RobinHood: Sharing the Happiness in a Wireless Jungle

Tarun Bansal†, Wenjie Zhou†, Kannan Srinivasan and Prasun Sinha
Department of Computer Science and Engineering, The Ohio State University, Columbus, OH 43210
{bansal, zhouwe, kannan, prasun}@cse.ohio-state.edu
†Co-primary Authors

ABSTRACT

Today's Enterprise Wireless LANs are comprised of densely deployed access points. This paper proposes RobinHood, an interference nulling scheme that leverages the high density of the access points to enable multiple mobile devices to transmit simultaneously to multiple access points (APs), all within a single collision domain. RobinHood also leverages the capability of the APs to communicate with each other on the wired backbone to migrate most of the complexity to the APs, while keeping the design at the mobile clients simpler. Finally, we leverage the static nature of the access points to make RobinHood more practical in networks where the mobility of clients inhibit the use of traditional interference alignment schemes. Results from our trace-driven simulations show that RobinHood obtains a throughput improvement of 6.08× and 24.2× over omniscient TDMA and IEEE 802.11, respectively.

1. INTRODUCTION

The recent explosive growth in the number of mobile devices and the data generated by these devices has led to a decrease in the channel resources available to each individual device. Network administrators have tried to tackle this problem by densely deploying the access points. However, the dense deployment of APs does not scale well with the throughput demands. In the existing network protocols [13, 18, 26], when one mobile client is transmitting uplink packets to an access point, the nearby clients have to remain silent to avoid causing interference to the ongoing transmission.

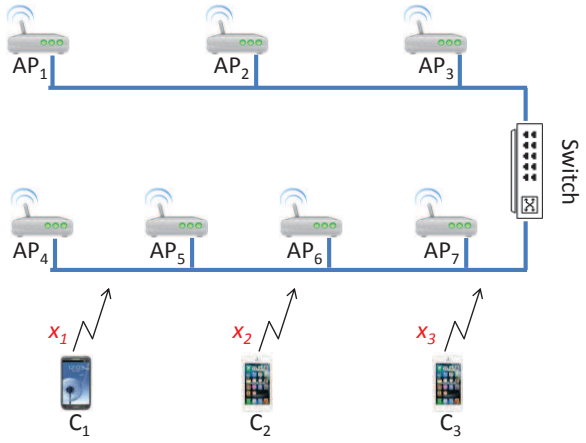
This paper proposes RobinHood, that enables multiple nearby access points to concurrently receive uplink packets from multiple mobile clients, all within a single collision domain. RobinHood does not increase energy consumption on the clients and executes exactly over two time slots. *RobinHood leverages the dense deployment of APs in enterprise networks (See Fig. 3), the capability of these APs to exchange packets with each other over the wired backbone and the immobility of APs resulting in relatively stationary channels (See Fig. 2).* Our proposed technique, RobinHood, makes use of the energy-rich access points to assist their clients (mobile devices) in decoding their packets at their respective access points. In RobinHood, the clients only participate in the first slot and the access points participate for the clients in the second slot.

ACM HotMobile'14, February 26–27, 2014, Santa Barbara, CA, USA.

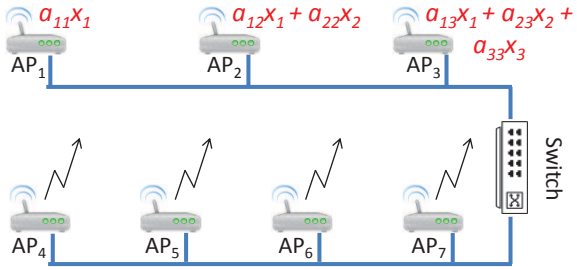
Consider the example enterprise WLAN shown in Fig. 1(a) where all Access Points (APs) and the three clients are in a single collision domain. Assume that the three users want to upload one packet each to the backbone. An omniscient TDMA scheduling algorithm with global knowledge would require three time slots to complete this upload. In RobinHood, in the first slot as shown in Fig. 1(a), all users will transmit at the same time. All the 7 APs will receive a combination of three transmitted packets. In the second slot, AP_4, AP_5 AP_6, and AP_7 will retransmit the received signals using zero-force nulling [15] such that the following two conditions are satisfied as shown in Fig. 1(b): (i) At AP_1, samples corresponding to x_2 and x_3 are canceled out; and, (ii) At AP_2, samples corresponding to x_3 are canceled out. Decoding happens in multiple steps as follows:

- At the end of the second slot, AP_1 simply decodes x_1 since it only received $a_{11}x_1$, which are samples corresponding to x_1. Then it transmits the decoded packet over the backbone to both AP_2 and AP_3.
- AP_2 recreates the samples corresponding to x_1 and subtracts them from $a_{12}x_1 + a_{22}x_2$. After subtraction, it can decode and obtain x_2 from the remaining samples, $a_{22}x_2$.
- AP_2 also transmits the packet x_2 to AP_3 using the wired backbone. In the second slot, AP_3 receives samples from all of x_1, x_2 and x_3. It recreates samples for x_1 and x_2 and subtracts them from $a_{13}x_1 + a_{23}x_2 + a_{33}x_3$. After subtraction, it decodes the remaining samples, $a_{33}x_3$, to obtain x_3.
- Afterwards, AP_1, AP_2 and AP_3 forward x_1, x_2 and x_3 to their destination APs using the wired backbone.

RobinHood enables the three transmitters with single antenna to upload three packets in two slots, improving the throughput by 50% compared to omniscient TDMA. In Section 2.1, we show that in networks with high enough density of APs, RobinHood enables N mobile clients to transmit N uplink packets in two slots resulting in unbounded throughput. Observe that unlike virtual MIMO [8], RobinHood requires the APs to exchange only the decoded packets instead



(a) RobinHood: First slot. Clients only need to transmit once in the first slot. x_1, x_2 and x_3 are the three packets transmitted by C_1, C_2 and C_3, respectively.



(b) RobinHood: Second slot. A subset of APs transmit in the second slot while the rest of the APs receive. a_{ij} are the final channel coefficients after two slots transmissions.

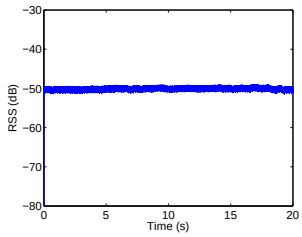
Figure 1: Working of RobinHood in two slots. (a) also shows the network topology. All the devices are in single collision domain.

of the raw samples. This property is quite useful since forwarding raw samples can overwhelm the bandwidth of the wired network [6]. Also, unlike [5] that precodes over an exponential number of time slots, RobinHood decodes all the packets in two slots resulting in lower latency. Further, in contrast with [5], RobinHood requires the mobile clients to transmit only once resulting in lower energy consumption.

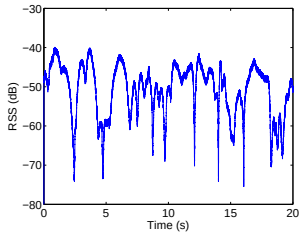
The focus of RobinHood is to increase throughput of the uplink traffic for clients with single antenna. This is in contrast with [16, 10] that focus on downlink traffic. Recently, the uplink traffic [7, 4] has been growing at a fast rate due to the emergence of a wide-range of computing paradigms and applications, such as cloud computing, video conferencing, online gaming, VoIP, and traffic generated from the mobile devices (e.g., location information or sensor readings). RobinHood makes extensive use of the wired backbone. Besides transmitting the decoded packets, the channel state information, which are required to do nulling in the second slot, are also exchanged through the backbone. Since RobinHood migrates most of the complexity from the mobile devices to the APs, it allows RobinHood to work even when the channel from clients to APs is rapidly changing due to client mobility. RobinHood works as long as the APs are time-synchronized with each other and places very few requirements on the clients. In wireless networks, it is possible that a client may use a data rate such that no AP is able to decode the packet. Typically, this requires the client to retransmit the packet resulting in higher energy consumption. RobinHood takes advantage of the receiver and transmitter diversity and increases the probability of decoding such packets without requiring packet retransmission.

2. ROBINHOOD DESIGN

In this section, we describe the design details of RobinHood. First, the two-slot nulling nulling scheme is described. Then, we formulate the AP matching problem and propose a polynomial-time solution.



(a) Channel between pairs of APs.



(b) Channel between mobile client and AP.

Figure 2: Plot of channel measurement shown over a period of 20 seconds in an office environment. The channel between APs is relatively stationary compared to channel between AP and mobile client.

2.1 Two Slot Nulling in RobinHood

Before discussing RobinHood in detail, we define a few notations. All of the clients and APs in RobinHood are assumed to have only one antenna. Let $\{C_i : i = 1, 2, \cdots, N\}$ be the set of wireless clients and N be the total number of clients. Let $\{AP_j : j = 1, 2, \cdots, M\}$ be the set of APs that are connected through a wired backbone. Let $h_{ij}^{(1)}$ be the channel state information between C_i and AP_j in slot 1. In the second slot, a subset of APs are selected to transmit.

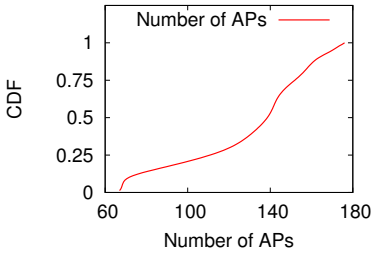


Figure 3: Shows the CDF of number of APs observed at a point. The data was collected at multiple places including OSU Medical Center (a large hospital), OSU Central library and a nearby apartment complex.

Let this set be $\{AP_k : k = N+1, N+2, \cdots, M\}$. Let $h_{kj}^{(2)}$ be the channel state information between AP_k and AP_j in slot 2. Let x_i be the signals sent by C_i in slot 1. In the following discussion, we neglect the presence of the noise since noise is unpredictable and thus, difficult to cancel out. However, we do take noise into account in our analysis (See Section 2.2.2). Let $y_{ik}^{(1)}$ be the component of x_i received by AP_k in slot 1. We have:

$$y_{ik}^{(1)} = h_{ik}^{(1)} x_i \tag{1}$$

Let v_k be the precoding coefficient for AP_k in the second slot. Let $y_{ij}^{(2)}$ be the component of x_i received by AP_j in slot 2. We have:

$$y_{ij}^{(2)} = \sum_{k=N+1}^{M} h_{kj}^{(2)} v_k y_{ik}^{(1)} = \sum_{k=N+1}^{M} h_{kj}^{(2)} v_k h_{ik}^{(1)} x_i \tag{2}$$

In the example network shown in Fig. 1, we want to ensure that components of x_2 and x_3 at AP_1 is nulled. Similarly, we want x_3 to be nulled at AP_2. Thus,

$$y_{21}^{(2)} = \sum_{k=4}^{M} h_{k1}^{(2)} v_k h_{2k}^{(1)} x_2 = 0 \tag{3}$$

$$y_{31}^{(2)} = \sum_{k=4}^{M} h_{k1}^{(2)} v_k h_{3k}^{(1)} x_3 = 0 \tag{4}$$

$$y_{32}^{(2)} = \sum_{k=4}^{M} h_{k2}^{(2)} v_k h_{3k}^{(1)} x_3 = 0 \tag{5}$$

Since, the right side of Eq. 3,4,5 are all 0, instead of 3, at least 4 variables (v_k) are required to obtain non-zero solutions. Thus, a total of 7 APs are required to support 3 clients as in Fig. 1. In general, for a system with N clients, we need to cancel $(N-1)$ signals at the first receiving AP, $(N-2)$ signals at the second receiving AP, and zero signal at the last receiving AP. In our technical report [1], we show that that the total number of APs required is: $\frac{N^2+N+2}{2}$.

In RobinHood, for the network shown in Fig. 1(a), at the end of slot 1, APs 4, 5, 6 and 7, solve Eq. 3, 4, 5 to obtain precoding vectors which are then used during slot 2 (See Eq. 2). Although this computation (done at the beginning of slot 2) may take time (due to communication among APs over backbone), it is not a problem for RobinHood since the channel between APs changes relatively slowly (See Fig. 2(a)).

2.2 Packet Decoding in RobinHood

In the example network discussed in Sec. 1, we assumed that AP_1, AP_2 and AP_3 decode x_1, x_2 and x_3, respectively. It is possible that a different mapping (or matching) between the APs and the packets may give different performance results. In the next subsection, we explain the effect of matching on the network performance.

2.2.1 *Why matching is important*

Consider the sample network shown in Fig. 1(a) with three clients and seven APs. Here, a matching of (AP_i, C_j) indicates that AP_i decodes the packet from C_j. To illustrate, we compare two possible matchings: $M_1 = \{(AP_3, C_1), (AP_2, C_3), (AP_1, C_2)\}$; and, $M_2 = \{(AP_1, C_1), (AP_2, C_2), (AP_3, C_3)\}$.

Observe that at the end of slot 1, AP_4 will receive x_1 at high signal-to-noise ratio (SNR) due to its closeness to C_1. Similarly, in the second slot, AP_1 will receive C_1 with high SNR due to its closeness to AP_4. At the same time, AP_2 will receive x_2 with high SNR due to its proximity to AP_5 and AP_6 that receive x_2 at high SNR. Thus, M_2 is a better matching than M_1 since in M_2, each AP decodes that packet which it received with high SNR. Our trace-driven simulations (detailed discussion in Sec. 3) show that M_1 gives a throughput of 7.55 Mbps while M_2 gives a throughput of 12.38 Mbps.

Thus, to maximize the SNR, RobinHood needs to solve the following problem statement:

$$\max_{f} \sum_{i=1}^{N} f_{ij} \times SNR_{ij} \tag{6}$$

$$\textit{such that} \quad \sum_{j=1}^{M} f_{ij} = 1, \ \forall i \in \{1 \cdots N\} \tag{7}$$

$$\sum_{i=1}^{N} f_{ij} \leq 1, \ \forall j \in \{1 \cdots M\} \tag{8}$$

$$||v_k \times y_k^{(1)}||^2 \leq P_0, \ \forall k \in \{1, 2, \cdots M\} \tag{9}$$

Here, f is a matching function such that $f_{ij} = 1$ indicates that AP_j will decode packets from C_i (Otherwise, f_{ij} is 0). SNR_{ij} denotes the SNR of x_i at AP_j during the second slot transmission. Eq. 7 constraints that each client must be decoded at exactly one AP. Eq. 8 prevents a single AP from decoding multiple clients. Eq. 9 puts a constrain on the transmission power level of each AP to comply with the FCC standard. This matching problem is combinatorial in nature and is NP-Hard to solve as we show in our technical report [1].

2.2.2 *Computing the best matching*

To determine matching between APs and packets, we need to compute the SNR of each packet to be received at each AP. The exact value of the SNR depends on the precoding vectors which in turn depends on the rest of the matching. This makes the problem combinatorial in nature. We simplify it by computing the expected SNR of a given packet (say x_i) at a given access point (say AP_j).

Let $n_k^{(1)}$ be the channel noise at AP_k in the first slot. Let $n_j^{(2)}$ be the channel noise at AP_j in the second slot and $N_j^{(2)}$ be the total noise (including the received noise) at AP_j in slot 2. Then, $N_j^{(2)}$ is the sum of the noise received from other

APs in slot 2 and the channel noise. We estimate $N_j^{(2)}$ and its power as follows:

$$N_j^{(2)} = \sum_{k=N+1}^{M} h_{kj}^{(2)} v_k n_k^{(1)} + n_j^{(2)} \quad (10)$$

$$||N_j^{(2)}||^2 = \sum_{k=N+1}^{M} ||h_{kj}^{(2)} v_k n_k^{(1)}||^2 + ||n_j^{(2)}||^2 \quad (11)$$

Observe that, here we do not need to consider interference from other clients at AP_j since either the transmissions from other clients would be canceled at AP_j due to joint precoding by other APs (e.g. x_2 is canceled at AP_1) or their samples will be subtracted from the received samples by AP_j (e.g. AP_2 subtracts x_1 from the received samples).

If P_0 is the transmission power level and $y_{ij}^{(2)}$ is defined as before (Eq. 2), then, we can estimate the RSS of x_i at AP_j,

$$RSS_{ij} = ||y_{ij}^{(2)}||^2 = || \sum_{k=N+1}^{M} h_{kj}^{(2)} v_k h_{ik}^{(1)}||^2 \times P_0 \quad (12)$$

In [1], we show:

$$|v_k| \approx \frac{1}{\sum_{i=1}^{N} h_{ik}^{(1)}} \quad (13)$$

So, using Eq. 12, Eq. 11 and Eq. 13, the SNR of x_i at AP_j can be estimated as:

$$SNR_{ij} = \frac{RSS_{ij}}{||N_j^{(2)}||^2} \quad (14)$$

Using the above equation, RobinHood computes the expected SNR of all clients at all APs. If R_i denotes the data rate chosen by C_i, then the throughput of C_i at AP_j can be computed as follows:

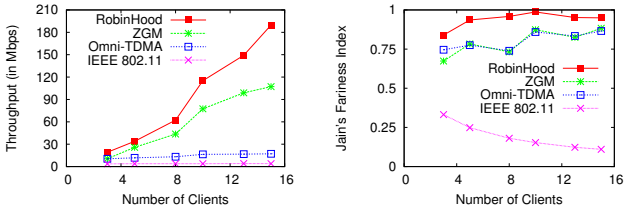
$$T_{ij} = \begin{cases} R_i, & \text{if } SNR_{ij} \geq \tau_i \\ 0, & \text{otherwise} \end{cases} \quad (15)$$

Here, τ_i represents the minimum SNR required to decode x_i. The value of τ_i depends on the physical layer data rate chosen by C_i. To determine which AP should decode which packet, RobinHood solves a polynomial-time maximum weight bipartite matching problem between the set of APs (AP_1, AP_2, $\cdots$, AP_M) and the set of packets (x_1, x_2, $\cdots$, x_N). During this computation, the weight of the edge (x_i, AP_j) is set to T_{ij}. The result of the bipartite matching indicates which AP should decode which packet.

2.2.3 *Computing the Packet Decoding Order*

Having determined the matching, we also need to determine the order in which the APs decode the packets. In the above discussion, we assumed that joint precoding leaves no residual noise. However, in practice, joint precoding is not always perfect and leaves some residual noise. Lets say AP_j decodes the i^{th} packet in the decoding sequence. Observe that decoding of i^{th} packet will experience residual noise from $(N - i)$ other packets that are canceled out at AP_j using joint precoding. Thus, packets decoded sooner experience higher residual noise (assuming that noise from nulling is higher than the noise from the packet cancellation).

To ensure proper delivery of messages, we first compute how much residual noise can a packet tolerate. Consider client C_i that transmits packet x_i at data rate R_i. Let AP_j be the receiving AP that decodes x_i. If τ_i is the minimum SNR required to decode x_i, then the residual noise that can be tolerated at AP_j during the decoding is given by [1]: $\frac{RSS_{ij}}{\tau_i} - ||N_j^2||^2$. Using this, RobinHood computes the maximum residual noise that each packet can tolerate. Finally, the decoding order is chosen by arranging the packets in an non-increasing order of the maximum noise they can tolerate. This ensures that packets that experience maximum residual noise have high noise tolerance, resulting in higher decoding probability.



(a) Total Throughput (b) Jain's Fairness

Figure 4: Trace-Driven Simulation Results.

3. TRACE-DRIVEN SIMULATION

To validate the performance of RobinHood, we implemented RobinHood in a simulator. We set up a field of size 60m × 60m. Apart from implementing RobinHood, we also implemented three other algorithms:

1. **Zeroforce nulling + Greedy matching (ZGM):** We present a simple greedy algorithm, denoted as ZGM. Like RobinHood, ZGM also uses all APs to simultaneously decode multiple uplink packets. However, unlike RobinHood, ZGM simply tries to match each client to its nearest AP. However, since an AP can only be paired with at most one client, this algorithm solves a corresponding matching problem. The weight between a client and AP is inversely proportional to the channel loss from the client to the AP. Finally, the matching between the clients and the APs is obtained by solving a bipartite matching problem.

2. **Omniscient TDMA algorithm:** This algorithm utilizes a central server that is aware of (i) packet queue at different clients; and, (ii) the channel between all clients and all APs. In each slot, it schedules one client (in a round-robin fashion) to send a packet to its nearest AP.

3. **IEEE 802.11 (without RTS/CTS).**

In our trace-driven simulation, for both RobinHood and ZGM, a subset of APs compute the precoding matrix such the interfering transmissions at the receiving APs are canceled out. However, due to different matching algorithms, the SNR of packets at the receiving APs may be different (See Sec. 2.2.1). We plot various metrics with varying number of clients. The number of APs were set to $\frac{N^2+N+2}{2}$ where N is the number of clients. To ensure that the simulation parameters reflect realistic environments, we used the SNR values from an existing testbed [19] and used those values to generate the channels in our simulation.

1. **Total Throughput:** Fig. 4(a) shows the variation in total throughput with variation in number of clients. By carefully matching the APs with the clients based on channel values, on an average, RobinHood is able to achieve

1.55× throughput as compared to ZGM. In RobinHood multiple packets can be simultaneously decoded using the wired backbone and thus the average throughput of RobinHood is 6.08× compared to the average throughput of TDMA. The throughput for both RobinHood and ZGM increases with increase in number of clients, since both of these algorithms allow multiple clients to transmit simultaneously. Finally, transmitters in IEEE 802.11 experience very low throughput due to backoffs and collisions. Compared to 802.11, RobinHood has an average throughput of 24.2×.

2. **Fairness:** Using simulations, we observed that the average value of Jain's fairness index for RobinHood, ZGM, TDMA and 802.11 was 0.93, 0.79, 0.80 and 0.19, respectively (See Fig. 4(b)). The fairness index of RobinHood is higher than other algorithms since RobinHood allows all clients to transmit. On the other hand, in 802.11, some clients may get starved due to their location with respect to other clients and APs in the network. RobinHood has higher fairness than TDMA since RobinHood performs precoding over transmissions from all clients. Thus, even the clients that are far away from all APs may experience high throughput due to zero-force nulling from multiple APs.

4. RELATED WORK

Although RobinHood builds on prior work done in the field of Wireless Networking, it differs from them in various ways.

Backbone usage: The idea of using the wired backbone to increase wireless throughput is not new. In MegaMIMO[16], multiple APs cooperatively precode the transmissions such that each client receives only the packets intended for it while the other transmissions are canceled out. However, MegaMIMO requires that transmitters exchange packets among themselves and thus, it works only for the downlink transmissions. On the other hand, RobinHood improves the throughput for the uplink traffic. Also, in contrast to MegaMIMO and OpenRF [10], transmitters in RobinHood perform joint nulling without knowing the actual contents of the packets.

A recently proposed protocol Symphony [4] also focuses on uplink traffic. However, in contrast to RobinHood, Symphony improves the network throughput only when the APs are in different collision domains. In Epicenter[7], authors propose that APs should exchange coarse representations of symbols to decode corrupted bits. Similarly, authors in [12, 23] also propose that APs exchange bits or raw samples on the backbone to facilitate packet decoding. In all these algorithms, the APs cooperate to decode the same packet whereas in RobinHood, APs encourage transmitters to collide and then cooperate to decode multiple packets simultaneously without exchanging the raw samples.

Interference Alignment: Previously, researchers (see [8] and references therein) have used interference alignment to improve the capacity of wireless networks. However, unlike RobinHood, they either require APs to exchange samples over the backbone [3], work only for the downlink traffic [20], assume presence of significant number of clients [14], require multiple antennas at transmitters or receivers [6], require the antennas to be physically moved [2] to a certain point, require the channel to change from one slot to another [5], precode over exponential number of time slots [5], or provide limited throughput gain [2, 6]. These assumptions are not practical in nature since if the client is stationary, the channel may not change [21] from one packet to another. In contrast to the previous works, RobinHood works even if the channel stays stationary.

Wireless Relays: Researchers [17, 9] have also looked at the problem of using special relay nodes to assist high speed communication between specific pairs of source and destination nodes. In contrast, the focus of RobinHood is to leverage the high density of APs and the wired backbone to carefully select the set of destination APs, determine which AP decodes which packet, and to use the wired backbone to migrate all the complexity away from the clients. Further, with previous works, it is possible that the destination AP is unable to decode a packet due to low SNR. However, in RobinHood, APs leverage the wired backbone to perform decoding over multiple rounds (See Sec. 5), thereby ensuring that the transmitted packets are eventually decoded.

5. CONCLUSIONS AND FUTURE WORK

In the previous sections, we discussed RobinHood: an interference alignment scheme that leverages the high density of access points to enable multiple mobile devices to transmit simultaneously. Using trace-driven simulations, we showed that on an average, RobinHood gives a throughput improvement of 6.08× and 24.2× over omniscient TDMA and IEEE 802.11, respectively. However, there are multiple other challenges that need to solved to make RobinHood practical:

1. **Multi-Collision domain:** The previous discussion assumes that all clients and all APs can hear each other directly. However, this may not be true for all networks. We are currently exploring extensions of RobinHood to multi-collision domain. One of the possible approaches is to divide the network into manageable smaller groups [25, 4] where each group works independently. Another possible solution is to combine RobinHood with Symphony[4], which focuses on uplink traffic in multi-collision domain. Symphony could be used first to decode packets from different collision domains, reducing the problem to several small single collision domains, which could be solved using RobinHood.

2. **Datarate adaption:** In RobinHood, the SNR of a client at its decoding AP depends not only on the relative channels between the different devices but also on the precoding vectors chosen by APs. The precoding vectors in turn, depend on the set of clients that are transmitting at the same time. Thus, it is not possible for a client to determine the best datarate to use. Further, traditional history based datarate adaptation algorithms such as Auto Rate Fall-back (ARF) are ineffective for RobinHood as the historical Packet Error Rate (PER) is no longer a good estimate for future PER. Thus, we need a new mechanism that helps clients in determining the best physical layer data rate to be used.

One of the possible approaches is that access points first compute the best physical layer datarate for each client and then they let the transmitters know the data rate to be used. Another possible approach involves clients predicting their data rate based on combination of history and the number of other clients that are transmitting simultaneously.

3. **Decoding a packet with insufficient SNR:** Even with a data rate adaption algorithm, it is possible that sometimes APs may not be able to decode packets from the clients due to low SNR. Traditional networks resolve this

problem by requiring the clients to retransmit. However, in RobinHood, we can take a different approach that leverages the transmitter and receiver diversity without requiring the clients to retransmit. We illustrate that with an example: Consider a network with 4 clients and 11 APs such that three of the packets are canceled at one AP, two at another AP and one packet is canceled at the third AP (a total of 6 cancellations). Let us assume that APs are unable to decode any of the four packets due to insufficient SNR. In that case, instead of requiring the clients to retransmit, ten of the 11 APs can retransmit the received samples with different choices of precoding vectors such that the 11th AP only receives samples corresponding to one client while the other three clients are canceled (a total of only 3 cancellations). The 10 APs can utilize transmitter and receiver diversity to choose the precoding vectors such that the SNR of the single packet at the 11th AP is maximized, thereby increasing the probability of decoding. Once one of the packets is decoded, then APs can subtract that packet from their received samples and try to decode the remaining packets. With only three unknowns remaining, the probability that APs can simultaneously decode all three of them also improves. As a future work, we plan to design algorithm for picking the "right AP" and the "right client" to bootstrap this process.

4. **Inconsistency in the AP density:** To decode N packets, RobinHood requires $\frac{N^2+N+2}{2}$ access points nearby. However, the actual number of APs present may be higher or lower than this number. If the number of available APs are higher, then RobinHood can make use of all of them and select precoding vectors such that the total SNR is further increased. On the other hand, if the number of available APs are smaller, than a mechanism is required to suppress some of the clients. This ensures that the packets transmitted can all be decoded by the given number of APs. The suppressing algorithm [4] should take both throughput and fairness into account.

5. **Channel Estimation:** To compute the precoding vectors, the APs in RobinHood require the knowledge of channel between all clients and APs as well as the channel between all APs. The problem of computing the channel from clients to APs has been well-studied in the context of MIMO networks [6, 24]. To compute the channel values, we are planning to use PN sequences to estimate the channel from multiple transmitters simultaneously[11, 22].

6. REFERENCES

[1] RobinHood. Tech. rep. `http://sites.google.com/site/bansaltarun/RobinhoodTechRep.pdf`.

[2] Adib, F., Kumar, S., Aryan, O., Gollakota, S., and Katabi, D. Interference Alignment by Motion. In *Proc. of ACM MobiCom 2013*.

[3] Annapureddy, V. S., El Gamal, A., and Veeravalli, V. V. Degrees of Freedom of Interference Channels with CoMP Transmission and Reception. *IEEE Transactions on Information Theory 58*, 9 (2012), 5740–5760.

[4] Bansal, T., Chen, B., Sinha, P., and Srinivasan, K. Symphony: Cooperative Packet Recovery over the Wired Backbone in Enterprise WLANs. In *Proc. of ACM MobiCom 2013*.

[5] Cadambe, V. R., and Jafar, S. A. Interference Alignment and the Degrees of Freedom for the K User Interference Channel. *IEEE Transactions on Information Theory* (2007).

[6] Gollakota, S., Perli, S. D., and Katabi, D. Interference Alignment and Cancellation. In *Proc. of ACM SIGCOMM 2009*.

[7] Gowda, M., Sen, S., Roy Choudhury, R., and S., L. Cooperative Packet Recovery in Enterprise WLANs. In *Proc. of IEEE INFOCOM 2013*.

[8] Jafar, S. A. *Interference Alignment: A New Look at Signal Dimensions in a Communication Network*. Now Publishers, 2011.

[9] Kuhn, M., Berger, S., Hammerstrom, I., and Wittneben, A. Power Line Enhanced Cooperative Wireless Communications. *IEEE Journal on Selected Areas in Communications 24*, 7 (2006), 1401–1410.

[10] Kumar, S., Cifuentes, D., Gollakota, S., and Katabi, D. Bringing Cross-Layer MIMO to Today's Wireless LANs. In *Proc. of ACM SIGCOMM 2013*.

[11] Li, T., and *et al*. CRMA: Collision-Resistant Multiple Access. In *Proc. of ACM MobiCom 2011*.

[12] Miu, A., Balakrishnan, H., and Koksal, C. E. Improving Loss Resilience with Multi-Radio Diversity in Wireless Networks. In *Proc. of ACM MobiCom 2005*.

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

[14] Nazer, B., and *et al*. Ergodic Interference Alignment. In *Proc. of IEEE ISIT 2009*.

[15] Peel, C., Hochwald, B., and Swindlehurst, A. A Vector-Perturbation Technique for Near-Capacity Multiantenna Multiuser Communication-Part I: Channel Inversion and Regularization. *IEEE Transactions on Communications 53*, 1 (2005), 195–202.

[16] Rahul, H., Kumar, S., and Katabi, D. MegaMIMO: Scaling Wireless Capacity with User Demand. In *Proc. of ACM SIGCOMM 2012*.

[17] Rankov, B., and Wittneben, A. Spectral Efficient Protocols for Half-Duplex Fading Relay Channels. *IEEE Journal on Selected Areas in Communications 25*, 2 (2007), 379–389.

[18] Shrivastava, V., and *et al*. CENTAUR: Realizing the Full Potential of Centralized WLANs Through a Hybrid Data Path. In *Proc. of ACM MobiCom 2009*.

[19] Stanford Information Networking Group (SING). SING Datasets. `http://sing.stanford.edu/srikank/datasets.html`.

[20] Suh, C., Ho, M., and Tse, D. N. Downlink Interference Alignment. *IEEE Transactions on Communications 59*, 9 (2011), 2616–2626.

[21] Vutukuru, M., Balakrishnan, H., and Jamieson, K. Cross-Layer Wireless Bit Rate Adaptation. In *Proc. of ACM SIGCOMM* (2009).

[22] Weikert, O., and ZÃűlzer, U. Efficient MIMO Channel Estimation With Optimal Training Sequences. In *Proc. of the Workshop on Commercial MIMO-Components and Systems (CMCS 2007)*.

[23] Woo, G. R., Kheradpour, P., Shen, D., and Katabi, D. Beyond the Bits: Cooperative Packet Recovery Using Physical Layer Information. In *Proc. of ACM MobiCom 2007*.

[24] Xie, X., Zhang, X., and Sundaresan, K. Adaptive Feedback Compression for MIMO Networks. In *Proc. of ACM MobiCom 2013*.

[25] Zhang, X., Sundaresan, K., Khojastepour, M. A., Rangarajan, S., and Shin, K. G. NEMOx: Scalable Network MIMO for Wireless Networks. In *Proc. of ACM MobiCom 2013*.

[26] Zhou, W., Li, D., Srinivasan, K., and Sinha, P. DOMINO: Relative Scheduling in Enterprise Wireless LANs. In *Proc. of ACM CoNEXT 2013*.