorem, inequality (b) is due to the service rate $1/X_i$ being strictly larger than the arrival rate $\lambda_i$ (as \eqref{eq:stability} enforces this constraint), inequality (c) is straightforward, and inequality (d) is due to the necessary stability condition for a $M/G/1$ queue. Applying Lemma \ref{lem:stability} to \eqref{eq:stability}, $u$ is strictly positive. This proves the lemma.

We point out that the linear approximation analysis only holds under certain regime, and is useful for tractable analysis. There are however limitations on the approximation. For example, increasing the number of nodes will increase service time. As future work, it is important to study other (nonlinear) approximations under which \eqref{eq:stability} can be further simplified.

**C. Convergence**

We establish below the convergence result related to the algorithms for the ARA and the PE.

**Theorem 4.4:** If $p^*$ exists, then starting from $\tilde{p}$, the ARA algorithm produces a monotone increasing sequence of vectors $p(k)$ that converges to a fixed point.

**Proof:** By Lemma \ref{lem:stability}, we know $\tilde{p} < p^*$. Note that $I_{ARA}(p)$ is a monotone non-decreasing function. Thus, starting from $\tilde{p}$, we have $p(1) = I_{ARA}(\tilde{p}) < I_{ARA}(p^*)$ and $p(1) = I_{ARA}(\tilde{p}) \geq p$. Suppose $p(1) \leq p(2) \leq \cdots \leq p(n) \leq p^*$. Then, monotonicity implies that

$$p^* = I_{ARA}(p^*) \geq I_{ARA}(p(n)) = p(n + 1) \geq I_{ARA}(p(n - 1)) = p(n).$$

That is, $p^* \geq p(n + 1) \geq p(n)$. Hence, the sequence $p(n)$ is nondecreasing and bounded above by $p^*$. Thus, $p(n)$ converges.

One can use a similar approach to prove the convergence of the PE algorithm \eqref{eq:fixed_point}, and the proof is omitted.

**V. APPLICATIONS**

**A. Feasibility Problem**

To demonstrate the utility of the proposed model, we use the above algorithm to address the following ARA question: In an IEEE 802.11 WLAN, suppose that the arrival rates $\lambda$ and the required delays $D = [D_1, D_2, ..., D_N]^T$ are given. Does there exist a set of access rates $[p_1, p_2, ..., p_N]^T$ such that the resulting delay for each node $i$ is guaranteed to be smaller than $D_i$? We refer to this problem as the average delay feasibility problem.

Formally, we say that $\{(\lambda_1, D_1), (\lambda_2, D_2), \ldots, (\lambda_N, D_N)\}$ is feasible if there exist $[p_1, p_2, \ldots, p_N]^T$ such that

$$Y_i(p) \leq D_i \forall i.$$

We argue that if there exists a $p$ such that the equality holds in the above (i.e., $Y_i \equiv D_i$, for $i = 1, 2, \ldots, N$), then $\{(\lambda_1, D_1), (\lambda_2, D_2), \ldots, (\lambda_N, D_N)\}$ is feasible. We implicitly assume in the following that if a vector of delays is feasible, then any set of component-wise larger set of delays is also feasible. Equivalently, we have the expected channel access delay as

$$X_i = \frac{2D_i}{2 - \lambda_i T + 2\lambda_i D_i},$$

where we substitute $Y_i = D_i$. Note that both $D_i$ and $\lambda_i$ are inputs, and hence $X_i$ can be completely determined by them. Consequently, $p_i = \lambda_i X_i$ is also determined. Substituting $p_i$ into \eqref{eq:fixed_point} yields a fixed point problem to yield the contention windows $p$. One can use the ARA algorithm proposed in the previous section to solve this fixed point problem. After obtaining the fixed point $p^*$, if $0 < p^*_i < 1$ for all $i$, then we can deduce that the flows are feasible, and a feasible contention window $CW_i$ is then given by the maximum integer that is smaller than $2/p^*_i$. Otherwise, we conclude that the flows are not feasible because if the fixed point had existed, the ARA algorithm is guaranteed to converge.
B. Minimization of Delay

We now consider a scheme for the delay minimization problem that is solved by a central controller, e.g., an access point in a WLAN, which collects the QoS requirements \{\{(\lambda_1, D_1), (\lambda_2, D_2), \ldots, (\lambda_N, D_N)\}\} from all the nodes. The WLAN access point first solves the feasibility problem in Section ??, and then optimizes the delay performance. Assume that the \(i\)th node has a cost function \(f_i(Y_i)\) that is differentiable, non-decreasing and strictly convex. Now, from (??), \(Y_i\) is convex in \(X_i\). We substitute (??) into \(f_i(Y_i)\) to yield a convex function in \(X_i\), which we denote as \(f_i(X_i)\). Consider the following optimization problem:

\[
\min \sum_{i=1}^{N} f_i(X_i) \tag{46}
\]

s.t. \(0 \leq X_i \leq \bar{X}_i := \frac{2D_i}{2 - \lambda_i} \forall i, \tag{47}
\]
\(X_i = f_i^\text{PE}(X(p)) \forall i, \tag{48}\)
\(0 < p_i \leq 1 \forall i. \tag{49}\)

In the above, the constraint (??) guarantees that the average delay is less than the required delay. However, the constraint (??) that relates \(p\) to \(X\) is nonconvex, thus (??) is generally hard to solve. To obtain numerical solution, we use the barrier method (interior-point method) in optimization theory [??] to compute a local optimal solution. Since the barrier method yields a solution in the interior of the feasible set, this can be useful to find a feasible solution that meets the delay requirements, i.e., the delay constraints (??) as they are satisfied at all the intermediate solution iterates.

From (??) and (??), we consider the following barrier function:

\[
B_i(p) := \frac{1}{X_i - X_i(p)} + \frac{1}{1 - p_i} + \frac{1}{p_i} \forall i. \tag{50}\]

Note that this barrier function increases to +\(\infty\) when any \(i\)th constraint approaches its boundary. Let \(\epsilon_i\) be a positive weight associated with \(B_i(p)\) for all \(i\). Consider the optimization problem:

\[
\max J(p) := \sum_{i=1}^{N} f_i(X_i(p)) + \sum_{i} \epsilon_i B_i(p). \tag{51}\]

The solution to (??) yields a suboptimal solution that is feasible to (??). We present the following algorithm based on the gradient method to solve (??) [??].

Gradient Algorithm

1) Obtain an initial point \(p^0\) by solving the feasibility problem in Section ??.

2) For a fixed \(p^k\) (solution of the feasibility problem), run the PE algorithm till convergence to obtain \(X^k\).

3) For fixed \(p^k\) and \(X^k\), obtain \(\frac{\partial J(p_k)}{\partial p_i}\) from (??) and (??).

4) Update \(p\) by

\[
p_i^{k+1} = p_i^k - \beta_i \frac{\partial J(p_k)}{\partial p_i} \forall i,
\]

where \(\beta_i\) is a positive diminishing stepsize [??].

5) Repeat from Step 2 until convergence to a small tolerance.

Due to the nonconvexity in (??), this gradient algorithm yields a solution that in general is not the global optimal solution of (??). However, by exploiting the linear system approximation in Section ??, we can obtain a relaxation to (??) that yields a lower bound to the (unknown) global optimal value of (??):

\[
\min \sum_{i=1}^{N} f_i(X_i) \tag{52}
\]

s.t. \(0 \leq X_i \leq \bar{X}_i \forall i, \tag{53}\)
\(X_i \geq ((I - F)^{-1}b(p))_i/p_i \forall i, \tag{54}\)
\(0 < p_i \leq 1 \forall i. \tag{55}\)

where \((Ax)_i\) denotes the \(i\)th element of the vector \(Ax\), and \(b(p) = [(T - \tau)p_1 + \tau, (T - \tau)p_2 + \tau, \ldots, (T - \tau)p_N + \tau]^T\). Note that (??) is obtained by relaxing the constraint (??) in (??) using (??). Now, (??) is still nonconvex. However, by making the logarithmic change of variable \(\tilde{p}_i := \log p_i\) for all \(i\), we obtain the following convex problem that is equivalent to solving (??):

\[
\min \sum_{i=1}^{N} f_i(X_i) \tag{56}
\]

s.t. \(0 \leq X_i \leq \bar{X}_i \forall i, \tag{57}\)
\(X_i \geq ((I - F)^{-1}b(e^{\tilde{p}}))_i/\tilde{p}_i \forall i, \tag{58}\)
\(\tilde{p}_i \leq 0 \forall i, \tag{59}\)

where \(e^{\tilde{p}} = [e^{p_1}, e^{p_2}, \ldots, e^{p_n}]^T\). In practice, it is observed through our simulations in the following section that (??) often yields an optimal value that is only slightly smaller than the objective value evaluated at the suboptimal solution obtained by the gradient algorithm. This demonstrates that the gradient algorithm can compute a near-optimal solution.

VI. Simulation Results

A. Simulation Setup

We perform our simulation using the NS-2 network simulator (version ns2.31) [??]. Table ?? summarizes the system parameters used in the simulation. As we do not employ the exponential back-off mechanism, after obtaining a \(CW\) from the analysis, we just set \(CW_{\min} = CW_{\max} = CW\) to disable the exponential back-off. These values of \(CW_{\min}\) and \(CW_{\max}\) shown in Table ?? are referred to as the default baseline settings for comparison purpose. No other parameters are changed in any simulation. Collocated topologies are created in which all nodes can carrier-sense one another. Each sender node is attached to a Poisson traffic generation agent in which packet inter-arrival times can be customized. The interface queues at each node use a Droptail policy and the queue size is set at 5000 packets. Each simulation runs for 400 seconds in simulation time. Two metrics, namely the service time and the queuing delays, are measured for each flow. For the service time, the time interval from the instant that the packet arrives at the head of the queue to the instant that the packet successfully
access delays increase. Even though $CW_2$ and $CW_3$ are not changed, their corresponding access delays decrease because $CW_1$ is increased. In the second scenario, $CW_1$ is changed, while maintaining the fixed ratio $CW_1 : CW_2 : CW_3 = 1 : 2 : 3$. The results are shown in Figure 2. One can observe that except for the nonlinear part where $CW_1$ is very small, the channel access delays agree with the theoretical values. The nonlinear part of the curves is due to the fact that the collision probability becomes larger when every node has a small $CW$ for contention resolution.

2) Service time and queueing delays under unsaturated conditions: For the unsaturated conditions, three scenarios are examined. The first scenario is intended to study how traffic arrival rates affect the service time. The inter-arrival time of flow 3 is varied, while keeping the other two links’ arrival rates fixed. The fixed packet inter-arrival times are $\lambda_1 = 0.03$ and $\lambda_2 = 0.005$. For the contention window, $CW_1 = CW_2 = CW_3 = 32$ is set. Figure 2 plots the results. The theoretical results are obtained by solving (2) using the PE algorithm.

In the second scenario, it is examined how $CW$ affects the channel access delays. The traffic arrival rates, and $CW_1$ and $CW_2$ are fixed. Only $CW_3$ is changed from 12 to 44. The results are shown in Figure 2. One can observe that as $CW_3$ is increased, the delays of flow 3 increase. One can also observe that the delays of flow 1 and flow 2 drop.

The third scenario is used to demonstrate how service time changes in response to the number of nodes. Each link has the same traffic rate and the same $CW$. In particular, $\lambda_3 = 0.03$ and $CW_i = 32$ for all $i$. Only the number of links is changed. Figure 2 plots the results. As expected, the access delays increase as the number of links grows.

3) Queueing delays: We repeat the same three scenarios for the queueing delays. One can observe similar trends in these figures to their counterparts for the service time. From the three scenarios, we conclude that the theoretical results are accurately verified by the simulation results.
is not only reflected in the trend but also in the quantitative values.

C. Performance Evaluation

In the following simulations, two case studies are examined to demonstrate the applicability of the model and to evaluate the performance of the proposed algorithm. Each point in the figures is a time-average of the queueing delay over every 50 simulation seconds.

1) Feasibility: In the first case, when the capacity is insufficient, the default 802.11 setting cannot meet the delay guarantees of all the QoS flows. But, with the proposed scheme, one can find an appropriate setting in which all the delay requirements are met. The three required delays are assumed to be 0.02 seconds. Note that this delay requirement is realistic according to [2]. The data rates of the UDP traffic flows are fixed as follows: $\frac{1}{\lambda_1} = 0.025, \frac{1}{\lambda_2} = 0.004$, and $\frac{1}{\lambda_3} = 0.003$. The ARA algorithm is run to obtain a set of feasible contention windows: $CW_1 = 66, CW_2 = 23, CW_3 = 18$. Figure ?? plots the simulation results. One can see that the baseline default IEEE 802.11 can guarantee the mean delay requirements only for flows 1 and 2, whereas the mean delay of flow 3 is much larger than the allowed mean delay. However, the WLAN can actually guarantee all the mean delays if the contention windows are appropriately adjusted. In fact, one sees that the mean delays of all flows are met when the WLAN uses the contention windows that are computed using our algorithms.

2) Minimizing delays: In this case study, the performance of the scheme that minimizes the average delays for UDP traffic flows, while preserving their delay guarantees, is evaluated. We consider the following cost function

$$\hat{f}_i(Y_i) = \frac{Y_i^2}{\lambda_i}. \quad (60)$$

The UDP traffic flows have fixed arrival rates $\frac{1}{\lambda_1} = 0.04, \frac{1}{\lambda_2} = 0.004$, and $\frac{1}{\lambda_3} = 0.003$. The delay requirements are still fixed at 0.02 seconds. Compared to the input of the first case, one can observe that the network capacity is sufficient for this input. Thus, there should be room for the flows to improve their performance (i.e., queueing delays in this case). Using the gradient algorithm presented in Section ??, the optimal $CW$s are computed to be $CW_1 = 19, CW_2 = 23$, and $CW_3 = 19$. The comparisons are plotted in Figure ???. We observe that when configured with the $CW$s computed by our algorithms, the WLAN does achieve the optimized mean delays and does at the same time provide a certain level of fairness. In contrast, in the baseline default IEEE 802.11, flow 3 suffers from bad delay performance and experiences serious unfairness.

3) Scaling up with nodes: In this case, we have studied how the number of nodes affects the performance of the proposed algorithm in the network. The delay requirements are still fixed at 0.02s. The data rates of the UDP traffic ($\frac{1}{X_i}$) are randomly generated between 0.05 and 0.1. Each simulation runs for 100 seconds. For a fixed number of nodes, we have repeated each simulation for 5 times and have compared different approaches using the averages. If the number of the contention window is not feasible (smaller than 1), CWmin is used for simplicity.

Figure ?? shows the percentage of the nodes that meet the delay requirement (0.02s) and Figure ?? shows the average queueing delay. From both figures, we observe that the proposed algorithm demonstrates the ability to lower the average queueing delay as comparing to the traditional 802.11 exponential backoff mechanism. Furthermore, we observe that the proposed algorithm can allow the nodes to meet the delay requirements as the number of nodes scaling up. Therefore, the proposed algorithm is applicable even when the number of nodes in the network increases beyond tens or twenties of nodes (a typical number in existing WiFi network).

VII. DESIGN OF DISTRIBUTED ALGORITHM

Our results in the previous sections address the feasibility question. In practice, it can be interesting to find the appropriate contention window allocations in a distributed manner. To this end, we derive a distributed algorithm that adapts the service rates to meet the demands through the contention window adjustment.
A. Derivation of the Distributed Algorithm

We have defined three probabilities $P^i_I$, $P^i_S$ and $P^i_O$ in (??) – (??). Let us assume that there is an observer who is monitoring the channel activities. The observer sees one of two possible states in a virtual slot: either the channel is idle during the virtual slot or the channel is busy due to the other nodes' transmission. Denote $\hat{P}^i_j$ as the probability that the observer sees an idle slot. Then, we have

$$\hat{P}^i_j = \prod_{j \neq i} (1 - \rho_j p_j).$$  

Denote the number of consecutive idle slots between any two transmissions as $n_i$. Since $n_i$ follows a geometric distribution, the probability that $n_i = k$ is

$$P[n_i = k] = (\hat{P}^i_j)^k (1 - \hat{P}^i_j).$$  

The expectation of $n_i$ is

$$E[n_i] = \frac{\hat{P}^i_j}{1 - \hat{P}^i_j}. \quad (63)$$

Rearranging (??), we have

$$\hat{P}^i_j = \frac{E[n_i]}{1 + E[n_i]}. \quad (64)$$

Now, substituting (??) into (??) yields

$$X_i = \frac{(1 - p_i)\hat{P}^i_j \tau + (1 - \hat{P}^i_j)T}{p_i \hat{P}^i_j} + T. \quad (65)$$

Replacing $\hat{P}^i_j$ in (??) by (??), we have

$$p_i(X_i - T + \tau) = \frac{T}{E[n_i]} + \tau. \quad (66)$$

An unbiased estimation of $E[n_i]$ is the average of its samples. In particular, the observer in node $i$ counts $n_i[k]$ and estimates
Fig. 11. Percentage of nodes satisfying the delay requirements (0.02s) using the gradient algorithm with respect to the number of nodes in the network.

Fig. 12. Average queueing delay using the gradient algorithm with respect to the number of nodes in the network.

Fig. 13. Transition of channel states.

B. Implementation Issues

The distributed algorithm has been implemented and tested in the ns-2 simulator. First, we introduce a counting mechanism to count \( \bar{n}_i \), the number of idle slots between two consecutive nodes’ transmissions. Then, the contention window can be adapted by (72) based on \( \bar{n}_i \).

\begin{equation}
E[n_i] \text{ by}
E[n_i] = \bar{n}_i = \frac{\sum_{k=1}^{K} n_i[k]}{K}.
\end{equation}

This suggests that one can use (72) to design a distributed algorithm because all the variables, \( p_i \), \( X_i \) and \( \bar{n}_i \), are locally available at node \( i \).

Rearranging (72) and substituting \( CW_i = \frac{2}{p_i} \) yields

\begin{equation}
X_i = \frac{1}{2} \left( \frac{T}{\bar{n}_i} + \tau \right) CW_i + T - \tau.
\end{equation}

If we consider \( CW_i \) as the variable, (72) defines a mapping from \( CW_i \) to \( X_i \). Assume that the target access delay is \( X_i^* \). In this regard, we consider the following optimization problem:

\begin{equation}
\min \sum_i (X_i^* - X_i(CW_i))^2
\end{equation}

s.t.

\begin{equation}
CW_i \geq 2 \; \forall i.
\end{equation}

The target access delay is met when \( X_i^* \) is equal to \( X_i(CW_i) \) for all \( i \). As the objective is a quadratic function of \( CW_i \), the problem is convex in the feasible region of \( CW_i > 2 \). Therefore, a gradient algorithm can readily be used to solve (72).

The derivative of the objective function in (72) with respect to \( CW_i \) is

\begin{equation}
\frac{d}{dCW_i} \left( X_i^* - X_i(CW_i) \right)^2 = -(X_i^* - X_i(CW_i)) \left( \frac{T}{\bar{n}_i} + \tau \right).
\end{equation}

Thus, we derive a distributed algorithm from:

\begin{equation}
CW_i(t + 1) = \max \{ CW_i(t) + \alpha (X_i^* - X_i(CW_i(t))) \left( \frac{T}{\bar{n}_i} + \tau \right), \ 2 \},
\end{equation}

where \( \alpha \) is an appropriately chosen stepsize [7].

In summary, \( CW_i \) is gradually driven to the optimal point such that the difference between \( X_i \) and \( X_i^* \) is minimized. In fact, as we adopt the gradient descent method to solve the distributed optimization problem, we can derive the convergence result using the property of gradient descent method.

1) Idle Counter: The channel state changes when a packet transmission begins or ends. Each node can carrier-sense the change of the channel state. At each instant, when channel state changes, the idle counter is triggered to update the count of the number of idle slots between two consecutive transmissions. To illustrate how the idle counter works, the transition of channel state is plotted in Figure 13. First, a wireless node senses the channel. When it detects that a
transmission has ended, the idle counter is reset and starts to increment by one for every slot time (e.g., 20 µs); when it detects a new transmission, the idle count stops and the counter is used to compute the average number of idle slots \( \bar{n}_i \). Denote the counter at the \( j \)-th count as \( c[j] \); then we can update \( \bar{n} \) by an exponential moving average update:

\[
\bar{n}[j + 1] = (1 - \eta)\bar{n}[j] + \eta c[j],
\]

(73)

where the parameter \( \eta \) is a constant smoothing factor that lies between 0 and 1.

2) Contention Window Adaptation: We assume that the target delays (or deadlines) \( Y \) are provided by the upper layer applications. The delay \( Y \) consists of buffering delay, service time and transmission delay. According to our analysis, the end-to-end delay is determined by the service time \( X \). Recall that we have derived (??) that maps \( Y \) to \( X \). Thus, applying (??) gives the target service time \( X_T \) if \( Y \) is specified. Finally, a sender node can adapt its contention window \( CW \) by using (??), where \( \bar{n} \) is provided by the idle counter.

3) Performance of Distributed Algorithm: We have performed a ns-2 simulation to evaluate the distributed algorithm.

In our simulation, the data rates of the UDP traffic flows are fixed as follows: \( \frac{1}{X_1} = 0.025, \frac{1}{X_2} = 0.004 \) and \( \frac{1}{X_3} = 0.003 \). Assume that the target delays of these flows are 0.02 second. The theoretical contention windows are computed by the centralized algorithm ARA for comparison: \( CW_1 = 66 \), \( CW_2 = 23 \) and \( CW_3 = 18 \). These contention windows are shown by the solid lines in Figure ???. The evolution of contention windows driven by the distributed algorithm is illustrated in the figure. One observes that starting from an initial condition 31, each contention window gradually moves towards its theoretical value. The contention windows of flow 2 and flow 3 converge in a few seconds. The converged values are also observed to be close to the theoretical values. However, the convergence of flow 1 is not as obvious as the other two. In fact, because the contention window of flow 1 is much larger than the other two, it fluctuates within a larger range. The other two contention windows also fluctuate, but within a smaller range and thus appear to remain constant after convergence. The fluctuation is an artifact of the gradient algorithm since in practice we use a constant stepsize \( \alpha \).

In Figure ???, the average delays of the three flows with the distributed algorithm are shown. One can compare the result of the delays with the centralized algorithm, as shown in Figure ???.

We have further studied the performance of the distributed algorithm when the number of nodes in the network increases. The configuration of the experiment is the same as that in Section V for studying the distributed algorithm. The experiment results show that the delay performance of the distributed algorithm is almost the same as that of the centralized algorithm except that the distributed algorithm converges slightly longer (after a few seconds).

VIII. VIDEO-BASED EXPERIMENTAL EVALUATION

In this section, we use a video transmission simulator EvalVid [?] to evaluate the performance of the algorithm when MPEG (moving picture express group) video is transmitted. The configuration of the experiment is the same as that in Section VIII for studying the distributed algorithm. The experiment results show that the delay performance of the distributed algorithm is almost the same as that of the centralized algorithm except that the distributed algorithm converges slightly longer (after a few seconds).

A. Overview of EvalVid

The structure of the EvalVid framework [?] is shown in Figure ?? that illustrates how EvalVid measures the video quality-of-service metrics. For more information, we refer the readers to [?], [?].

B. Video Distortion

We focus on three metrics associated with video distortion for three types of video frames namely I-frames, P-frames and B-frames and their calculation.

- Packet loss: For each type of data, we compute the packet loss rate as follows. Let \( T \) denote the type of data in the packet (one of I, P, B frames), \( PR_T \) denote the number of type \( T \) packets received, and \( PS_T \) denote the number of type \( T \) packets sent, then the packet loss rate \( PL_T \) is
Fig. 16. Overview of the video transmission simulation of EvalVid system.

given by

\[ PL_T = \frac{PR_T}{PS_T} \times 100\%. \quad (74) \]

- Frame loss: The frame loss calculation is introduced because we cannot easily deduce the frame loss rate from packet loss rate. Let \( FR_T \) denote the number of type \( T \) frames received and \( FS_T \) denote the number of type \( T \) frames sent, then the frame loss rate is

\[ FL_T = \frac{FR_T}{FS_T} \times 100\%. \quad (75) \]

- Delay and Jitter: Frames in digital videos have to be displayed at a constant rate. Displaying a frame before or after the defined time leads to a phenomenon called “jerkiness” [?]. This issue is addressed by play-out buffers that can be used to absorb the jitter introduced by network delays. The buffer size is predefined based on the playback time in our experiment.

C. Experiment Design

Our experiments are based on real video data using raw data downloaded from the Video Trace Library [?]. The video traces are provided in two formats, namely YUV QCIF (176 x 144) and YUV CIF (352 x 288). Since the two formats have different resolutions, the amount of data that needs to be transmitted per unit time are different for the same video sequence. For example, the size of 300-frame-long Foreman video in YUV CIF is 44M, while the size of the same video in YUV QCIF is 11M.

For comparison purpose, we conduct two experiments using different algorithms in EvalVid. We use the default IEEE 802.11 DCF for the first experiment while using the proposed algorithm for the second experiment. In both experiments, we use both format video sources. Furthermore, we set MTU to 1000 bytes. The packet arrivals are shown in Figure ?? and ??.

The first experiment is described as follows: we assume two groups of users coexisting in a wireless network. Users in Group 1 download while playing a video in YUV CIF format. Users in Group 2 download while playing the same video in YUV QCIF format. Each group consists of two users. All users adopt the default IEEE 802.11 DCF. Each video provider first encodes the YUV file to obtain the compressed MPEG-4 file. Then, users use MP4.exe (from Evalvid) to record the tracefile for a sender. The tracefiles are linked to a ns-2 UDP agent attached in the sender node. Simulated packets are generated based on the tracefile and then sent to the receiver. The sender records when these packets are sent out and the receiver records when these packets are received. After the simulation, the two records are compared to produce the received compressed MPEG-4 file. Finally, the decoder decodes the file and reconstructs the received video.

The second experiment differs from the first only in using the proposed algorithm. The improvement can be observed from the quality of the received video. The comparison is shown in Figure ??.

Three consecutive frames are selected from the received video in group 1. The upper three frames are taken from the video transported in the default IEEE 802.11 DCF network, and the lower three frames are taken from the video reconfigured by parameters derived from our scheme. Observe that the lower three frames have negligible distortion, because the contention window is appropriately adjusted to guarantee the delay requirement of the video in CIF format. In contrast, we see that the upper three frames are corrupted. Furthermore,
some frames are displayed incorrectly. If a frame gets lost, the decoder fills the gap with the most recent successfully-decoded frame. Thus, we may see an old picture appearing repetitively because the receiver does not receive the correct frames.

We also compare the PSNR (Peak Signal Noise Ratio) results for the two experiments as shown in Figure ???. These experiments demonstrate that our algorithm can improve the quality of service even if the arrival process does not follow a Poisson distribution and that it works well for realistic applications such as live video players.

IX. CONCLUSIONS

We have presented a simple and accurate model to analyze queueing delays in non-homogeneous IEEE 802.11 MAC based WLANs. Our queueing analysis allows us to study the feasibility problem of whether the network can provide the mean delay guarantees required by several non-homogeneous QoS flows. Furthermore, in order to optimize the performance of QoS flows, we have proposed centralized and distributed algorithms to minimize the mean delays, which are measured by a certain cost function, for a set of UDP traffic flows while preserving mean delay guarantees. Extensive NS-2 simulations have been conducted to verify the accuracy of the model and to evaluate the performance of the algorithms.

Yan Gao received his Bachelor degree from University of Science and Technology of China in 2003; and the Master degree in information engineering at the Chinese University of Hong Kong in 2006. He received the Ph.D. degree in Computer Science at the University of Illinois at Urbana-Champaign in 2011. He currently serves as a researcher at Accenture Technology Labs.

Chee Wei Tan (M ’08–SM ‘12) received the M.A. and Ph.D. degrees in electrical engineering from Princeton University, Princeton, NJ, in 2006 and 2008, respectively. He is an Assistant Professor at the City University of Hong Kong. Previously, he was a Postdoctoral Scholar at the California Institute of Technology (Caltech), Pasadena, CA. He was a Visiting Faculty at Qualcomm R&D, San Diego, CA, in 2011. His research interests are in networks, inference in online large data analytics, and optimization theory and its applications. Dr. Tan was the recipient of the 2008 Princeton University Wu Prize for Excellence and was awarded the 2011 IEEE Communications Society AP Outstanding Young Researcher Award. He was a selected participant at the U.S. National Academy of Engineering China-America Frontiers of Engineering Symposium in 2013. Dr. Tan currently serves as an Editor for the IEEE Transactions on Communications and the Chair of the IEEE Information Theory Society Hong Kong Chapter.
Shan Lin received the Bachelor degree in the department of computer science and technology from the University of Science and Technology of China, Hefei, Anhui, China, in 2014. She is pursuing her Ph.D. degree in the department of computer science at City University of Hong Kong. Her research interests center around information theory and optimization theory.

Ying Huang received her Bachelor of Science degree from Department of Electronics Engineering and Computer Science at Peking University, Beijing, China in 2004. She received her Master of Science degree in Department of Computer Science at University of Illinois at Urbana-Champaign in 2008. Her research interests include vehicular networks, proximity social networks, mobile wireless networks, smartphone applications, distributed systems, content sharing and data dissemination, Quality of Service (QoS), topology control, mobility model, and security. She received both IEEE MASS09 Travel Grant Award and CS Department Conference Travel Grant Award from UIUC in 2009. She currently works at Google.

Zheng Zeng received the B.E. degree from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 2005; and the PhD degree from the Department of Computer Science, University of Illinois at Urbana-Champaign in 2011. She is currently working at Apple Inc.

P. R. Kumar obtained his B. Tech. degree in Electrical Engineering (Electronics) from I.I.T. Madras in 1973, and the M.S. and D.Sc. degrees in Systems Science and Mathematics from Washington University, St. Louis, in 1975 and 1977, respectively. From 197784 he was a faculty member in the Department of Mathematics at the University of Maryland Baltimore County. From 19852011 he was a faculty member in the Department of Electrical and Computer Engineering and the Coordinated Science Laboratory at the University of Illinois. Currently he is at Texas A&M University, where he holds the College of Engineering Chair in Computer Engineering. Kumar has worked on problems in game theory, adaptive control, stochastic systems, simulated annealing, neural networks, machine learning, queuing networks, manufacturing systems, scheduling, wafer fabrication plants and information theory. His current research interests are in wireless networks, sensor networks, and networked embedded control systems. His research is currently focused on wireless networks, sensor networks, cyberphysical systems, and the convergence of control, communication and computation. Kumar is a member of the National Academy of Engineering of the USA, and the Academy of Sciences of the Developing World. He was awarded an honorary doctorate by the Swiss Federal Institute of Technology (Eidgenossische Technische Hochschule) in Zurich. He received the IEEE Field Award for Control Systems, the Donald P. Eckman Award of the American Automatic Control Council, the Fred W. Ellersick Prize of the IEEE Communications Society, and the Outstanding Contribution Award of ACM SIGMOBILE. He is a Fellow of IEEE. He was a Guest Chair Professor and Leader of the Guest Chair Professor Group on Wireless Communication and Networking at Tsinghua University, Beijing, China. He is an Honorary Professor at IIT Hyderabad. He was awarded the Distinguished Alumnus Award from IIT Madras, the Alumni Achievement Award from Washington University in St. Louis, and the Daniel C. Drucker Eminent Faculty Award from the College of Engineering at the University of Illinois.