Compute-and-Forward for Random-Access: The Case of Multiple Access Points

Shwan Ashrafi, Chen Feng, Member, IEEE, and Sumit Roy, Fellow, IEEE

Abstract—Compute-and-forward (C&F) recently finds new applications in random-access networks focusing on the single access point (AP) scenario. In this paper, we extend the use of C&F from the single AP scenario to the multi-AP scenario. To achieve this, we identify two major challenges and propose two novel solutions. First, we introduce an AP cooperation problem and develop an efficient distributed algorithm. Second, we introduce a joint channel estimation and active user recovery problem and propose a solution based on spare recovery techniques. In addition, we provide accurate throughput and delay expressions for C&F-based CSMA random-access protocols. These expressions, together with our trace-driven simulations, demonstrate the significant advantages of C&F-based CSMA over conventional CSMA in the multi-AP scenario.

Index Terms—Compute-and-forward, inhomogeneous CSMA, multiple access points, performance analysis

I. INTRODUCTION

Compute-and-forward (C&F) has emerged as a powerful physical-layer network coding technique [1], [2]. C&F enjoys two significant theoretical advantages: First, it achieves close-to-optimal information-theoretic performance in various important channel models, such as interference channels [3], two-way relay channels [4], and Gaussian multi-hop networks [5]. Second, it admits highly efficient encoding and decoding methods, with a favourable performance-complexity tradeoff [6]. Hence it has attracted attention from several communities - communications [7], information theory [3], [4], and coding theory [6] - that have sought to advance fundamentals and system applications.

In our previous work [8] and [9] we sought to integrate C&F within a typical random-access scenario such as slotted-ALOHA and CSMA, with a particular focus on a single WLAN network, i.e., one access point (AP) with clients. In this paper, we aim to study the theoretical benefits of C&F in a network consisting of multiple APs that deploy CSMA. This scenario is motivated by the specific (and one that is largely work-in-progress) challenge of optimizing dense WLAN deployments in the enterprise, characterized by overlapping coverage and corresponding increase in co-channel interference. Since the multiple co-channel WLAN networks lie within a single collision domain, throughput does not scale merely by increasing density\(^{1}\) and hence alternative solutions - notably those based on enabling multi-packet co-channel reception are necessary.

Current enterprise WLANs employ a RF controller device for network management; this accepts network data from the APs it controls and in turn, can push decisions down to them\(^{2}\). Such intelligent controllers effectively enable AP collaboration that has the potential for solving the network scaling problem in traffic hot-spot locations. Our proposed C&F-based multi-packet reception is readily mapped to the system mock-up shown in Fig. 1 and represents a novel contribution for throughput enhancement as explained next. Here each AP can choose to decode a linear combination of the transmitted packets that it receives. The decoded linear combinations are sent to a central controller where the original packets can be recovered by solving a system of linear equations as long as these two combinations are linearly independent.

Although the basic idea of using C&F in a multi-AP setting seems straightforward at first glance, there are two major technical challenges in the design of practical protocols. First, unlike the single-AP case where the AP resolves collisions by itself, APs in a multi-AP system need to collaborate with each other. This poses a new question: which linear combinations shall each AP decode? Intuitively, if each AP in Fig. 1 simply decodes the linear combination with the highest reliability, then the linear combinations collected at the central controller could be linearly dependent, making it impossible to recover the original packets. In other words, the APs should cooperate with each other to ensure that the decoded linear combinations are as reliable as possible and the collection of combinations can be used to reconstruct the original packets.

\(^{1}\)In such scenarios, a single user accessing the channel will cause all other co-channel users to remain silent to avoid interference and packet collision.

\(^{2}\)Cisco 2500 Controller Series has a baseline configuration that controls up to 75 APs.
In Section IV, we formally define this AP cooperation problem and develop a distributed solution which achieves the best possible reliability under the linear independence (full-rank) constraint. Our distributed solution is computationally efficient, especially when the number of colliding packets is less than 5.

Second, the APs need to identify colliding users and estimate their channel conditions before decoding linear combinations. This problem is of great importance in practice, but has been neglected in our own previous work [8], [9]. In this work, we note that although the number of potential users can be very large in a dense network with multiple APs, the number of colliding users is typically only a small fraction under random-access protocols. This sparsity allows us to draw upon sparse recovery techniques developed in compressed sensing to perform channel estimation.

In Sec. V, we formally define the channel estimation and active user recovery problem. We then propose an algorithm based on sparse recovery techniques that makes use of Reed-Muller signatures and has computational complexity that is sub-linear in the number of users.

Once the above two challenges have been addressed, we are able to provide accurate throughput and delay expressions for C&F-based CSMA protocols through a unified perspective on the single-AP and multi-AP case. Finally, we conduct extensive simulations driven by a real-world channel condition using traces collected from a 44-node wireless network [10]. Our simulation results show that C&F-based CSMA achieves a multi-fold increase in network throughput over a wide range of system parameters, and can also greatly reduce average delays.

The main contributions of this work can be summarized as follows:

1) Two major challenges are identified in applying C&F to random-access networks with multiple APs.
2) The AP cooperation problem is formulated and a distributed solution is developed which is computationally efficient (when the number of colliding packets is less than 5).
3) The joint channel estimation and active user recovery problem is formulated and an algorithm based on sparse recovery techniques is proposed.
4) Trace-driven simulations are conducted to demonstrate the significant advantages of using C&F in multi-AP scenarios.

We believe these contributions provide novel solutions and insights towards a practical design of random-access protocols with C&F in the multi-AP scenario. Our work is complementary to an exciting development of cloud radio access networks (C-RAN) [11] which are expected to play an important role in 5G cellular systems. The capacity analysis of the uplink C-RAN can be found in, e.g., [12] and [13], which are based on network compress-forward [14] and noisy network coding [15], respectively. An interesting case of the uplink C-RAN with oblivious relaying (i.e., the relays don’t know the information of the users’ codebooks) has been analyzed in [16]. Very recently, C&F has been applied to the uplink C-RAN by Aguerri and Zaidi, who have developed effective coding schemes by combining C&F with compress-forward and noisy network coding [17].

Although the multi-AP scenario looks similar to the uplink C-RAN scenario, they differ in the following two fundamental ways. First, the multi-AP scenario is an instance of random-access networks where users observe random data arrivals, whereas the uplink C-RAN scenario is an instance of two-hop relay networks where data is always available at the encoders. Second, the multi-AP scenario is based on slow fading channels (where the outage probability rather than the capacity is an important metric), whereas the uplink C-RAN scenario is based on AWGN channels (where the capacity is an important metric). For the above reasons, our solution is quite different from the centralized solutions in [17], and, in particular, our solution only involves point-to-point coding schemes.

II. A PRIMER ON COMPUTE-AND-FORWARD

In this section, we briefly describe the basic ideas underlying C&F. As the most important ingredient of C&F, a lattice is a discrete set of points in a (multidimensional) Euclidean space with the fundamental property that the sum and the difference of any two lattice points are also lattice points [18]. That is, if $\mathbf{x}_1$ and $\mathbf{x}_2$ are two lattice points, so are $\mathbf{x}_1 + \mathbf{x}_2$ and $\mathbf{x}_1 - \mathbf{x}_2$. This implies that every integer linear combination $a_1 \mathbf{x}_1 + a_2 \mathbf{x}_2$ ($a_1, a_2 \in \mathbb{Z}$) is also a lattice point. More generally, every integer linear combination of a set of lattice points is again a lattice point. This is the key property that C&F exploits, as we will see shortly.

![Fig. 2. Illustration of the architecture of compute-and-forward decoding.](image)
received signal $y$. In order to do so, the receiver applies a
scaling operation $\alpha y = \sum_{\ell=1}^{n} a_\ell x_\ell + \alpha z$ to steer the “effective”
channel gains $\{a_\ell\}$ towards integer values, without unduly
amplifying the noise $z$. Let $a_\ell$ be the closest integer to $a_\ell$, for $\ell = 1, \ldots, n$. Then, we have

$$\alpha y = \sum_{\ell=1}^{n} a_\ell x_\ell + \mathbf{n}$$

where $\mathbf{n} = \sum_{\ell=1}^{n} (a_\ell - a_\ell) x_\ell + \alpha \mathbf{z}$ is the effective channel
noise. The receiver can decode $\sum_{\ell=1}^{n} a_\ell x_\ell$ from $\alpha y$ by using a (standard) lattice decoder. Intuitively, the decoding is correct
if the effective noise $\mathbf{n}$ is sufficiently small.

To characterize the decoding performance, [1] introduces
the following computation rate $R_{\text{comp}}(\mathbf{a}, \alpha; \mathbf{h})$ for any scalar $\alpha \in \mathbb{R}$ and any coefficient vector $\mathbf{a} = (a_1, \ldots, a_n)^T$:

$$R_{\text{comp}}(\mathbf{a}, \alpha; \mathbf{h}) = \frac{1}{2} \log \left( \frac{P}{\alpha^2 + P \|\mathbf{a} - \mathbf{h}\|^2} \right)$$

(2)

where $P$ is the power constraint and $\mathbf{h} = (h_1, \ldots, h_n)^T$ and
$\log^+(x) = \max\{0, \log(x)\}$. The main result of [1] says that the
receiver can decode an integer combination $\sum_{\ell=1}^{n} a_\ell x_\ell$ with coefficient vector $\mathbf{a}$ and scalar $\alpha$ as long as the computation rate $R_{\text{comp}}(\mathbf{a}, \alpha; \mathbf{h})$ is larger than the message rate $R$, i.e.,

$$R_{\text{comp}}(\mathbf{a}, \alpha; \mathbf{h}) > R. \text{ This has two implications as follows.}$$

1) For any given coefficient vector $\mathbf{a}$, the optimal scalar $\alpha^*$ that maximizes the computation rate $R_{\text{comp}}(\mathbf{a}, \alpha; \mathbf{h})$ is given by

$$\alpha^* = \frac{P \mathbf{h}^T \mathbf{a}}{1 + P \|\mathbf{h}\|^2}$$

and the resulting best computation rate $R_{\text{comp}}(\mathbf{a}; \mathbf{h})$ is given by

$$R_{\text{comp}}(\mathbf{a}; \mathbf{h}) = \frac{1}{2} \log \left( \frac{1 + P \|\mathbf{h}\|^2}{(1 + P \|\mathbf{h}\|^2) \|\mathbf{a}\|^2 - P (\mathbf{h}^T \mathbf{a})^2} \right).$$

2) If there are multiple coefficient vectors whose computation rates are larger than $R$, the receiver can recover multiple integer combinations of the transmitted signals.

Remark 1. The above results assume that the length of the codewords is asymptotically long. In reality, the codeword length is constrained by the coherence time. For finite codeword length, the larger the difference $R_{\text{comp}}(\mathbf{a}; \mathbf{h}) - R$, the smaller the error probability. Hence, the difference $R_{\text{comp}}(\mathbf{a}; \mathbf{h}) - R$ indicates the “reliability” of the decoding.

Once an integer combination $\sum_{\ell=1}^{n} a_\ell x_\ell$ of transmitted signals is decoded, it can be mapped to a linear combination $\sum_{\ell=1}^{n} a_\ell w_\ell$ of the corresponding packets (see, e.g., [6] for details). Finally, we emphasize again that all the above results can be easily extended to the case of multi-antenna systems with complex-valued channel gains.

III. System Model and Existing Results

In this section, we present our system model of C&F for random-access networks. We then review some of our previous results on the single-AP scenario [8], [9].

A. System Model

Consider a random-access network with $N$ users and $K$ APs. Each user belongs to one of the $V$ possible classes $\mathcal{V} = \{1, \ldots, V\}$. Let $N_v^{(N)}$ be the number of users in class $v \in \mathcal{V}$. We assume that all class-$v$ users have identical arrival rate and transmission probability. Assume that time is slotted and data packets are of constant length requiring $\kappa$ time slots. Each user is equipped with an infinite buffer for storing packets in a FIFO manner. Data packets arrive at the buffer of a class-$v$ user according to a Bernoulli process with rate $\lambda_v^{(N)}$. That is, at each time slot, a new packet arrives at the buffer of a class-$v$ user with probability $\lambda_v^{(N)}$. The arrival processes are assumed to be independent across users.

In CSMA random-access protocols, a class-$v$ user senses the channel continuously until it finds the channel in idle state and then transmits a packet with probability $p_v^{(N)}$ (if its buffer is nonempty). Note that the behavior of this user is independent of the behavior of other users in the network. So, if the channel is in busy state, there is either a packet transmission or a packet collision. If a packet transmission is successfully decoded by one AP, an ACK will be sent to all the users. If a packet collision occurs, APs decode linear combinations of transmitted packets and then forward them to a central decoder. The central decoder can recover the original packets if these decoded linear combinations are full rank.

The AP decoding process can be abstracted by a sequence of success probabilities $(q_1, q_2, \ldots, q_M)$ defined as follows. Suppose that there are $n$ active users contending the channel. $q_n$ denotes the probability that the central decoder successfully recovers the original packets. Furthermore, $q_n = 0$ for $n > M$ where $M$ is a threshold determined by practical constraints. Clearly, $q_n$ depends on the channel condition as well as the way each AP chooses its linear combination.

B. Existing Results

Using mean-field analysis, we have derived asymptotically exact throughput and delay expressions for the single-AP scenario in [9] and [19]. Our key observation is that in the large-systems regime, the queue evolution of a class-$v$ user can be viewed as a discrete-time random process with packet arrival rate $\lambda_v^{(N)}$ and utilization probability $\rho_v^{(N)}$ (i.e., the probability that the queue is non-empty in a time slot).

Define a super slot as either an idle period or a busy period which is the number of time slots required for the successful transmission of a packet ($\kappa$ time slots) with its overheads, e.g., an acknowledgment (ACK) packet. We assume that an idle period is exactly one time slot and a busy period consists of $\tau$ time slots. For any fixed $N$ denote by $N_v^{(N)}$ the number of class-$v$ users and let $N_v^{(N)} \to \beta_v$, as $N \to \infty$.

Let $\rho^{(N)} = \left(\rho_1^{(N)}, \ldots, \rho_V^{(N)}\right)$. Then, the average throughput (rate function) of a class-$v$ user is given by

$$R_v\left(\rho^{(N)}\right) = \frac{P_v}{\rho^{\text{idle}}\left(\rho^{(N)}\right)} + \tau \left(1 - \rho^{\text{idle}}\left(\rho^{(N)}\right)\right),$$

(3)

where

$$\rho^{\text{idle}}\left(\rho^{(N)}\right) \triangleq \prod_{v=1}^{V} \left(1 - \rho_v^{(N)} p_v^{(N)}\right)^{N_v^{(N)}}$$

(4)
is the probability that a super slot is idle, and
\[
P_v(\rho^{(N)}) \triangleq \sum_{\beta_1, \ldots, \beta_N \geq 0} \frac{N_v}{N_v^N} \beta_1^{n_1} \cdots \beta_N^{n_N} \prod_{v=1}^V \frac{1}{N_v} \tag{5}
\]
is the probability that a packet of a given class-\(v\) user is successfully transmitted in a super slot. Note that when \(V = 1\) (i.e., a single class), our system model reduces to the homogeneous persistent CSMA system. To better understand Eq. (5), one shall notice that the product term gives the probability that there are \(n_v\) active users from class \(j\) transmitting their packets simultaneously and the summation (over all combinations of \(n_j\)’s) is the probability that a packet from a class-\(v\) user is successfully decoded (the probability of successful decoding is uniformly distributed among \(n_v\) class-\(v\) users, hence the term \(n_v/N_v^{(N)}\)).

1) Aggregate Throughput:

**Proposition 1.** [9] The aggregate throughput of the inhomogeneous persistent CSMA with C&F in a \(V\)-class network can be approximated by
\[
R_{\text{CSMA}} = \sum_{v=1}^V N_v^{(N)} R_v(\rho^{(N)}) + o(1)
\]
where the \(o(1)\) term is understood as \(N \to \infty\). In particular, the saturated throughput is given by \(\sum_v N_v R_v(1)\), where \(1\) is the all-one vector of length \(V\).

In order to apply Proposition 1, we need to obtain the value of the utilization probabilities \(\rho_v^{(N)}\) for any given \(\{\lambda_v^{(N)}\}\) and \(\{\rho_v^{(N)}\}\). This can be done by approximating \(\rho_v\) with the limiting utilization probabilities when \(N\) tends to infinity. Specifically, we set \(\lambda_v = \lambda_v^{(N)} N\) and \(\tilde{\rho}_v = \rho_v^{(N)} N\) and solve the system of equations given in (7) to obtain a solution \(\rho_v\) which can be achieved by using the algorithm presented in [9]. We have
\[
\tilde{\lambda}_v = \rho_v \tilde{\rho}_v \chi(y(\rho)) e^{-\gamma(\rho)} e^{-\gamma(\rho)} + \tau (1 - e^{-\gamma(\rho)})
\]
where
\[
\chi(x) = q_1 + \frac{q_2}{x^2} + \frac{q_3}{x^3} + \cdots + \frac{q_M}{(M-1)!} x^{M-1}.
\]
and \(\gamma(\rho) = \sum_{v=1}^V B_v \rho_v P_v \rho_v\).

2) Total Packet Delay: Total delay of a packet is defined as the time it takes for the packet to be successfully decoded after it arrives in the buffer of a user.

**Proposition 2.** [9] Suppose that, \(\rho_v < 1\), i.e., the system is globally stable [9]. Total delay of a packet in the homogeneous persistent CSMA system with C&F is
\[
T_v,\text{CSMA} = \frac{\rho_v}{1 - \tau} + \frac{\tau - 1}{2} (1 - \rho_v \rho_v) + o(1)
\]

As shown in [9], aggregate throughput and delay expressions are asymptotically exact when the number of users tends to infinity, and are very accurate even for systems with a small number of users. The reader is referred to [9] for the general case of \(V > 1\) and further details.

C. Extension to the Multi-AP Scenario

As stated in the previous section, the success probabilities (\(q_1, q_2, \ldots, q_M\)) “abstract” the physical-layer behavior of AP cooperation. In other words, we can use the results in Propositions 1 and 2 to characterize the throughput and delay performances of C&F-based CSMA systems with \(K\) APs as long as we know how to obtain \(\{q_v\}\). When \(K = 1\), the single AP can simply use existing methods (e.g., Algorithm 1 in [6] and Section III.C in [20]) to choose the “best” linear combinations to decode. However, when \(K > 1\), it is unclear how APs should choose their linear combinations in a distributed fashion. In addition, each AP needs to identify colliding users and estimate their channel conditions before decoding linear combinations. This problem is largely overlooked in the previous work [8], [9]. We will address these two problems in the following sections.

IV. C&F FOR THE MULTI-AP CASE: AP COOPERATION

In this section, we first define the AP cooperation problem and develop a distributed solution.

A. AP Cooperation Problem

As before, consider a network with \(N\) users and \(K\) APs. Suppose that there are \(n\) active users involved in a packet collision, and the received signal at the \(k\)th AP is given by
\[
y_k = \sum_{v=1}^n h_k x_v + z_k
\]

Upon receiving \(y_k\), the \(k\)th AP decodes as many linear combinations as needed. Let \(L_k\) be the number of linear combinations decoded by the \(k\)th AP. Let \(a^{k}_i\) be the coefficient vector for the \(i\)th decoded linear combination, where \(1 \leq i \leq L_k\). Let \(A_k\) be a matrix with \(a^{k}_i\) as its \(i\)th column. Let \(b_k = [h_{k1}, \ldots, h_{kn}]^T\) be the channel-gain vector. Then the difference \(\min R_{\text{comp}}(a^{k}_i; b_k) - R\) indicates the reliability of the decoding, as explained in Sec. II. For notational convenience, if the \(k\)th AP decodes nothing, we set \(L_k = 0\). \(A_k\) be an empty matrix, and \(\min R_{\text{comp}}(a^{k}_i; b_k) = \infty\).

Now, let \(A = [A_1, \ldots, A_K]\). Clearly, if \(A\) is of rank \(n\), then the central decoder can recover all the original packets by solving a system of linear equations. This suggests the following optimization problem:
\[
\max_{k=1, \ldots, K} \min_{i=1, \ldots, L_k} R_{\text{comp}}(a^{k}_i; b_k) \tag{9}
\]
subject to \(\text{rank}(A) = n\).

**Remark 2.** Problem (9) maximizes the worst-case reliability of AP decoding. This applies to both the infinite-length regime and the finite-length regime.

B. Our Distributed Solution

Here, we develop a distributed algorithm for the AP cooperation problem (9), which consists of two steps. In the first step, the \(k\)th AP computes a so-called dominant solution as defined in [6]:
\[
\Omega_k = \{a^{(1)}_k, a^{(2)}_k, \ldots, a^{(n)}_k\} \tag{10}
\]
where \( a_k^{(1)}, \ldots, a_k^{(n)} \) have the following property
\[
\begin{align*}
    a_k^{(1)} &= \arg \max \{ R_{\text{comp}}(a; h_k) \mid a \text{ is nonzero} \} \\
    a_k^{(2)} &= \arg \max \{ R_{\text{comp}}(a; h_k) \mid a \text{ and } a_k^{(1)} \text{ are linearly indep.} \}
\end{align*}
\]
and so on. That is, \( a_k^{(1)} \) has the largest computation rate among all non-zero vectors, and \( a_k^{(2)} \) has the largest computation rate among all vectors linearly independent of \( a_k^{(1)} \), and so on. Moreover, we have
\[
R_{\text{comp}}(a_k^{(1)}; h_k) \geq R_{\text{comp}}(a_k^{(2)}; h_k) \geq \cdots \geq R_{\text{comp}}(a_k^{(n)}; h_k),
\]
as shown in [6].

When \( n \leq 4 \), this step can be implemented by using the lattice-reduction method proposed in [21], which is remarkably fast compared to alternative methods such as the Fincke-Pohst method used in [7]. When \( n > 4 \), this step can be implemented by using the greedy search algorithm presented in [6].

In the second step, the destination collects all the solution sets \( \Omega_k \), \( k \in \{1, \ldots, K\} \), and then tries to pick up \( n \) vectors from the solution sets such that these \( n \) vectors are linearly independent and the minimum value of their computation rates is maximized.

**Example 1** \((n = 2 \text{ and } K = 3)\). Suppose that
\[
\begin{align*}
    \Omega_1 &= \left\{ a_1^{(1)} = (1, 1)^T, a_1^{(2)} = (0, 1)^T \right\}, \\
    \Omega_2 &= \left\{ a_2^{(1)} = (1, 1)^T, a_2^{(2)} = (0, 1)^T \right\}, \\
    \Omega_3 &= \left\{ a_3^{(1)} = (1, 0)^T, a_3^{(2)} = (2, 1)^T \right\}.
\end{align*}
\]
Suppose that the computation rates are ordered as follows
\[
a_1^{(1)} > a_2^{(1)} > a_3^{(1)} > a_2^{(2)} > a_3^{(2)}
\]
where \( a_1^{(1)} > a_2^{(1)} \) means \( R_{\text{comp}}(a_1^{(1)}; h_1) \geq R_{\text{comp}}(a_2^{(1)}; h_1) \).

**Example 2** \((n = 3 \text{ and } K = 3)\).
\[
\begin{align*}
    \Omega_1 &= \left\{ a_1^{(1)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \ a_1^{(2)} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \ a_1^{(3)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right\}, \\
    \Omega_2 &= \left\{ a_2^{(1)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \ a_2^{(2)} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \ a_2^{(3)} = \begin{bmatrix} -1 \\ 0 \end{bmatrix} \right\}, \\
    \Omega_3 &= \left\{ a_3^{(1)} = \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \ a_3^{(2)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \ a_3^{(3)} = \begin{bmatrix} 1 \\ -2 \end{bmatrix} \right\}.
\end{align*}
\]
Suppose that the computation rates are ordered as follows
\[
a_2^{(1)} > a_2^{(2)} > a_3^{(1)} > a_3^{(2)} > a_2^{(3)} > a_1^{(1)} > a_1^{(2)} > a_3^{(3)} > a_1^{(3)}.
\]
Then, the second step will output \( A = \left[ a_2^{(1)}, a_2^{(2)}, a_3^{(2)} \right] \).

Based on the above examples, we are ready to present an algorithm for the second step as shown in Algorithm 1.

**Algorithm 1** The Second Step of Our Distributed Solution

**Input:** \((\Omega_1, \ldots, \Omega_K)\) together with a chain of the coefficient vectors in the non-increasing order of their computation rates.

**Output:** Matrix \( A \).

1. **Initialization.** Set \( \ell = 0 \). Let \( A \) be an empty matrix. Set \( \text{rank}(A) = 0 \).
2. **while** \( \text{rank}(A) < n \) **do**
3. \( \ell = \ell + 1 \).
4. **if** the span of \( A \) doesn’t include the \( \ell \)th coefficient vector in the chain **then**
5. Augment \( A \) by adding the \( \ell \)th vector as a new column.
6. **end if**
7. **end while**
8. Reorder the columns of \( A \) to ensure the format \( A = [A_1, \ldots, A_K] \).

**C. Proof of Correctness**

Let \( A^* = [A_1^*, \ldots, A_K^*] \) be an optimal solution to Problem (9). Let \( A = [A_1, \ldots, A_K] \) be an output of Algorithm 1. We will show that the solution \( A^* \) is no better than \( A \). To this end, we consider two cases as follows.

1. **For all** \( k \), the columns of \( A_k^* \) belong to \( \Omega_k \). In this case, every column of \( A^* \) is in the chain of the coefficient vectors (which is an input to Algorithm 1). Let \( a_k^* \) be the column of \( A^* \) with the largest order (i.e., the smallest computation rate) in the chain. Suppose that \( a_k^* \) is the \( \ell \)th coefficient vector in the chain. Then the first \( \ell \) vectors in the chain must span the entire space (because otherwise \( \text{rank}(A^*) < n \)). Hence, Algorithm 1 will stop at \( \ell \) or earlier. That is, \( A^* \) is no better than \( A \).

2. **There exists some** \( k \) such that a column of \( A_k^* \) doesn’t belong to \( \Omega_k \). In this case, we will show that we can always replace \( A_k^* \) with some \( A_k \) whose columns belong to \( \Omega_k \) and the new solution is no worse than \( A^* \). This allows us to reduce Case 2) into Case 1).

For ease of presentation, we assume that \( a_k^i \) is the only coefficient vector in \( A_k^* \) such that \( a_k^i \notin \Omega_k \). (This assumption can be easily relaxed through mathematical induction). Recall that \( \Omega_k = \left\{ a_1^{(1)}, a_2^{(1)}, \ldots, a_n^{(1)} \right\} \). Let \( i \) be the smallest integer such that the span of \( \left\{ a_k^{(1)}, \ldots, a_k^{(i)} \right\} \) contains \( a_k^i \). First, we will show that
\[
R_{\text{comp}}(a_k^{(i)}; h_k) \geq R_{\text{comp}}(a_k^i; h_k)
\]
in Lemma 3. Second, we notice that there exists some \( i' \) (\( 1 \leq i' \leq i \)) such that by replacing \( a_k^{(i)} \) with \( a_k^{(i')} \), the new A is full rank, because otherwise \( \text{rank}(A^*) < n \). Since
\[
R_{\text{comp}}(a_k^{(i')}; h_k) \geq R_{\text{comp}}(a_k^{(i)}; h_k) \geq R_{\text{comp}}(a_k^i; h_k),
\]
the new solution is no worse that the original solution \( A^* \). This completes the proof.

**Lemma 3.** \( R_{\text{comp}}(a_k^{(i)}; h_k) \geq R_{\text{comp}}(a_k^i; h_k) \).

**Proof.** Recall that \( i \) is the smallest integer such that the span of \( \left\{ a_k^{(1)}, \ldots, a_k^{(i)} \right\} \) contains \( a_k^i \). This implies that \( a_k^i \) is linearly
independent of \( \{ a_k^{(1)}, \ldots, a_k^{(i-1)} \} \), because otherwise the span of \( \{ a_k^{(1)}, \ldots, a_k^{(i-1)} \} \) contains \( a_k^i \). Now, by the definition of \( a_k^i \), we have \( R_{\text{comp}}(a_k^i; h_k) \geq R_{\text{comp}}(a_k^i; h_k) \). □

Finally, we note that we can achieve better performance by using successive C&F [20], [22] in a similar way as in [23]. Specifically, in the first step, the \( k \)-th AP still computes the dominant solution set \( \Omega_k \), but then obtains the computation rates based on successive C&F formulas in [20]. The second step remains the same.

V. JOINT CHANNEL ESTIMATION AND ACTIVE USER RECOVERY

Previously, we assume that the channel coefficients and the set of active users are known to the APs. (This assumption is also made in [8] and [9] for the single-AP case.) In this section, we show how to relax this assumption by making use of sparse recovery techniques.

A. Joint Channel Estimation and Active User Recovery

Let \( N \) be the number of clients in the network. Let \( s_k \in \mathbb{C}_T \) be a unique codeword (signature) of length \( T \) symbols, assigned to the \( k \)-th client. This signature is used as the header for every packet of the \( k \)-th client. Clearly, the smaller \( T \) is, the lower the overhead. In particular, we assume that \( T \ll N \) in a dense network. The header of the received signal at the \( k \)-th AP is

\[
y_k = \sum_{i=1}^{N} h_k \zeta_i s_k + n = \mathbf{S} \mathbf{h}_k \mathbf{z} + \mathbf{n}
\]

where \( h_k \) is the channel coefficient from the \( k \)-th client to the \( k \)-th AP, \( \mathbf{S} = [s_1 \cdots s_N] \) is a matrix whose columns are signatures, \( \mathbf{h}_k \in \mathbb{C}^{N \times N} \) is a diagonal matrix representing channel coefficients from users to the \( k \)-th AP, \( \mathbf{z} \in \{0, 1\}^N \) represents client activity state, i.e., \( \zeta_i = 1 \) if client \( i \) is active, and \( \zeta_i = 0 \) if it is inactive, and lastly \( \mathbf{n} \sim \mathcal{CN}(0, \sigma^2) \) is additive white Gaussian noise (AWGN).

The objective of the \( k \)-th AP is to estimate the set \( \mathbf{z} \) of active users as well as their corresponding channel coefficients. Hence, we call it the joint channel coefficient estimation and active client recovery problem.

B. Connection with Sparse Recovery

We note that Eq. (11) can be rewritten as

\[
y_k = \mathbf{S} \mathbf{v}_k + \mathbf{n}
\]

where \( \mathbf{v}_k = \mathbf{H}_k \mathbf{z} \). The \( k \)-th AP tries to estimate \( \mathbf{v}_k \), which contains the information of active users and their channel coefficients. Clearly, this is a sparse recovery problem as we need to reconstruct vector \( \mathbf{v}_k \) which a \( N \times 1 \) high dimensional vector and the columns of matrix \( \mathbf{S} \) are \( T \times 1 \) low dimensional vectors (\( T \ll N \)) representing the signatures.

Compressed sensing (CS) and sparse recovery techniques aim at reconstruction of a high dimensional sparse vector using a small number of random linear measurements [24]. CS-based Multi-User Detection (MUD) in the random-access channels have been investigated in a number of works [25], [26]. In [25] the high dimensional vector space is the message space and the measurements are collided signatures taken from an i.i.d. Gaussian codebook. Capacity bounds and minimum number of measurements are then characterized for the on-off random multiple access channel.

Fletcher et. al. [25] used an i.i.d random Gaussian codebook with sequential orthogonal matching pursuit (SeqOMP) which is analogous to successive interference cancellation decoding. For practical applications, however, random codebook matrix incurs large memory size. Furthermore, the active power shaping proposed in [25] is not a realistic assumption as users are not aware of the channel coefficients. Alternatively, deterministic codebook matrices have a much more efficient storage requirement and fast reconstruction/decoding time. Reed-Muller codes with a very fast reconstruction algorithm are proposed in [27]. The complexity of Reed-Muller decoding depends on the number of measurements \( T \) and not on the message space dimension \( N \) [27]. So, we propose to use deterministic signatures constructed from second-order Reed-Muller codes where the decoding complexity is sub-linear in the dimension of the message space \( N \).

C. MUD with Reed-Muller Signatures

The \( r \)-th order Reed-Muller code \( \text{RM}(r, m) \) over \( \mathbb{F}_2 \) for \( r \geq 2 \) is a linear error-correcting code whose message space consists of degree \( r \) polynomials over \( \mathbb{F}_2 \) in \( m \) variables [28]. Thus, each codeword is of length \( T = 2^m \) symbols over \( \mathbb{F}_2 \). For the details of Reed-Muller code construction see [29].

Howard et. al. [27] used \( \text{RM}(2, m) \) to construct a deterministic sensing matrix and proposed a fast reconstruction algorithm. Based on their construction, we assume that each client is assigned a unique signature of length \( n \) bits consisting of two binary vectors \( \mathbf{b} \in \mathbb{Z}^n \) and \( \mathbf{c} \in \mathbb{Z}^m \). \( \mathbf{c} \) is then mapped to a \( m \times m \) symmetric matrix \( \mathbf{P}(\mathbf{c}) \) as follows

\[
\mathbf{P}(\mathbf{c}) = \sum_{i=1}^{n_2} c_i \mathbf{B}(i) \mod 2
\]

where \( c_i \) is the \( i \)-th entry of vector \( \mathbf{c} \) and \( \mathbf{B}(i) \) is the \( i \)-th element of the ordered set \( \mathbf{B} \) of basis for the space of \( m \times m \) binary symmetric matrices [30].

Let \( \phi_{\mathbf{b}, \mathbf{c}} \) be a \( \text{RM}(2, m) \) codeword, we then have

\[
\phi_{\mathbf{b}, \mathbf{c}}(\mathbf{a}) = \exp \left( j \pi \frac{1}{2} \mathbf{d}^\mathsf{T} \mathbf{P}(\mathbf{c}) \mathbf{d} + \tilde{\mathbf{b}}^\mathsf{T} \mathbf{d} \right)
\]

where \( \tilde{\mathbf{b}} \in \mathbb{Z}^m \) is formed by padding \( \mathbf{b} \) with \( m-n_1 \) zeros, and codewords are indexed by \( \mathbf{d} \in \mathbb{Z}^m \).

For the codebook \( \text{RM}(2, m) \), the set of basis \( \mathbf{B} \) of size \( m(m+1)/2 \) can be calculated in advance and stored at each client where it is used to compute \( \mathbf{P}(\mathbf{c}) \) requiring \( n_2 m^2 \) operations.

For decoding \( \text{RM}(2, m) \) we modify the scheme described in Algorithm 4 in [30]. The main steps of the decoding algorithm are as follows:

1) The first \( 2^m \) symbols of a client’s packet constitutes its signature which is input to the Reed-Muller decoder where \( \mathbf{\tilde{b}}_j \) and \( \mathbf{\tilde{c}}_j \) are then estimated and used to detect an active client. The detected active client that appears
Algorithm 2 Channel coefficient estimation and active client set detection algorithm

**Input:** $y_1, \ldots, y_K$

**Output:** Channel coefficients $\hat{h}_1, \ldots, \hat{h}_K$, active client set $\hat{D}$.

1. Initialize: $r_k = y_k$, $\hat{D}_k = \emptyset$, $\hat{D} = \emptyset$
2. **for all** $k \in \{1, \ldots, K\}$ **do**
3. **while** $i \leq i_{\text{max}}$ and $\|r_k\|_2 > \epsilon$ **do**
4. Run RM decoder with $r_k$ as input to detect the index of a client $d$.
5. $\hat{D}_k \leftarrow \hat{D}_k \cup \{d\}$
6. Form the matrix $S_{\hat{D}_k}$ by placing the codewords corresponding to $\hat{D}_k$ as the columns.
7. Find the channel coefficients:
   $$\hat{h}_k \leftarrow \arg\max_{h_k} \|y_k - S_{\hat{D}_k} \hat{h}_k\|_2$$
8. Update the residual $r_k \leftarrow r_k - y_k$
9. **end while**
10. **end for**
11. APs’ pass $\hat{D}_k$ among themselves over a backhaul link
12. Each AP computes the final set of active users as $\hat{D} = \cap_k \hat{D}_k$
13. **for all** $k \in \{1, \ldots, K\}$ **do**
14. Refine channel coeffs. $\hat{h}_k \leftarrow \arg\max_{\hat{h}_k} \|y_k - S_{\hat{D}_k} \hat{h}_k\|_2$
15. **end for**
16. return $\hat{h}_1, \ldots, \hat{h}_K$ and $\hat{D}$

more than any other at the output of the $K$ APs (majority vote) is then added to the set of current detected active users whose corresponding codewords form the columns of the matrix $S_{\hat{D}}$.

2) Channel coefficients $\hat{h}_k$ for all $k \in \{1, \ldots, K\}$ are estimated using the set of current detected active users. The estimation is based on minimization of $\|y_k - S_{\hat{D}} \hat{h}_k\|_2$. Update the residual signal $r_k = y_k - S_{\hat{D}} \hat{h}_k$.

3) If $\|r_k\| < \epsilon$ for some pre-set small $\epsilon > 0$ then $\hat{D}_k$ is the final estimate of the set of active users observed by the $k$th AP. Otherwise the residual signal is used as the input to the RM decoder. This procedure is repeated for no more than a pre-determined maximum number of iterations $i_{\text{max}}$ or until the energy of the residual signal is below $\epsilon$; whichever is satisfied earlier.

4) Finally the APs exchange their set of active users with each other and their intersection will be the set of current active users based on which the channel coefficients are re-estimated.

The proposed channel coefficients estimation and active client set detection scheme is summarized in Algorithm 2.

**VI. TRACE-DRIVEN SIMULATIONS**

To understand the performance of C&F-based CSMA protocols under realistic channel variations, we conduct simulations using a real-world channel condition trace [10]. The trace, collected from a 44-node wireless network in an office area, contains over 9300 measured channel impulse responses (CIR) for a total of $44 \times 43 = 1892$ pair-wise links.

We strategically pick 2 nodes as 2 APs, and the remaining 42 nodes as wireless users. Furthermore, we assume that 2/3 of the users are in class 1 ($N_1 = 28$) and the rest are class-2 users ($N_2 = 14$). The transmission of a packet is assumed to take $\tau = 10$ time slots. During the subsequent simulations, we generate channel coefficients for these sender-receiver (client-AP) pairs based on the statistics obtained from the trace. Specifically, for each client-AP pair, we first collect all the measured CIRs from the trace, based on which we estimate the variance of its channel gains. The estimated variance captures the large-scale fading effect (i.e., the path loss) [31]. To model the small-scale fading effect, we use the Rayleigh fading model [31]. Based on the traces we compute the conditional probabilities for the 2-AP scenario as $q_1 = 0.9946$, $q_2 = 0.8800$. Note that the effect of channel estimation is not captured in $q_1$ and $q_2$.

We first look at the network aggregate throughput for single AP and multi-AP scenarios. Fig. 3 illustrates the network aggregate throughput for class-1 users while varying the transmission probability of class-1 users with the transmission probability of class-2 users fixed at $p_2 = 1/42$. It can be seen that depending on the traffic load in class-1 and class-2, either class or both can become stable or unstable, and the aggregate throughput closely matches our analytical result. In addition, the multi-AP scheme offers a better aggregate throughput compared to single-AP case as a result of AP cooperation.

We then compare the delay performance for single AP and multi-AP scenarios. Fig. 4 depicts the total delay experienced by a packet in the two-class network. It is evident that the delay analysis matches the simulations for both classes quite well. The multi-AP scenario also supports higher arrival rates for a given finite delay. Note that the delay performance is given for each class separately. In other words, the simple expression provided for the total delay enables us to compute the delay for a user in any class which is helpful in system design.

Next, we compare the aggregate throughput performance of C&F-based CSMA with that of conventional CSMA in Fig. 5.
The performance gain we obtain by using C&F is considerably good. For instance, in $\lambda_1 = 0.015$ and $\lambda_2 = 0.015$, the CSMA with C&F gives a much higher aggregate throughput for a larger range of transmission probabilities of class-1 users. As an example, at $p_1 = 0.1$, aggregate throughput offered by the multi-AP scenario with C&F-based CSMA is 4.7 times greater than that of conventional CSMA, indicating that the throughput scales well with our C&F-based scheme.

Finally, we look into the benefit of AP cooperation as opposed to a non-cooperative multi-AP case where each AP decodes the most reliable linear combination independently and then forwards it to the central controller. As can be seen in Fig. 6, the cooperation between the two AP’s improves the network aggregate throughput significantly. The reason is that when AP’s cooperate with each other their decoded linear combinations are more likely to be linearly independent and thus collision can be resolved by the central controller. However, in the non-cooperative scheme there is a higher chance of linear dependence between the decoded linear combinations. With a small cooperation overhead, therefore, the throughput scales well thanks to the C&F decoding technique.

VII. RELATED WORK

A. C&F for Random-Access Networks

The study of C&F for random-access networks was initiated by the work of [8] and [9] very recently, with a particular focus on the single-AP scenario. Our work extends [8] and [9] to the multi-AP scenario by identifying two technical challenges and proposing novel solutions. In particular, we are able to characterize the benefits of adding more APs in terms of throughput and delay performances.

B. C&F for Multi-Source Multi-Relay Networks

The use of C&F in multi-source multi-relay networks has been studied in [7], [32]. Although their setup looks similar to ours, their focus is on the information-theoretic analysis, whereas our focus is on the networking performance under practical constraints. Also, the network-coding design problem in [7], [32] assumes that each relay decodes at most one linear combination. In contrast, our AP cooperation problem relaxes this constraint and allows each AP decodes more than one linear combinations.

C. Channel Estimation via Sparse Recovery

While the high-level idea of applying sparse recovery techniques to channel estimation problems is not new (e.g., [33]), our application focuses on a dense network with concurrent transmitters, whereas existing approaches focus on a single transmitter-receiver pair (where the sparsity comes from the multi-path effect). Moreover, we make use of second-order Reed-Muller codes, leading to a recovery algorithm whose complexity is sub-linear of the number of users.
D. Alternative Physical-Layer Techniques

Reasonable alternatives to C&F may include physical-layer network coding (PNC) [34], multiuser detection via successive interference cancellation (SIC) [35], and ZigZag decoding [36]. However, they are not particularly suitable for multi-AP scenarios. Conventional PNC decodes an XOR-combination from a collision [34]. In a two-AP setup, even if both APs are able to decode the XOR of the colliding packets, the controller cannot recover the individual packets due to the linear dependency.

An SIC receiver attempts to recover all packets from a single collision. In many instances, this may not be possible, even though it might still be possible in such instances to decode one linear combination of the colliding packets (clearly an “easier” problem, and the one solved by C&F).

ZigZag decoding makes use of multiple collisions by exploiting distinct temporal offsets in different collisions. If ZigZag decoding is used in a two-AP setup, each AP must wait for at least two collisions in order to recover the colliding packets. In sharp contrast, C&F exploits spatial diversity and can reconstruct the colliding packets from a single collision.

VIII. CONCLUSIONS

C&F is a promising physical-layer technique with a potential to significantly improve wireless networks performance. In contrast to the existing work on C&F which is mainly focused on information-theoretic analysis and encoding-decoding methods, we explored the advantages of C&F in wireless random-access networks by taking into account several important practical challenges. Our work has extended some very recent results from the single-AP case to the multi-AP scenarios. In particular, accurate throughput and delay expressions for C&F-based CSMA protocols in multi-AP settings are derived and verified by trace-driven simulations, demonstrating the significant advantages of C&F-based CSMA over conventional CSMA in multi-AP scenarios.

REFERENCES