DirCast: A Practical and Efficient Wi-Fi Multicast System

Ranveer Chandra[‡], Sandeep Karanth[⋆], Thomas Moscibroda[‡],
Vishnu Navda[⋆], Jitendra Padhye[‡], Ramachandran Ramjee[⋆], and Lenin Ravindranath[†]
[⋆]Microsoft Research India [‡]Microsoft Research Redmond [†]MIT

Abstract—**IP multicast applications such as live lecture broadcasts are being increasingly used in enterprise and campus networks. In many cases, end hosts access these multicast streams using Wi-Fi networks. However, multicast over Wi-Fi suffers from several well-known problems such as low data rate, high losses and unfairness vis-a-vis other contending unicast transmissions. In this paper we present DirCast, a system to solve many of these problems. DirCast requires no changes to the 802.11 MAC protocol or the wireless access points. Software changes are required on clients only if they wish to participate in multicast sessions. The aim of DirCast system is to minimize the airtime consumed by the multicast traffic, while simultaneously improving client experience. To meet these goals, the DirCast converts multicast packets to unicast packets targeted to certain selected clients; other clients receive these packets by listening in promiscuous mode. The target clients are carefully selected to minimize loss rate experienced by the non-targeted clients. If necessary, clients are forced to change the AP they are associated with. In addition, DirCast uses proactive adaptive FEC to further reduce the loss rate and implements a novel virtual multicast interface in order to be compatible with the security needs of the enterprise. We demonstrate the effectiveness of DirCast using extensive experiments in a Wi-Fi prototype implementation and through large-scale simulations.**

I. Introduction

Multimedia IP multicast applications such as live lecture broadcasts are increasingly being used in enterprise and campus networks. For example, in a recent enterprise traffic study [20], 5-10% of all payload bytes were attributed to multicast streaming. At the same time, an increasingly large number of hosts in enterprise environments are using Wi-Fi for their basic connectivity [16]. Thus, enterprise Wi-Fi networks need to efficiently support IP multicast.

This ought to be an easy task, given the inherently broadcast nature of the wireless medium. However, the peculiarities of 802.11 protocol create several well-known problems [8]. First, the 802.11 protocol does not require an ACK for a multicast transmission. So, lost multicast packets are not retransmitted at the MAC layer, and some of the receivers can experience high loss rate. Second, since there are no acks, the MAC-level sender (generally, the access point) does not know whether the frame might have collided with another frame and thus, does not implement backoffs resulting in unfairness to other unicast clients that backoff. Third, the link data rate is static and, in order to guarantee coverage to all associated clients, is typically fixed to one of the low basic rates available (e.g. 1/6Mbps for 802.11b/g). This limits the rate at which multicast data can be sent - for example, high-def video can not be sent via multicast. Fourth, the low link data rate coupled with the packet-fair Wi-Fi MAC can significantly reduce the throughput of any contending unicast client. This is known as the rate-anomaly problem [12]. Fifth, the background multicast chatter can reduce energy savings for clients operating in power-save mode (PSM) [11], since clients that are not subscribed to any multicast group have to wake up unnecessarily to receive these packets.

These problems are so severe, that many organizations simply do not permit multicast traffic over Wi-Fi links. Our own organization has recently made the same decision.

One way to solve these problems is to convert the multicast traffic into a group of unicast sessions, one for each client. Since unicast packets are acknowledged and are subject to retransmission and rate adaptation by the AP, we see lower losses, better MAC fairness and effective high data rate transmission for the multicast stream.

However, this simple approach would require too much bandwidth, and defeat the very purpose of multicast. In wired networks, multicast problems are often solved by application level multicast, but as we shall explain in Section V, this solution is not feasible for WiFi networks.

As a result, a number of clever solutions have been proposed [6], [8], [15] for mutlicast in WiFi networks. However, most of these solutions require significant changes to the 802.11 protocol, making it difficult, if not impossible to deploy them in the real world. Even prototyping such solutions can be difficult, and thus most of them have been evaluated via analysis or simulations only.

In this paper, we present DirCast, a practical and efficient system for improving multicast performance over corporate Wi-Fi networks. DirCast does not require any changes to the 802.11 standard. Furthermore, it does not require any changes to the APs. While some software needs to be installed on the clients, only the clients that wish to participate in multicast sessions need software changes. Clients that do not wish to join multicast sessions require no changes at all. We have built a prototype DirCast implemenation and evaluated it in a testbed consisting of 12 clients and 7 APs spread across one floor of a typical office building.

The goal of the DirCast system is to provide good service to all multicast clients while simultaneously minimizing the air time consumed by multicast traffic. DirCast uses a central controller to intercept multicast packets and convert them to unicast packets targeted to certain selected clients. These clients are called *target clients*. Other clients receive these packets by listening in promiscuous mode. This approach is known as psuedo-broadcast [14].

To minimize the amount of airtime consumed, DirCast controller groups the multicast clients at APs in a way that minimizes the amount of air time consumed. This is called *association control*. The association control algorithm employed in DirCast takes a non-trivial greedy-approach. We analytically prove it to be a $4 \log n$-approximation algorithm, which is asymptotically optimal. Next, to be conservative the DirCast controller selects the client with the "worst" connectivity at each AP, and makes it the target client. The target is updated dynamically, to compensate for changes in Wi-Fi environment and traffic patterns. This is called *destination control*. Since non-target clients listen to packets in promiscuous mode, they may not receive some of the packets. To overcome these losses, DirCast proactively adds FEC packets to the multicast stream. DirCast uses an adaptive algorithm to determine the amount of FEC needed. Our algorithm is successful in reducing losses to all multicast clients with little overhead. Besides reducing airtime and losses, DirCast also cuts down multicast chatter in the network since it converts all multicast traffic to unicast. APs do not advertise DirCast traffic in their beacons as multicast traffic, thus preventing unnecessary wakeups for PSM clients that are not part of any multicast group.

DirCast requires non-target clients to operate in promiscuous mode. However, most corporate Wi-Fi networks use 802.11x, which encrypts traffic with a unique key for each Wi-Fi client. So while other clients can sniff the packets, they can not decrypt them. DirCast implements a novel mechanism called virtual multicast interface, where a VirtualWiFi [4] interface is created for each of the client's multicast sessions. The security session key for the virtual multicast interface is shared among the clients of the respective multicast group, allowing those clients to decrypt (only) the appropriate multicast group packets received in promiscuous mode.

In summary, we have designed, implemented and evaluated DirCast, a practical WiFi multicast system for enterprise environments. DirCast does not require any changes to the 802.11 protocol, or to the WiFi infrastructure. The DirCast system converts multicast packets to trageted unicast transmissions. To minimize the amount of air time consumed, DirCast uses destination control and a novel greedy algorithm for association control. DirCast includes an adaptive, proactive FEC scheme to reduce overall packet losses. Finally, to overcome the challanges posed by encryption protocols such as 802.1x, DirCast uses a novel mechanism called virtual multicast interface.

II. DirCast Architecture

Before we describe the DirCast architecture, we list the practical constraints that DirCast has to satisfy in order to be practically deployable in today's enterprise scenarios.

A. Preliminaries

Throughout the rest of the paper, we will need to distinguish between application data rate, and the data rate of the wireless link. We use the term *data rate* to describe the former, and the term *transmission rate* to describe the latter. The *data rate* depends on the application and can take any arbitrary value, while the *transmission rate* depends on the wireless technology and can take only certain fixed values (e.g. 6MBps, 12Mbps etc. for 802.11a).

B. Constraints, Assumptions and Requirements

The critical *constraints* any practical solution to the Wi-Fi multicast problem should satisfy are:

C1: Access points are off-the-shelf and cannot be modified.

C2: There will be legacy Wi-Fi clients which operate outside the control of DirCast.

Constraint C1 is critical to ensure easy deployability and cost-effectiveness. Enterprise network administrators spend a lot of money on their APs, and are generally loath to replace them. Constraint C2 simply reflects reality, and underscores the need for incremental deployment. If either of these constraints can be relaxed, performance of DirCast can be improved further. In addition to the above constraints, DirCast is designed for an environment that satisfies the following additional *assumptions*, which are generally met in a typical enterprise scenario.

A1: Neighboring APs operate on different channels.

A2: Multicast traffic never exceeds available AP capacity.

Assumption A2 can be relaxed with minor changes to our design – we use them only for simplicity of design and ease of explanation. To relax A1, we would need to perform admission control and load balancing on multicast traffic, which is beyond the scope of this paper. We do not address the issues of client mobility and power consumption in this paper. This is because most Wi-Fi clients in enterprise environments are laptops, and people tend to use their laptops in nomadic fashion: they move from place to place (e.g. between conference rooms), but generally spend significant time at each location [19].

Clients that wish to participate in DirCast multicast sessions must satisfy the following minimal requirements.

R1: The client must run a DirCast software component.

R2: The wireless interface must support promiscuous mode.

R3: The wireless interface can be instructed to associate with a specific AP.

R1 is easy to achieve in an enterprise environment – the central IT department can require the users to install the software by fiat. Furthermore, notice that all modern Wi-Fi interfaces and their drivers support R2 and R3.

C. Overview

The DirCast architecture consists of a client component and a server component. The client component is a software module that runs on all clients that wish to receive multicast traffic. We call them DirCast clients. The server component runs on a single separate machine called the DirCast server. For the purpose of this discussion, we will assume that there is only one DirCast server per enterprise.

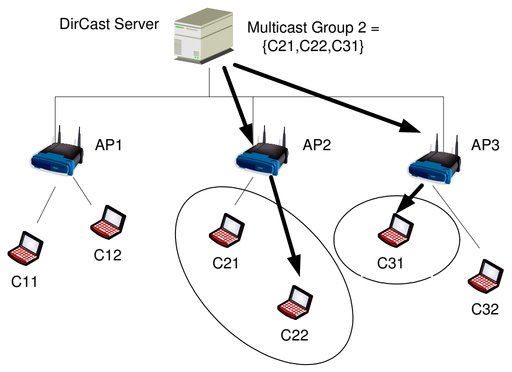


Fig. 1. **Operational overview**

The basic operation of the DirCast system is as follows. The software module on the DirCast client intercepts the requests to join and leave an IP multicast group, and sends the request to the DirCast server. The DirCast server subscribes to the multicast group on client's behalf, if it has not already subscribed to it on behalf of some other client. The server maintains the membership in the multicast group as long as at least one client in the system is subscribed to it.

When a multicast packet is received, the DirCast server prepares a list of all clients that have subscribed to that multicast group, and groups them according to the APs they are associated with. For each AP, it selects a particular client as the "target client", and unicast the packet to that client. The target client at each AP is generally the client that has the *worst* connectivity to that AP. Other clients associated with that AP receive the packet by monitoring the channel in a promiscuous mode. Since the non-target clients listen to packets in promiscuous mode, they may not receive all the packets. To overcome these losses, the DirCast server also sends extra FEC packets. These FEC packets allow non-target clients to recover any data they have missed.

The DirCast server periodically re-evaluates if it needs to change the target client at each AP and/or more drastically, needs to reassign clients among APs (destination control), or change the amount of FEC it is adding. The re-evaluation is also carried out when clients join or leave multicast groups.

Figure 1 illustrates DirCast using an example system with 3 APs with two associated DirCast clients each. Clients C_{21}, C_{22}, C_{31} form a multicast group G. Since C_{22} is more distant to the AP than C_{21}, the central controller determines C_{22} as the target client for AP_2 with regard to this multicast group. Client C_{21} receives the packets in promiscuous mode. On AP_3, only one client, C_{31}, is interested in the multicast group and hence, this client becomes the target client. Finally, AP_1 has no clients belonging to this multicast group and hence, there is no target client. The DirCast server need not inform AP_1 of these multicast messages.

We now describe the destination and association control mechanisms in detail, followed by a brief overview of our proactive FEC mechanism. Finally, we will describe how DirCast handles security mechanisms such as 802.1x.

For the purpose of the discussion below, we will assume that the central controller knows everything about the APs and multicast clients in the system, and the condition of channels between them. In Section II-G, we will describe how we gather this information in practice.

D. Destination control

As described earlier, DirCast sends multicast packets as unicast to a target client at each AP. All the other clients at that AP listen to the data in promiscuous mode.

The primary challenge in this approach is to select the right target client at each AP (i.e. *destination control*. Since the packets are sent as unicast to the target client, the transmission rate is determined by the autorate algorithm on the AP (recall that DirCast does not modify APs), and the condition of the wireless channel between the AP and the target client.

If the client selected as the target client has good channel conditions, the autorate algorithm on the AP will send the packets at a higher transmission rate. This will reduce the consumption of air time, but some of the other clients with worse channel conditions, who are listening in promiscuous mode may not be able to decode the packets.

Thus, DirCast selects the *"worst"* client at each AP as the target client. The *worst* client is the one to which the AP will send with the least transmission rate. In Section II-G we will describe how the controller determines what unicast data rate the AP uses for each client. This choice is re-evaluated every 30 seconds. In addition, whenever a new client arrives, or if the loss rate at any client exceeds 10%, the central controller re-evaluates the choice of the target client for that AP.

With the approach of destination control described above, it is clear that it is essentially the worst (least transmission-rate) client belonging to an AP group that determines the impact this group will have on overall air time consumption and thus, how it will affect other contending wireless traffic. For this reason, associating clients and multicast sessions to APs becomes a non-trivial optimization problem. This is the problem of *association control*, which we describe next.

E. Association control

The goal of this optimization problem is to minimize the air time consumed by all multicast sessions in the system. For example, if client A is likely to get 24Mbps at AP1, and 12 Mbps at AP2, it makes sense for client A to associate with AP1. However, if AP1 has no other clients associated with it, and AP2 already has a multicast client associated with it and is receiving data at a rate of 12Mbps or less, it makes sense to associate client A with AP2, as no additional airtime would be consumed. The actual optimization problem definition and the DirCast controller algorithm are described below.

Practically, DirCast re-evaluates client and multicast assignments every 30 seconds, or whenever a new client arrives. After determining the appropriate mappings using our algorithm, the server asks the clients to change their associations if necessary. This is accomplished by sending the BSSID of the AP to the DirCast client, which then makes appropriate calls to the wireless driver on the client to enforce the choice.

1) Formal problem definition: If every client can be a member of multiple multicast groups, the question of how to assign clients to APs becomes an interesting and difficult combinatorial optimization problem. Intuitively, the difficulty stems from the fact that once we assign a client to an AP, this AP must host *all* multicast groups which this client is a member of. Some of these multicast groups may be well-suited to be hosted on this AP (in the sense that other clients belonging to these groups are already associated with this AP), but others may not. The challenge is to appropriately disentangle these dependencies and find a good partitioning of clients to APs, such that all clients are assigned to good APs. In all of this, given our destination control procedure, it is always the client with least data-rate to the AP that determines this AP's airtime utilization for its multicast groups.

Problem Definition: Formally, we can model the multicast client association problem as follows. Given a set of n clients $\mathcal{C} = \{C_1, \ldots, C_n\}$, m APs $\mathcal{AP} = \{AP_1, \ldots, AP_m\}$ and g multicast groups $\mathcal{M} = \{M_1, \ldots, M_g\}$. Every multicast group consists of a subset of the clients, i.e., $M_k \subseteq \mathcal{C}$ for all $1 \leq k \leq g$. Notice that one client can be a member of multiple multicast groups. We denote by T_i the set of multicast groups that a client C_i is interested in, i.e., $T_i = \{M_k \in \mathcal{M} \mid C_i \in M_k\}$. Let R_{ij} be the transmission rate client C_i achieves when associating to AP AP_j. The goal is to assign clients and multicast groups to APs such that

1) every client is associated to exactly one AP and
2) if a client C_i is associated to AP_j, then all multicast groups in T_i must be assigned to AP_j.

For every multicast group that is assigned to an AP, its airtime is determined by the least transmission rate of all clients in this multicast group that are associated to this AP. Formally, the airtime a_{jk} of multicast group M_k on AP AP_j is defined as $a_{jk} = \max_{C_i \in C^j} \frac{1}{R_{ij}}$, where C^j is the set of clients associated to AP_j. Notice that if no client in a group M_k is assigned to an AP_j, then $a_{jk} = 0$. Finally, since we want to minimize the overall airtime across all APs, we seek to minimize the quantity $AT = \sum_{AP_j \in \mathcal{AP}} \sum_{M_k \in \mathcal{M}} a_{jk}$.

ILP Formulation: It is easy to formulate the problem as a (mixed) Integer Linear Program (ILP). Let x_{ij} be an indicator variable that is 1 if C_i is assigned to AP_j, and 0 otherwise.

$$\begin{aligned}
\min AT = & \sum_{AP_j \in \mathcal{AP}} \sum_{M_k \in \mathcal{M}} a_{jk} \\
\text{s.t.} \sum_{AP_j \in \mathcal{AP}} x_{ij} &= 1 \qquad , \forall C_i \in \mathcal{C} \\
a_{jk} &\geq \frac{1}{R_{ij}} \cdot x_{ij} \quad , \forall AP_j \in \mathcal{AP}, C_i \in M_k \\
a_{jk} &\geq 0, \forall AP_j \in \mathcal{AP}, M_k \in \mathcal{M} \\
x_{ij} &\in \{0,1\}, \forall C_i \in \mathcal{C}, AP_j \in \mathcal{AP}.
\end{aligned}$$

The above ILP computes an optimal client association. The problem with this ILP is that its number of constraints is $2^{\mathcal{M}}$, which can be exponential if $g \in \omega(\log n + \log m)$, and that is generally unclear how to solve it efficiently or appropriately relax it to an efficiently solvable LP. In the sequel, we propose a greedy heuristic for the problem and show that—as long as the number of multicast groups is within reasonable bounds—, the heuristic is in fact an $4 \log n$ approximation algorithm for the problem, i.e., its solution is guaranteed to be within at most a logarithmic factor of the theoretical optimum.This is asymptotically optimal.

2) Greedy Algorithm: It has been shown that if all clients belong to exactly one multicast group, a variant of the *multi-cast group association control problem* can be reduced to an instance of the set cover problem [6]. Intriguingly, our more complex problem in which every client can be in multiple multicast groups can also be reduced to a known problem, albeit with a different transformation.

Given an instance I of the assignment problem, we define a set $\mathcal{S}$ consisting of sets $S_{G,j,\kappa}$, where $G \subseteq 2^{\mathcal{M}}$ denotes a set of multicast groups, j denotes AP_j, and κ is the maximum airtime that is allowed. Intuitively, the set $S_{G,j,\kappa}$ contains all clients for the transmission rate is sufficiently high ($1/R_{ij} \leq \kappa$) and that, if all multicast groups in G are assigned to AP_j, can associate to AP_j. Formally, we define $S_{G,j,\kappa}$ and $\mathcal{S}$ as

$$\begin{aligned}
S_{G,j,\kappa} &:= \{C_i \in \mathcal{C} \mid T_i \subseteq G \wedge \frac{1}{R_{ij}} \leq \kappa\} \\
\mathcal{S} &:= \{S_{G,j,\kappa} \mid G \subseteq 2^{\mathcal{M}}, AP_j \in \mathcal{AP}, \exists R_{ij} \text{ s.t. } R_{ij}^{-1} = \kappa\}.
\end{aligned}$$

Notice that the cardinality of $\mathcal{S}$ is at most $|\mathcal{S}| \leq 2^{|\mathcal{M}|} \cdot |\mathcal{AP}| \cdot |\mathcal{C}|$ because the number of different transmission rates R_{ij} and hence κ is at most $|\mathcal{C}|$.

For each set $S_{G,j,\kappa}$ we define its cost as

$$cost(S_{G,j,\kappa}) = \sum_{M_k \in G} \max_{\substack{C_i \in M_k \\ R_{ij}^{-1} \leq \kappa}} R_{ij}^{-1}.$$

The cost of $S_{G,j,\kappa}$ indicates how much adding this set (i.e., the corresponding multicast groups) costs in terms of additional airtime.

Algorithm 1 Approximation Algorithm

```
1: Round up all $R_{ij}$ to the next power of two. ($\rightarrow \widehat{R}_{ij}$);
2: $U = \mathcal{C}$;  $S^* = \{\}$;
3: while  $|U| > 0$  do
4:     Compute $cf(S_{G,j,\kappa}) = \frac{|U \cap S_{G,j,\kappa}|}{cost(S_{G,j,\kappa})}$ , $\forall S_{G,j,\kappa} \in \mathcal{S}$.
5:     Let $S'$ be the set with maximum $cf(S_{G,j,\kappa})$.
6:     $U := U \setminus S'$;   $S^* := S^* \cup \{S'\}$;
7: end while
```

The Greedy Algorithm 1 employed in DirCast first rounds up all transmission rates R_{ij} to the next higher power of two, and it then uses these rounded values $\widehat{R}_{ij}$ throughout the remainder of the algorithm. It then proceeds like the greedy set cover algorithm in that in every iteration, it chooses the set that has maximum cost-efficiency $cf(S_{G,j,\kappa})$.

3) Analysis: We can prove the following worst-case bound on the performance of our greedy heuristic. The detailed proof is omitted due to lack of space. The time complexity can easily be seen by observing that the only computationally intensive operation is the computation of $cf(S_{G,j,\kappa})$ for all $S_{G,j,\kappa} \in$

$\mathcal{S}$, because this takes time $|S|$ which can be as large as 2^g. However, for any $g \in O(\log n + \log m)$, the time complexity is polynomial. As for the approximation, it can be shown that the set-cover problem over the sets $\mathcal{S}$ forms an upper bound on the problem.

Theorem 2.1: For $g \leq C(\log n + \log m)$, Algorithm A computes in polynomial time a solution to the client association problem that is within a factor of $4 \log n$ of the optimum solution. This is asymptotically optimal.

F. Proactive FEC

Since non-target clients receive packets by listening in promiscuous mode, they may not receive all the packets. To combat these losses, the DirCast server adds FEC packets to the data stream. It uses the well-known Reed-Solomon codes to construct parity packets from a block of recent packets, using typical block sizes of 16 or 32 packets. Since wireless losses at different clients are typically uncorrelated [8], Reed-Solomon codes allow the transmission of optimal amount of parity packets in order for all clients to recover from their losses. While Reed-Solomon codes are computationally expensive, our focus is mostly on the ability of DirCast to adapt to variations in wireless channel conditions and mobility. We could easily replace the encoder with other codes such as Tornado codes that allow for better trade-off between efficiency and computational overhead.

Algorithm 2 MIMD-based Proactive Coding

```
1: if (insufficient parity) /* loss */
2:     if (Parity*2 > loss) Parity = loss
3:     else Parity = Parity *2;
4: if (excessive parity)
5:     if (Parity/2 < (Parity - gain)) Parity -= gain
6:     else Parity = Parity / 2;
```

The DirCast server has to estimate the current level of packet losses seen by the clients subscribed to the multicast group at a particular AP, and proactively send the requisite amount of parity packets in order to reduce latency in loss recovery. The amount of parity packets sent is determined by the losses seen by the clients for previous blocks. We use Algorithm 2 for computing the parity packets, which is based on a multiplicative increase multiplicative decrease (MIMD) algorithm. The rationale behind the choice of an MIMD algorithm is that, we found packet losses exhibited spike-like characteristics[1] in our experimental traces. Thus, the MIMD algorithm helps DirCast react quickly to both sudden increase in losses as well as sudden decrease in losses.

G. Client Feedback

In the discussion above, we assumed that the DirCast central controller has full knowledge of the network. In practice, it needs to know the following items for each client: *(1)* The AP the client is associated with *(2)* The loss rate the the client is currently experiencing for the multicast traffic *(3)* The *unicast* transmission rate a client may get at an AP. It derives this information from periodic feedback sent by the clients.

- The event that triggered the feedback.
- The AP the client is associated with.
- Loss rate of multicast packets. This feedback is given on a per-group basis if the client is associated with multiple multicast groups.
- Average RSSI of packets received from the AP the client is associated with.
- RSSI of beacons heard from other nearby access points

Let us now consider each item in turn. The loss rate of the multicast packets is computed as a running average over past 5 seconds. The average RSSI of the packets received from the AP is computed in a similar manner. To gather RSSI of beacons heard from nearby APs, we rely on the fact that most modern WiFi cards perform periodic background scans to gather this information. The feedback is sent back to the DirCast server as a normal unicast packet.

The DirCast server can use the loss rate information directly. Estimating the transmission rate a client is likely to get at various APs is more difficult. We currently use the RateMap approach described in [19] to convert signal strength information into estimates of transmission rates. The RateMap approach relies on a profile generated from historical data, that tries to correlate signal strength with transmission rate. It is essentially a way to reverse-engineer the autorate algorithm on the AP, with only signal strength information. The results are (obviously) not fully accurate, but they are sufficient for our purpose. We present more details in Section IV.

The feedback from the clients is generally triggered by a periodic timer. However, if at any time, the loss rate exceeds a certain threshold (set to 10% in our current implementation), the client sends a feedback report immediately. The feedback mechanism also helps detect whether the target client has left the network. Imagine a scenario in which the target client simply shuts down or walks away, without first leaving the multicast group. In this case, there are two possibilities. If the client dissociates with the AP, the AP will simply stop forwarding the unicast stream immediately. Even if the client does not disassociate, the AP will generally stop forwarding traffic using some internal timeout. Either way, the stream of unicast packets will stop, and this will result in 100% loss rate for all other clients associated with that AP. The clients will immediately trigger a feedback, and the DirCast server will pick another client to be the target client.

H. Virtual Multicast Interface

Recall that non-target clients receive packets by listening to them in promiscuous mode. Multicast security in IEEE 802.11 is performed using a separate global encryption key. This key is known to all devices in the network and is different from the client's unicast key. The sender encrypts multicast packets using the global key, which is then decrypted at each node that is part of the multicast group. It is challenging to achieve the same level of security in DirCast. The APs will encrypt

[1]Part of the reason for this is specific to DirCast since the AP is performing rate adaptation with respect to one of the clients while losses at the clients in promiscuous mode can vary significantly

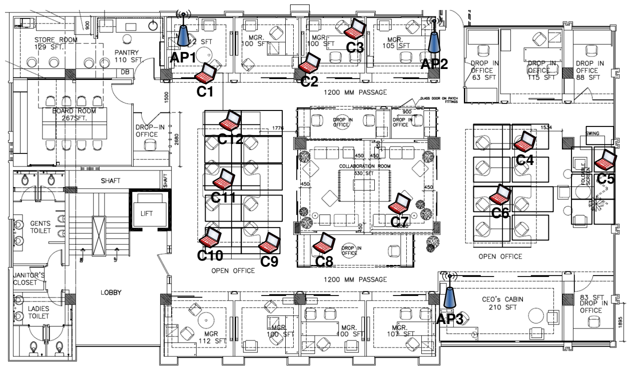


Fig. 2. **Testbed**

packets with the destination client's unicast key and hence other clients in the multicast group will be unable to decode the packets. We note that simple solutions might not work. Revealing the unicast key to other clients will compromise security. Adding logic to the AP such that it uses non-unicast keys for DirCast packets will require modifications to the AP, which is against our design goals.

Our solution achieves the functionality of a global multicast key by leveraging the VirtualWiFi idea from [4]. DirCast creates a VirtualWifi interface at the client for each multicast group that the client is subscribed to. Each virtual interface has a separate MAC address and hence appears to the AP as a separate wireless card. Multicast traffic at the client for each multicast group is routed through the respective virtual multicast interface, while regular unicast traffic is sent over its own distinct virtual interface. Every client shares the security session keys for each of its virtual multicast interfaces with the DirCast server, which then multicasts the keys to the clients of each of the respective multicast groups. The other clients then use the received keys to decode their multicast group's packets that they receive (as unicast) in promiscuous mode. In this manner, we allow each multicast group to have distinct encryption keys which are also separate from the unicast encryption keys of each client.

Note that this technique does not suffer from VirtualWiFi's switching overhead since all the virtual interfaces are associated to the same AP and thus operate on the same WiFi channel. Furthermore, most commercial WiFi cards allow the card to send ACKs for multiple MAC addresses. More importantly, this technique enables multicast security for DirCast without modifying the APs.

III. Implementation

We have built a prototype implementation of DirCast server and DirCast client on Windows Vista OS platform. The DirCast server is implemented as a user-level application that is hosted on a machine on the wired network. The DirCast client software that is installed on the client laptops consists of two components: a filter driver and an user-level daemon. The filter driver sits below the IP layer in the kernel and it intercepts IGMP packets and redirects them to the user-level daemon. The user-level daemon sets up a control channel with the DirCast server on which feedback and other control messages are exchanged. In addition, it also manages WiFi interface configurations and collection of wireless statistics for feedback generation. We disabled the wireless auto-configuration service that is provided by Windows and we managed all wireless configurations such as management of virtual WiFi interfaces, promiscuous mode setup, and AP associations, using appropriate device IO control commands. During a multicast session, the filter driver also takes care of modifying the incoming pseudo-broadcast packets into appropriate multicast packets. The filter driver uniquely identifies a pseudo-broadcast packet using the source IP and the port number, which are notified to the user-level daemon by the DirCast server. The IP header is modified so that the packet appears as multicast packet and it is delivered to the application via the networking stack.

IV. Experimental Evaluation

In this section, we present an experimental evaluation of DirCast over a WiFi LAN testbed with off-the-shelf access points (AP). We have also studied DirCast using extensive simulations, especially to ensure that DirCast can work with a large number of clients. We omit simulation results due to lack of space.

The layout of the testbed used for many of the experiments in this section is shown in Figure 2. The testbed consists of three APs and 12 clients distributed on one floor of a typical office building. The floor has "full" offices along the outer walls, while most of the space in the middle is occupied by cubicles. The experiments were carried out in the 5GHz band (802.11a), since there is an active b/g network in the building. The APs were assigned to different channels.

The section is organized as follows. We first demonstrate the validity of the RateMap approach used by the DirCast controller. Then, we present an experiment with static clients, that demonstrate the overall gains achieved by DirCast. Next, we present a series of experiments that quantify the gain provided by individual components of DirCast. We then characterize performance of DirCast in more dynamic settings. Finally, we present some results related to scalability.

A. Validity of RateMap approach

The association control algorithm uses the RateMap [19] technique to estimate the transmission rate a client may get if it associates with a particular AP. The idea is to look at the signal strength of AP's beacons as seen by the clients, and map the observed signal strength to expected transmission rate. To demonstrate the validity of this approach, we conducted the following experiment. We moved one of the clients at 7 different positions on the map. At each location, the client was stationary for several minutes. The client associated with whichever AP it chose to (normal WiFi operation). We measured the average RSSI of 1000 beacons of AP it had associated with. Then, we did a short unicast data transfer from the AP to the client, and measured the average transmission rate of 200 packets. The results of the experiment are shown in Figure 3. The error bars (both X and Y) represent standard deviation. The figure demonstrates

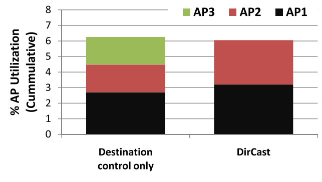
DirCast against Legacy (or normal) Wi-Fi operation using the following experiment.

We use all 12 clients, which are shown in Figure 2. Initially, the clients operated without any DirCast support: i.e. they chose the APs they wished to associate with. The client-AP associations are shown in Table I. Each client then joins a multicast session, which lasts for 60 seconds. We measured the airtime consumed at each of the APs, as well as the average loss rate experienced by the clients. Next, we repeated the same experiment, but with DirCast support enabled. The client-AP associations are shown in Table II. Notice that DirCast association control grouped the clients to just two APs. The fraction of airtime consumed by each of the APs, and the average packet loss rate experienced by the clients is shown in Figures 4. The loss rates are averaged across all clients, and the error bars show the standard deviation.

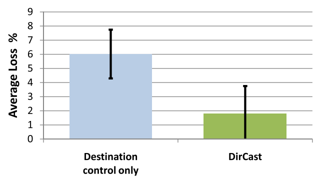
The results show that DirCast reduces airtime utilization by a factor of 8 (from 40% to just over 5%), while also reducing the average loss rate by a factor of 2 (from 4% to under 2%). Thus, DirCast offers significant performance improvement over traditional WiFi.

We note two points about computation of airtime consumption and loss rate. First, the airtime consumed under DirCast *includes* the control traffic generated by clients. Second, the loss rate is the loss rate seen by the application. In case of traditional multicast, it is the same as the PHY-layer loss rate. In case of DirCast, it includes the impact of both retransmissions done for target client as well as the included proactive FEC.

While these results are impressive, they lead to some questions. Recall that DirCast uses several techniques to improve overall performance: pseudo-broadcast, association control, destination control and proactive FEC. What is the contribution

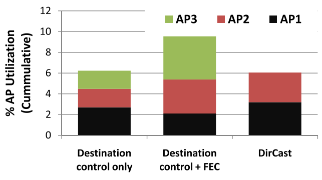


(a) Airtime consumption

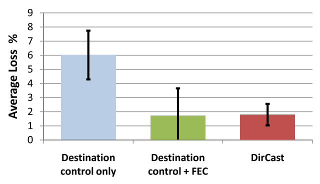


(b) Loss Rate

Fig. 5. **Impact of destination control**



(a) Airtime consumption



(b) Loss Rate

Fig. 6. **Impact of Proactive FEC**

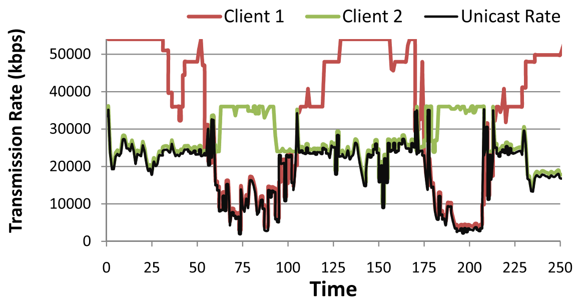


Fig. 7. **Effective multicast data rate in presence of mobility**

affected by noise and channel conditions that are independent of what the target clients are experiencing.

To tease apart impact of destination control from pseudo-broadcast, we would have to retain the mechanism of pseudo-broadcast and transmit to either a randomly selected client, or perhaps the best client. This may reduce the airtime further (because the AP may use higher transmission rate), but it may also increase the average loss rate, since some client may not be able to decode any packets at all. We have not investigated this tradeoff, since complete (or significant) starvation of a client is unlikely to be acceptable to most users.

These results point us to the next question: would it be sufficient to just add proactive FEC, and dispense with association control altogether? To answer this question, we repeat the previous experiment, but we enable proactive FEC. Association control remains disabled. The results of the experiment are shown in Figure 6.

We make a few observations regarding these results. First, note that as expected, we continue to use three APs instead of two. Second, the loss rate is as low as one sees with full DirCast. Thus, proactive FEC is quite effective in combating losses. In other words, proactive FEC is DirCast's primary mechanism for combating losses.

However, note that the airtime utilization is almost double than what it is under DirCast. This is because we when association control is enabled, DirCast central controller realizes that it can make do with just two APs - thereby reducing the airtime utilization. Thus, while most of the reduction in airtime utilization comes because we use unicast, association control also helps, by grouping clients across fewer APs whenever possible.

In summary, these results have shown is the impact of the three individual mechanisms employed in DirCast. Furthermore, we see that all three mechanisms are necessary to obtain the full benefit of DirCast. Notice that these results were obtained under stationary conditions. The natural question to ask is how well does DirCast perform under dynamic conditions. We answer this question below.

D. Performance under dynamic conditions

In a dynamic wireless environment where there is a lot of background traffic, client mobility etc., the DirCast server needs to do two things on a frequent basis. First, it needs to correctly keep track of the "worst" client at each AP, so that it can do proper destination control. Second, it needs to adjust the amount of FEC it adds to the traffic based on the loss rate reported by the clients. We evaluate these two aspects using specially constructed scenarios.

1) Dynamic selection of target client: The feedback mechanism described in Section II allows the DirCast server to track the worst client under dynamic conditions. We illustrate this using the following scenario consisting two clients and one AP. The clients belong to the same multicast session. In addition, we sent periodic unicast probes to both clients, so that we could directly measure the transmission rates used by the AP.

Client 1 is mobile while client 2 is stationary. Initially, the mobile client is closer to the AP and has higher signal quality. The mobile client then walks to the edge of the cell and back to its original location and repeats the movement. Figure 7 shows the effective data rate for the two clients, as well as for the multicast session. The graph shows that the DirCast server correctly picks the client with the lowest transmission rate at any instant using the feedback mechanism. Initially, client 2 is the target until around 50 seconds, when the client 1's data rate falls below client 2's rate and client 1 becomes the new target.

2) Proactive FEC: To evaluate how well proactive FEC works in the presence of dynamically varying channel conditions, we conduct the following experiment.

We establish a streaming multimedia multicast session with a data rate of 512Kbps to two clients associated to the same AP. In this application, 64 1KB packets are sent at periodic intervals every second. DirCast uses a block size of 16 packets and adds parity packets to each block using the redundancy adaptation algorithm described in Section II.

The top and bottom graphs in Figure 8 shows the maximum instantaneous loss rate percentage of two clients without and with coding, respectively. The y-axis shows the amount of loss as a percent for each block. In this experiment, after time = 100 seconds, we purposefully occlude the line-of-sight path of one of the clients, resulting in very high loss rates of 10-60% without coding. One can see from the graph depicting the

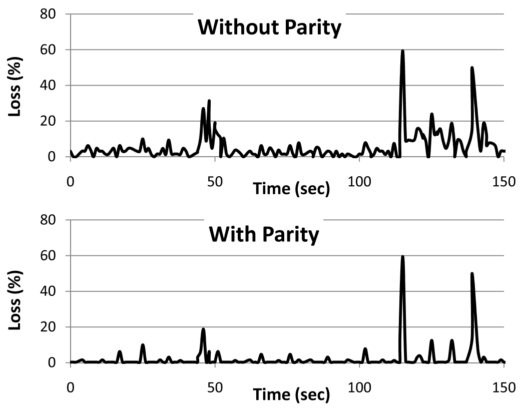


Fig. 8. **Proactive Coding.**

maximum loss sustained with coding that DirCast's MIMD algorithm is able to react quickly to the dynamically varying channel conditions. We see that DirCast reacts to bad channel conditions by increasing parity immediately once the loss rate of a block increases and bringing the client loss rate down quickly. DirCast also reacts quickly to improved channel conditions by reducing parity overhead while maintaining low-loss rates. In this experiment, the average loss rate with and without coding for the first 100 seconds were 3.9% and 0.8% respectively while the loss rate with and without coding for the next 50 seconds, emulating the extremely variable channel conditions, were 10.2% and 4%, respectively.

E. Scalability

There are three potential areas of concerns regarding scalability. First, we need to consider the load on the DirCast server. Second, we need to consider the impact of control traffic introduced by DirCast. Third, we need to consider the additional load placed on the clients by running the DirCast client component. We now address these concerns.

1) Load on DirCast server: We have measured the CPU load on the DirCast server as both the number of clients it supports increases, and the number of multicast sessions it supports increases. On a typical PC-class machines, the load is negligible for up to a dozen clients. We believe that a PC-class machine will be easily able to support a few hundred clients. We omit detailed results due to lack of space. If server load does becomes a concern, we can reduce the load by deploying several servers and making each one responsible for a small number of APs. We are currently exploring this approach.

2) Volume of Control traffic: The control traffic primarily consists of the traffic reports generated by the clients. The extra traffic is quite small. Each client generates, on average, only about 10Kbps of control traffic. The traffic is sent as unicast packets to the DirCast server, so the airtime consumption is low. We note again that the airtime utilization results presented earlier, include this traffic.

3) Client load: The client load primarily comes from having to listen in promiscuous mode. This load varies depending on the amount of background traffic on the channel, but most modern laptops and wireless cards can handle the load easily. When 802.1x-like security protocols are enabled, the clients

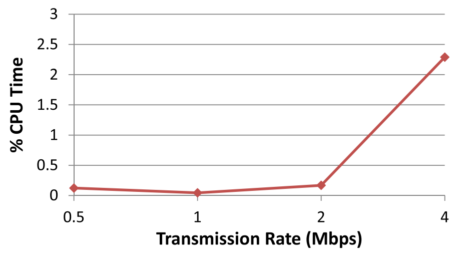


Fig. 9. **Overhead of decryption**

have to decode the multicast packets in software (as explained in Section II-H). The CPU overhead of the software-decoding process varies with the data rate (not transmission rate!) of the multicast session. We measured this overhead for various data rates on a typical laptop client. The results shown in Figure 9. As we see, the overhead is quite low even for high-quality multicast traffic (data rate of 4Mbps).

V. Related Work

There has been a lot of prior work in the areas of both wired and wireless multicast. For most part, we do not discuss wired multicast research in this section, although some results are applicable in wireless arena as well. Even when we focus on wireless multicast alone, it is not possible for us to give full coverage to all the work done in the area. Instead, we focus, by necessity, on a very small sample of the published results that we believe are directly related to our work.

The problem of optimal association of multicast clients to access points has been studied recently in [6]. The authors define three problems, maximizing the number of users (MNU), balancing the load among APs (BLA) and minimizing the load of APs (MLA), show that these problems are NP-hard, devise approximate centralized and distributed algorithms to solve these problems, and using simulations evaluate the performance of these algorithms. They restrict themselves to the case where each user may subscribe to only one multicast flow.

In DirCast we consider a more general optimization problem. In fact , it can be viewed as a generalization of the MLA problem definition that allows clients to subscribe to multiple multicast flows; this allows unicast flows to be easily captured in the formulation as a multicast flow with one group member.

Rate adaptation for broadcast/multicast is a hard problem because of the lack of ack-based feedback in broadcast/multicast transmissions. On the other hand, enabling acks for multicast would result in the ack implosion problem, a large increase in the MAC overhead and the need for sophisticated schemes to coordinate the acks sent by the clients of the multicast group. Several researchers [25], [15] have looked at solving the feedback problem, albeit by changing the 802.11 MAC. In [25], the sender sends a RTS addressed to all its groups members and waits to receive a CTS from them. As long as the sender is able to decode the CTS or detects that the channel is busy during the expected CTS time interval, it proceeds with the data transmission. In [15], the authors provide a solution to the feedback problem by electing a leader that is responsible for generating an ACK. Packet losses at other group members are communicated by sending negative ACKs (that may collide with the ACKs by the leader) triggering retransmission by the access point. Apart from requiring changes to the 802.11 MAC, they cater only to the case of single-rate wireless LANs. Authors in [7] develop an algorithm for achieving low latency broadcast in multi-rate mesh networks assuming that the MAC layer of future wireless meshes would support rate adaptation for multicast. They point out that the dual-tone-based MAC proposed in [5] can be adapted for designing a multi-rate MAC. In contrast, DirCast requires no changes to the 802.11 MAC or to the wireless access points.

There has been a lot of work [26], [18], [17], [24], [22], [23], [21], [3] on reliable multicast and use of FEC on improving reliability. Authors in [26] combine FEC with channel estimation and NAK-based feedback for retransmissions and show using simulations that their technique increases reliability without sacrificing channel efficiency. In [18], the authors use a network-based proxy that implements block erasure codes [17] and NAK feedbacks from receivers to enable adaptive FEC support for multicast in collaborative computing applications. RMDP [22] is a FEC-based reliable multicast protocol that uses both FEC and retransmissions for use over the MBone and wireless mobile networks. Recently, the authors in [23], evaluate the effectiveness of using XOR-based network-coding for retransmission in wireless broadcast/multicast applications.

DirCast uses well-known and optimal Reed-Solomon codes for error correction and evaluate its efficacy in a practical setting where DirCast needs to adapt to dynamic channel conditions and client mobility.

Several application layer multicast protocols [2], [1] have been proposed, since IP multicast is not widely used in the Internet. In [2] data delivery occurs over an overlay multicast network consisting of end-hosts. In [1], use of application-layer servers is proposed. Although such protocols consume lower bandwidth compared to using multiple unicast flows to every client, they are not designed for wireless multicast. In WLAN setting, forwarding data between wireless clients can unnecessarily increase airtime, since every wireless transmission goes through the AP.

There is a lot of work [10], [13], [9] to improve performance of wireless multicast in the context of mobile and multi-hop wireless networks. In [10], multicast transmission happens over an overlay network that efficiently adapts to changes in network topology with minimal control traffic overhead. In [13], extensions to IEEE 802.11 MAC have be proposed to reduce multicast packet losses using a distributed channel reservation protocol. In contrast, DirCast focuses on single hop, infrastructure wireless networks and requires no changes to the MAC protocol.

VI. Conclusion and Future work

In this paper, we presented the DirCast system for efficient transmissions of multicast traffic over WiFi networks. Unlike many previous schemes, DirCast does not require any changes to the 802.11 MAC, and can be deployed in existing WiFi networks. DirCast uses pseudo-broadcast, and augments it with destination control, association control and proactive FEC to improve multicast performance. Using testbed experiments we demonstrated the efficacy of our approach. In future work, we plan to investigate the impact of mobility on DirCast, and also incorporate power constraints in the decision-making algorithms.

References

[1] E. Al-Shaer, H. Abdel-Wahab, and K. Maly. Application-Layer Group Communication Server for Extending Reliable Multicast Protocols Services. In *ICNP*, 1997.

[2] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable Application Layer Multicast. In *SIGCOMM*, 2002.

[3] G. Cao and Y. Wu. Reliable Multicast Via Satellites. In *ITCC*, 2002.

[4] R. Chandra, V. Bahl, and P. Bahl. MultiNet: Connecting to multiple IEEE 802.11 networks using a single wireless card. In *INFOCOM*, 2004.

[5] P. Chaporkar, A. Bhat, and S. Sarkar. An Adaptive Strategy for Maximizing Throughput in Mac Layer Wireless Multicast. In *MobiHoc*, 2004.

[6] A. Chen, D. Lee, and P. Sinha. Optimizing Multicast Performance in Large-Scale WLANs. In *ICDCS*, 2007.

[7] C. Chou and A. Misra. Low Latency Multimedia Broadcast in Multi-rate Wireless Meshes. In *IEEE Workshop on Wireless Mesh Networks*, 2005.

[8] D. Dujovne and T. Turletti. Multicast in 802.11 WLANs: An Experimental Study. In *ACM MSWIM*, 2006.

[9] A. Garyfalos and K. Almeroth. A Flexible Overlay Architecture for Mobile IPv6 Multicast. *IEEE JSAC*, November 2005.

[10] C. Gui and P. Mohapatra. Overlay Multicast for MANETs using Dynamic Virtual Mesh. *Wireless Networks*, 13(1):77–91, 2007.

[11] Y. He, R. Yuan, X. Ma, J. Li, and C. Wang. Scheduled PSM for Minimizing Energy in Wireless LANs. In *ICNP*, 2007.

[12] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda. Performance anomaly of 802.11 b. In *INFOCOM*, 2003.

[13] S. Jain and S. Das. MAC layer Multicast in Wireless Multihop Networks. In *COMSWARE*, 2006.

[14] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft. Xors in the air: Practical wireless network coding. In *SIGCOMM*, 2006.

[15] J. Kuri and S. Kasera. Reliable Multicast in Multi-Access Wireless LANs. *Wireless Networks*, 2001.

[16] M. Lopez. The State of North American Enterprise Mobility in 2006. *Forrester Research*, 2006.

[17] A. McAuley. Reliable Broadband Communications Using Burst Erasure Correcting Code. In *SIGCOMM*, 1990.

[18] P. McKinley, C. Tang, and A. Mani. A Study of Adaptive Forward Error Correction for Wireless Collaborativecomputing. *IEEE Transactions on Parrallel and Distributed Systems*, Sept 2002.

[19] R. Murty, R. Chandra, J. Padhye, A. Wolman, and B. Zill. Designing High Performance Enterprise Wi-Fi Networks. In *NSDI*, 2008.

[20] R. Pang, M. Allman, M. Bennett, J. Lee, and V. Paxson. A First Look at Modern Enterprise Traffic. In *ACM IMC*, 2005.

[21] P. Radoslavov, C. Papadpoulos, R. Govindan, and D. Estrin. A Comparison of Application-Level and Router-Assisted Hierarchical Schemes for Reliable Multicast. In *INFOCOM*, 2001.

[22] L. Rizzo and L. Vicisano. RMDP: an FEC-based Reliable Multicast protocol for wireless environments. *Mobile Computer and Communication Review*, 1998.

[23] E. Rozner, A. Iyer, Y. Mehta, L. Qiu, and M. Jafry. ER: Efficient Retransmission Scheme for Wireless LANs. In *ACM CoNEXT*, 2007.

[24] D. Rubenstein, S. Kasera, D. Towsley, and J. Kurose. Improving Reliable Multicast Using Active Parity Encoding Services. *Computer Networks Journal*, January 2004.

[25] K. Tang and M. Gerla. MAC Layer Broadcast Support in 802.11 Wireless Networks. In *MILCOM*, 2000.

[26] Y. Xu and T. Zhang. An Adaptive Redundancy Technique for Wireless Indoor Multicasting. In *ISCC*, 2000.