Wireless Video Streaming for Single-Hop Networks

Monchai Lertsutthiwong, Thinh Nguyen, and Alan Fern

Abstract

Limited bandwidth and high packet loss rate pose a serious challenge for video streaming applications over wireless networks. Even when packet loss is not present, the bandwidth fluctuation as a result of an arbitrary number of active flows in an IEEE 802.11 network, can significantly degrade the video quality. This paper aims to enhance the quality of video streaming applications in wireless home networks via a joint optimization of video coding technique, admission control algorithm, and Medium Access Control (MAC) protocol. Using an Aloha-like MAC protocol, we propose a novel admission control framework, which can be viewed as an optimization problem that maximizes the average quality of admitted videos, given a specified minimum video quality for each flow. We present some hardness results for the optimization problem under various conditions, and propose some heuristic algorithms for finding a good solution. In particular, we show that a simple greedy layer-allocation algorithm can perform reasonably well, although it is typically not optimal. Consequently, we present a more expensive heuristic algorithm that guarantees to approximate the optimal solution within a constant factor. Simulation results demonstrate that our proposed framework can improve the video quality up to 26% as compared to those of the existing approaches.

Index Terms

Admission control, Layered video coding, Optimization, Submodular function, Video streaming, WLAN.

I. INTRODUCTION

Recent years have witnessed an explosive growth in multimedia wireless applications such as video streaming and conferencing [1]. One of the reasons for this tremendous growth is the wide deployment of the IEEE 802.11 wireless LANs (WLANs) in both private home and enterprise networks. Despite of these seemingly successes, many fundamental problems of transmitting multimedia data over wireless networks remain relatively unsolved. One of the challenges is how to efficiently guarantee a specified bandwidth for a video flow in a wireless network. The popular WLAN, particularly Distributed Coordination Function (DCF) in typical IEEE 802.11 [2] which operates under a contention-based channel access mechanism, does not provide a mechanism to guarantee minimum bandwidth for multiple concurrent flows. As a result, a video application may experience significant quality degradation due to free admission of an arbitrarily large number of flows. Nevertheless, Point Coordination Function (PCF) in typical IEEE 802.11 and HCF Controlled Channel Access (HCCA) in IEEE 802.11e [3] are able to provide a polled access mechanism to guarantee the minimum bandwidth. However, the use of PCF and HCCA mechanisms are rather limited, and often result in high latencies. Furthermore, a scheduler and queuing mechanism at the AP is needed to control to regulate the polling frequency in HCCA and PCF to provide flows with the requested throughputs. That said, this paper considers the a contention-based approach to admission control, similar to the work of Banchs et al. [4] in which, the parameters of the IEEE 802.11e in the contention-based mode are set appropriately to enable flows to achieve their requested throughputs, or maximum delay.

Admission control prevents a new flow from joining the network in order to maintain a reasonable quality of the existing flows. The decision to admit or reject a new flow that requests to enter a wireline link is arguably easier to make, compared to that of a wireless link. A simple admission control algorithm for a wireline link can keep track

of the total used bandwidth. The available bandwidth is then equal to the difference between the link capacity and used bandwidth. A new flow is admitted if its requested bandwidth is smaller than the available bandwidth of the link by some threshold, otherwise it is rejected. Theoretically, the same algorithm can be applied to a wireless link if a Time Division Multiple Access (TDMA) scheme is used to allocate bandwidth for each flow. Using a TDMA scheme, each flow is assigned a set of exclusive time slots for transmitting its data, thus eliminating the multi-user interference associated with a wireless link. As a result, the admission control algorithm can determine its available bandwidth precisely and make the decision to admit or reject a new flow accordingly.

However, such a protocol may require a centralized scheduling algorithm, which may not be feasible in a distributed environment. Therefore, existing Medium Access Control (MAC) protocols such as the IEEE 802.11, employs a random access approach that allows the flows to compete for shared channel efficiently. That is, IEEE 802.11 protocol enables the flows to achieve high throughputs, while minimizing their collisions. Thus, characterizing the wasted bandwidth from collisions is specific to a MAC protocol.

The problem of MAC protocols such as the IEEE 802.11 is the multi-user interference, i.e., the collisions between the new flow and the existing flows, which reduce all the flow's throughputs. The number of these collisions increases nonlinearly with the number of competing flows, making it harder for an admission control algorithm to determine the resulted throughputs of all the flows in order to make the right decision [5]. In particular, for a simple single-hop wireless network, to decide whether or not to admit a new flow, the admission control algorithm must ensure that the available bandwidth is at least $K + H$ kbps, where K is the total requested bandwidth including that of the new flow, and H is the incurred overhead from the collisions. While K is given to the algorithm, determining H is non-trivial when using a typical MAC protocol. Computing H is even more difficult in a multi-hop wireless network.

Even when an algorithm can determine precisely the collision bandwidth, it is not always beneficial to employ the traditional admission control framework in which, the decision to admit a new flow is solely based on the bandwidth and delay requirements of all the flows. Instead, with the advance in video coding techniques, we argue that the criterion for flow admission should be the visual quality of the video streams. That is, the inputs to the admission control algorithm are the minimum visual quality of the video streams, not their bandwidth and delay requirements. The former approach assumes that each video is coded at a certain bit rate, thus any lesser rate provided by the network, is unacceptable since the video playback will be interrupted frequently. On the other hand, with scalable video coding techniques, a video can be transmitted at different bit rates, albeit at different visual qualities. The advantage of this approach is that a larger number of flows can be allowed to enter a network as long as the video quality of each flow does not fall below a specified minimum threshold. The objective is then to maximize the average quality of all the admitted videos, given a specified minimum video quality for each stream, and the current available bandwidth.

That said, our paper aims to enhance the quality of video streaming applications in wireless home networks via a joint optimization of video coding technique, admission control algorithm, and MAC protocol. While it is possible to extend our framework to multi-hop wireless ad-hoc environment, for clarity, our discussion is limited to

a one-hop wireless network, e.g., the network of all the wireless hosts (devices) within a home or a small building such that every host can hear the transmissions of all other hosts. Using an Aloha-like MAC protocol [6], we present a novel admission control framework, which can be viewed as an optimization problem that maximizes the average quality of admitted videos, given a specified minimum video quality for each flow. In particular, using scalable video streams, our framework allows more flows to enter the network, as long as the video quality of each flow does not fall below a specified minimum threshold. We then present some hardness results for the optimization problem under various conditions, and propose two heuristic algorithms for obtaining a good solution. We show that a simple greedy layer-allocation algorithm can perform reasonable well, although it is typically not optimal. Consequently, we present a more expensive heuristic algorithm that guarantees to approximate the optimal solution within a constant factor.

The outline of paper is as follows. We first discuss a few related work on admission control for wireless networks and scalable video coding in Section II. In Section III, we describe a MAC protocol to be used in conjunction with the admission control algorithm. We then formulate the admission control framework as an optimization problem in Section IV. In Section V, we provide some hardness results for the optimization problem, and corresponding heuristic algorithms for obtaining good solutions. Simulation results will be given in Section VI. We then summarize our contributions and conclude our paper with a few remarks in Section VII.

II. Related Work

Providing QoS for flows on the Internet is extremely difficult, if not impossible, due to its original design to scale with large networks. The current design places no limit the number of flows entering the network, or attempt to regulate the bandwidth of individual flows. As a result, bandwidth of multimedia applications over the Internet often cannot be guaranteed. To that end, many scalable coding techniques have been proposed for video transmission over the Internet. Scalable video coding techniques are employed to compress a video bit stream in a layered hierarchy consisting of a base layer and several enhancement layers [7]. The base layer contributes the most to the visual quality of a video, while the enhancement layers provide successive quality refinements. As such, using a scalable video bit stream, the sender is able to adapt the video bit rate to the current available network bandwidth by sending the base layer and an appropriate number of enhancement layers [8],[9],[10],[11],[12]. The receiver is then able to view the video at a certain visual quality, depending on network conditions.

Scalable video coding techniques can mitigate the insufficient bandwidth problem, but the fundamental issue is the lack of bandwidth to accommodate all the flows. Thus, admission control must be used. While it is difficult to implement admission control on a large and heterogeneous network, e.g., the Internet, it is possible to implement some form of control or regulation in small networks, e.g., WLAN. Consequently, there have been many researches on providing some form of QoS for media traffic in WLANs [13],[14],[15],[16],[17],[18],[19].

Many existing admission control algorithms for WLANs have been proposed. Gao et al. [20] provided an admission control by using a physical rate based scheme in IEEE 802.11e. They use the long-term average physical rates to compute the reservation of the channel for some amount of time called the Transmission Opportunity

(TXOP) for each station then distribute TXOP's to everyone. Their framework provides some certain level of admission control. Xiao and Li [21] used the measurements to provide flow protection (isolation) in the IEEE 802.11e network. Their algorithm is simple, yet effective. The algorithm requires the Access Point (AP) to broadcast the necessary information to other wireless stations. In particular, the AP announces the budget in terms of the remaining transmission time for each traffic class (there are 4 traffic classes in the IEEE 802.11e) through the beacon frames. When the time budget for a class is depleted, the new streams of this class will not be admitted. Xiao and Li's work set a fixed limit on the transmission time for the entire session, resulting in low bandwidth utilization when not every traffic class approaches its limit. Recently, Bai et al. [22] improved the bandwidth utilization of Xiao and Li's work by dynamically changing the transmission time of each class based on the current traffic condition. There are also other admission control schemes implemented at different layers of the network stack. For example, Barry et al. [23] proposed to monitor the channel using virtual MAC frames and estimate the local service level by measuring virtual frames. Shah et al. [24] proposed an application layer admission control based on MAC layer measurement using data packets. Valaee et al. [25] proposed a service curve based admission procedure using probe packets. Pong and Moors [26] proposed admission control strategy for QoS of flows in IEEE 802.11 by adjusting the contention windows size and the transmission opportunity. All these admission control schemes do not take quality of the traffic, particularly video quality in our framework, into consideration directly. On the other hand, we advocate a direct cross-layer optimization of video quality, admission control algorithm, and MAC protocol, simultaneously. Most similar to our work is that of Banchs et al. [4]. Since we will be using this scheme for performance comparisions, we delay the discussion until Section VI.

III. MAC Protocol

As discussed previously, the amount wasted bandwidth from collisions in a wireless network is different when using different MAC protocol. In this section, we describe an Aloha-like MAC protocol [6] to be used in the proposed admission control framework that aims to maximize the average quality of admitted videos, given a specified minimum video quality for each flow.

In order to contrast the advantages of the new MAC protocol, we first briefly describe the existing IEEE 802.11e protocol, more specifically in a contention-based channel access scheme called Enhanced Distributed Channel Access (EDCA), which defines a set of QoS enhancements for WLAN applications through modifications to the MAC layer. To access the channel, a host first senses the channel. If the channel is idle for more than the Arbitration Interframe Space (AIFS) time, it starts sending the data. Otherwise, it sets a backoff timer for a random number of time slots between $[0, CW_{min}]$ where CW_{min} is the minimum contention window size. The backoff timer is decremented by one for each idle time slot after the AIFS time, and halts decrementing when a transmission is detected. The decrementing resumes when the channel is sensed idle again for an AIFS time. A host can begin transmission on the channel as soon as its backoff timer reaches zero. If a collision occurs, i.e., no acknowledgment packet is received after a short period of time, the backoff timer is chosen randomly between $[0, (CW_{min}+1)2^i - 1]$ where i is the number of retransmission attempts. In effect, the contention window size is doubled for each retransmission

in order to reduce the traffic in a heavily loaded network. Every time a host obtains the channel successfully, it can reserve the channel for some amount of time (TXOP). Unlike the IEEE 802.11b, IEEE 802.11e can tune the transmission parameters (e.g., CW_{min}, CW_{max}, TXOP, AIFS) to provide QoS support for certain applications. We note again that while PCF and HCCA can guarantee bandwidth for a flow, they require an AP (infrastructure mode) and tend to result in high latencies. As such, the focus of our paper is to provide admission control in a contention-based mode which is also applicable in adhoc mode settings.

However, the advantage of the existing IEEE 802.11 protocol is that it is bandwidth efficient. That is, based on the current traffic condition, each host adjusts its rate to achieve high throughput while minimizing the number of collisions. On the other hand, the rate of a flow cannot be controlled precisely unless we use PCF or HCCA. Often, this is problematic for video applications. Consequently, we argue for a different MAC protocol which, when used, would produce a stable throughput for a flow. Furthermore, it is preferable to implement the new MAC protocol with minimal hardware modification to the existing IEEE 802.11 devices. Indeed, this is possible.

In the new MAC protocol, the contention window size is not doubled after every unsuccessful retransmission attempt. Instead, depending on the rate requested by a host, it is assigned a fixed value. All other operations are exactly identical to those of the IEEE 802.11 protocol. We argue that when a proper admission control is employed, eliminating the doubling of CW in the IEEE 802.11 protocol, helps to increase the bandwidth efficiency since the rate of each host is not reduced unnecessarily. Based on the above discussion, it is crucial for an admission control algorithm to determine whether or not there exists a set of CW's for each host that satisfies their requested rates without doubling CW's. To answer this question, we now proceed with an analysis of the new MAC protocol.

We assume the use of reservation packets, i.e., Request-To-Send/Clear-To-Send (RTS/CTS) packets. RTS/CTS packets are employed to reduce the collision traffic as well as eliminating the hidden terminal problem [27]. The main idea is to send small packets to reserve the channel for the actual data transmission. By doing so, collisions only occur with the small packets, hence reducing the amount of wasted bandwidth. Since we assume that all the hosts can hear each other's transmissions, we do not have the hidden terminal problem. Our use of RTS/CTS is simply to reduce the collision bandwidth.

Our analysis is based on time-slotted, reservation based protocols similar to the Aloha protocol, where the time taken to make a reservation is a geometrically distributed random variable with parameter p. On significant difference between the our protocol and the Aloha protocol is that all the hosts in our network are assumed to be able to hear transmissions of other. Therefore, a host will not attempt to transmit if it determines that the channel is busy, i.e., some host is sending. Thus, a host will attempt to send an RTS packet with probability p only if it determines that the channel is idle.

Assume the host transmits the packets with some probability p. To translate the transmission probability p back to the contention window size used in IEEE 802.11 protocol, CW can be set to $1/p$. We note that this is only an approximation since CW in the IEEE 802.11 protocol is not reset at every time slot. To simplify the analysis, we further assume that every host can start at most one flow at any point in time. A straightforward generalization to support multiple flows per host is to consider all the flows from one host as one single large flow with the

transmission probability p. Whenever a host successfully obtains the channel, it selects a packet from one of its flows to send. The probability of a packet selected from a particular flow then equals to the ratio of that flow's throughput to the total throughput of all the flows on the same host. This approach would result in the correct average required throughputs for all the flows.

For a network with N flows, our objective is to determine whether or not there exists a set of p_1,p_2,...,p_N and for each flow such that all the flows achieve their specified throughputs R_1,R_2,...,R_N, taking into account of collisions. Since the rates R_i's depend on the percentages of successful slots, we first characterize the percentages of collided, successful, and idle slots, given p_i's for each flow i. To that end, let us denote

- I: percentage of idle slots
- S_i: percentage of successful RTS slots for flow i
- C: percentage of collided slots
- R_i': throughput of flow i as a fraction of the channel capacity.

Note that $I + C + \sum_i S_i = 1$. Suppose the transmission probability for a new flow is p, then for $C - type$ slots, in which collisions occur, the new traffic would have no impact on it. For $S - type$ slots, with probability p, it may cause a collision. For an $I - type$ slots, with probability p, it would become a $S - type$ slot. Otherwise it stays the same. Using the above argument, we can calculate I, S, and C after the new flow starts. In particular, the new idle, collided, and successful probabilities can be calculated using the current I, C, S, and p as:

$$S_{new} = S_{current}(1-p) + I_{current}p \quad (1)$$

$$I_{new} = I_{current} - I_{current}p \quad (2)$$

$$C_{new} = 1 - I_{new} - S_{new}. \quad (3)$$

Here, we denote $S = \sum_i S_i$. Similarly, we can calculate the successful probability S_i as

$$S_{i,new} = S_{i,current}(1-p), \quad (4)$$

for any existing flow i, and the successful probability for the new flow (S_N) as

$$S_N = I_{current}p. \quad (5)$$

Using the equations above, one can compute the I's, C's, and S_i's for N flows, given the transmission probabilities $p_1, p_2, ..., p_N$. In particular, the following algorithm can be used to compute the collision probability C, which will be used in the admission control algorithm later.

Algorithm 1: *Computing C, given the transmission probabilities p_i's*

$C = Compute\_C(p_1, p_2, ..., p_N, N)$

$I = 1$

$C = 0$

$S = 0$

for i = 1 to N **do**

$S = S \times (1 - p_i) + I \times p_i$

$I = I - I \times p_i$

$C = 1 - I - S$

end for

$return\ C$

Algorithm 1 enables us to compute the successful probability precisely based on the given transmission probabilities p_i's. On the other hand, one typically wants to determine the transmission probabilities p_i's, given the requested rates R_i''s from each flow i. Since the rate R_i' is proportional to the corresponding successful probability S_i, we now show how to compute p_i's based on S_i's. We then show how to relate S_i's to R_i''s, completing our objective.

In principle, (1)-(5) enable us to write down a set of N equations with N unknown variables p_1,p_2,...,p_N in terms of the known variables S_i's, and solve for p_i's. Unfortunately, these equations are not linear, and therefore difficult to solve. We propose an algorithm to find the p_i's given S_i's based on the following observation: When a flow i stops, I will increase by S_i. If flows i starts again with the same transmission probability p_i as before, its successful probability remains S_i as before. Hence, the following equations hold:

$$(I + S_i)p_i = S_i$$
$$p_i = \frac{S_i}{I + S_i} \tag{6}$$

This is true because $I + S_i$ is the probability of idle slots without flow i. Hence, after the flow i starts, its successful probability is $(I + S_i)p_i$ which should also equal precisely to S_i, the successful probability before it stops. Thus, we have N such equations corresponding to N flows. We also have the constraint:

$$I + C + S = 1 \tag{7}$$

where C and S are the collision and successful probabilities for all the flows. We note that I is the same for every equation since it is the probability of idle slots when all flows are active. Now, we can solve for $N + 1$ unknowns, i.e. N for p_i's and one for I. Solving this set of $N + 1$ equations is simple since each equation is linear except (7). Equation (7) is non-linear in p_i because C and S are polynomials in p_i which are the results from (1)-(5). However, (7) will be used as a constraint. Since $I \in [0, 1]$, one can systematically try different values of I from large to small, i.e., 1 to 0. For each value of I, we compute p_i's according to (6). All the p_i's are then input to Algorithm 1 to compute C. We then test to see whether or not $I + C + S$ approximately equals to 1. If so, we have an admissible set of solutions. If not, we increase I by a small value and repeat the procedure. If the algorithm cannot find such I for the entire range of $I \in [0, 1]$, then the solution does not exist. This indicates invalid S_i's.

Typically, R_i''s, not S_i's, are given. Therefore, to use the procedure above, we first calculate the S_i's in terms of R_i''s. With minimal modification from IEEE 802.11e standard, our framework uses IEEE 802.11e frame formats and the timing diagram as shown in Fig. 1. After the channel is idle for a period time equal to a Distributed Interframe

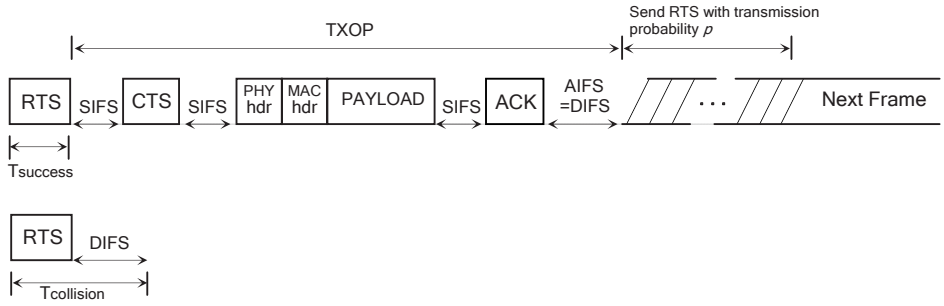


Fig. 1. Timing diagram with RTS/CTS for our proposed framework.

Space (DIFS) time slots, instead of counting down CW before beginning a new transmission, the host sends RTS with probability p to reserve the shared channel. That means we set AIFS=DIFS as same as the one in typical IEEE 802.11 standard [2]. Because all hosts can listen each other transmissions, the collision will occur only if there are more than one hosts initiating RTS's at exactly the same time slot. Otherwise, the host successfully reserves the channel then that host can begin the transmission for $TXOP$ time slots without further collision. A host detects unsuccessful transmission of an RTS if none of the CTS arrives within DIFS time slots. Note that a host needs to wait for a short period of time called Short Interframe Space (SIFS) where SIFS $<$ DIFS before sending an ACK as shown in Fig. 1. To be fair among all the flows with the same traffic class [3], i.e. video streams, everyone uses the same $TXOP$ where $TXOP$=CTS+PHY$_{hdr}$+MAC$_{hdr}$+PAYLOAD+ACK+3SIFS+DIFS.

Suppose after T time slots where T is large, we observe that there are K_i successful transmissions of RTS and $K_i \times TXOP$ slots of data transmission for each flow i. Then by definition, we have:

$$\begin{aligned} S_i &= \frac{K_i}{T - \sum_i^N K_i \times TXOP} \\ &= \frac{K_i \times TXOP/T}{TXOP \times (1 - \sum_i^N K_i \times TXOP/T)} \\ &= \frac{R_i'}{TXOP \times (1 - \sum_i^N R_i')} \end{aligned} \tag{8}$$

where $R_i' = K_i \times TXOP/T$ can be thought of as the host i's requested bandwidth in terms of a fraction of the channel capacity and N is the number of flows. If the channel capacity is BW, then the transmission rate R_i can be computed such that $R_i = R_i' \times BW$. For example, if channel capacity (BW) is 54 Mbps, and host i requests the rate (R_i) of 27 Mbps, then R_i' = 0.5.

Using (8), given the specified rates R_i''s, one can compute the corresponding S_i's, which are then used in the following algorithm to determine the transmission probabilities p_i's, if there are such p's.

Algorithm 2: *Compute p_i's given all R_i''s*

$[p_1, p_2, ..., p_N, success] = Compute\_p(R'_1, R'_2, ..., R'_N, N)$

$\epsilon = 0.01$

$I' = 1$

$\{I'$ is the percentage of idle slots$\}$

$search\_step = 0.01$

$success = 0$

for $i = 1$ to N **do**

$S_i = \frac{R'_i}{TXOP \times (1 - \sum_i^N R'_i)}$

end for

while $I' < 1$ **do**

for $i = 1$ to N **do**

$p_i = \frac{S_i}{I' + S_i}$

end for

{run Algorithm 1 to compute collision probability C}

$C = Compute\_C(p_1, p_2, ..., p_N, N)$

$total = I' + C(RTS + DIFS) + \sum_i^N S_i(RTS)$

{check for boundary condition smaller ϵ results in higher accuracy}

if $(abs(total - 1) < \epsilon)$ **then**

$success = 1$

$return\ [p_1, p_2, ..., p_N, success]$

end if

$I' = I' - search\_step$

end while{fail to find p, success = 0}

$return\ [0, 0, ..., 0, success]$

We note that for each unsuccessful RTS transmission, we waste the channel equal to RTS+DIFS time slots. On the other hand, each successful RTS transmission uses only RTS time slots. Furthermore, Algorithm 2 explicitly considers the percentage of collided, successful, and idle slots with respect to RTS transmissions to reserve the channel. This results in $total = I' + C(RTS + DIFS) + \sum_i^N S_i(RTS)$ where $total$ is close to (or equal to) 1. We now describe our proposed admission control framework.

IV. Admission Control Framework

A. Architecture

Due to a typical small size of a single-hop network, our admission control algorithm runs at the AP or an elected host. We assume that all hosts are collaborative. That is, each host obeys the admission protocol which operates as follows.

For simplicity, in this paper, we assume that there is no cross-traffic of any kinds except videos. In general, to accommodate other non-time sensitive traffic in the proposed framework, one can perhaps set the minimal throughput requirements for the traffic. Each host can send videos to and receive from the Internet, or they can send videos among each other. When a host wants to inject a new video stream into the network, it first requests to join the network by sending a message to the AP. For video streaming applications, the message may contain the rate distortion profile of the video, which specifies the different distortion amounts and the corresponding number of layers used. The message also contains the maximum allowable distortion for a video. Note that for live video applications, the rate-distortion profile is typically not known ahead of time, but can be estimated. That said, the focus of this paper will be on streaming applications. Upon receiving the request, the AP (or some elected host) will run the admission control algorithm to produce a set transmission probabilities p_i's for each flow i that maximizes the average visual quality of all the flows, given the maximum distortion levels for each video and overall bandwidth constraint. If such transmission probabilities exist, the AP will broadcast the transmission probabilities p_i's to all the hosts.

Upon receiving AP's instructions, each host i begins to transmit its packets with probability p_i (or roughly setting its contention window to $1/p_i$) when it observes that the channel is idle. Each transmission probability p_i corresponds to a particular rate (or number of layers). If there is no feasible set of transmission probabilities, the AP will inform the new flow that it cannot join the network at this moment. Note that our proposed protocol is able to extend to serve the services for other flow types while the AP requires to know their rate-distortion profiles in order to minimize overall distortion effectively.

B. Problem Formulation

We are now at the position to formulate a rate-distortion optimization problem for multiple layered video streams under bandwidth and distortion constraints. We note that the average throughput per unit time or transmission rate R_i for flow i can be achieved by setting its transmission probability p_i. When there is enough bandwidth for everyone, p_i's are set to large values so that all the layers of all the video streams would be sent. When there is not enough bandwidth, e.g. due to too many flows, the layers from certain videos are dropped resulting in the least average distortion over all the videos. For a simple scenario, we assume that there is no packet loss during transmission. The transmission rate $R_i(l_i)$ for flow i is proportional to the number of transmitted video layers l_i.

The optimization problem studied in this paper is to select the optimal number of video layers to transmit for each of N hosts (or N flows) while maximizing the overall video quality. Furthermore, the inclusion of the bandwidth overhead term $H = C + S$ due to channel contention access (collision and reservation bandwidth used for RTS/CTS packets) makes our optimization problem distinct from other optimization problems. In particular, the problem is specified by giving: for each host, a function $D_i\left(l_i\right)$, that gives the reduction in distortion when using l_i layers at host i; a rate function $R_i\left(l_i\right)$, that gives the required bandwidth for transmitting l_i layers from host i; an overhead function $H\left(l_1, ..., l_N\right)$, that gives the amount of bandwidth consumed by overhead (e.g., due to the channel contention) for a given assignment of layers to hosts; lower bounds on the reduction in distortion for each

i denoted by Z_i; and finally a bound on the total bandwidth BW. Given these quantities, the optimization problem is as follows:

$$\begin{aligned} &\textbf{maximize} && \sum_{i=1}^{N} D_i\left(l_i\right) \\ &\textbf{over} && l_i \\ &\textbf{subject to} && D_i\left(l_i\right) \geq Z_i \\ & && \sum_{i=1}^{N} R_i\left(l_i\right) + H\left(l_1, ..., l_N\right) \leq BW \end{aligned} \tag{9}$$

That is, we must find the optimal assignment of layers to each host that maximizes the reduction in total distortion subject to bandwidth and local minimum reduction in distortion constraints. In particular, there exists a solution iff we are able to compute a set of transmission probabilities p_i's corresponding to an optimal assignment of layers for everyone. A necessary condition is that each flow i is required to maximize its total reduction in distortion D_i at least Z_i. Nevertheless, the way to select the layer depends on what layer-selection strategies we use (e.g., greedy algorithm, exhaustive search). Note that propagation delay and processing delay can be negligible due to operating in a single-hop network. However, the delay variation or jitter, would likely affect the performance of the protocol. The detail analysis of throughput jitter is discussed in Section VI-C. Next, we will study the computational properties of the layer-selection problem and show that while in general the problem is computationally hard, under certain reasonable conditions, a simple greedy layer-allocation algorithm can be guaranteed to perform close to optimal.

V. Computational Complexity of Layer Optimization

In this section, we study the computational complexity of the layer allocation problem described above, showing both hardness results and conditions under which optimal and approximate solutions can be guaranteed in polynomial time. Our optimization problem is distinct from most other bandwidth optimization problems by its inclusion of the overhead term H in the bandwidth constraint. Thus, existing algorithms and complexity proofs do not directly apply to our problem. Below we first consider the complexity of solving the problem optimally and then we consider efficient approximation algorithms.

A. Computing Optimal Solutions

Here we analyze the computational complexity of problem classes with the form given in the previous section. We begin by stating three assumptions about the optimization problem and consider the complexity under various subsets of these assumptions.

Assumption 1: Uniform rate increase per level

$$R_i\left(l+1\right) - R_i\left(l\right) = R_j\left(l'+1\right) - R_j\left(l'\right) \text{ ; for any } i, j, l, \text{ and } l' \tag{10}$$

Assumption 2: Diminishing returns

$$D_i(l+1) - D_i(l) \leq D_i(l) - D_i(l-1) \ ; \text{for any } i \text{ and } l \tag{11}$$

Assumption 3: Invariant overhead

$$H(..., l_i + 1, ...) = H(..., l_j + 1, ...) \ ; \text{for any } l_1, ..., l_N \tag{12}$$

Below we will also refer to the property of additive overhead which means that $H(l_1, ..., l_N)$ can be factored as a sum of the individual overhead function H_i. That is,

$$H(l_1, ..., l_N) = \sum_{i=1}^{N} H_i(l_i) \tag{13}$$

Intuitively the first assumption states that the amount by which the rate function increases is constant across all layers of all streams. The second assumption states that within a particular stream, higher layers may never reduce distortion more than the lower layers. Thus, it will never be the case that a stream must include many lower layers with low distortion reduction in order to get a big distortion reduction at a higher layer. The third assumption states that given a particular layer allocation across layers, incrementing any layer by one produces the same increase in the overhead function. This means that the overhead function is impartial to both the particular stream that is incremented and the current number of layers allocated to that stream.

Our first result is that given the above three assumptions (10)-(12), we can solve the optimization problem using an efficient greedy layer-allocation algorithm. The algorithm proceeds as follows:

1) For each stream i, we initialize the layer count l_i to the smallest l_i such that $D_i(l_i) \geq Z_i$. If for some i this is not possible, then return "no solution".
2) If it is not possible to increment the layer count of any stream without violating the bandwidth constraints then terminate and return the current layer counts. In other words, it is not possible to find a feasible set of transmission probabilities for each host using the Algorithm 2 in Section III.
3) Increment the layer count of stream i by 1, where stream i is the stream that when incremented produces the greatest reduction in distortion without violating bandwidth constraints.

Proposition 1 *The greedy layer-allocation algorithm is optimal for any problem where Assumptions 1, 2, and 3 hold.*

Proof: We first introduce some notation. We will use an allocation vector $L = \langle l_1, \ldots, l_N \rangle$ to specify the layer allocation l_i to each host i where N is the total number of hosts. We will denote by $D(L)$ the reduction in distortion resulting from allocation vector L. A layer increment sequence is a sequence of host indices $(i_1, \ldots, i_k)$, indicating the order of layers to increment finally arriving at a final allocation vector where k is the total increments in layers.

Note that with invariant overhead and uniform rate increase, each increment in the layer counts results in exactly the same increase in bandwidth. This means that all optimal layer allocations will satisfy $\sum_i l_i = k$ for some value

k. That is, all optimal layer allocations will be a result of exactly k increments to layer counts. Thus, finding the optimal layer allocation is equivalent to finding a length k layer increment sequence that results in the best layer allocation starting from the null allocation.

Now consider any layer allocation L and let i^* be the index of the host that would be selected by the greedy algorithm starting from L and let Δ^* be the reduction in distortion resulting from the greedy step. Now consider any layer increment sequence $(i_1, \ldots, i_v)$ starting at L resulting in an allocation vector L_v. We say that the sequence is an optimal v-step completion of L if the value of $D(L_v)$ is the maximum possible when starting from L and incrementing v layers.

Our key claim is that there is always an optimal v-step completion to L that includes an increment to i^*. Assume that this was not the case and that the above sequence was an optimal completion, implying that it does not contain an increment to i^*. We show that this leads to a contradiction. First, let Δ_j equal the reduction in distortion resulting after adding the j'th layer increment and note that $D(L_v)$ is equal to the sum of this sequence. By the diminishing returns assumption we have that $\Delta^* \geq \Delta_j$ for all j. This is true because the greedy algorithm selected the index i^* with the largest decrease in distortion across all layers and thus any further decreases resulting from incrementing any layer must not be greater than that, otherwise this would violate diminishing returns. Given this fact consider the new layer increment sequence $(i^*, i_1, \ldots, l_{v-1})$ and let L^* equal the result of applying this sequence starting at L. It can be easily verified that this is a legal sequence and that the corresponding sequence of reductions in distortion is equal to $(\Delta^*, \Delta_1, \ldots, \Delta_{v-1})$. Since $D(L^*)$ is simply the sum of this sequence and we know that $\Delta^* \geq \Delta_v$ this implies $D(L^*) \geq D(L_v)$. Thus, we have shown an optimal k-step completion that includes an increment to i^*, which gives a contradiction.

Using the above fact, it is straightforward to show by induction on the number of greedy steps k that the greedy algorithm always maintains an optimal k-step completion of the null set, which completes the proof. ■

We now show that in a sense the above assumptions are necessary for the greedy algorithm to be optimal. Furthermore, they are necessary for there to exist any efficient solution algorithm, unless P=NP. In particular, the next series of propositions show that if we remove any one of the assumptions the problem becomes NP-hard. Below our results concern the decision-problem version of the above optimization problem. That is, the problem of deciding whether there is a feasible solution given a particular distortion threshold as input. Note that if the optimization problem can be solved efficiently then so can the decision problem. Thus, hardness results about the decision problem pertain to optimization as well.

Our hardness proofs are all based on reductions from the 0-1 knapsack problem [28], [29]. A 0-1 knapsack problem provides us with a finite set of objects, each having a specified value and cost, along with a total value goal and a total cost limit. We are asked to decide whether there is a subset of objects whose total value meets the goal, but has cost less than the specified limit. More formally, a 0-1 knapsack problem is a 4-tuple,

$$\langle \{v_1, ..., v_N\}, \{c_1, ..., c_N\}, V, C \rangle \tag{14}$$

where v_i and c_i, and give the value and cost of the i'th item, V is the value goal and C is the cost limit. We are

asked to decide whether there is a subset S of $1, ..., N$ such that

$$\sum_{i \in S} v_i \geq V, \quad \sum_{i \in S} c_i \leq C \tag{15}$$

In this section, we will specify our layer-allocation decision problem by the following 6-tuple,

$$\langle \{M_1, ..., M_N\}, \{D_1(l_1), ..., D_N(l_N)\}, \{R_1(l_1), ..., R_N(l_N)\}, H(l_1, ..., l_N), D, BW \rangle \tag{16}$$

where for each flow i, M_i is the maximum number of layers, $D_i(l_i)$ is the reduction in distortion function, $R_i(l_i)$ is the rate function, H is the overhead, D is the (reduction in) distortion bound, and BW is the bandwidth limit. For the purposes of complexity analysis, we will make the reasonable assumptions that the M_i are encoded in $\log M_i$ bits and that the D_i, R_i, and H functions can be evaluated in polynomial time in the size of the problem encoding.

Given an instance of the layer-allocation decision problem as in (17), we are asked to decide whether there is an assignment of non-negative integers to variables $\{l_1, ..., l_N\}$ such that

$$\begin{gathered} l_i \leq M_i \\ \sum_i D_i(l_i) \geq D \\ \sum_{i=1}^{N} R_i(l_i) + H(l_1, ..., l_N) \leq BW \end{gathered} \tag{17}$$

Note that this formulation of the problem ignores the local distortion constraints $D_i(l_i) \geq Z_i$ that were present in our original problem. It turns out that a problem with such constraints can be easily normalized to one without the constraints[1] and hence for simplicity we ignore the constraints in this section.

We now consider in Propositions 2-4, the complexity of this problem when each one of the above assumptions is lifted.

Proposition 2 *The class of decision problems for which Assumptions 2 and 3 hold but Assumption 1 does not is NP-complete even if we restrict the overhead to be the constant zero function.*

Proof: Our problem is clearly in NP, as it is possible to enumerate possible layer allocations and check them in polynomial time. Each layer-allocation certificate is polynomial-size implying the decision problem is in NP. To show NP-hardness we reduce from 0-1 knapsack problem. Given an instance of 0-1 knapsack as shown in (14), we will form the following version of our problem as in (18),

$$\langle \{1, ..., 1\}, \{D_1(l_1), ..., D_N(l_N)\}, \{R_1(l_1), ..., R_N(l_N)\}, H(l_1, ..., l_N), V, C \rangle \tag{18}$$

where $D_i(l_i) = v_i$, $R_i(l_i) = c_i$, and $H(l_1, ..., l_N) = 0$ for all inputs. That is, we have only one layer to allocate. The reduction in distortion and the rate function for that layer are equal to v_i and c_i of the 0-1 knapsack problem, respectively. Note that this problem does not satisfy the constant rate increase since c_i can be different for each i.

[1]In particular, one can easily determine the minimum number of layers required by each stream to meet the local distortion constraint. One can then formulate the problem of optimizing the layer allocation beyond those minimal layers, which does not need to include local distortion constraints.

However, it does satisfy Assumptions 2 and 3 trivially. Considering (17) it is straightforward to show that the answer to the layer-allocation problem given by (18) will be "yes" if and only if the answer is "yes" for the corresponding 0-1 knapsack problem. ■

Proposition 3 *The class of problems for which Assumptions 1 and 3 hold but Assumption 2 does not is NP-complete even if we restrict the overhead to be the constant zero function.*

Proof: The problem is in NP for the same reasons as above. For the purposes of this problem we will only consider the 0-1 knapsack problem with integer values of v_i, c_i, V, and C. We can do this without loss of generality since we can always multiply all numbers by the appropriate power of 10. Given an instance of the 0-1 knapsack problem as specified in (14), our reduction constructs the following layer allocation problem,

$$\langle \{c_1, ..., c_N\}, \{D_1(l_1), ..., D_N(l_N)\}, \{R_1(l_1), ..., R_N(l_N)\}, H(l_1, ..., l_N), V, C \rangle \tag{19}$$

where $D_i(l_i) = 0$ for $l_i < c_i$, $D_i(c_i) = v_i$, $R_i(l_i) = l_i$, and $H(l_1, ..., l_N) = 0$. The intuition here is that we have many layers in each stream, but only the last layer actually reduces the distortion. This type of behavior violates Assumption 2. Each layer adds exactly one unit of bandwidth which satisfies Assumption 1 and the overhead is zero, which satisfies Assumption 3. Note also that the number of layers for flow i is c_i. So in order to get a reduction in distortion of v_i, we must pay a bandwidth of c_i, which aligns with the 0-1 knapsack problem.

Given this reduction, it shows that there is a "yes" answer for the constructed layer allocation problem iff there is a "yes" answer for the 0-1 knapsack instance. ■

Finally, we show that Assumption 3 is also necessary in some sense. In particular, when it is lifted the problem becomes computationally hard even when restricted to the class of problems with additive overhead.

Proposition 4 *The class of problems for which Assumptions 1 and 2 hold but Assumption 3 does not is NP-complete even if we restrict to additive overhead.*

Proof: Again here we will only consider the integer knapsack problem. Given an instance of an integer 0-1 knapsack problem, we construct the following instance of the layer allocation problem,

$$\langle \{v_1, ..., v_N\}, \{D_1(l_1), ..., D_N(l_N)\}, \{R_1(l_1), ..., R_N(l_N)\}, H(l_1, ..., l_N), V, C \rangle \tag{20}$$

where $D_i(l_i) = l_i$, $R_i(l_i) = 0$, and $H(l_1, ..., l_N) = \sum_i H_i(l_i)$ where $H_i(l_i) = c_i$ for $l_i > 0$ and $H_i(0) = 0$. So here we have the diminishing return property since for each layer we add a single unit of reduction in distortion. We have the constant bandwidth property trivially. But we do not have invariant overhead since when we move a layer l_i from 0 to 1 we get an increase in c_i. Only the overhead function occupies the bandwidth since all the rates are equal to zero. Given this reduction it is easy to verify that there is a "yes" answer to the 0-1 knapsack instance iff there is a "yes" answer to the constructed layer allocation problem. ■

Together these complexity results show that if we remove any one of three assumptions, the problem becomes NP-hard and hence is not likely to be solved by an efficient algorithm, in particular the greedy algorithm. The results also show that this is true even if we place strict restrictions on the form of the overhead function. Even if the overhead is additive the problem is hard as shown by Proposition 4.

In practice, it may often be possible to satisfy assumptions 1 and 2. Unfortunately, we can show that the overhead function arising in our protocol is not invariant as required by Assumption 3 and hence an efficient optimal solution is still unlikely.

Proposition 5 *The overhead of the proposed MAC protocol is not invariant.*

Proof: We will show that the overhead of proposed MAC protocol is not invariant by contradiction. Assuming that the bandwidth overhead $H = C + S$, bandwidth involved in reservation, is invariant, then:

$$H'(i) = H'(j) \tag{21}$$

where $H'(i)$ and $H'(j)$ denote the overhead resulted from adding δ bps (one layer) into flow i and flow j, respectively. Since $I + C + S = I + H = 1$, we have

$$I'(i) = I'(j) \tag{22}$$

where $I'(i)$ and $I'(j)$ denote the idle slots resulted from adding δ bps into flow i and flow j, respectively. In particular, adding δ bps into flow i results in the increase of S_i by Δ. We can represent I in terms of S_i as:

$$\begin{aligned} I &= \prod_i (1 - p_i) \\ &= \prod_i \left(1 - \frac{S_i}{I + S_i}\right) \\ &= \prod_i \left(\frac{I}{I + S_i}\right) \end{aligned} \tag{23}$$

That is,

$$\left(\frac{I'(i)}{I'(i) + (S_i + \Delta)}\right) \prod_{k \neq i} \left(\frac{I'(i)}{I'(i) + S_k}\right) = \left(\frac{I'(j)}{I'(j) + (S_j + \Delta)}\right) \prod_{k \neq j} \left(\frac{I'(j)}{I'(j) + S_k}\right) \tag{24}$$

Expand the product on both sides of (24). All product terms, except the one with either S_i or S_j, will be canceled out, leading to:

$$\left(I'(i) + (S_i + \Delta)\right)\left(I'(i) + S_j\right) = \left(I'(j) + S_i\right)\left(I'(j) + (S_j + \Delta)\right) \tag{25}$$

Since $I'(i) = I'(j)$, (25) is true iff $S_i = S_j$. Since S_i is directly proportional to R_i, this implies that in order to achieve $H'(i) = H'(j)$, the current sending rate of any two video stream must be equal to each other. This is not true in general, therefore the bandwidth overhead of the proposed MAC protocol is not invariant. ■

The result shows that adding a video layer from different flows causes an uncertain amount of overhead bandwidth. Thus, the bandwidth overhead of the MAC protocol is neither additive nor invariant. It is correlated to all active flows in the system, not individual flows. This shows that the greedy allocation algorithm is not guaranteed to be optimal for the proposed MAC protocol, though it may still provide practically useful solutions. These results motivate investigating whether there are variants of the protocol that are overhead invariant to allow for tractable optimization in the presence of Assumptions 1 and 2, which will often be satisfied in practice. Alternatively, our results below show that a slight variation on the greedy algorithm allows for the efficient computation of approximately optimal solutions under only Assumption 1.

B. Computing Approximate Solutions

Of the three assumptions above, Assumption 2, diminishing returns, is the one that is most likely to be satisfied in application settings, since most coding schemes exhibit diminishing returns behavior. Assumption 1, uniform rate increase, will also often be satisfied, though it will rule out non-uniform rate coding schemes. Assumption 3, invariant overhead, as we have seen is violated by the our protocol and we are unaware of other protocols that satisfy the assumption. In this light, it is interesting to consider what can be said theoretically when only Assumption 2 is satisfied. We know from Propositions 2 and 3 that in general the problem is NP-hard to solve optimally with only Assumption 2, however, this does not rule out the existence of efficient approximation algorithms. In this section, we show that small extensions to the greedy layer-allocation algorithm can result in algorithms that are guaranteed to return a solution within a factor of $(1 - e^{-1}) \geq 0.63$ from the optimal solution.

Our approximation results are based on an extension to recent approximation results for the constrained maximization of submodular set functions. Below we first introduce the concept of submodular functions, describe existing approximation results, our extensions to them, and how they apply to the layer allocation problem.

Optimizing Submodular Functions. Given a finite set E of elements and a function F on subsets of E, we say that F is *non-decreasing* if $F(A) \leq F(A')$ for any subsets A and A' such that $A \subseteq A'$. We say that F is *submodular* if

$$F(A \cup \{x\}) - F(A) \geq F(A' \cup \{x\}) - F(A') \tag{26}$$

for any subsets A and A' such that $A \subseteq A'$ and any $x \in E$. Intuitively speaking, a function is submodular if it exhibits a diminishing returns property, where the increase in F resulting from adding an element to A is at least as large as adding the same element to a superset A' of A.

Submodular functions have nice properties with respect to optimization. A classic result concerns finding the subset $A \subseteq E$ of size k that maximizes a submodular and non-decreasing function F. It has been shown [30] that the simple greedy allocation algorithm we described earlier—i.e. greedily add the next element from E that increases F the most—achieves an approximation factor of $(1 - e^{-1})$. That is, for any F that is submodular and non-decreasing, the set A that results from the greedy algorithm will satisfy $F(A) \geq (1 - e^{-1})F(A^*)$, where A^* is the optimal solution to the maximization problem.

Note that the above approximation result concerns finding the best subset of k elements, which implicitly means that the cost of each element in E is the same and equal to 1. In many cases, we are rather interested in the situation where each element x has a cost $c(x)$ and the goal is to find the subset A that maximizes F subject to the constraint that the total additive cost of elements in A is less than a budget B. We will call this problem the *additive-cost submodular maximization problem*. The original approximation result does not apply to this problem and in fact the greedy algorithm can be shown to yield arbitrarily poor results [31]. Recent results, however, have shown that slight extensions to the greedy algorithm can result in approximation bounds for additive-cost submodular maximization [31], [32].

For the first approximation result, from [32], consider a modified version of the greedy algorithm, where at each step instead of adding the element x that most increases F, we add the element x that achieves the largest ratio of the increase in F to $c(x)$, provided that adding x does not violate the budget B. That is, we initialize the set A to null and then at each iteration add the x with maximum value of $\left(F(A \cup \{x\}) - F(A)\right)/c(x)$, such that the cost of $A \cup \{x\}$ is less than B. The iteration repeats until no element can be added to A without violating the budget. Thus, in order to justify adding an element x with a large cost to the current set, it must yield a large increase in F. It can be shown that even this modified algorithm can yield arbitrarily poor results compared to the optimal solution. However, if one returns the best solution found by either the original greedy algorithm or the modified greedy algorithm then one can guarantee an approximation factor of $0.5(1 - e^{-1})$. We will call this algorithm the *Double-Greedy algorithm*. This shows that by increasing the computation time by a factor of two over the original greedy algorithm (i.e. we must now run both greedy and modified greedy) it is possible to achieve a non-trivial approximation bound.[2]

It turns out that by increasing the computation further, but remaining polynomial time, it is possible to improve the approximation bound to $(1 - e^{-1})$ for the constrained optimization problem, which matches the result for the unconstrained problem, as proven in both [31], [32]. The new algorithm simply enumerates all triples of elements from E that do not violate the budget constraint. For each triple T, the algorithm runs the modified greedy algorithm initialized to the set T resulting in a final set A_T. The algorithm then returns the set A_T among all of the triples that achieved the highest value of F. We will refer to this algorithm as the *All-Triples-Greedy algorithm*. This algorithm increases the runtime by a factor of $O(|E|^3)$ over the original greedy algorithm, but yields a much stronger approximation result.

Unfortunately, we found that it was not possible to cast our layer-allocation problem as an instance of additive-cost submodular optimization. The primary reason is the inclusion of the overhead term in the bandwidth constraint, which does not allow for an additive cost model in the general case. This led us to extend the results of [31], [32] to handle a more general maximization problem sufficient for our purposes. In particular, we now assume in addition to a submodular, non-decreasing function F that we have a monotonically increasing cost function C on subsets of E. That is, adding an element to a set always increases the cost of the set. Note that additive cost functions are a special case of this formulation, when all element costs $c(x)$ are positive, but that many other cost functions are now also included. Our problem now is to find the set A that maximizes the function F subject to the constraint $C(A) \leq B$. We will call this problem the *monotonic-cost submodular maximization problem*. It turns out that one can prove identical approximation results to the above with only slight modifications to the *Double-Greedy* and *All-Triples-Greedy* algorithms. In particular, rather than greedily select elements according to

[2]Technically it is not necessary to run the full greedy algorithm in order to achieve the approximation bound. Rather one need only run the original greedy algorithm for one iteration (i.e. selecting the best single element of the set) and then return the maximum of the best single element and the result of the modified greedy algorithm. The results of *Double-Greedy* are guaranteed to be at least as good as this and often better, but with an increase in computation time.

$(F(A \cup \{x\}) - F(A))/c(x)$, one selects elements according to

$$\frac{F(A \cup \{x\}) - F(A)}{C(A \cup \{x\}) - C(A)} \tag{27}$$

That is, rather than normalize according to $c(x)$ we normalize according the increase in the cost associated with adding x to A. Note that when the cost function C is additive this difference is always equal to $c(x)$ and hence the algorithms are identical to those described above. We will refer to these modified algorithms as *Double-Greedy** and *All-Triples-Greedy** respectively. We can prove the following,

Theorem 1 *The Double-Greedy* algorithm achieves a constant factor approximation bound of* $0.5(1 - e^{-1})$ *for the monotonic-cost submodular maximization problem.*

Theorem 2 *The All-Triples-Greedy* algorithm achieves a constant factor approximation bound of* $(1 - e^{-1})$ *for the monotonic-cost submodular maximization problem.*

We do not include the proofs of these theorems in this paper as they are quite involved and require only very minor modifications to the proofs provided in [32]. In particular, one need only verify that replacing occurances of $c(x)$ with $C(A \cup \{x\}) - C(A)$ in the proof is correct provided that C is a monotonically increasing set function. The main contribution of our work with respect to submodular optimization is to recognize that the existing approximation results can be easily extended to the useful and much more general case of arbitrary monotonic cost functions. To the best of our knowledge this has not been previously observed.

Application to Layer Optimization. Using the above theorems it is straightforward to give approximation algorithms for our layer optimization problem. The key is to show that the problem can be viewed as one of optimizing a submodular function, upon which we can apply the above results. To do this consider a problem with N streams each of which, without loss of generality, has m layers. Now define the set $E = \{x_{i,j} | 1 \leq i \leq N, 1 \leq j \leq m\}$, where we think of each stream i as containing m elements in E indexed by $x_{i,1}$ through $x_{i,m}$. Consider any set $A \subseteq E$ and let $l_i(A)$ denote the number of elements in A that correspond to stream i, i.e. $l_i(A) = |\{x_{i,j} \in A | 1 \leq j \leq m\}|$, and let $L(A) = \langle l_1(A), \ldots, l_N(A) \rangle$. In this way, we can view each $A \subseteq E$ as specifying a layer assignment vector $L(A)$ across the streams. Also note that for any possible layer assignment vector L, there is a corresponding set $A \subseteq E$ such that $L(A) = L$. Thus, searching over subsets of E can be viewed as searching over possible layer assignments.

Using the above definitions we now define the function $F(A) = \sum_i D_i(l_i(A))$, which is simply the total reduction in distortion resulting from the layer assignment $L(A)$. Also define the function $C(A) = \sum_i R_i(l_i(A)) + H(l_1(A), \ldots, l_N(A))$, which equals the total bandwidth, including overhead, required by the layer assignment $L(A)$. It is easy to see that finding the set A that maximizes $F(A)$ subject to $C(A) \leq B$ yields a layer assignment $L(A)$ that optimally solves the layer assignment optimization problem. Given this equivalence we are now ready to apply the tools of submodular optimization by showing that $F(A)$ is a submodular, non-decreasing function.

Lemma 1 *If Assumption 2, diminishing returns, holds then F(A) is a submodular, non-decreasing set function over* E.

Proof: The fact that $F(A)$ is non-decreasing follows from the fact that the reduction in distortion functions D_i never decrease with the layer index and that adding elements to A never decreases $l_i(A)$. To show submodularity, consider any two sets A and A' such that $A \subseteq A'$. Now consider any element $x_{i,j} \in E$ and let $l = l_i(A \cup \{x_{i,j}\})$ and $l' = l_i(A' \cup \{x_{i,j}\})$. We can have either one of two cases. If $l_i(A) = l_i(A')$ is true then we know that $l = l' = l_i(A) + 1$ and hence the increase in F when adding $x_{i,j}$ to A and A' is identical, which satisfies the submodularity property. Otherwise, since $A \subseteq A'$ we must have that $l_i(A) < l_i(A')$. We get that the increase in F for A and A' is $F(l_i(A)+1) - F(l_i(A))$ and $F(l_i(A')+1) - F(l_i(A'))$ respectively, which by Assumption 2 shows that the the increase for A must be at least as large as for A'. ■

In addition to F being submodular and non-decreasing, we also have that C monotonic, since the bandwidth must increase as we allocate more layers. Thus, the assumptions required for *Double-Greedy** and *All-Triples-Greedy** to achieve their approximation bounds are satisfied by the layer allocation problem. In particular, these algorithms will return an A that approximately optimizes F subject to $C(A) \leq B$ with a factor given in the above theorems. The resulting set A corresponds exactly to a layer allocation $L(A)$ that achieves this same approximation bound for the layer allocation problem. . Note that in our experimental results we only run the *Double-Greedy** algorithm as the results of this algorithm were very close to optimal and so did not justify the increased runtime of the *All-Triples-Greedy** algorithm.

VI. Simulation Results

In this section, we provide a comprehensive evaluation of the proposed optimized framework for video streaming in single-hop networks. In particular, the simulations provide the visual quality of video streams, as measured by Mean Square Error (MSE), when admission control is employed in conjunction with different layer allocation algorithms and the proposed MAC protocol. The layer allocation algorithms of interest are the optimal, the equal rate, the greedy, and the double greedy algorithms. We intentionally omit the results for the triple greedy algorithm since in our simulations, they are observed to be identical to those of the double greedy algorithm. This suggests that perhaps for the bit rates and distortion levels of typical video layers, the double greedy algorithm is sufficient to obtain a good solution.

We note that the optimal algorithm employs an exhaustive search scheme. That is, it examines all the possible combinations of video layers and chooses the one that results in the lowest distortion, i.e., smallest MSE that satisfies the bandwidth and distortion constraints. Thus, the optimal algorithm is prohibitively expensive when the numbers of video layers and hosts are large. The optimal algorithm, however produces the smallest MSE, and is thus used to evaluate the goodness of other algorithms. The equal rate algorithm allocates an equal amount of bandwidth to every video (hosts), layer by layer in a round robin fashion until the constraint on total used bandwidth is no longer satisfied. The greedy and double greedy algorithms are previously described in Section V-A and V-B, respectively.

In all our simulations, we use two sets of standard video profiles, each set consists of three layered videos, as

shown in Tables I and II [7],[33],[34],[35],[36]. Depending on the scenarios, a simulation may use either one or both sets of the video profiles.

TABLE I

STANDARD VIDEO PROFILES - SET I

LAYER	AKIYO			COASTGUARD			FOREMAN		
	Bit Rates (kbps)	Distortion (MSE)	Reduction in Distortion (MSE)	Bit Rates (kbps)	Distortion (MSE)	Reduction in Distortion (MSE)	Bit Rates (kbps)	Distortion (MSE)	Reduction in Distortion (MSE)
1	64	–	–	32	123.90	–	64	–	–
2	128	83.77	–	112	103.06	20.84	128	71.30	–
3	192	63.54	20.23	160	87.72	15.34	192	56.63	14.67
4	256	50.48	13.06	208	78.18	9.54	256	46.03	10.60
5	320	38.29	12.19	256	71.30	6.88	320	39.18	6.85
6	384	32.59	5.70	304	65.03	6.27	384	33.35	5.83
7	448	27.74	4.85	352	57.95	7.08	448	29.05	4.30
8	512	23.61	4.13	400	51.65	6.30	512	25.89	3.16

TABLE II

STANDARD VIDEO PROFILES - SET II

LAYER	FOREMAN 1 (FGS-temporal scalability mode)			COASTGUARD (FGS)			FOREMAN 2 (FGS-AFP mode)		
	Bit Rates (kbps)	Distortion (MSE)	Reduction in Distortion (MSE)	Bit Rates (kbps)	Distortion (MSE)	Reduction in Distortion (MSE)	Bit Rates (kbps)	Distortion (MSE)	Reduction in Distortion (MSE)
1	140	57.03	–	110	60.68	–	384	42.96	–
2	240	33.05	23.98	160	51.65	9.03	512	31.12	11.84
3	340	21.89	11.16	240	43.96	7.69	640	26.43	4.69
4	440	16.30	5.59	300	38.29	5.67	768	22.55	3.88
5	540	11.46	4.84	360	31.12	7.17	896	19.19	3.36
6	640	8.67	2.79	420	26.48	4.63	1024	17.91	1.28
7	740	7.24	1.43	490	24.16	2.33	1152	15.96	1.95
8	840	5.87	1.37	590	20.56	3.60	1280	14.22	1.74
9	940	4.59	1.28	730	17.50	3.06	1408	13.27	0.95
10				850	14.89	2.61			
11				900	11.56	3.33			

A. Protocol Evaluation

We first compare the performance of the proposed MAC protocol against the standard IEEE 802.11 without admission control [2] and the IEEE 802.11e with admission control. In particular, for the IEEE 802.11e with admission control, we use the mechanism proposed by Banchs et al. [4]. This mechanism is somewhat similar to ours, in the sense that each flow i achieves its throughput by setting the contention window CW_i to an appropriate size. For our proposed MAC, we can approximate $CW_i = 1/p_i$. The fundamental difference, however is in the formulation which leads to two different algorithms, and consequently different behaviors. In the IEEE 802.11e with admission control, Banchs et al. formulated the admission control process as maximizing the total throughput from all the flows subject to the constraint on the relative throughput for each flow. In particular, the algorithm tries to set the values of the contention window for each flow in such a way to maximize $R = R_1 + R_2 + ...R_N$, while

ensuring that $R_1/R_1 = n_1$, $R_2/R_1 = n_2$, ... $R_N/R_1 = n_N$ where N, R_i, and $\{n_1, ..., n_N\}$ are the number of flows, the throughput rate for flow i, and a set of given requirements, respectively. Note that we randomly choose R_1 as a reference flow. As a direct result, the throughput obtained by each flow might be higher than what is specified, especially when the specified aggregate throughput is much smaller than the network capacity. On the other hand, our algorithm produces precisely the specified rate for each flow. As will explained shortly, the ability to precisely control the rate will enable an efficient cross-layer optimization.

Our simulator is a time-driven, packet-based simulator written in MatLab. It is designed to mimic as close as possible to the real operations using all the critical parameters in the IEEE 802.11e protocol. For all the simulations, the parameters specified for Frequency Hopping Spread Spectrum (FHSS) PHY layer with the channel capacity of 1 Mbps are shown in Table III. We note that the MAC_{hdr} for IEEE 802.11e contains 2 bytes for a QoS field in addition to that of IEEE 802.11. We also assume that the processing and propagation delays are zero.

TABLE III

FHSS SYSTEM PARAMETERS FOR IEEE 802.11E USED TO OBTAIN NUMERICAL RESULTS

PARAMETER	**V**ALUE
Packet payload	1500 bytes
MAC header (MAC_{hdr})	36 bytes
PHY header (PHY_{hdr})	16 bytes
RTS	20 bytes+PHY_{hdr}
CTS	16 bytes+PHY_{hdr}
ACK	14 bytes+PHY_{hdr}
Channel capacity (BW)	1 Mbps
Slot time	50 μs
SIFS	28 μs
DIFS	128 μs
RTS timeout	(RTS/BW$\times 8 \times 10^6$)+DIFS μs

We first show the simulation results for a single-hop wireless network consisting of 3 hosts. Each host sends exactly one video to other host over a limited channel capacity (BW) of 1 Mbps. These flows are assumed to be in the same traffic class. The minimum throughput requirements for flows 1 (R_1) and 2 (R_2) are set to 200 kbps and 300 kbps, respectively. The minimum throughput of flow 3 (R_3) increases linearly from 115 kbps to 370 kbps with a step size of 15 kbps. For the IEEE 802.11e with admission control, CW_i's are set according to the admission control algorithm while for the standard IEEE 802.11 without admission control, CW_{min} and CW_{max} are set to 15 and 1023 respectively.

Fig. 2(a)-(c) show the observed throughputs for IEEE 802.11 without admission control, IEEE 802.11e with admission control, and our proposed admission control as a function of the flow 3's throughput. As seen, the standard IEEE 802.11 performs well when the total requested throughput is smaller than the network capacity. Without admission control, however, flow 3 cannot achieve its requested throughput of greater than 320 kbps. On

the other hand, the IEEE 802.11e with admission control performs very well as the throughput for each flow is consistently above its specified minimum requirement. Unlike the IEEE 802.11e, our proposed admission control produces a precise the requested throughput for each flow. As a result, the collision rate (wasted bandwidth) is much smaller than that of the standard and the IEEE 802.11e as shown in Fig. 2(d), even when the total useful throughputs in both schemes are approximately the same. This is an advantage of using the proposed MAC protocol. Not surprisingly, the total bandwidth usage for our algorithm is much smaller than those of other protocols as shown in Fig. 2(e) for a specified set of rates.

We now show the simulation results when applying cross-layer optimization for transmitting 3 video flows. First, to provide some intuitions about the interactions between the proposed MAC protocol and the layer allocation algorithms described in Section V-B. Specifically, we present the simulation results for various quantities, e.g. throughputs, transmission probabilities, when using a simple greedy layer allocation. The simulation parameters are shown in Table III. Since all video streams are in the same traffic class, they use the same $TXOP$ where $TXOP$=CTS+PHY$_{hdr}$+MAC$_{hdr}$+PAYLOAD+ACK+3SIFS+DIFS. We use standard video profile set I in this simulation. Fig. 3(a) shows the average throughputs of different video streams increase with the normalized bandwidth usage. From left to right, each point in the graph represents an additional layer being added to one of the videos according to the greedy algorithm. The rightmost point denotes the final number of layers for each video. Adding a layer to any video on each graph at this point would violate the bandwidth constraint. In other words, with the addition of a new layer, the Algorithm 2 in Section III will not able to find a set of transmission probabilities that satisfies the requested rates for all the videos. We note that, at this point, the total bandwidth usage is 95%, indicating a relatively high bandwidth utilization.

Fig. 3(b) shows the transmission probabilities for each host as a function of normalized bandwidth usage. As expected, as the number of layers increases for each video, their transmission probabilities also increase accordingly to ensure a higher chance for data transmissions. It is interesting to note that the transmission probabilities increase almost exponentially to compensate for roughly linear increase in the overall throughput. Fig. 3(c) shows the corresponding increases percentage of successful slots (over the number of non-data slots) for different video streams, as a direct result of increase in transmission probabilities.

However, as the transmission probabilities increase, the percentage of collision slots also increases substantially as shown in Fig. 3(d). Of course, the percentage of idle slots decreases accordingly. This agrees with our intuition about the proposed MAC protocol. We note that using this MAC protocol, one is able to control the rate of the flows precisely by tuning their transmission probabilities. These rates, in turn, control the visual quality of the video streams. Fig. 3(e) shows the visual quality of the three video streams in terms of MSE as a function of normalized bandwidth usage. In this case, the greedy algorithm which minimizes the total MSE for all the flows given the bandwidth constraint, yields an MSE of 38, 71, and 46 for Akiyo, Coastguard, and Foreman sequences, respectively. Fig. 3(f) shows the actual bandwidth percentage for various packet types. As seen, only minimal bandwidth overhead (2%) is incurred when using the RTS of 36 bytes and packet payload of 1500 bytes.

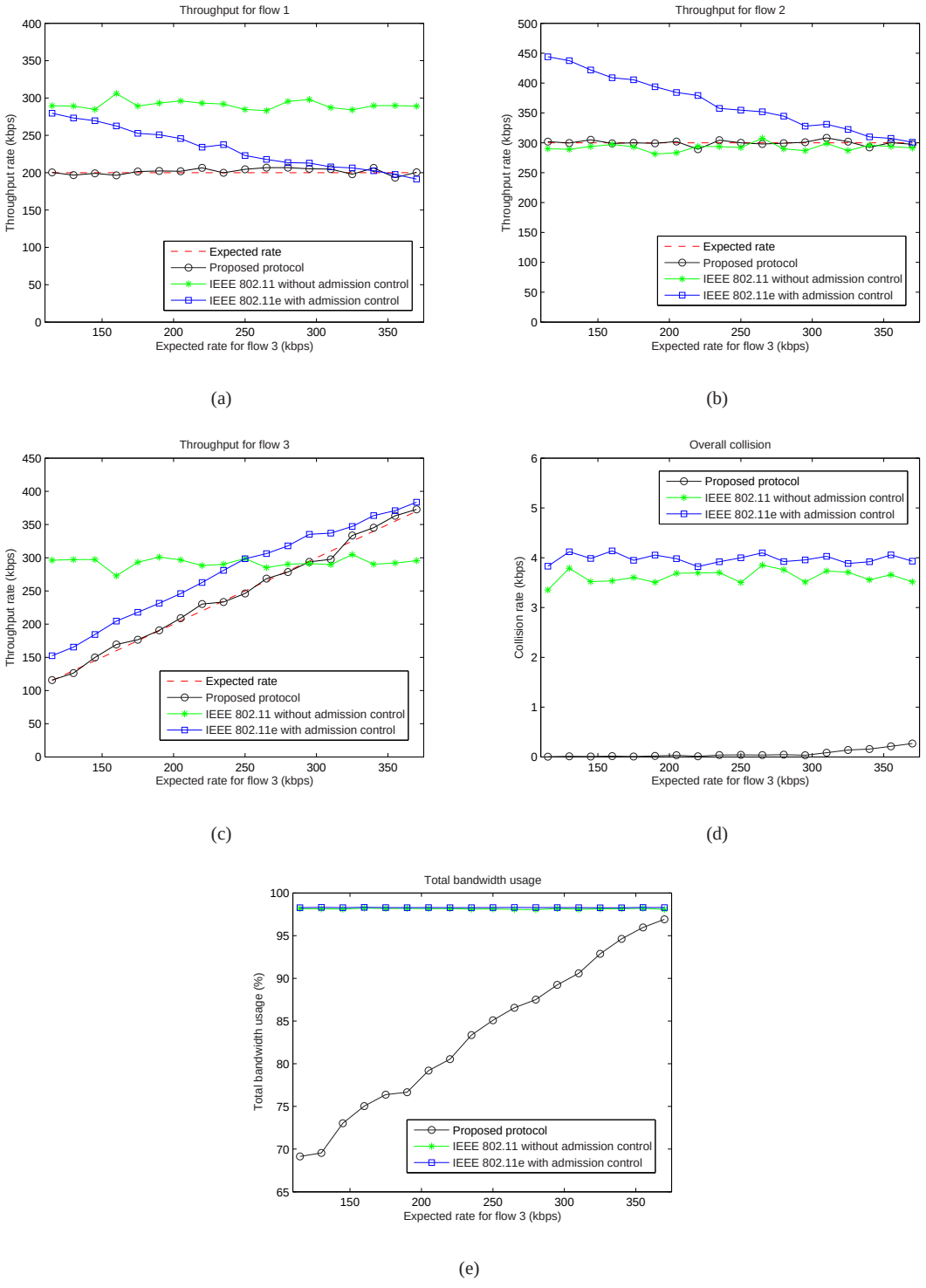


Fig. 2. Proposed protocol validation comparing to IEEE 802.11 and IEEE 802.11e protocols. (a) Throughput for flow 1; (b) Throughput for flow 2; (c) Throughput for flow 3; (d) Overall collision; (e) Overall bandwidth usage.

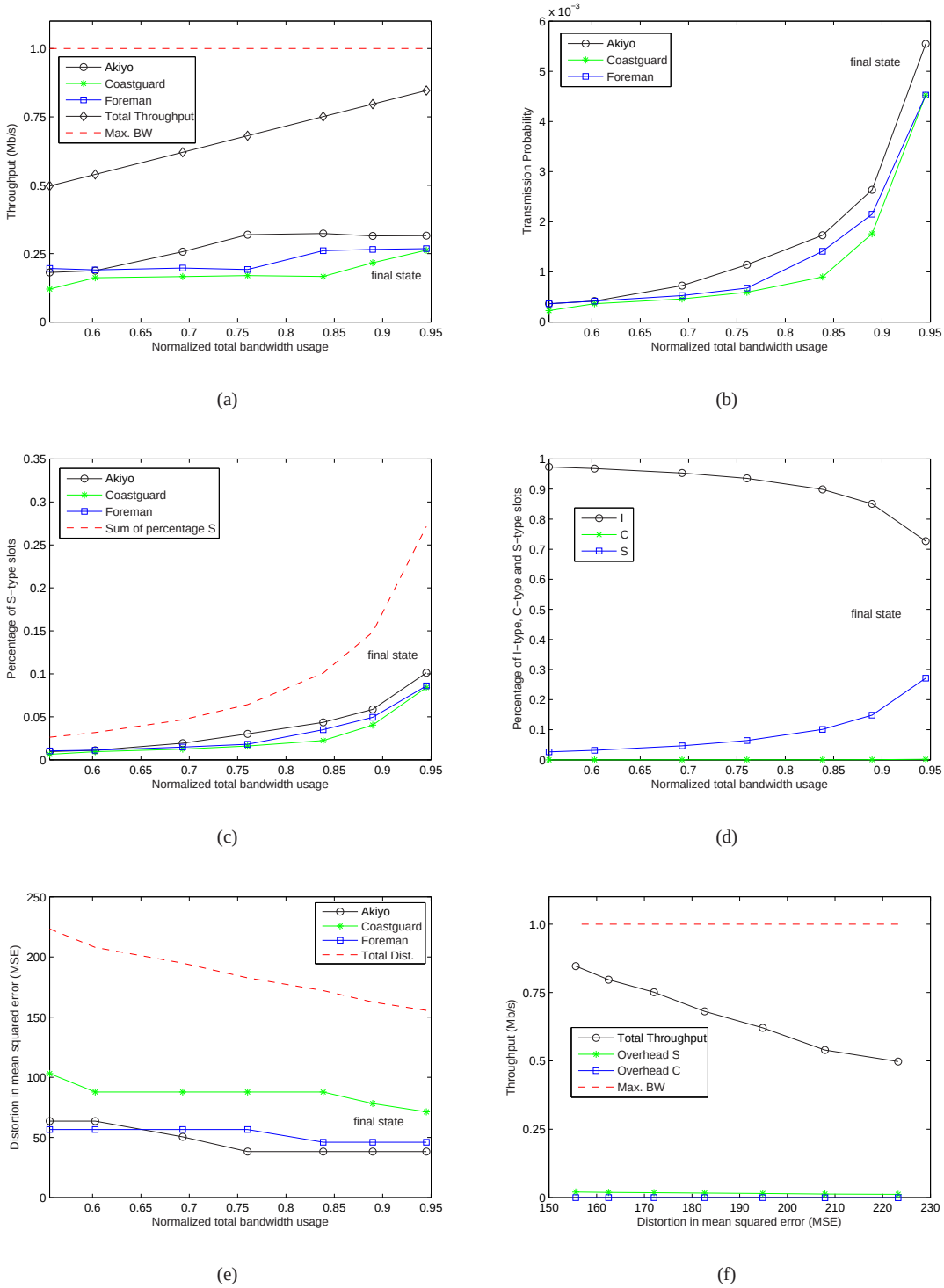
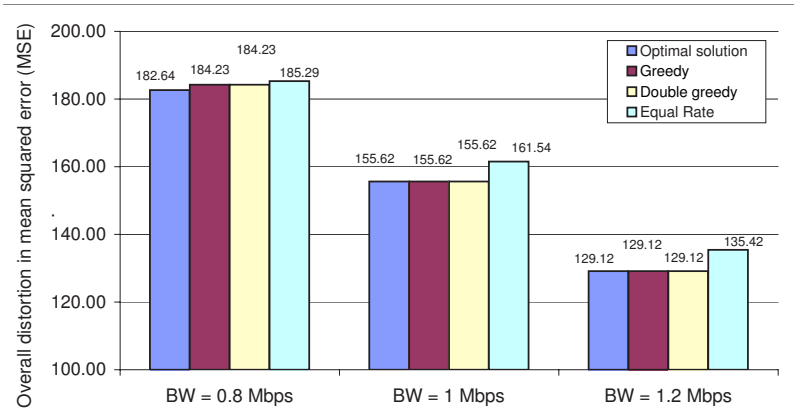
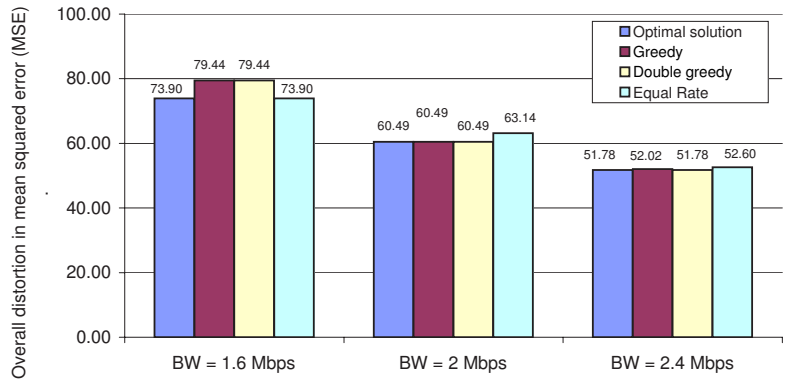


Fig. 3. Performance results using the greedy algorithm. (a) Throughput; (b) Transmission probability; (c) Percentage of S slots; (d) Percentage of I, C, and S slots; (e) Distortion in MSE; (f) Rate-distortion characteristic.



(a)



(b)

Fig. 4. Distortion performances of different algorithms for (a) Video profiles in Table I; (b) video profiles in Table II.

B. Layer Allocation Algorithm Performance

We now show the performance of different layer allocation algorithms. For simplicity, we assume there is no packet loss. Furthermore, by using standard video profiles in Table I ,we require that the distortion levels (MSE) for Akiyo, Coastguard, and Foreman cannot be greater than 63, 103, and 56, respectively. For this simulation, these MSE values are chosen rather arbitrarily, but in practice a user can specify his or her visual quality requirement. Fig. 4(a) and Fig. 4(b) show the distortions resulted from using different algorithms for video profiles in Tables I and II, respectively. For Table II, the maximum distortion requirements for Foreman 1, Coastguard, and Foreman 2 are 21, 51, and 31, respectively. As expected, the optimal (exhaustive search) always produces the lowest distortion, albeit has the highest computational cost.

Fig. 4(a) shows that the performances of the greedy and the double greedy are all identical for video profiles in Table I. At 0.8 Mbps, the greedy and the double greedy algorithms fail to find the optimal solutions. However, the performances of the greedy, the double greedy, and the optimal algorithms are all identical at the capacity channel

of 1 Mbps and 1.2 Mbps. This suggests that greedy and double greedy algorithms perhaps are sufficient in practice. As expected, the equal rate algorithm performs worst since it does not even try to minimize the overall distortion.

On the other hand, Fig. 4(b) shows that the greedy algorithm fails to find the optimal solutions in two instances (BW= 1.6 Mbps and 2.4 Mbps). In contrast, the double greedy algorithm typically finds the optimal solutions. This is, however, not guaranteed as the double greedy algorithm does not converge to optimal solution for BW=1.6 Mbps. We note that as described in Section V-B, the computational cost of the double greedy algorithm is twice that of the greedy algorithm since it must run the greedy and modified greedy algorithms, and picks the best one. The modified greedy algorithm is the basic greedy algorithm, where at each step instead of adding the layer that most decreases the distortion, we add the layer that achieves the largest ratio of the reduction in distortion to the layer bit rate, provided that adding the layer does not violate the bandwidth constraint. The modified greedy algorithm might or might not produce a better solution than that of the greedy algorithm. Both the greedy and modified algorithms can produce an arbitrarily bad solution, while the double greedy algorithm which returns the best solution (lower distortion) of the greedy and modified greedy algorithm, guarantees that its solution is a constant approximation factor to the optimal solution.

Tables IV and V show the detail information associated with different algorithms for the video profiles I and II, respectively. As seen, the optimal algorithm always achieves in the lowest distortion. In all cases, the bandwidth overhead is relatively small, indicating the ability of the framework to utilize the bandwidth efficiently. Note that there are some other overhead bandwidth (i.e., PHY$_{hdr}$, MAC$_{hdr}$, SIFS, DIFS, ACK) that amount to 8 to 10% of the total bandwidth.

TABLE IV

DETAIL INFORMATION ASSOCIATED WITH DIFFERENT ALGORITHMS FOR VIDEO PROFILES I (BW=1.2MB/S)

	Distortion	Normalized Bandwidth Usage (%)					Transmission probability		
	(MSE)	Total	Overhead S	Overhead C	Throughput	Others	AKIYO	COASTGUARD	FOREMAN
Optimal Solution	129.12	98.47	2.11	0.07	87.86	8.43	0.0215	0.0269	0.0215
Greedy	129.12	98.47	2.11	0.07	87.86	8.43	0.0215	0.0269	0.0215
Modified Greedy	129.12	98.47	2.11	0.07	87.86	8.43	0.0215	0.0269	0.0215
Double Greedy	129.12	98.47	2.11	0.07	87.86	8.43	0.0215	0.0269	0.0215
Equal Rate	135.42	93.56	2.00	0.02	83.52	8.02	0.0046	0.0051	0.0046

TABLE V

DETAIL INFORMATION ASSOCIATED WITH DIFFERENT ALGORITHMS FOR VIDEO PROFILES II (BW=2.4MB/S)

	Distortion	Normalized Bandwidth Usage (%)					Transmission probability		
	(MSE)	Total	Overhead S	Overhead C	Throughput	Others	FOREMAN 1	COASTGUARD	FOREMAN 2
Optimal Solution	51.78	95.86	2.01	0.06	83.74	10.05	0.0140	0.0129	0.0165
Greedy	52.02	97.14	2.04	0.08	84.84	10.18	0.0203	0.0155	0.0280
Modified Greedy	51.78	95.86	2.01	0.06	83.74	10.05	0.0140	0.0129	0.0165
Double Greedy	51.78	95.86	2.01	0.06	83.74	10.05	0.0140	0.0129	0.0165
Equal Rate	52.60	96.47	2.02	0.07	84.26	10.12	0.0165	0.0186	0.0165

C. Throughput Jitter Evaluation

The proposed optimization framework guarantees that each flow will achieve its required throughput when it is averaged over a long period of time. However, the throughputs of the flows may fluctuate within a short period of time due to the probabilistic nature of the channel contention access. These throughput fluctuations may prevent smooth playback for many audio and video streaming applications. To alleviate this problem, many streaming applications employ the prebuffering technique in which, the receiver puts the received data into a buffer for a short period of time before starting to playback. A longer buffer results in a smoother playback session. On the other hand, using a larger buffer results in larger initial delay and required memory. Interactive applications such as video conferencing may not tolerate such large delay, and the low power wireless hosts may not have enough memory for buffering. Thus, it is important to characterize the throughput jitter resulting from using the protocol.

Fig. 5(a) shows the ratio of the actual throughputs to the requested throughputs of different flows in video profiles I, averaged over every 300 kbytes. The channel capacity is set to 1 Mbps. Fig. 5(b) shows throughput ratios for different flows in the video profiles II, averaged over every 600 kbytes, with the capacity channel set to 2 Mbps. As seen, the throughput ratios fluctuate around 1, indicating all the flows achieve their required throughputs. The magnitudes of these fluctuations are also small, e.g. 0-20%, suggesting that one can use a small streaming buffer for smooth playback.

In this simulation, we repeatly transmit three videos from video profiles I for 20 minutes. We want to quantify how long a streaming buffer should be in order to prevent lost packets due to late arrival. To prevent throughput fluctuation, we also request a slightly larger bandwidth than the recorded video bit rate. This bandwidth safety margin provides robustness against possible throughput depletion during a session. Fig. 6(a) to Fig. 6(d) show the number of late packets as a function of streaming buffer length for various bandwidth safety margins.

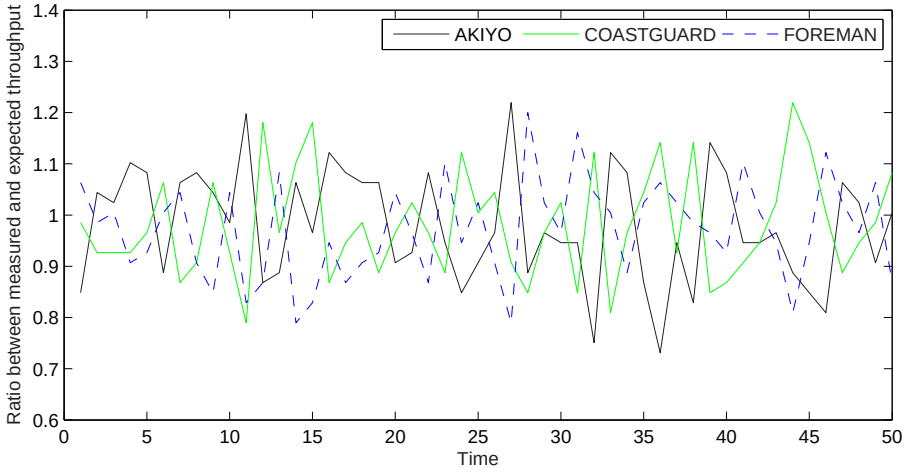
As seen, to have no late packet with no bandwidth safety margin, a user receiving Akiyo needs to wait on average, 48 seconds to smoothly playback a 20-minute video. Users receiving Coastguard and Foreman need to wait up to 88 and 40 seconds, respectively. However the required waiting time reduces with the increase in the bandwidth safety margin. For example, using the bandwidth safety margin of 3%, the waiting times for Akiyo, Coastguard, and Foreman users are reduced to 24, 32, and 12 seconds, respectively.

We also quantify the throughput jitter of a flow as the normalized standard deviation of its fractional throughput within a number of time slots. Specifically,

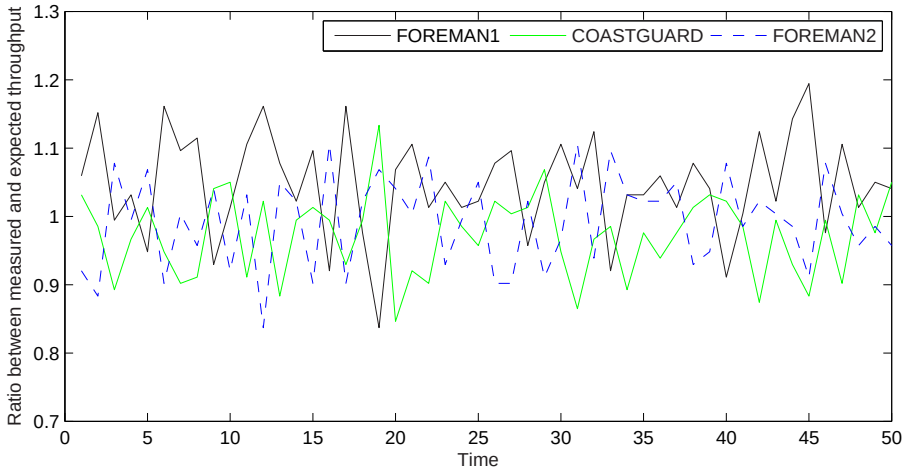
$$Stdev_n(X_i(T)) = \frac{Stdev(X_i(T))}{R'_i}, \tag{28}$$

where $X_i(T)$ is a fraction of the data slots of the flow i measured within T time slots, and R'_i denotes its average long term fractional throughput. Clearly, a larger T should result in a smaller normalized throughput standard deviation since we average the throughput over a longer period of time. It is straightforward to show that

$$Stdev_n(X_i(T)) = TXOP_i \times \frac{\sqrt{(1-\sum_i^N R'_i) \times S_i \times (1-S_i)}}{R'_i\sqrt{T}}, \tag{29}$$



(a)



(b)

Fig. 5. Throughput jitter for each flow for (a) Video profiles in Table I; (b) Video profiles in Table II.

where N denotes the number of flows and S_i' denotes the number of successful time slots for flow i. [3].

To quantify the normalized throughput standard deviation, we simulate three flows with R'_1, R'_2, and R'_3 being set to 0.1, 0.27, and 0.4, respectively. Fig. 7 shows the normalized throughput standard deviations for three different flows as a function of buffer size (T). As expected, as T increases, the normalized throughput standard deviation decreases. However, increasing T implies an increase in playback delay.

[3]This result can be obtained by noticing that the number of successful transmission slots within a period T is binomially distributed with parameters S_i and $T(1-\sum_i^N R'_i)$.

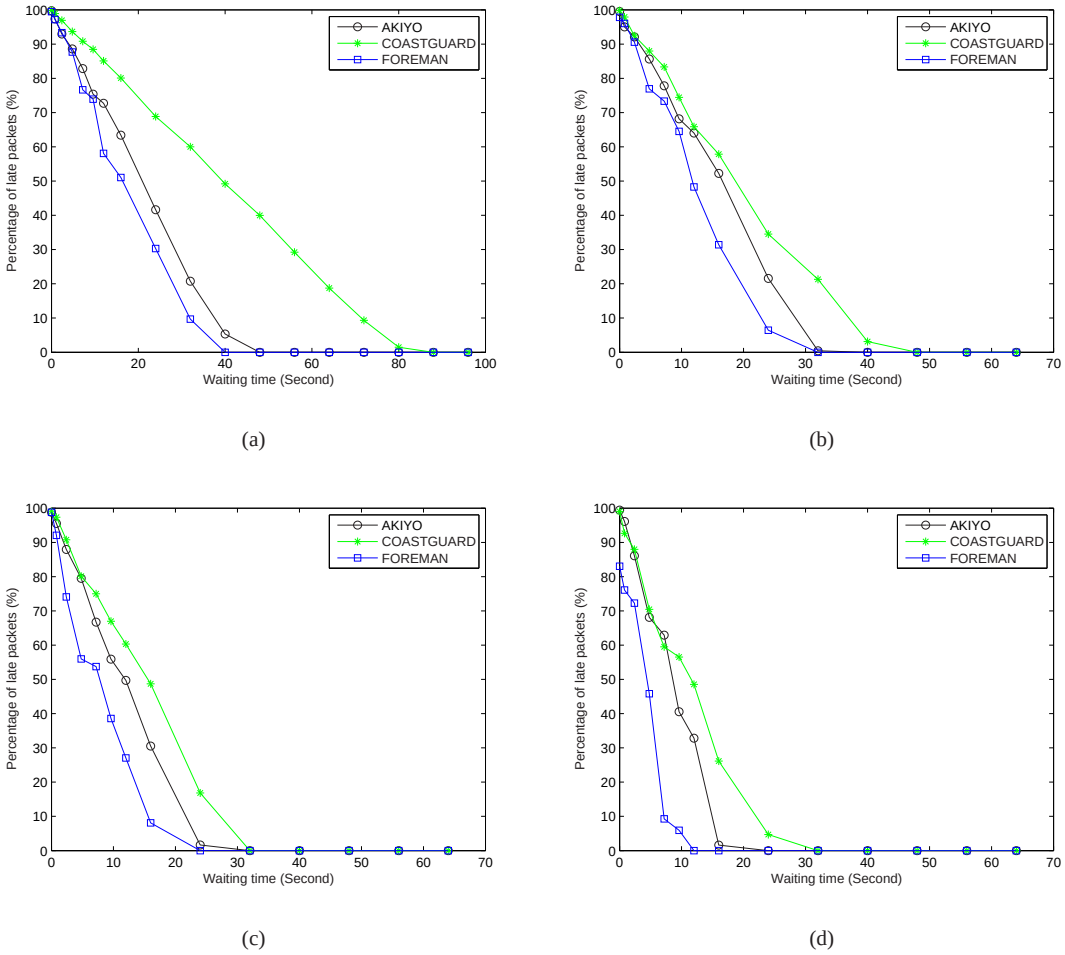


Fig. 6. Percentage of late packets as a function of streaming buffer length (in seconds); (a) 0% bandwidth safety margin; (b) 1% bandwidth safety margin; (c) 2% bandwidth safety margin; d) 3% bandwidth safety margin. The bit rates of Akiyo, Coastguard, and Foreman are 192, 160, and 256kbps, respectively.

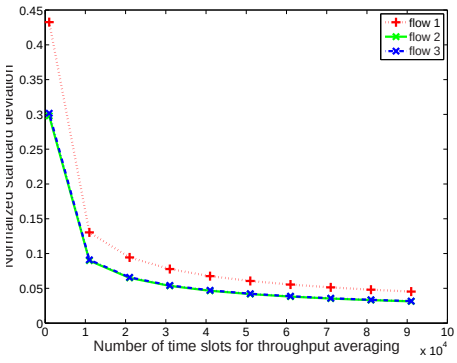


Fig. 7. Normalized throughput standard deviation as a function of buffer size (T).

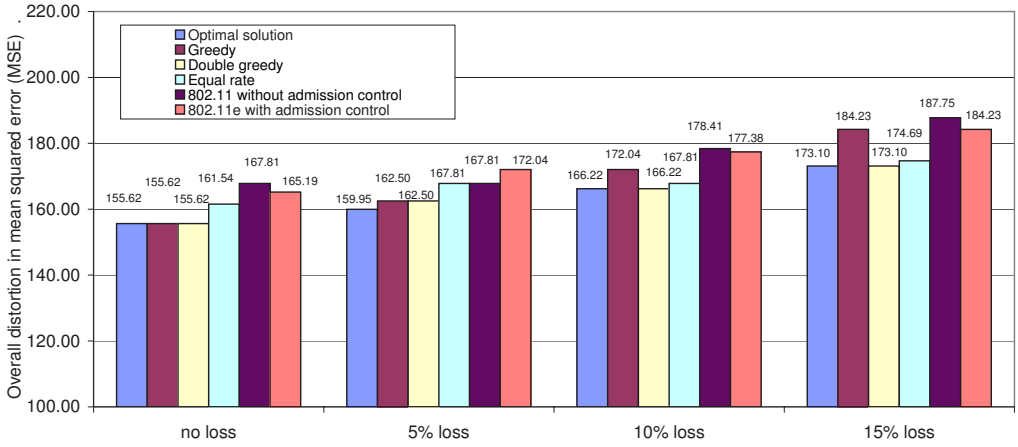
D. Overall Performance

We now compare the performance of our proposed framework against the existing IEEE 802.11 without admission control and IEEE 802.11e with admission control proposed by Banchs et al. [4]. To simulate realistic settings, packet losses are introduced into the simulations. For simplicity, we assume that packet loss rates are identical for all the receivers. We also assume that a packet will be transmitted repeatedly until it is received successfully. As a result, the useful throughput reduces with an increase in packet loss rate. The admission control module is assumed to be able to measure the packet loss rate, and thus can determine the overall effective throughput. Using the overall effective throughput and the video profiles as inputs, it can use different optimization strategies to allocate bandwidths for receivers so as to minimize the average distortion.

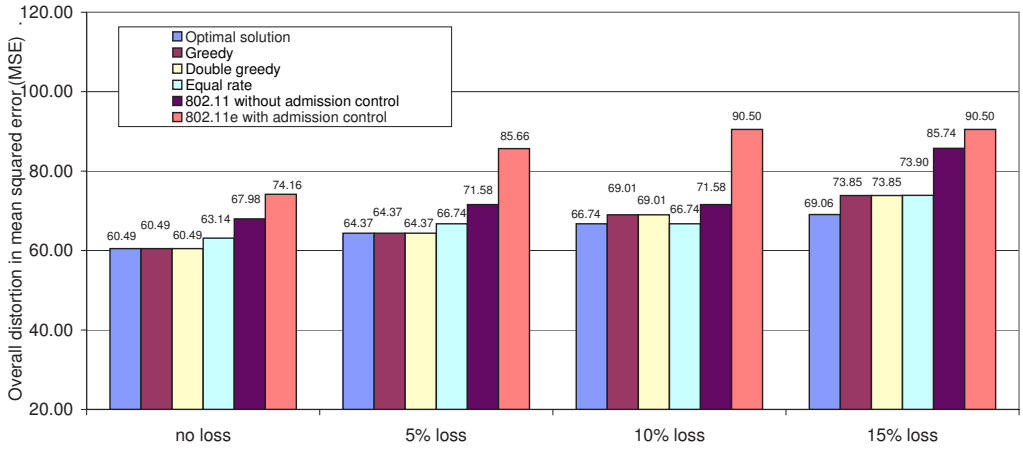
Fig. 8(a) shows the average distortion for various strategies when streaming the videos in profile I. As expected, the optimal strategy, i.e., exhaustive search, always provides the smallest average distortion. The double greedy algorithm also performs equally well. The greedy algorithm performs slightly worse, followed by the equal rate algorithm, the IEEE 802.11 protocol without admission control, and IEEE 802.11e with admission control. The main reason for the worse performances when using the IEEE 802.11 and the IEEE 802.11e is the lack of layer allocation optimization. For the IEEE 802.11, due to the random contention-based access where each flow has an equal chance of accessing the shared channel, all three flows obtain approximately the same throughputs. However, this is not the optimal allocation for minimizing the overall distortion. Intuitively, the IEEE 802.11 should perform as well as the equal rate allocation. On the other hand, the equal rate allocation scheme uses the proposed admission control which results in smaller collision bandwidth as compared to that of IEEE 802.11. As a result, the average distortion of the equal rate scheme is typically lower than that of the IEEE 802.11.

However, admission control mechanism alone does not improve the quality of the video. To see this, let us consider the performance of the IEEE 802.11e with admission control. Using this scheme, one is able to guarantee the minimum throughput rate for each flow. However, if after providing these minimum throughputs for the flows, there is still much bandwidth available, then as designed, the IEEE 802.11e protocol would allow each flow to increase its throughput proportionally until the wireless capacity is reached. The final throughput of each flow as obtained by the admission control algorithm then dictates the quality of a video. In other words, the layer allocation employed in this case is to allocate the rate proportionally according to the initial conditions. For example, if the minimum rates (or equivalently maximum distortions) for three flows are specified initially 200 kbps, 400 kbps, and 400 kbps, then with a channel capacity of 2 Mbps, the IEEE 802.11e protocol will roughly allocate 400 kbps, 800 kbps, and 800 kbps for these flows (for sake of illustration, we assume no collision bandwidth). Clearly, this allocation is not optimal as it does not take into the account the distortion profile for each video. That said, there is no reason why the IEEE 802.11e with admission control should perform better than the IEEE 802.11 or other schemes when there is enough bandwidth. In fact, Fig. 8(b) shows that the IEEE 802.11e results in a larger average distortion consistently when using video profiles II. Our proposed framework which integrates the admission control with layer allocation optimization enables us to achieve the lowest distortion. Overall, our proposed framework improves

the video quality up to 26% over that of a typical IEEE 802.11 based network.



(a)



(b)

Fig. 8. Distortions resulted from using various protocols for (a) Using video profiles I with channel capacity set to 1 Mbps; (b) Using video profiles II with channel capacity set to 2 Mbps.

VII. Conclusions

We have proposed a framework to enhance the quality of video streaming applications in wireless home networks via a joint optimization of video coding technique, admission control algorithm, and MAC protocol. Using an Aloha-like MAC protocol, our admission control framework which can be viewed as an optimization problem that maximizes the average quality of admitted videos, given a specified minimum video quality for each flow. We provided some hardness results for the optimization problem under various conditions, and proposed two heuristic algorithms for obtaining a good solution. In particular, we showed that a simple greedy layer-allocation algorithm can perform reasonable well, although it is typically not optimal. Consequently, we presented a more expensive

heuristic algorithm that guarantees to approximate the optimal solution within a constant factor. Simulation results demonstrated that our proposed framework can improve the video quality up to 26% as compared to those of the existing approaches.

REFERENCES

[1] E. Setton, X. Zhu, and B. Girod, "Congestion-optimization scheduling of video over wireless ad hoc networks," *IEEE int. conf. circuits and systems*, pp. 3531–3534, May 2005.

[2] *Wireless LAN medium access control (MAC) and physical later (PHY) specifications*, IEEE std. 802.11 Std., 1999.

[3] *Wireless LAN medium access control (MAC) and physical later (PHY) specifications*, IEEE std. 802.11e Std., 2005.

[4] A. Banchs, X. Perez-Costa, and D. Qiao, "Providing throughput guarantees in ieee 802.11e wireless lans," *18th International teletraffic congress*, Sep. 2003.

[5] M. Ergen and P. Varaiya, "Throughput analysis and admission control for ieee 802.11a," *Mobile Networks and Applications 10*, pp. 705–716, 2005.

[6] T. Nguyen, K. Nguyen, , and L. He, "Collaborative distributed admission control (cdac) for wireless networks," *CDS*, p. 75, Jun. 2007.

[7] W. Li, "Overview of fine granularity scalability in mpeg-4 video standard," *IEEE trans. on circuits and systems for video tech*, vol. 11, pp. 301–316, Mar. 2001.

[8] P. Tang and T. Tai, "Network traffic characterization using token bucket model," *INFOCOM*, vol. 1, pp. 51–62, Mar. 1999.

[9] L. Kondi, F. Ishtiaq, and A. Katsaggelos, "On video snr scalability," *IEEE Int. conf. on image processing*, vol. 3, pp. 934–938, Oct. 1998.

[10] J. Chakareski and P. Frossad, "Rate-distortion optimized packet scheduling of multiple video streams over shared communication resources," *IEEE trans. on multimedia*, vol. 8, pp. 207–218, Apr. 2006.

[11] S. Wenger, "Temporal scalability using p-pictures for low latency applications," *IEEE 2nd workshop on multimedia and signal processing*, pp. 559–564, Dec. 1998.

[12] W. Zhou, D. Sarkar, and S. Ramakrishnan, "Traffic models for mpeg-4 spatial scalable video," *GLOBECOM*, vol. 1, pp. 256–259, Nov. 2005.

[13] A. Lindgren, A. Almquist, and O. Schelen, "Evaluation of quality of service schemes for ieee 802.11," pp. 348–351, Nov. 2001.

[14] H. Zhu, M. Li, I. Chlamatac, and B.Prabhakaran, "Survey of quality of service in ieee 802.11 networks," *IEEE wireless comm.*, vol. 11, pp. 6–14, Aug. 2004.

[15] I. Aad, Q. Ni, C. Castelluccia, , and T. Turletti, "Enhancing ieee 802.11 performance with slow cw decrease," *IEEE 802.11e working group document*, Nov. 2002.

[16] L. Romdhani, Q. Ni, and T. Turletti, "Adaptive edcf: enhanced service differentiation for ieee 802.11 wireless ad hoc networks," *WCNC*, vol. 2, pp. 1373–1378, Mar. 2003.

[17] V. Kanodia, C. Li, A. Sabharwal, B. Sadeghi, and E. Knightly, "Ordered packet scheduling in wireless ad hoc networks: mechanisms and performance analysis," *ACM int. symposium on mobile ad hoc networking and computing*, pp. 58–70, 2002.

[18] E. Setton, T. Yoo, X. Zhu, A. Goldsmith, and B. Girod, "Cross-layer design of ad hoc networks for real-time video streaming," *IEEE wireless comm.*, pp. 59–65, Aug. 2005.

[19] M. Li and B. Prabhakaran, "Mac layer admission control and priority re-allocation for handling qos guarantees in non-cooperative wireless lans," *Mobile networks and applications*, vol. 10, pp. 947–959, Dec. 2005.

[20] D. Gao, J. Cai, and K. N. Ngan, "Admission control in ieee 802.11e wireless lans," *IEEE network*, vol. 19, pp. 6–13, Jul. 2005.

[21] Y. Xiao and H. Li, "Evaluation of distributed admission control for the ieee 802.11e edca," *IEEE radio comm.*, vol. 42, pp. s20–s24, Sep. 2004.

[22] A. Bai, B. Selvig, T. Skeie, and P. Engelstad, "A class based dynamic admitted time limit admission control algorithm for 802.11e edca," in *6th International workshop on applications and services in wireless networks*, Berlin, Germany, May 2006.

[23] M. Barry, A. Campell, and A. Veres, "Distributed control algorithms for service differentiation in wireless packet networks," *INFOCOM*, vol. 1, pp. 582–590, Apr. 2001.

[24] S. Shah, K.Chen, and K. Nahrstedt, "Dynamic bandwidth management for single-hop ad hoc wireless networks," *PERCOM*, pp. 195–203, Mar. 2003.

[25] S. Valaee and B. Li, "Distributed call admission control for wireless ad hoc networks," *VTC*, vol. 2, pp. 1244–1248, Sep. 2002.

[26] D. Pong and T. Moor, "Call admission control for ieee 802.11 contention access mechanism," *GLOBECOM*, vol. 1, pp. 174–178, Dec. 2003.

[27] C. Fullmer and C. Garcia-Luna-Aceves, "Solutions to hidden terminal problems in wireless networks," *ACM SIGCOMM Review*, vol. 27, pp. 39–49, Oct. 1997.

[28] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack Problems*. Heidelberg:Springer-Verlag, 2004.

[29] A. Goldberg and A. Marchetti-Spaccamela, "On finding the exact solution of a zero-one knapsack problem," *ACM symposium on theory of computing*, pp. 359–368, Dec. 1984.

[30] G. Nemhauser, L. Wolsey, and M. fisher, "An analysis of the approximations for maximizing submodular set functions," *Mathematical Programming*, pp. 14:265–294, 1978.

[31] M. Sviridenko, "A note on maximizing a submodular set function subject to knapsack constraint," *Operations Research Letters*, pp. 32:41–43, 2004.

[32] A. Krause and C. Guestrin, "A note on the budgeted maximization of submodular functions," *Technical report, CMU-CALD-05-103*, 2005.

[33] X. Sun, F. Wu, S. Li, W. Gao, and Y. Zhang, "Macroblock-based progressive fine granularity scalable (pfgs) video coding with flexible temporal-snr scalabilities," *IEEE Int. conf. on image processing*, vol. 2, pp. 1025–1028, Oct. 2001.

[34] F. Wu, S. Li, R. Yan, X. Sun, and Y. Zhang, "Efficient and universal scalable video coding," *IEEE int. conf. image processing*, pp. 37–40, Sep. 2002.

[35] H. Radha, M. Schaar, and Y. Chen, "The mpeg-4 fine-grained scalable video coding method for multimedia streaming over ip," *IEEE trans. on multimedia*, pp. 53–68, Mar. 2001.

[36] S. Chen, C. Chang, and C. Lin, "Mpeg-4 fgs coding performance improvement using adaptive inter-layer prediction," *ICASSP*, vol. 3, pp. 265–268, May 2004.