Scalable WiFi Media Delivery through Adaptive Broadcasts

Sayandeep Sen, Neel Kamal Madabhushi, Suman Banerjee
University of Wisconsin-Madison

Abstract

Current WiFi Access Points (APs) choose transmission parameters when emitting wireless packets based solely on channel conditions. In this work we explore the benefits of deciding packet transmission parameters in a content-dependent manner. We demonstrate the benefits specifically for media delivery applications in WiFi environments by designing, implementing and evaluating a system, called *Medusa*. In order to keep the APs relatively simple, we implement the *Medusa* functions in a media-aware proxy. More specifically, when forwarding our media traffic, *Medusa* requires that APs simply use the WiFi broadcast feature, and that they refrain from making decisions on which wireless packets to retransmit, or what PHY rates such packets should be transmitted at. Instead we combine these typical link layer functions with a few other content-specific choices, in the proxy. Through detailed experiments across diverse mobility and interference conditions we demonstrate the advantages of this scheme for both unicast and multicast media delivery applications. The advantages are particularly substantial in multicast scenarios, where *Medusa* was able to deliver a 20 Mbps HD video stream simultaneously to 25 clients, using a single 802.11 AP, with good to excellent PSNR.

1 Introduction

Robust delivery of rich media content over wireless links is an increasingly important service today. As more and more high quality media content becomes available through the Internet, the user expectation of accessing such content over their wireless enabled devices continue to grow. Examples include users watching on-demand shows and movies from sources such as Hulu and NetFlix, students in a university campus interested in following online lectures while sitting in the cafeteria, and employees in a company watching a company-wide presentation by their CEO, whether at home, at work, or while sitting in a coffee shop. While the widely deployed WiFi technology can provide adequate performance for relatively lower quality media streams, the user experience when watching high quality streams (e.g., HD quality content) leaves much to be desired. In this paper, we attempt to push the envelope of media delivery performance for WiFi systems, by exploiting some knowledge of media content in making transmission parameter selection at the wireless transmitter (APs). While our proposed approach applies equally to unicast as well as to multicast scenarios, the biggest advantage of the system arises in the multicast case where multiple users are interested in the same content.

A target campus application and challenges: The IT department of the UW-Madison campus is interested in providing high quality broadcast of specific educational content through its intranet. Such capability would allow students sitting in dormitory rooms, in union buildings, in cafeterias, and in libraries, to follow the classroom. Further, the system would also allow easy and convenient dissemination of live guest lectures from remote locations, without requiring the guest lecturer to visit the campus. While the wired backhaul has sufficient capacity to carry such media traffic, initial experiments by the IT department revealed obvious performance problems on the WiFi based last hop. In particular, they observed that even if 3 users connected to a single 802.11g AP attempted to watch the *same* HD video stream, the performance of the system was abysmally poor [1].

The poor performance of media delivery over WiFi for multiple users requesting the same content, is a combined effect of three factors: (i) HD quality video places a high bandwidth demand on the wireless medium — commercial HD encoders, such such as the Streambox SBT3-9200 [6], create content with data rates ranging from 512 Kbps to 30 Mbps, (ii) WiFi typically employs 802.11 unicast mode for sending similar content separately to each user, and (iii) the WiFi transmitter makes various configuration choices, e.g., PHY transmission rates, number of re-transmission attempts, etc., for each wireless packet, without any knowledge of the relevance of its(packet's) contents to end applications. In this paper, we design and implement a system called *Medusa* — Media delivery using adaptive (pseudo)-broadcasts, that can efficiently address issues (ii) and (iii).

1.1 *Medusa* approach

The 802.11 transmitter typically transmits packets in the FIFO order and makes multiple decisions for each wireless packet transmission. This include channel contention, i.e., when to attempt wireless transmission, selection of the PHY rate for the packet, and the number of re-transmission attempts to make in case of failures. In this paper we contend that many of these decisions, namely PHY rate selection, number of re-transmission

[1] An 802.11n AP can potentially scale performance to upto 10-12 users watching such HD content simultaneously. The typical number of users in busy parts of campus is often much higher.

attempts for each packet, and the order of packet transmissions, are better made by taking the "value" of the wireless packet to the application into account as well. Let us consider MPEG-encoded [4] video content that consists of I-, P-, and B- video frames. Given that a packet carrying I-bits is more important, the PHY rate of such a packet can be picked more conservatively, than value-unaware rate adaptation algorithms, e.g., SampleRate [8]. This would ensure that the loss probability of packets carrying I-bits are particularly low. Similarly, if the wireless channel capacity is scarce and packet errors are high, then it is more important to devote greater re-transmission effort for packets carrying I-bits, than packets carrying P- and B-bits.

Hence in *Medusa*, we offload these decisions for our media traffic to a media-aware proxy that can interpret the value of the data. More specifically, APs in *Medusa* no longer perform link-layer re-transmissions or PHY rate selections. Instead, the proxy examines the value of each packet to applications, and instructs APs to (re)transmit these packets in a certain order and at specified PHY rate.

Prior work, e.g., Trantor [21], has considered a model in which a centralized controller decides transmission parameters, e.g., PHY rates, transmit power, etc. for different APs and clients in an entire enterprise WLAN. At a high level, our proposal of proxy-based PHY rate selection and re-transmissions may appear similar. However, there is a fundamental difference between the two proposals. Trantor suggests a centralized rate selection in a content-agnostic manner, based solely on potential interferences between different conflicting wireless links. The approach in *Medusa* augments this decision by incorporating knowledge about *value of the packet contents to applications* in deciding the PHY rate of packets, the order in which they should be transmitted, and the number of re-transmission attempts to be made. It also optionally utilizes simple network coding approaches [23] to improve efficiency.

Unicast vs broadcast vs pseudo-broadcast: In a scenario where multiple users are requesting the same content (the same media stream, for example), we advocate the use of 802.11 standard's broadcast mode of operation. The choice is motivated by the observation that a 802.11 broadcast packet can be used to communicate content simultaneously to all receivers. This would substantially reduce the load on the wireless medium. However, MAC-layer broadcast packets do not elicit MAC-layer acknowledgments from receivers, leaving the transmitter unaware of losses on the wireless channel. This is a problem for any broadcast-based wireless system, as the transmitter can no longer decide which packets require re-transmissions. Further, absence of loss information makes it impossible to determine an appropriate

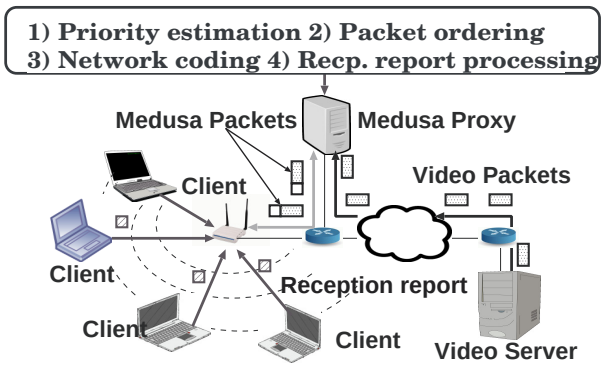


Figure 1: Schematic of the *Medusa* shared media delivery system.

PHY rate adaptation mechanism. Hence, we incorporate higher layer acknowledgments from the clients to the proxy, that help the latter in making these decisions for the APs in a content-dependent manner.

In a single user scenario, we could use 802.11 unicast transmissions. However, even in such settings, it is possible to exploit ER-style network coding opportunities when transmitting wireless packets [23]. Such network coded packets, by definition, have multiple intended recipients. Broadcast-based 802.11 packets are suitable to facilitate this enhancement as well.

Lack of MAC-layer acknowledgments is problematic, however, for one more link layer decision that has to be taken by the APs — channel contention. Lack of acknowledgments indicating packet losses, prevent APs from inferring how the MAC layer backoff counters should be adjusted. Given the lack of MAC-layer acknowledgments for broadcast packets, the existing methods for channel contention are likely to fail. Hence, we use pseudo-broadcast packets in *Medusa* to communicate all broadcast media content, analogous to what was proposed in COPE to deliver network coded multicast data [18]. In pseudo-broadcast, 802.11 unicast mode is used where one of the receivers is picked at random to be the explicitly stated (unicast) recipient, while other intended recipients simply overhear the packet. MAC-layer acknowledgments arrive from the explicit recipient, and backoff parameters can be appropriately adjusted. Note that these MAC-layer acknowledgments are interpreted by the APs solely for the backoff adjustment function, and not used by the proxy to decide PHY rate, transmission order, or eligibility for re-transmission of packets.

***Medusa* system overview:** Figure 1 shows a schematic of different aspects of the *Medusa* system, including an unchanged media server, a media-aware *Medusa* proxy, and APs and clients with minor software-level changes. The *Medusa* proxy intercepts all IP packets corresponding to various video frames and relays

them to the AP for transmission. The proxy instructs the AP to use 802.11 pseudo-broadcast for each wireless packet (irrespective of whether it is part of a multicast or a unicast stream) and also informs the AP what specific PHY rate to transmit the packet at. The *Medusa* proxy makes these decisions, using a combination of four mechanisms: *(i) WiFi reception reports:* Each client provides a periodic reception report to the proxy about various wireless packets that it did or did not receive. While analogous to Reception Reports in RTCP [24], the reception reports in *Medusa* differs from those in RTCP in the detailed MAC layer information that is carried in *Medusa* for rate adaptation and re-transmission purposes. *(ii) Estimating the value of a packet to media applications:* Not all packets are of equal importance to receivers. When the wireless channel capacity is scarce, the value of each packet determines the choice of PHY rate and the number of re-transmission attempts to be made to deliver the packet. We use a simple per packet value assignment function at the *Medusa* proxy to determine the priority level of each packet encapsulating a media stream. *(iii) PHY rate adaptation and re-transmissions for broadcast packets:* We design a PHY rate adaptation and re-transmission strategy for broadcast wireless packets that cannot depend on MAC-layer acknowledgments. Our rate adaptation scheme is *two-paced*. A conservative baseline PHY rate is identified at a slow timescale, and an individual PHY rate for each packet is chosen subsequently using an algorithm called *Inflate-Deflate. (iv) Packet order selection and network coded re-transmissions:* We modify the ordering of media packets from traditional FIFO, especially when the channel quality is poor. Under bad channel conditions, it is beneficial to prioritize packet transmission based on packet "value". This would increase the probability of successful reception of important packets. Further, the transmission order is also selected such that there are proxy-initiated re-transmission opportunities for the more important packets. During re-transmissions, we also leverage the gains of ER-style network coding.

Key contributions: Summarizing, the key contributions of this work is two-fold:

(i) We propose an intuitive design of a pseudo-broadcast based WiFi system for media delivery at high video rates. The design incorporates various aspects of rate adaptation, re-transmission, and packet priorities, coupled with higher-layer feedback.

(ii) We integrate all of these ideas together into a functional *Medusa* system and present detailed evaluation of this system. Our results show that *Medusa* provides robust, high-bandwidth (upto 20 Mbps), HD quality media delivery to tens of co-located WiFi users interested in the same content, all sharing the same WiFi AP, across a range of channel conditions and mobility scenarios. Our technique is applicable to unicast media delivery scenarios as well.

2 *Medusa* design overview

We describe the design of *Medusa* by using the example of MPEG4-encoded [4] video delivery over wireless. In MPEG4 the video content is partitioned in an independently decodable sequence of pictures, called Group of Pictures (GOP). Each GOP has frames of three different types: I, P and B. Each frame, in turn are broken down into multiple packets which are then transmitted over wireless channel.

At the receiver, the I frames can be correctly decoded, as long as all constituent packets are received. For decoding P packets, the successful reception of previous I or P frame in sequence is also necessary. Finally, to decode a B frame, the previous as well as the next I or P frame in sequence are needed. Put another way, an I frame does not depend on any other frame, while P frames depend on another frame and B frames depend on two other frames.

As described, *Medusa* involves an unchanged media server, a media-aware proxy, and APs and clients, with small software modification. The only change in the AP is to create a single functionality — for each packet forwarded by the proxy to the AP, the proxy should be able to specify the PHY rate of transmission. The AP simply accepts each such packet (which can be an original packet, a previously transmitted packet, or a network coded packet) and transmits them using the pseudo-broadcast mode using the PHY rate specified by the proxy. The AP continues to perform back-off decisions independently based on MAC-layer acknowledgments received for its pseudo-broadcasts. The client is modified to incorporate WiFi reception reports targeted to the proxy. These reception reports include a higher layer acknowledgment (ACK) bitmap, and is sent infrequently by the clients, roughly once every 100 packets or 100 ms. Since the proxy knows the PHY rates at which different packets were transmitted, it can use these reception reports to infer packet losses observed by different clients at different PHY rates.

Based on this simple setup, the design problem of *Medusa* can be stated as,

For a set of k video packets (including both original packets and packets that need re-transmissions), determine the order of transmission and PHY rates for the packets, and pass this information along with the packets to the APs. Further, determine whether some of these packets should be network coded, and whether some of them should be discarded.

We present a particular solution to the above problem in the rest of this section that exploits knowledge of the value of each packet to applications.

While we describe our scheme for MPEG4 video, our approach generalizes to any other media encoding, where the content is structured in layers, and there are different levels of priority (value) for each layer.

2.1 Determining value of packets

As mentioned above successful decoding of video frames might need reception of other video frames. Hence, all else being equal, the value of each video packet depends on how many other packets (or bytes) depend on this packet for correct decoding of various video frames. Naturally, I-frame packets become more important than P- or B-frame packets. The value of video packets is also influenced by its impending playback deadline and that of other dependent video frames. Packets for video frames that are approaching display deadline are more important. Finally, given that many packets are relevant to more than one client (true for original and re-transmitted and network coded packets), the value of a packet should also grow with the number of intended recipients.

Previous research on video encoding for streaming [9, 12, 19, 26] has proposed LP based techniques for determining the priority of video frames. Such techniques typically utilize a directed acyclic graph (DAG) of video frame dependencies along with a (empirical/theoretical) channel error model to determine relative value for frames.

While such sophisticated designs of packet value can certainly be used, in this paper we consider a relatively easy to compute function to determine packet value to applications that illustrates its usefulness in making rate adaptation, re-transmission, packet ordering, and network coding decisions based on the worth of packet contents.

In our scheme, we assign a weight, X, to each video frame, based on how many bytes the frame can help decode (including itself). This weight is given to all constituent packets of this frame. Our media-aware proxy knows the video encoding process, and can calculate X by buffering and observing packets corresponding to each GOP before making transmission decisions. We next assign a weight, C, proportional to the number of intended recipients of each packet. Finally, we assign a third weight, D, based on the delay until the display deadline of this frame. We normalize all these weights to the same scale, and assign the value of the packet to be the product of these normalized weights, thus,

Value $= \bar{X} \times \bar{C}/\bar{D}$.

2.2 Determining a base PHY rate

Our PHY rate selection process is two paced. Initially, we compute a conservative PHY rate for all the packets. If channel capacity is abundant, all packets will simply be transmitted at this rate to enhance the possibility of successful reception. However, if the channel is error prone, then some of these rates will be updated, as described in Section 2.3. The timescale for adapting base PHY rate depends on the reception report frequency of clients (roughly every 100 ms in our current implementation).

We pick the highest PHY rate such that the expected error probability of the packets at all clients would be below a certain threshold (set to a low value) as the base PHY rate. By retaining PHY rate information for all transmitted packets, the proxy, on receiving ACK bitmaps from client, can infer the necessary error rates. In case statistics for certain PHY rates are missing (possibly because no packets are sent at these rates), standard interpolation techniques can be used to estimate the expected error rates.

An important distinction of our broadcast rate assignment from typical 802.11 unicast rate assignment algorithms([8, 30]) is that, it does not favor a higher rate to merely increase the channel utilization. Instead it tries to ensure high reception probability across multiple broadcast receivers (who might have diverse channel conditions).

Another distinction of *Medusa* rate adaptation from unicast stems from inability of *Medusa* to adapt quickly to changed network conditions, due to delayed ACKs. This can result in *Medusa* persisting at high data rate even when channel quality has deteriorated resulting in high errors. To ensure that such a situation does not occur, we update the error characteristic of the PHY rates with a EWMA function with heavy weight on history(thus preventing it from reacting to transient fades of the channel).

Transmitting packets at base PHY rate ensures that the packets have a good likelihood of successful reception. However, if the base rate of the system is too low (say, due to presence of clients with bad channel quality) which limits successful transmission of all video packets before their respective deadlines. Under such circumstances, we selectively increase the transmit rate of different packets in a certain order as described next.

2.3 Packet order and actual PHY rate

The problem of deciding the video packet schedule while maximizing the delivered quality across a group of users is known to be NP-complete [12]. Hence, we use a heuristic algorithm for packet ordering. We now describe our mechanism to determine the transmission order and PHY rate of packets, using an algorithm, we call *Inflate-Deflate*. The heuristic schedules a batch of packets in each round. For ease of exposition we assume the presence of a virtual timeline and our goal is to place packets on this timeline and determine their PHY rate. Placing the packets at a given timeslot signifies *schedul-*

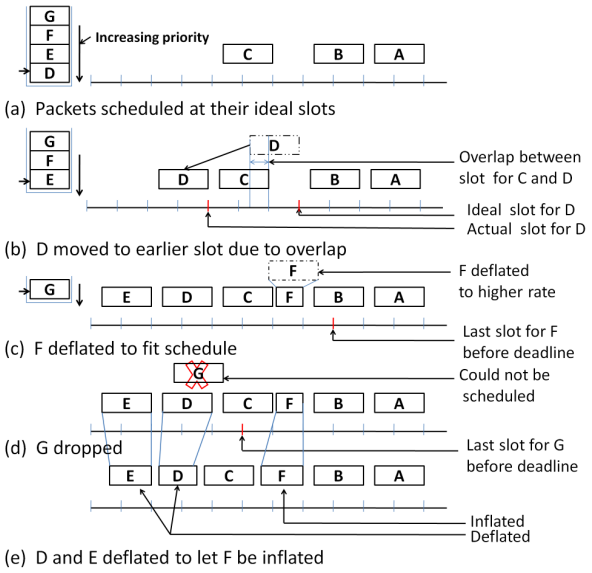


Figure 2: Packet ordering and final rate assignment carried out by *Medusa*.

ing the transmission of the packet at that instant. Packets (including retransmission candidates) are added onto the timeline in the decreasing order of packet value (as calculated in Section 2.1), i.e., higher valued packets are placed first, and lower valued packets later.

We describe our algorithm for ordering packets next and illustrate it with a toy example. We consider A, B, C, D, E, F and G as the seven packets which need to be transmitted (Figure 2). The width of a packet signifies the time required to transmit the packet at its base PHY rate. Initially we try to place each packet at its *ideal* timeslot — a time by which it needs to be transmitted so that it can be re-transmitted multiple times in case of consecutive losses. Also, the latest timeslot at which a packet can be placed is when it just makes its playback deadline for the slowest client, called the *deadline slot* for the packet. The reason for not placing a packet at the earliest available slot, is to ensure that packets which have lower value but have an earlier deadline than the more important packets still get a chance to be scheduled. When the current time is past a packet's deadline slot, and it is not required for decoding any other packet at any receiver, the packet can be discarded. We now walk through the example of how different packets get placed on the scheduling timeline using the following cases.

Case-I (Sufficient time is available to schedule all packets at ideal time-slot): This scenario is depicted in Figure 2(a). Here, packets A, B, and C are scheduled at their ideal timeslot. Also, the packets are assigned their base rates.

Case-II (Ideal slot occupied by a higher valued packet): Under realistic settings, contention from other traffic sources and the necessity to retransmit different packets would mean that scheduling all packets at their ideal slot might not be possible. We depict such a situation in Figure 2(b), in this case scheduling packet D at its ideal slot would lead to an overlap with packet C. We mandate that the packet with the lower value be the one which gets moved around. We find a best fit timeslot in the schedule for the lower valued packet D. Note by considering packets in order of their value implies that only the current packet needs to be moved.

Case-III (No slots left at current PHY rate): In extreme case, we might be unable to find a big enough time slot for a packet, for example in Figure 2(c) we are unable to find a timeslot big enough to fit packet F at its base rate, before its deadline. In such a case, we increase the data-rate of the packet, we call this operation *Deflate* as it results in shortening the packet dimensions on the transmission timeline. For example, we were able to fit F on the timeline after deflating operation. In this operation, we keep trying to find a best-fit timeslot for the packet by increasing its rate. This process continues till we find a slot to fit the packet, or we exhaust all rates without being able to fit the packet on the timeline. This would imply the inability to schedule the packet in the current round. We show this in Figure 2(d). Packet G could not be placed in the timeline even at the highest PHY rate and hence, had to be left out from current iteration.

Case-IV (Compensating for rate optimization): Packets with only a few intended recipients(say, ones with bad channel quality) would have lower value. Thus, such packets would potentially be transmitted at higher rates. This might lead to drastic degradation in received video at clients with bad channel quality. To remedy this, we carry out a round of rate re-assignment before sending out the packets. We call this operation *Inflate* as it decreases the rate of some packets, thus, increasing their size on the transmission timeline. Note that the inflate process does not increase the size of the timeline itself. Inflating is carried out by going through the list of active clients and calculating the expected distortion in video quality, they would suffer if the packets are sent out according to current plan. In case the expected quality of a client falls below a certain minimum threshold, we find out the packet which can increase the expected quality of reception the most and we decrease the rate of transmission of the packet. We then try and compensate by deflating a few other packets which would minimally decrease the quality of video at clients. This process is illustrated in Figure 2(e). In this example packet F is deemed a valuable packet for a client with poor channel quality and hence, its rate is reduced, while the transmission rates of D and E are increased as a compensation.

Note that *Inflate*, might lead to overall reduction in quality of video received over all clients. We keep it in

order to ensure that a minimal quality of video is served to each client.

We would like to note that once this order has been determined, *we do not delay the transmission of packets until the scheduled timeslot*. Instead the packets are sent out at the next transmission opportunity. This ensures that we get even more opportunities to retransmit the packets before its deadline expires. The virtual timeline and time slots are, thus, used only to determine the order of transmissions and the corresponding rates.

An interesting aspect of the PHY rate selection using Inflate-Deflate is that many packets can get transmitted at distinct rates based on the rate assignment algorithm. As a consequence, the proxy can get feedback on a large range of PHY rates from the clients, without having to explicitly raise the base PHY rate. This is another difference between the rate adaptation and error rate estimation technique employed by *Medusa* from unicast rate adaptation techniques such as SampleRate [8].

2.4 Re-transmission planning

We discuss the three inter-related components of re-transmission planning next.

Timeout estimation: A key issue in planning for re-transmissions is to determine the timeouts accurately — under-estimating would lead to redundant packet transmission, while over-estimation would lead to video packets missing their playback deadline. Since each client reception report acknowledges a block of packets, we have to adjust the round trip time and the timeout calculations, to account for additional delays incurred in clients. We adopt a TCP-like Exponentially Weighted Moving Average (EWMA) mechanism for RTT estimation, which takes into account this change. Furthermore, we re-compute the value for all packets that become eligible for re-transmissions, and use this new value to determine the packet transmission ordering and PHY rate.

Network coded re-transmissions: As packet errors at different locations occur independently, multiple clients would potentially (not) receive different packets from a set of consecutive transmissions. This allows us to deploy a simple XOR-based coding [23] of packets to be re-transmitted, to further optimize channel utilization. In our system, we XOR-code a group of packets, only if they satisfy the following rule: *Out of a set of packets to be re-transmitted, if a subset of packets can be found such that each intended recipient of a specific packet has received all other packets in the subset , then the subset can be network coded.* Such coding opportunities occur frequently in the proxy, as MAC-layer re-transmissions are not used in *Medusa* . The algorithm for network coded re-transmissions is shown in Algorithm 2.1.

A key decision in our design is to determine the set of packets to be coded after the packet order has been

Algorithm 2.1: NETCODE(P)

INPUT P: set of coding candidates,
arranged in decreasing order of packet values
$Coding\_set$: Set of packets to be coded
$P_i.client\_set$: Set of clients interested in P_i
OUTPUT S: set of coded packets
for each $P_i \in \mathcal{P}$
  do $Coding\_set \leftarrow \phi$
for each $P_j \in \mathcal{P}, j > i$
  do $\begin{cases} \textbf{if } is\_coding\_worthy(P_j, Coding\_set) = \textbf{ true} \\ \quad \textbf{then } Coding\_set \leftarrow P_j \end{cases}$
if $Coding\_set \neq \phi$
  then $\begin{cases} X \leftarrow make\_coded\_packet(P_i, Coding\_set) \\ X.rate \leftarrow P_i.rate \\ S \leftarrow X \end{cases}$
  else $S \leftarrow P_i$
return (S)
procedure IS_CODING_WORTHY($P_j, Coding\_set$)
 for each $C_i \in Coding\_set$
  do $\begin{cases} \textbf{if } P_j.client\_set \cap C_i.client\_set \neq \phi \\ \quad \textbf{do return } (\textbf{ false }) \end{cases}$
 return (**true**)

decided. This is done to keep the packet ordering algorithm simple, as otherwise the algorithm would have to deal with coded packets (with multiple constituent packets of different values), while deciding the sending order and rate. A coded packet is always transmitted with the intended PHY rate of the first packet in the set. This ensures that the probability of error in receiving the first packet at its intended receivers is not hampered, while opportunistically delivering other packets in the coded set to their respective clients. At the client side all received packets (natively or from network coded packets) packets are maintained till their deadline expires. This is done to ensure that packets coded with previously received packets can be recovered. The client sends back acknowledgment for packets which are successfully decoded as part of reception reports.

Delayed Packet discard: The deadline for packet delivery shifts over the duration of a streaming session. We initially set it to the playback deadline of the frame, which is calculated using the following formula,

$$\textit{Deadline} = \frac{\text{Frame}_{\text{seqno}}}{\text{Frame Rate}} + \textit{Playback buffer size} + \delta,$$

where, the deadline is number of seconds from the transmission time of the first video frame, $\text{Frame}_{\text{seqno}}$ is the sequence number of the frame. Frame Rate is the number of frames that the video player needs to display in a second. Playback buffer size is the amount of time

(in seconds) that the receiver can store the video before it needs to start decoding the frames. And, δ is a small time constant added to account for initial frame delay.

Once the playback deadline of a packet expires, we reset the deadline for delivering its constituent packets to that of the next frame which depends on the successful reception of the packet for decoding. This goes on until the packet is delivered to all clients, or the deadlines of all the frames which depend on the current packet have expired. We drop the packet from our system at that instant.

Similarly, at client, we discard a packet only if its playback deadline has expired and the packet is not useful for decoding any other frame.

3 Putting it all together

We have implemented the *Medusa* proxy and client. The implementation consists of about 3.5K lines of C code. We stream video using the Evalvid tools package [1]. We modified Evalvid to provide information about dependency structure of video frames, frame type of the generated packet and the deadline of the packet. The *Medusa* proxy runs as an application level process. We modified the MadWiFi driver to carry out per-packet rate assignment. Per packet rate assignment is achieved by specifying the target rate in a header of the video packet and then extracting it out of the packet inside the AP's driver.

At the client, the *Medusa* module keeps information regarding number of packets received and the channel quality. The module passes the received video packets to video playback software such as VLC [7] and MPlayer [5], for displaying. It also keeps a copy of received packets, till the expiry of their deadline for decoding other packets.

4 Evaluation

To study the performance of *Medusa* we have experimented with upto 25 users that are associated to a single AP(operating in 802.11g mode) and attempting to receive HD quality video from the *Medusa* proxy. Our setup consists of 30 laptops with Atheros wireless driver running Linux operating system.

Wireless conditions: The experiments were done on a university building floor. We broadly classify our wireless environment into three types: (i) *Low-loss* environment - corresponding to specific client locations where the packet error rates were 5% or less; (ii) *Medium-loss* environment - corresponding to locations where packet error rates were in the range of 5-15%; and (iii) *High-loss* environment - corresponding to locations where packet error rates were in excess of 15%. For the set of experiments reported, an experiment location did not shift from one to another in the course of experiment.

MOS Rating of video quality	PSNR range
Excellent	> 37
Good	31-37
Fair	25-31
Poor	20-25
Bad	< 20

Table 1: Table mapping the MOS based user perception of video quality to the PSNR range

Video setup: We experimented with different video clips, in this paper we present results for the Mobile calender video clip [3] replayed back to back to run for 2 minutes. The video was encoded at rates of 5, 10, 15 and 20 Mbps using FFmpeg [2] tool with H264 codec. We have repeated each experiment for 20 runs. For our experiments, we used a fixed playback buffer of 10 seconds at clients. We intend to evaluate the benefits of adaptively modifying the playback buffer size in future.

Metrics: We compare the performance of different schemes in terms of Peak Signal-to-Noise Ratio (PSNR), jitter, and overall network load imparted.

PSNR: Is a standard metric for measuring the relative quality of video streams [13, 20]. The PSNR of a video is well correlated with the perceived quality of video experienced by the user. The relationship between user perception expressed in Mean Opinion Score (MOS)and the PSNR range were detailed in [17, 22] and are summarized in Table 1.

Jitter: We measure the Instantaneous Packet Delay Variation(IPDV) [14] of received packets as a measure of jitter of the delivered video stream. This metric complements the PSNR metric which is oblivious to the delay and jitter of the delivered video, as it assumes the presence of an infinite playback buffer. High jitter value signifies a bad performance.

Network load: We measure the load placed on the network by different schemes in terms of the a) number of packets transmitted in air and also in terms of amount of air-time occupied by the packets sent by different schemes.

Compared schemes: We compare the performance of *Medusa* to the following alternate schemes.

BDCST: This scheme uses WiFi broadcast to transmit packets. However, unlike normal WiFi broadcast, the PHY rate is chosen to maximize the video PSNR performance averaged across all clients. The PHY rate is selected by sending about 30 seconds of traffic at different rates.

UCAST-INDIV: In this scheme we send the video stream to each client using isolated WiFi unicast, in sequence. For example, if there are two clients, we first send the entire video to client 1 and then the same video

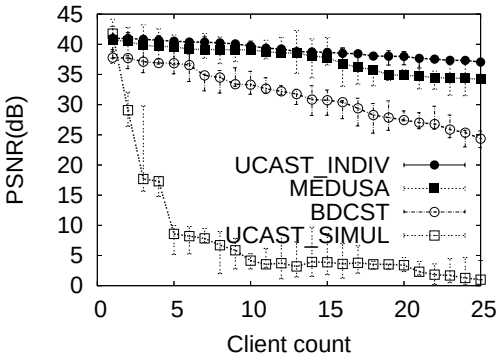


Figure 3: Plot showing the average per client PSNR of different scheme when serving 25 clients with a 20 Mbps video stream, under medium loss conditions. The mean and the variance (errorbars) are shown.

separately to client 2 using WiFi unicast. This scheme gives a quality bound for *Medusa*.

UCAST-SIMUL: In this scheme we send the video traffic to all the clients simultaneously using normal WiFi unicast with SampleRate rate adaptation. This is the traditional method for wireless data delivery.

Note that in all of these alternate schemes described above, there is no proxy and the APs and clients are unmodified, i.e., the APs take PHY rate adaptation and packet re-transmission decisions, while clients do not need to send out reception reports.

To evaluate *Medusa* we first look at the overall system performance in the multicast (Section 4.1) and the unicast (Section 4.2) cases. We then look at contribution of various *Medusa* components to the overall performance in Section 4.3. Specifically, we investigate benefits of rate adaptation in Section 4.3.1 and the performance benefits due to retransmissions in Section 4.3.2.

4.1 Overall performance (multicast)

We begin by evaluating the performance of the *Medusa* system in terms of its scalability for multicast traffic scenarios. We do so by — increasing number of clients and increasing video rates.

Scalability in the number of clients: We compare how different schemes can support HD video delivery to a large number of co-located WiFi clients. Figure 3 shows the performance of a highly loaded system with 25 clients (all receiving the same 20 Mbps video stream) at medium loss locations. We find that *Medusa* performs close to *UCAST-INDIV* (difference of 3-4 dB with 25 clients) with increasing client count, and is significantly superior to all other schemes.

Also, we find that there is a graceful degradation in *Medusa* performance when the number of clients is increased from 1 to 25. But even with 25 clients, the average PSNR value is around 37 while BDCST performance is around 27 (a 10 dB difference). The gradual degradation in performance of *BDCST* is because of the almost similar nature of errors experienced at each client(10-15% packet error).

The performance of *UCAST-SIMUL* suffers as 802.11a/g technology cannot support more than 2 streams with 20 Mbps rate(20 + 20 = 40 Mbps net load).

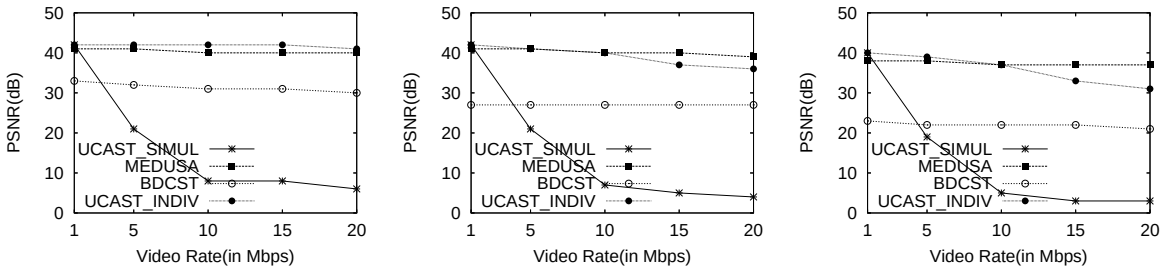
Scalability in video rate: We fix the number of clients to 10 and evaluate how the performance scales with increasing video rate — from 1 Mbps to 20 Mbps. We show the results separately for clients in good, medium and bad channel conditions in Figures 4(a), (b) and (c) respectively.

For good channel condition we observe that *UCAST-SIMUL* quickly degrades in performance with increase in video rates. Even at 5 Mbps (where the aggregate load is expected to be 5×10 Mbps = 50 Mbps), a lot of packet losses and buffer underflows occur. *BDCST* performs better and provides a more gradual performance degradation across the different rates. However, *Medusa* outperforms both and performs identical to *UCAST-INDIV*.

With worsening channel condition as shown in Figure 4(c) the performance of all schemes suffered. An interesting observation is that *the performance of* UCAST-INDIV *became worse than* Medusa as the traffic load (video rate) increased above 15 Mbps. This is due to "head-of-line" blocking in AP wireless NICs in the *UCAST-INDIV* case. Essentially, when various P- or B-packets are encountering losses, the AP spent significant effort in re-transmitting these packets, while more important I-packets waited behind. The lack of knowledge about the value of different packets, prevented the AP from devoting an appropriate amount of re-transmission effort for more important packets. *Medusa* explicitly addresses this problem and hence, led to improved performance.

4.1.1 Jitter variation of *Medusa*

We present the results for Jitter(measured as IPDV) in Figure 5. The experiment involved 10 users. Jitter increases with an increase in the number of clients, for all the schemes. However, the jitter of *Medusa* is significantly lower than both *BDCST* and *UCAST-SIMUL*. The jitter of *UCAST-SIMUL* increases exponentially with the number of clients. This can be attributed to the fact that with increasing number of clients the amount of data necessary to be transmitted becomes more than the network capacity. This results in a cascade of video packet drops in AP buffers and missing of deadlines. The jitter for *BDCST* also grows with the number of clients, as the number of candidates who can loose packets has also increased. Also, we note that the slope of increasing jit-



(a) *Good* channel (b) *Medium* channel (c) *Bad* channel

Figure 4: Average PSNR for 10 clients averaged over 20 runs as a function of the video rate under varying channel conditions.

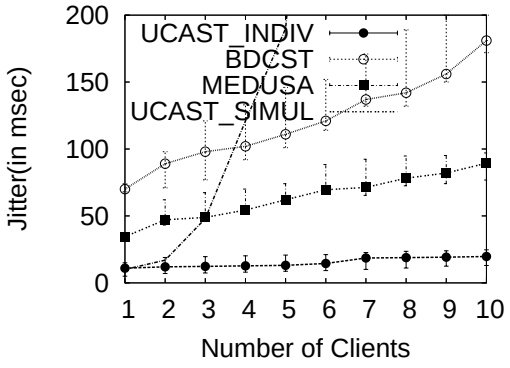


Figure 5: Average jitter experienced by 10 clients under medium channel conditions for 20 runs.

ter for *Medusa* is lower than that of *BDCST*, signifying a more gradual increase.

4.1.2 Induced network load

Apart from providing video quality commensurate with each user's channel quality, a good video multicast system should induce minimal additional network load. We compare the network load imparted by *BDCST*, *UCAST-INDIV* and *Medusa* in Table 2. We calculate the additional load placed in terms of the amount of airtime occupied by the packets (product of data-rate and packet size) which were transmitted using the different schemes. The results are normalized by the amount of airtime taken by *BDCST*. Table 2 shows that *Medusa* has an overhead of 4% for good channel conditions, which goes up to 30% under bad channel conditions. This overhead is to compensate for the 1-5% of errors that occur in good channel conditions. The channel induced losses go upto 15%-25% when the channel conditions are bad in our settings, forcing *Medusa* to inject an extra 30% traffic into the network. Hence, *Medusa* does not place unecessarily high traffic load over the network.

Channel Cond.	*BDCST*	*UCAST-INDIV*	*Medusa*
Good	1	10.12	1.04
Medium	1	10.26	1.1
Bad	1	11.40	1.3

Table 2: Airtime occupied by different schemes normalized to that of *BDCST* for 10 clients watching a 5 Mbps video, averaged over 20 runs, under varying channel conditions.

The above observation would seem to contradict with the fact that *Medusa* uses a conservative rate-adaptation mechanism which should significantly increase its network resource usage. However, we find that conservative rate-adaption while increasing the relative time occupied by individual packets also suffers less packet loss. Thus, keeping the overall network utilization low. We present further results in support of this statement in Section 4.3.

4.1.3 Interaction with other traffic

We investigate the performance of *Medusa* in presence of multiple uncorrelated traffic sources in Section 4.1.3. Since we do not introduce any new end-to-end congestion control mechanism in *Medusa* we do not present in depth results on the interaction of *Medusa* with TCP flows. In our experiments, introducing a *Medusa* flow without congestion control along with multiple TCP flows results in *Medusa* flow forcing the TCP flows to share only the residual bandwidth amongst themselves. We plan to implement a congestion controlled version of *Medusa* as part of our future work. We depict the impact of UDP flows on *Medusa* performance, in Figure 4.1.3.

We vary the number of background UDP flows, each at 4 Mbps, and compare the behavior of *Medusa* operating with a 10 Mbps video for 10 clients. The presence of multiple UDP streams causes a reduction in the quality of video seen at the clients for all schemes. However, *Medusa* outperforms *UCAST-INDIV* as the number of background flows is increased (around 7dB better for

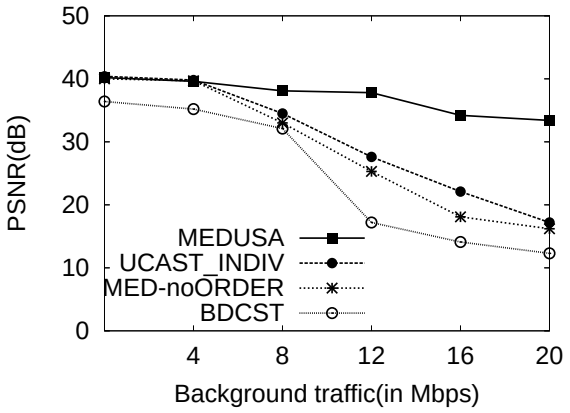


Figure 6: Average PSNR for 10 clients averaged over 20 runs in presence of background UDP flows under medium channel conditions. Video rate is 10 Mbps.

4 background flows). We find that these gains of *Medusa* are mainly due to our intelligent packet (re)-transmission ordering that mitigates the head-of-line blocking problem in *UCAST-INDIV*. We show this explicitly by also introducing a new scheme, called *Medusa*-noORDER, in which the packet re-ordering mechanism of *Medusa* is disabled. The performance of *Medusa*-noORDER is quite similar to that of *UCAST-INDIV*.

4.2 Overall performance (unicast)

To evaluate the performance of *Medusa* in serving multiple unicast video streams, we have increased the number of video flows from one to four in increments of one. We select the client randomly from a pool of 15 clients. Each experiment was run 20 times with a 5 Mbps video stream. There were other uncorrelated background flows (total of 5 Mbps) running during each experiment We plot the results of our observation in Figure 7. In unicast traffic settings, the gains in *Medusa* arrive from content-dependent rate selection, intelligent packet ordering and re-transmissions, as well as network coded re-transmissions. The broadcast advantage is available only for these network-coded re-transmissions, and not for original packets. With unicast video destined to 4 clients, the aggregate load is 20 Mbps, which is quite significant. Under good channel conditions, *Medusa* still delivers an average PSNR of 40 dB, which is very similar to *UCAST-INDIV* and is 9 dB greater than *UCAST-SIMUL*. This gain is even larger (18 dB) under bad channel conditions.

4.3 Micro-benchmarks of *Medusa* components

We now evaluate the effect of individual design choices on overall system performance. We look into performance of rate adaptation under diverse channel conditions in Section 4.3.1. The performance of network coded retransmissions is evaluated in Section 4.3.2 and the overall contribution of different components is summarized in Section 4.3.3.

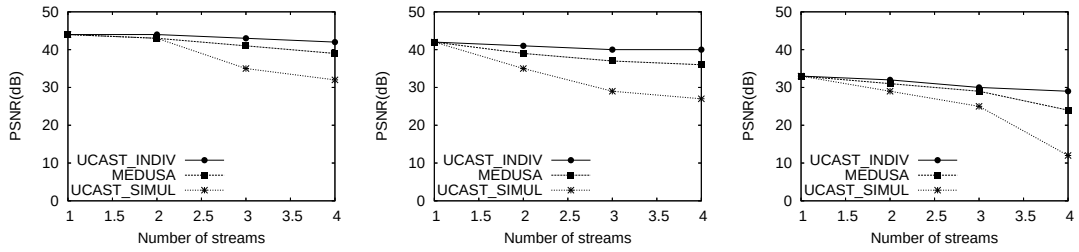
4.3.1 Rate adaptation in *Medusa*

An important aspect of *Medusa* is its ability to adapt the PHY rate based on channel conditions. We investigate the performance of these mechanisms next.

Impact of conservative base PHY rate adaptation: We look at the effects of using a conservative rate adaptation algorithm in *Medusa* on the overall system performance. We conduct experiments with a 5 Mbps video rate to 10 clients in good, medium and bad channel conditions. We ran *Medusa* with a conservative ($err\_thresh = 0.02$) and an aggressive ($err\_thresh = 0.18$) rate adaptation algorithm. Here, err_thresh signifies the maximum expected error rate which we are willing to tolerate for any PHY rate. We also ran the experiment with *UCAST-INDIV*. All the experiments we repeated for 20 runs. Figure 8(a, b) show the CDF of PHY rates assigned by different schemes under the good and the bad channel conditions. We observe, *Medusa*-conservative assigns lower PHY rates to packets than unicast, while the aggressive algorithm assigns data rates higher than the conservative scheme, but lower than the unicast scheme. To highlight the benefits of conservative rate adaptation, we plot the number of extra bytes transmitted, as a fraction of overall video size, and the PSNR of the resulting video under different channel conditions when using conservative, aggressive and unicast rate adaptation in Figure 8(c). For the *UCAST-INDIV* we plot the number of packets averaged by number of clients present. The following observations can be made from the plot,

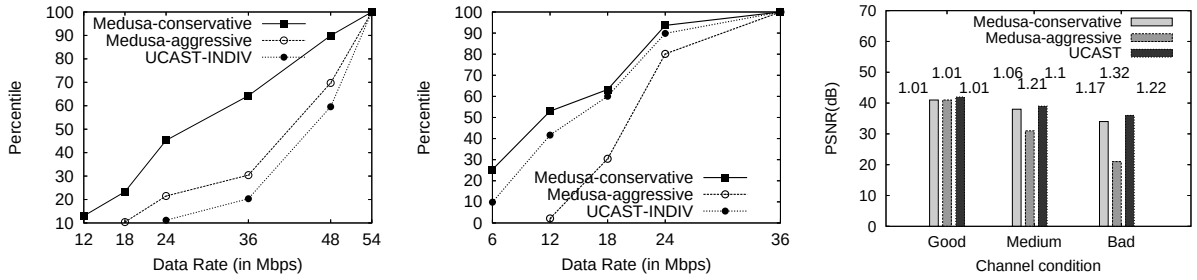
- Under good channel conditions, an aggressive as well as a conservative scheme would lead to similar number of packet losses(1%). Under such circumstances, all three schemes offer similar video quality and send similar amount of traffic over the network. From Figure 8(a), we find that around 80% (74%)of packets were transmitted at 24 Mbps or higher rate in *UCAST-INDIV*(*Medusa*-aggressive), in contrast to only 30% form *Medusa*-conservative. Hence, using an aggressive rate adaptation would had been beneficial in this case, as it would lead to network bandwidth conservation.

- For medium and bad loss environments, *Medusa*-conservative sends around 20% and 10% packets at 24 Mbps or higher. *UCAST-INDIV* sends about 40% and 11% packets at 24 Mbps of higher. In contrast, *Medusa*-aggressive sends about 60% and 20% of its packets at 24 Mbps or higher. This is because of the slowness of the feedback process which



(a) *Good* channel (b) *Medium* channel (c) *Bad* channel

Figure 7: Overall performance of *Medusa* for unicast-only media traffic. Upto 4 clients shown, each requesting a separate media stream with 5 Mbps video rate, under different channel conditions.



(a) *Good* channel (b) *Bad* channel (c) Performance under various channel conditions

Figure 8: CDF of packet rates assigned when transmitting 5 Mbps video to 10 clients in different channel conditions using different rate adaptation mechanisms. The bars in plot (c) shows performance of the schemes in terms of PSNR. The numbers in plot (c), on top of each bar, depict the normalized extra traffic in number of bytes sent by each scheme, relative to *BDCST*.

makes the *Medusa*-aggressive algorithm slow to react to changes in channel conditions. The performance of the *Medusa*-aggressive scheme suffers because of its inability to adapt quickly as shown in Figure 8(c). The number of packets transmitted by the conservative algorithm is around 15% less than *Medusa*-aggressive. This is expected, as the high threshold value ensures that we would make very few errors. The aggressive algorithm also leads to worse video quality(in PSNR) when the network resouces are scarce, precisely because of their inefficient network resource usage. Worsening channel conditions makes the difference in video quality about 6-12 dB(*Medusa*conservative and *UCAST-INDIV* have an advantage over *Medusa*-aggressive).

Thus, except under good channel conditions, keeping a conservative rate leads to better network resource utilization, while the quality is maximized in all conditions by adopting a conservative rate adaptation.

Impact of mobility on rate adaptation: To study the effect of mobility and its impact of adaptation mechanisms, we performed targeted mobility experiments, where we repeatedly moved one user between a high-loss and a low-loss location, while all the other clients stayed stationary at the low-loss location. The mobile client moved from the low loss to the high loss location (across a wall) quickly, stayed there for about 4 seconds, and returned. We show the adaptation performance of *Medusa* in comparison to *UCAST-INDIV* and *BDCST* in Figure 9. The *UCAST-INDIV* scheme running its MAC-layer rate adaptation technique adapts the fastest. *Medusa* with its intent of making rate adaptation decisions (of its PHY base rate) slowly, adapts somewhat slower. It takes *Medusa* about 0.4 seconds to adapt to the change in channel condition for the mobile user. This occurs in both cases —

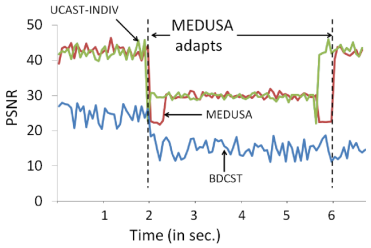


Figure 9: Adaptation of different scheme with targeted mobility, for video at 20 Mbps rate.

when the user moves away from the low loss location, and when it returns to the low loss location. This can be attributed to the higher layer reception reports and slower timescales in which they occur. However, once *Medusa* adapts, it provides the user with the same performance as the *UCAST-INDIV* in this case. The *BDCST* scheme has no adaptation mechanism and does not adapt when the user moves.

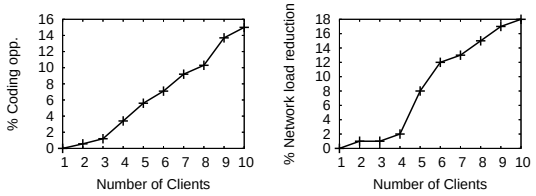
Impact of interference on rate adaptation: We next study the performance of these schemes under targeted interference from an external 802.11 source, that was a hidden terminal to the *Medusa* clients. Figure 10(a) shows the relative performance of *Medusa*, *UCAST-INDIV*, and *BDCST*, when the video rate was 5 Mbps. The interferer used UDP to download a large file starting at time 2 seconds. The performance impact of this interference is similar to that of mobility. *Medusa* performed similar to *UCAST-INDIV* and much superior to *BDCST*. However, it experienced a slight delay in adapting its rate when compared to *UCAST-INDIV.*

As shown in Figure 10(b), at a video rate of 20 Mbps, a similar effect happens with the hidden terminal interference. However, hidden terminal has a significantly greater interference impact and at this high video rate, the PSNR of both *Medusa* and *UCAST-INDIV* drops. Further, at 20 Mbps and with hidden terminal interference, the performance of *UCAST-INDIV* falls slightly below *Medusa*. Examining this performance of *Medusa* more closely, we see that at time 2.4 seconds, the inflate-deflate algorithm kicks in to help improve performance. The table in Figure 10(c) shows the number of I, P, and B frames that *Medusa* had to discard, inflate, deflate, and their channel occupancy time in the three phases (initial no interference, interference starts, and inflate-deflate starts).

4.3.2 Network coded re-transmissions

We evaluate the benefits of using network coded retransmissions with varying number of clients in the system (Figure 11). Panel (a) figure shows the percentage of all packet transmissions in each case that were actually network coded. As can be seen from the plot with increasing number of clients the number of network coding opportunities increases. Also, we would like to note that the computation overhead is never more than 1% of CPU time in any of our experiments. The actual performance gains from network coding can be seen in Figure 11(b) which indicates the reduction in airtime load that occurred due to network coding opportunities.

Finally, we evaluate the benefits of network coded retransmissions under varying channel conditions. We experiment with 5 clients receiving a 5 Mbps video under good, medium, and bad channel conditions respectively. We report the coding opportunities and the airtime reduction due to the network coded scheme in Table 3. The table shows that worsening channel condition leads to higher benefits from network coding. This is expected, as the number of packet losses increases as the channel condition becomes bad, this in turn leads to higher number of retransmissions and thus more coding opportunities.



(a) Coding opportunities (b) Traffic savings

Figure 11: Coding opportunities and percentage traffic reduction as a function of number of clients under medium channel condition with 5 Mbps video averaged over 20 runs.

Chnl. cond.	*Good*	*Medium*	*Bad*
% of coded packets	3.1	7.4	12.6
% Airtime load reduction	5	8	13

Table 3: Coding opportunities and normalized traffic injected as a function of channel condition for five clients with 5 Mbps video averaged over 20 runs.

We note that using network coding also leads in improving PSNR with increasing number of clients or worsening channel error conditions. We do not present the results for sake of brevity.

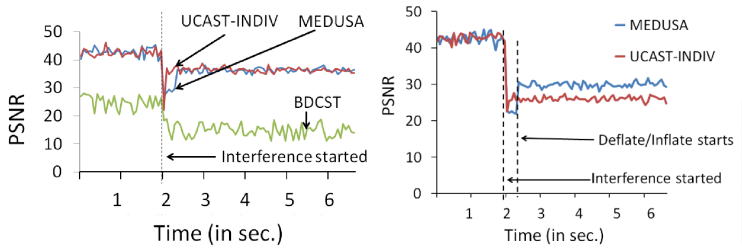
4.3.3 Component contribution

The *Medusa* system employs content aware rate adaptation, selective retransmissions and transmission (re)ordering to provide quality enhancements over broadcast based media delivery. Figure 12 shows the relative contribution of different design components in *Medusa*, over and above standard WiFi broadcast. In the low-loss and the high-loss environments various mechanisms in *Medusa* (re-transmissions, rate adaptations, ordering, rest – from integration of all the components). provides a nearly 9 and 10 dB improvement in PSNR over plain *BDCST*.

5 Related Work

There has been a significant amount of research in the area of video streaming over wireless networks, both in video and systems community (see [29] for a summary). We comment on the most related pieces in this section.

Dynamic transcoding is a standard technique for enhancing the quality of the streaming video. It involved



	(I,P,B)	(I,P,B)	(I,P,B)
Lost/Dropped	(0,0,0)	(3,15,23)	(0,2,8)
Inflated	(0,0,0)	(0,0,0)	(1,11,0)
Deflated	(0,0,0)	(0,0,0)	(0,2,39)
Chnl. time	(.2,.3,.5)	(.3,.3,.4)	(.5,.3,.2)

(a) Video at 5 Mbps rate. (b) Video at 20 Mbps rate. (c) Video at 20 Mbps rate.

Figure 10: Adaptation of different schemes with external hidden terminal interference with video at 5 and 20 Mbps rate. The table shows the number of I, P, and P frames that were discarded, inflated, deflated, and the channel occupancy time for *Medusa* in the three phases.

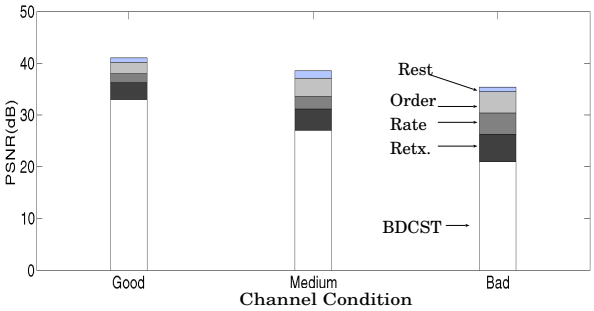


Figure 12: Performance breakdown between rate adaptation and retransmission components of *Medusa* system for 10 clients averaged over 10 runs under varying channel conditions.

estimating the bandwidth available in the medium and then change the video rate itself to ensure the best quality video that the channel can support is delivered to the receivers. Chou et.al. [9, 11, 12] in their seminal work propose a rate-distortion optimization technique to adjust the rate of the transmitted video based on channel quality. However, this body of work depends on the wireless hardware to pick the rate at which the video is to be transmitted. A second set of prior work dealing with identifying the optimal video rate as well as the amount of redundancy to be added to the video stream is represented by the [25, 26]. The authors formulate a complex optimization problem for the same and provide heuristic algorithms which show the performance benefits of the designed algorithms. Such FEC based mechanisms are orthogonal to the set of techniques used in our work. In *Medusa* we mostly leverage understanding from such prior work, and tailor our solutions to the needs of WiFi-based media delivery and specific issues therein.

Authors in [19] use a scalable video codec and optimally determine the amount of FEC required. This is a representative of a large body of literature in the area. This approach is, however, complementary to ours, as we focus on rate adaptation and re-transmission based techniques for WiFi broadcasts in video delivery systems.

In general wide-area network settings, the OxygenTV project [15, 16] has considered performing selective end-to-end re-transmissions of packets based on the video frame type, focusing more on unicast video delivery. They propose the SR-RTP protocol for the such selective retransmissions [16]. In contrast, our work explores various wireless link adaptation mechanisms that leverage packet content information.

In [31], authors present a measurement study different application-layer video streaming mechanisms in multi-hop wireless context. They do not explore interactions between the value of content to applications, and link adaptation mechanisms as we do in this work. In [27], authors present mechanisms to improve the quality of the video while operating in a lossy wireless environment. However they focus on low bit-rate video streams, while our solutions are stylized to deliver HD quality video in WiFi environments.

The authors of [28] present an end-to-end video rate control protocol for mobile media streaming on Internet paths involving wireless links. They implement the control functionalities in the receiver, which is charged with proving feedback to the server. The video server uses this information to change the video codecs used to match the available capacity of the end to end path. The proposed approach is complementary to ours, as we focus on adapting video delivery on the WiFi link, by making link adaptation decisions for WiFi transmitters.

A recent mechanism, SoftCast [17], uses the notion of compressed sensing to create equal priority video packets. This allows users to extract information proportional with their own channel quality. The core of this work focuses on the complementary aspect of compressed sensing. Furthermore, SoftCast also requires changes to the wireless radio hardware (and the PHY layer), while our system makes no changes to the current 802.11 standards.

Finally, DirCast [10] also design and implement a system for WiFi multicast. They advocate the use of pseudo-broadcasts in their system. However, the main difference between our *Medusa* approach and DirCast is that we propose a content-dependent PHY rate selection, re-transmissions, and packet order selection. This is an issue that is not considered by DirCast. DirCast focuses on some complementary problems for the multicast case only (e.g., intelligent client-AP association decisions, FECs, etc.), and is agnostic of value of packets to applications.

We believe that the main contribution of *Medusa* is in combining some understanding of packet contents with various WiFi link layer functions to improve the quality of media delivery. WiFi link layer decisions, until now, have been considered in a mostly content-agnostic manner. *Medusa* suggests an interesting design point for combining application-layer information in making decisions at the link layer.

Various other new techniques can be brought to improve performance of *Medusa* even further. In the future we therefore plan to investigate the use of other complementary but related mechanisms, such as application layer FEC for proactive error recovery, and a congestion control mechanism for co-existence with TCP flows.

6 Conclusions

Media delivery over wireless systems is a growing area of importance. We present the design and implementation of the *Medusa* system which allows efficient delivery of high quality media to one or more WiFi clients. The key contribution of this work is in recognizing that certain link layer functions, e.g., re-transmissions, PHY rate selection, packet transmission order, can be implemented better by having some knowledge about the value of packets to applications. In order to be minimally invasive to existing systems, we implement this function in a proxy. Our results indicate that our collection of techniques can facilitate HD video delivery of 20 Mbps to 25 clients while maintaining a good viewing quality.

7 Acknowledgements

We thank Sanjeev Mehrotra for initial discussions on the state-of-art in media streaming. We acknowledge Sateesh Addepalli for his comments and suggestions on our evaluation plan that helped improve the quality of this paper. We also acknowledge Sriram Subramanian for co-developing a preliminary version of this system from which we learned a number of important lessons. Finally, we thank our shepherd Srinivasan Seshan and other anonymous reviewers whose feedback helped bring the paper to its final form.

Sayandeep Sen and Suman Banerjee were supported in part by the US NSF through awards CNS-0916955, CNS-0855201, CNS-0751127, CNS-0627589, CNS-0627102, and CNS-0747177.

References

[1] Evalvid - a video quality evaluation tool-set. `www.tkn.tu-berlin.de/research/evalvid/`.
[2] Ffmpeg - digital audio converter. `www.ffmpeg.org/`.
[3] Mpeg-2 hd test patterns. `www.w6rz.net/`.
[4] Mpeg-4, moving picture experts group-4 standard. `www.chiariglione.org/mpeg/standards/mpeg-4/mpeg-4.htm`.
[5] Mplayer - the movie player. `www.mplayerhq.hu`.
[6] Streambox. `www.streambox.com/products/hd/hd\_9200\_main.html`.
[7] Vlc media player. `www.videolan.org/vlc/`.
[8] BICKET, J. Bit-rate selection in wireless networks. MIT Master's Thesis, 2005.
[9] CHAKARESKI, J., AND CHOU, P. A. Radio edge: rate-distortion optimized proxy-driven streaming from the network edge. *IEEE/ACM Transactions on Networking* (2006).
[10] CHANDRA, R., KARANTH, S., MOSCIBRODA, T., NAVDA, V., PADHYE, J., RAMJEE, R., AND RAVINDRANATH, L. Dircast: A practical and efficient wi-fi multicast system. ICNP 2009.
[11] CHOU, P., AND MIAO, Z. Rate-distortion optimized streaming of packetized media. *IEEE Transactions on Multimedia* (2006).
[12] CHOU, P., MOHR, A., WANG, A., AND MEHROTRA, S. Fec and pseudo-arq for receiver-driven layered multicast of audio and video. *Proceeding of Data Compression Conference* (2000).
[13] DAVID, S. A guide to data compression methods, 2002.
[14] DEMICHELIS, C., AND CHIMENTO, P. Ip packet delay variation metric for ip performance metrics (ippm). RFC 3393, IETF.
[15] FEAMSTER, N. Adaptive delivery of real-time streaming video. MIT M.Eng. Thesis, 2001.
[16] FEAMSTER, N., AND BALAKRISHNAN, H. Packet loss recovery for streaming video. 12th International Packet Video Workshop, 2002.
[17] JAKUBCZAK, S., RABUL, H., AND KATABI, D. Softcast: One video to serve all wireless receivers. MIT-CSAIL-TR-2009-005, 2009.
[18] KATTI, S., RAHUL, H., HU, W., KATABI, D., MÉDARD, M., AND CROWCROFT, J. Xors in the air: practical wireless network coding. *IEEE/ACM Transaction Networking.*
[19] MAJUMDAR, A., SACHS, D., KOZINTSEV, I., RAMCHANDRAN, K., AND YEUNG, M. Multicast and unicast real-time video streaming over wireless lans. *IEEE Transactions on Circuits and Systems for Video Technology,* (2002).
[20] MARTINEZ-RACH, M., AND ET. AL. Quality assessment metrics vs. PSNR under packet loss scenarios in MANET wireless networks.
[21] MURTY, R., WOLMAN, A., PADHYE, J., AND WELSH, M. An architecture for extensible wireless lans. *HOTNETS-VII* (2008).
[22] ORLOV, Z. Network-driven adaptive video streaming in wireless environments. *PIMRC* (2008).
[23] ROZNER, E., IYER, A. P., MEHTA, Y., QIU, L., AND JAFRY, M. ER: efficient retransmission scheme for wireless lans.
[24] SCHULZRINNE, H., CASNER, S., FREDERICK, R., AND JACOBSON, V. RTP: A transport protocol for real-time applications. RFC 3350, IETF.
[25] SEFEROGLU, H., ALTUNBASAK, Y., GURBUZ, O., AND ERCETIN, O. Rate distortion optimized joint arq-fec scheme for real-time wireless multimedia.
[26] SEFEROGLU, H., GURBUZ, O., ERCETIN, O., AND ALTUNBASAK, Y. Rate-distortion based real-time wireless video streaming. *Image Communications* (2007).
[27] TAN, K., RIBIER, R., AND LIOU, S.-P. Content-sensitive video streaming over low bitrate and lossy wireless network.
[28] TAN, K., ZHANG, Q., AND ZHU, W. An end-to-end rate control protocol for multimedia streaming in wired-cum-wireless environments. *ISCAS '03.*.
[29] WANG, Y., AND ZHU, Q.-F. Error control and concealment for video communication: a review. *Proceedings of the IEEE* (1998).
[30] WONG, S. H. Y., YANG, H., LU, S., AND BHARGHAVAN, V. Robust rate adaptation for 802.11 wireless networks.
[31] XIAOLIN, C., PRASANT, M., SUNG-JU, L., AND SUJATA, B. Performance evaluation of video streaming in multihop wireless mesh networks.