AutoMAC: Rateless Wireless Concurrent Medium Access

Aditya Gudipati
adityag1@stanford.edu
Stanford University
Stanford, CA, USA

Stephanie Pereira
sfp@stanford.edu
Stanford University
Stanford, CA, USA

Sachin Katti
skatti@stanford.edu
Stanford University
Stanford, CA, USA

Abstract

Current wireless network design is built on the ethos of avoiding interference. In this paper we question this long-held design principle. We show that with appropriate design, successful concurrent transmissions can be enabled and exploited on both the uplink and downlink. We show that this counter-intuitive approach of encouraging interference can be exploited to increase network capacity significantly and simplify network design. We design and implement AutoMAC, a novel MAC and PHY protocol that exploits recently proposed rateless coding techniques to provide such concurrency. We show via a prototype implementation and experimental evaluation that AutoMAC can provide a 60% increase in network capacity on the uplink compared to traditional Wifi that does omniscient rate adaptation and a 35% median throughput gain on the downlink PHY layer as compared to an omniscient scheme that picks the best conventional bitrate.

Categories and Subject Descriptors

C.2 [**Computer Systems Organization**]: Computer - Communication Networks

General Terms

Algorithms, Performance, Reliability, Design

Keywords

Wireless Communication, MAC Protocol, Interference exploitation, Successive Interference Cancellation

1. INTRODUCTION

A basic precept of current wireless design is that at any instant only one transmission should be happening on any given frequency. For example, WiFi networks attempt to ensure that at any time a single client transmits to the AP via a random access MAC protocol. Similarly, cellular networks use synchronous scheduling techniques such as OFDMA, and the AP ensures that on any given subcarrier, only one node is transmitting at any instant. Correspondingly, when the AP is transmitting to clients, it ensures that only one client's packet is transmitted on any given frequency at any time [1].

The key reason for this design approach is of course to avoid the harmful effects of interference. If concurrent transmissions are allowed on the same frequency, packets interfere and the receiver cannot decode the constituent packets. Avoiding such interference when the basestation is transmitting is simple since the basestation controls to whom it transmits. But, when client nodes are transmitting, current network designs spend significant overhead (in the form of control traffic such as RTS/CTS, backoffs) to sense and avoid interfering with each other. Consequently, such an interference avoidance approach limits throughput, both fundamentally since only one node can transmit at any time, as well as due to protocol overheads such as RTS/CTS that are needed to avoid interference.

In this paper, we take the opposite approach. Instead of avoiding interference, we show that with the appropriate design, networks can systematically exploit interference to significantly increase throughput. We design and implement AutoMAC, a new PHY/MAC design that encourages multiple clients to transmit simultaneously to the AP on the uplink and similarly allows the AP to transmit packets to multiple clients concurrently on the downlink. Inevitably inerference results and the current design would be forced to throw away the receptions since they are undecodeable. Instead in AutoMAC, we design a novel *rateless PHY layer encoding and decoding algorithm* that allows the receivers to decode all the constituent packets from interfered receptions. The rateless code provides another critical benefit: it allows successful concurrent transmissions without requiring that the sender know either their channel to the receiver nor the identities of the other concurrent senders, each node simply sends a continous stream of rateless encoded symbols until the receiver decodes. As we will show in Sec. 2, traditional PHY layer modulation and coding techniques would incur enormous MAC layer coordination overhead to realize the benefits of exploiting interference. In AutoMAC, the rateless property of its PHY ensures that concurrency benefits can be realized with a simple, low overhead MAC.

We provide detailed intuition on how AutoMAC exploits interference in the following section 2, but we first summarize our main contributions. Theoretically, we show that AutoMAC is asymptotically optimal, i.e. it achieves the capac-

[1]The above discussion is for a single antenna. MIMO allows multiple transmissions on the same frequency, but separates them on the spatial dimension, and our approach naturally generalizes on the spatial dimension.

ity of the Gaussian Multiple Access channel on the uplink, and the Gaussian broadcast channel on the downlink. For experimental evaluation, we have prototyped AutoMAC in the GNURadio [3] SDR platform and evaluated it in an indoor testbed via experiments using USRP2s, as well as trace drive simulations. We compare AutoMAC with the standard 802.11 MAC which performs omniscient rate adaptation. This scheme has perfect channel knowledge, and always picks the optimal bitrate. However, it suffers from the inefficiencies of the 802.11 MAC protocol. Our evaluation shows that:

- On the uplink, AutoMAC's decoding algorithm can handle and decode an interfered signal that consists of up to 3 concurrent transmissions. AutoMAC thus completely eliminates hidden terminals. Similarly, on the downlink, AutoMAC can send 3 packets concurrently to different nodes in the same transmission.
- AutoMAC achieves a median throughput that is 35% *better than the omniscient scheme* on the downlink due to its ability to multiplex concurrent packets.
- In comparison with 802.11 carrier sense with omniscient rate adaptation, we show that AutoMAC achieves nearly 60% throughput improvement on the uplink in dense networks. The source of the gains come from both being able to exploit interference, as well as having an efficient MAC protocol.

Finally, we note that there has been considerable interest in recent work in combating collisions and interference [4]. However, all of them still consider interference as a necessary evil that can be dealt with in smarter ways when it inevitably happens, but the main goal is still to avoid it. AutoMAC makes a fundamental leap: it shows that interference should be encouraged and exploited to increase capacity. We believe AutoMAC thus represents the next step in the evolution of our understanding of how to handle interference, from avoiding it in current designs with CSMA [23, 7], to dealing with it [11, 4, 9, 8], to exploiting it.

2. INTUITION

To give intuition about AutoMAC, we use the classic example of two nodes Alice and Bob connected to an AP. Throughout this section, we assume that the channel attenuation between Alice and Bob to the AP is h_1 and h_2 respectively. For exposition simplicity, we assume channel reciprocity and hence the channels from the AP to the two nodes have the same parameters. We assume that all nodes transmit with power P, and the noise at all nodes is N.

2.1 Exploiting Interference on the Uplink

To avoid interference on the uplink, the optimal current system will schedule Alice and Bob to transmit one after the other in separate time slots. Assuming no MAC overhead, a fair (equal) time allocation to both Alice and Bob, and capacity achieving codes, information theory tells us that the throughput of the system is given by:

$$R_{nint} = \frac{1}{2}\left(\log_2\left(1+\frac{|h_1|^2P}{N}\right)+\log_2\left(1+\frac{|h_2|^2P}{N}\right)\right). \quad (1)$$

In our approach, Alice and Bob are explicitly asked to interfere. In other words, Alice and Bob transmit concurrently and the AP receives

$$y = h_1x_1 + h_2x_2 + n \quad (2)$$

How might the AP decode Alice or Bob's signals? One possible approach is for the AP to try and decode one of the packets while treating everything else (including the other interfering transmission) as noise, and if successful, subtract the decoded packet out and then decode the second packet interference free. This approach is referred to in the literature as Successive Interference Cancellation (SIC).

Assuming Alice's transmission is decoded first, the optimal throughput that Alice can achieve is given by:

$$R_A = \log_2\left(1+\frac{|h_1|^2P}{|h_2|^2P+N}\right) \quad (3)$$

Once Alice's transmission is decoded, the AP can subtract her contribution to the received signal h_1x_1 (feasible now since the AP knows both h_1 and x_1), to obtain $y' = h_2x_2 + N$. The AP can now decode Bob's signal at a rate which is exactly the same as when Bob alone was transmitting and there was no interference from Alice:

$$R_B = \log_2\left(1+\frac{|h_2|^2P}{N}\right). \quad (4)$$

The benefits of exploiting interference are now apparent. Specifically, Bob is able to decode his signal interference-free, as if Alice had not transmitted at all! Hence the extra throughput that Alice obtained is over and above the throughput Bob would have obtained if there was no interference, and thus strictly better than the interference-free case. If the decoding order is reversed (i.e. if Bob's packet is decoded first and then subtracted out to decode Alice's), Alice's transmission would see an interference free channel and thus any throughput Bob would achieve would be essentially for free compared to the non interference case. Assuming the AP equally alternates between these two decoding orders, the sum throughput of the system can be shown to be:

$$R_{sic} = \log_2\left(1+\frac{(|h_1|^2+|h_2|^2)P}{N}\right) \quad (5)$$

As we can see $R_{sic} > R_{nint}$, i.e. the system throughput achieved by SIC is strictly greater than that achieved by the optimal interference-avoidance scheme.

Said another way, for every transmission that the optimal interference-avoidance system makes (either from Alice or Bob), the other node can also transmit concurrently and get extra throughput without hurting the ongoing transmission which can be decoded interference free after SIC. The above intuition is not our contribution, information theory calls this the multiple access channel and its well known that exploiting interference in this manner provides significantly higher performance than avoiding interference. Fig. 1 plots the achievable throughput combinations for Alice and Bob when they are exploiting interference compared to the conventional interference avoidance scheme.

2.2 Exploiting Interference on the Downlink

In the downlink, the AP is transmitting to Alice and Bob. The current practice is to send to Alice and Bob one after the other using power P for each transmission. In that case, the downlink system throughput of the optimal interference

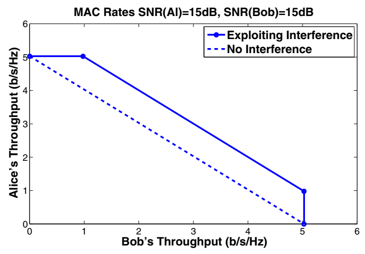

Figure 1: Throughputs Alice and Bob can achieve by exploiting interference compared to the optimal interference avoidance scheme in the uplink.

avoidance scheme is

$$R_{nint} = \frac{1}{2}\left(\log_2\left(1+\frac{P|h_1|^2}{N}\right)+\log_2\left(1+\frac{P|h_2|^2}{N}\right)\right). \quad (6)$$

We show that we can do better using ideas from information theory [1] where the problem of downlink transmission is termed the broadcast channel and the best possible rates for each of the users has been characterized analytically. Wireless is a broadcast medium, so every transmission from the AP reaches both Alice and Bob, even if it is not meant for them. AutoMAC takes advantage of this property to send packets concurrently for both Alice and Bob in the same transmission.

Suppose the AP needs to send the symbols x_1 and x_2 for Alice and Bob respectively. It has a power budget of P per transmission that must be allocated between x_1 and x_2. Since the AP has no channel knowledge, we decide to split the power equally among Alice and Bob. Thus, the AP sends $(x_1 + x_2)/\sqrt{2}$ so that each symbol has power $P/2$.

How can Alice and Bob decode? Bob receives the signal $h_2x_1/\sqrt{2} + h_2x_2/\sqrt{2} + N$. Notice that this is similar to the uplink interference case, except that both symbols experience the same channel and the channel depends on the receiver, i.e. Alice or Bob. One can imagine using the same successive interference cancellation (SIC) approach to decode. However, there is one critical difference. In the downlink, Alice or Bob require only the packet intended for them, while in the uplink, the AP needed to decode all the packets. This implies that the node with the weaker channel from the AP should simply try to decode his packet treating everything else as interference. The reason is that it will take him the same amount of time to decode either his or the other packet, since both of them are allocated equal power and experience the same channel. Hence there is no reason to try decoding anything else, the node with the weak channel just decodes the packet intended for it.

However, the node with the stronger channel doesn't have to do the same. Instead, since the AP has to transmit enough to ensure that the weaker channel node can decode, the number of transmissions will be greater than that required by the stronger channel node to decode the packet of the weak node. Hence, the strong node first decodes the weak node's packet, and then applies SIC to cancel the interference, and decode his own packet interference free.

To understand rigorously why allowing users to interfere with each other produces higher rates, we refer the reader to [21]. The intuitive (but less precise) reason is that the Shannon capacity function is concave with increasing power: increasing the power of a transmission provides diminishing returns beyond a certain level. Hence instead of using all the transmission power for one node, an AP is best served by dividing it and sending to multiple nodes concurrently. To show the effect visually, Fig. 2 plots the optimal rates achieved using the above scheme by Alice and Bob for various realistic SNR combinations. The scheme achieves higher sum throughput when Alice and Bob have different SNRs and the same sum throuhgput as traditional interference-free schemes when Alice and Bob have equal SNRs.

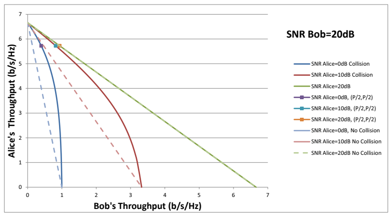

Figure 2: Throughputs Alice and Bob can achieve by exploiting interference compared to the optimal interference avoidance scheme in the uplink.

2.3 What's the Catch?

While the above intuitions on how exploiting interference can help improve throughput in both the uplink and the downlink have been well understood, realizing them in practice has been challenging due to one basic challenge: rate adaptation. Modern wireless networks perform rate adaptation, i.e. the senders estimate the channel quality to the receiver and pick the best bitrate that maximizes throughput while ensuring that the receiver can decode. Even assuming that nodes can do perfect rate adaptation (i.e. pick the optimal bitrate) given the channel to their receiver, the above scheme fails in the presence of interference. For example in the uplink, if multiple clients transmit concurrently, the effective SINR each of their transmissions will see will be lower than the channel SNR the transmitters had estimated earlier because the receiver has to treat the other interfering transmissions as noise when decoding. Consequently, none of the transmissions will be decoded since bitrates exhibit a threshold behavior [5], if the actual SINR is lower than the minimum required to decode that bitrate, decoding fails. This is consistent with the observations in [18] which doubts the future of SIC based protocols. The other design choice is of course to coordinate, i.e. each node tries to learn in advance the identity of the concurrent senders, their channels to the AP, and then appropriately encode its own transmission for the SINR his transmission will experience at the receiver. However, that approach adds enormous coordination overhead and is at odds with the random access nature of WiFi.

The same effect holds on the downlink since clients are using the same SIC approach to decode their respective packets, and the SINR a packet experiences is a function of both the channels as well as the interference. Remember that the strong channel node is using SIC to decode its packet, while the weak channel node decodes assuming everything else is noise. To ensure that they can decode, the AP has to encode their respective packets such that they are decodeable at the SINR of their respective transmissions. To do so, the AP would need to have accurate knowledge of the channels to both clients, else once again decoding will fail due to the threshold effect of current encoding techniques. Such channel estimation adds overhead and might be impractical in dense or mobile networks.

2.4 AutoMAC

The basic challenge in exploiting interference is that the sender does not know the SINR his packet will experience at the receiver in advance, yet it has to be able to encode its data such that the receiver can decode even when there are collisions and bootstrap SIC. In other words, we want the sender's encoding to have a *rateless* property, i.e. the sender should be able to generate a stream of rateless transmissions which enable the receiver to decode the original data after a certain number of rateless transmissions commensurate with the actual channel SINR are received, and in effect achieve the same throughput as one would achieve if the sender had known the SINR in advance and had encoded its data at the correct rate.

AutoMAC leverages recent work on rateless codes [6, 15] to solve the above problem. These codes allow the receiver to bootstrap SIC, since the senders transmit a stream of rateless coded transmissions until the receiver can decode. Hence eventually one of the colliding packets will be decoded and SIC kickstarts. So a simple MAC protocol would be to let both senders keep transmitting with the rateless code until their transmissions get ACKed. The receiver would employ SIC and ACK the senders as soon as their respective packets are decoded.

However, the above simple MAC protocol fails. Consider the case when the channels from Alice and Bob to the AP are equally strong, e.g. 10dB. Using the above MAC protocol they would keep transmitting concurrently until both their packets could be decoded. For simplicity, lets assume the AP tries to decode Alice's packet first. Since it is treating Bob's packet as noise when decoding Alice's, the effective SINR for Alice's transmission is 0dbB. Consequently Alice has to transmit quite a few rateless coded transmissions before the AP can decode. For example, with the Strider rateless code [6], Alice has to send 15 rateless transmissions before the AP can decode her packet. However, once the AP decodes, it can completely cancel Alice's contribution and decode Bob's signal interference free. In effect, this means Bob's transmissions see a SINR of 10dB, and can be decoded with far fewer transmissions compared to Alice. Again with Strider's rateless code, with a SNR of 10dB, Bob would need to only send 5 rateless transmissions. But since the AP does not ACK until it can decode Bob's packets fully, which it cannot until Alice's packet is decoded, Bob ends up sending 10 wasteful transmissions or nearly 3$\times$ the number he actually needs to transmit. Thus the naive MAC protocol leads to tremendous waste.

AutoMAC's second main contribution is a novel PHY/MAC protocol that solves the above challenges and allows one to exploit collisions through rateless codes. The key technique is *speculative ACKing*. Intuitively, in the above example, if the AP could speculatively ACK Bob's transmissions after it has received a sufficient number (around 5) to ensure it can decode Bob's packet after eventually decoding Alice's packet, then Bob can move on to his next packet and ensure that channel time is not wasted. The basic problem is that the AP has to estimate even before it has decoded either Alice or Bob's packets, when to speculatively ACK. The decision is dependent on the SNR of the channel from Bob which the AP now has to estimate in the presence of interference. AutoMAC designs a novel detection and prediction algorithm that allows it to accurately estimate channels under interference, and speculatively ACK to avoid waste.

3. DESIGN

In this section we describe the design of AutoMAC.

3.1 MAC Protocol

AutoMAC uses a simple AP driven MAC protocol. In AutoMAC the AP has an opportunity to transmit after every packet transmission in the network. In the uplink, after receiving the concurrent transmissions, the AP sends an ACK/NACK packet to inform the clients if any of their packets have been decoded. In the downlink, the AP is of course transmitting a packet. The basic idea is that whenever the AP is transmitting a packet (either data or ACK), if it wants to accept transmissions from the clients in the next slot, the AP appends a short contention advertisement at the end of its transmission. The contention advertisement informs nodes of the number, n of users that may transmit concurrently in the next slot. The advertisement may also include requests from the AP for retransmission of specific packets from particuar nodes, which as we will see in Sec. 3.2.1 is needed to ensure robustness when decoding fails after speculative ACKing.

After hearing the contention advertisement, nodes that have outstanding traffic enter into a contention period to determine which n nodes may transmit concurrently. We build on the recently proposed frequency domain backoff technique [17] that allows nodes to pick a random subcarrier out of the 64 available ones, and send a single tone on that subcarrier. The contention period lasts for 4 time slots, and the clients randomly pick one slot out of the four and one subcarrier out of the 64 to transmit the tone. Next, the AP just retransmits the information about the subcarriers and time slots on which it received tones in the contention period. The n nodes with the n smallest subcarriers and time slots win the contention period, and transmit concurrently after the contention period ends. Since there are a total of 256 time-frequency slots, the probability of two clients picking the same slot and subcarrier in a reasonably sized network is low. However, if network size grows, the contention period can always be increased. Note that the contention period is only needed when the AP wants to accept transmissions from new clients. If not, the existing set of clients continue to transmit concurrently after receiving the ACK/NACK packet.

Note that the AP can choose to not advertise any contention spots. In this case, two events can occur. One the AP itself may chose to transmit in the next slot. Or, the AP might ask the nodes which were concurrently transmitting in the previous slot to continue. The contention advertisement contains a special field to inform all nodes about these cases. The above MAC protocol provides each node with an equal amount of channel transmission time. However that by itself does not ensure fairness, as the discussion in Sec. 2 shows, the throughput a node achieves is also a function of the order in which it is decoded by SIC and the SNRs of the other colliding transmissions. In Sec. 3.6 we describe a novel algorithm that ensures fairness even under collisions.

3.2 Exploiting Interference on the Uplink

The n nodes that win the contention period transmit concurrently after the contention period ends. Each node uses the standard OFDM PHY with 64 subcarriers spread over

20MHz. As with normal WiFi, data symbols are sent only on 48 subcarriers, the remaining 16 are used for pilots and padding. The only change we make is replacing convolutional coding with the rateless code. We first describe how each node generates its packet.

3.2.1 Packet Encoding

Each node uses a rateless code to encode its data. AutoMAC is *orthogonal to the choice of the rateless code*, and can use either Strider [6] or spinal codes [15]. Since Strider has linear decoding complexity and is simpler to implement, we choose to use it in our current design.

Strider encodes data using the following steps:

1. Packet data is divided into chunks of size S KB. In each block we have D data blocks of length L bytes each. Current default values are $D = 30$, $L = 200$ and $S = 6$.
2. Each of the D data blocks is passed through the *base code* (Strider's current implementation uses a 1/5 rate tubo code and a QPSK constellation as the base code), to produce D blocks with $5L/2$ complex symbols each.
3. The D blocks are passed through Strider's encoding matrix R, which is a $M \times D$ matrix, where M stands for the number of transmissions. To generate the i'th transmission, the sender uses the i'th row of the encoding matrix and so on. The sender can generate as many transmissions as it needs since Strider is a rateless code.

The output of the Strider encoder is a frame of $5L/2$ complex symbols. These are divided into 48 equal sized sets which are mapped to the 48 OFDM subcarriers typical in WiFi. Next the standard OFDM PHY is used to generate the actual transmitted wireless signal, which includes doing a IFFT, adding a cyclic prefix, adding pilots on four subcarriers and adding a preamble at the start of the frame.

3.2.2 Decoding All the Constituent Packets with SIC

The concurrently transmitted frames go through the wireless channel and are received at the AP. Assuming h_{ij} is the channel for the j'th rateless transmissions from the i'th node. The signal at the AP for the j'th transmission is

$$y_j = \sum_{i=1}^{K} h_{ij} x_{ij} + N \qquad (7)$$

where x_{ij} is the rateless symbol transmitted by the i'th node in the j'th slot and N is the noise.

AutoMAC uses successive interference cancellation to decode the transmissions from the K nodes. To kickstart the process, the AP attempts to decode the first node's packet treating everything else as interference. Hence the first node's transmissions at the AP can be modeled as

$$y_j = h_{1j} x_{1j} + \sum_{i=2}^{K} h_{ij} x_{ij} + N \qquad (8)$$

Lets assume we need M rateless transmissions from the first node to decode its packet using Strider's rateless code. To decode the original data, Strider first needs to estimate the channel for each of the M rateless transmissions.

Per Frame Channel Estimation: AutoMAC uses known training symbols at the start of the transmitted OFDM frames. The estimation uses the least square algorithm due to its low complexity. Since the training symbols are defined in the frequency domain – each OFDM subband is narrow enough to have a flat frequency response, AutoMAC estimates the channel in the frequency domain as a complex scalar value at each subcarrier. Specifically, let $\mathbf{X} = (P_{ij}[0], \cdots, P_{ij}[64])$ be the vector of the training symbols used across the 64 subcarriers for a single OFDM symbol for the j'th transmission from node i, and C be the number of such OFDM training symbols. Let $\mathbf{Y_j}^{(c)}$, $c = 1, \cdots, C$, be the corresponding values at the receiver after going through the channel. The least squares algorithm estimates the channel frequency response of each subcarrier k for the j'th transmission as, $\hat{H}_{ij}[k]$, as follows:

$$\hat{H}_{ij}[n] = \frac{1}{C} \left[\frac{1}{P_{ij}[n]} \left(\sum_{c=1}^{C} Y_j^{(c)}[n] \right) \right]$$

Decoding the Packet using Strider: Next, AutoMAC decodes the data from the first node using the Strider rateless code. Since the AP has received M transmissions from node 1, it estimates the channel for all M transmissions as above. Since we are using OFDM, the received symbol on the n'th subcarrier can be written as

$$Y_j[n] = H_{1j}[n] X_{ij}[n] + N \qquad (9)$$

It then uses Strider's rateless decoding technique to decode the original packet from node 1. We skip the details of the rateless decoding algorithm and refer the reader to [6], but for our discussion it suffices to assume that the packet from node 1 is now decoded.

Subtracting Node 1's contributions from the collision: Next, the AP subtracts node 1's transmissions from the received collisions. Note that the AP has to do this for all the M rateless transmissions from node 1. However, since the actual interference happens in the time domain, we have to subtract the decoded signal in the time domain even though the rateless decoding happens after the FFT in the frequency domain.

To do so, we first compute the time domain channel response by applying the inverse fast Fourier transform (IFFT) to the frequency response of the channel estimated above. This is done for all M transmissions. Next, the AP re-encodes the decoded data using Strider's rateless code and OFDM to re-generate the M rateless transmissions. Each rateless transmission is then passed through the time-domain channel response computed above to obtain the exact interference contribution in the received signal.

The time domain channel response is represented using a standard finite impulse response (FIR) filter in the digital domain. To generate the digital samples for cancellation, AutoMAC convolves the known signal with the FIR filter representing the channel. Let $s_{1j}[n]$ be the known transmitted digital sample at time n by node 1 in the j'th rateless transmission. These digital samples are fed into the FIR filter. The output $x_{1j}[n]$ of the filter is the linear convolution of $\hat{h}_{1j}[n]$ (the time domain FIR representation of the channel response) $s_{1j}[n]$:

$$x_{1j}[n] = \sum_{k=0}^{F-1} \hat{h}_{1j}[k] s[n-k]$$

After this step, the radio subtracts the estimates of the transmit signal from the received samples $y_j[n]$:

$$\hat{y}_j[n] = y_j[n] - x_{1j}[n]$$

The above process is repeated recursively until all n packets are decoded. As soon as a particular node's packet is decoded, a spot opens up in the next slot for another node to transmit concurrently. The AP signals this open slot via the contention advertisement discussed in Sec. 3.1.

3.3 Accurate Channel Estimation Under Interference

To apply the above SIC technique, the AP needs to now estimate channels from all the n nodes. The above Least Squares algorithm works very well when there is a single node transmitting, but can be inaccurate if the training symbols used for channel estimation collide with other training symbols. To ensure that we obtain accurate channel estimates, AutoMAC leverages the AP driven MAC protocol.

Specifically, after the contention period ends, the succesful nodes transmit training blocks in separate time slots in order of their contention winning sequence. In other words, the node with the smallest contention subcarrier and time slot index transmits 4 OFDM symbols, the first one is a symmetric pseudorandom sequence for time and frequency synchronization via Schmidl-Cox [16], and the next two symbols are the preamble used for channel estimation. The fourth symbol is used for identifying the node and the packet. Next, there is a short guard interval of $2\mu s$, and then the node with the next highest contention subcarrier index transmits the training symbols and so on. Such clean training symbol transmission one after the other is possible because the nodes can synchronize on the reception of the contention advertisement, which essentially acts like a common clock for the network. AutoMAC is now free to use the standard least squares algorithm for all nodes since pilots and preambles are obtained cleanly for all of them. After n such training block transmissions, all the nodes start transmitting their rateless encoded frames concurrently.

Estimating Wireless Hardware Distortions: Apart from the channel, the training symbols help estimate two other practical hardware effects that distort the signal. The first one is *carrier frequency offset* (CFO) which exists because the oscillators at the sender and receiver are never perfectly synchronized to the same frequency. Consequently, there is always a small frequency offset δf_i, between the i'th transmitter and receiver. The CFO causes a linear displacement in the phase of the received signal that increases over time, i.e.

$$y_j[n] = \sum_{i=1}^{n} h_{ij} e^{j2\pi n \delta f_i T} x_{ij} + N \tag{10}$$

In AutoMAC, the receiver calculates the CFO for every transmitter thats part of a collision. We use the standard Schmidl-Cox algorithm [16] on training symbols. Remember that the first symbol is pseudorandom repeating sequence that is used by the Schmidl-Cox algorithm for packet detection. By correlating shifting halves of samples, the receiver can figure out the start of the packet where the peak occurs. The phase of the correlation peak also provides an estimate of the fractional CFO, defined as a fraction of the OFDM subcarrier width.

When decoding a packet, CFO correction is applied on all the received samples. However, once decoded AutoMAC has to cancel the decoded packet's contribution from the received signal. Hence it has to apply back the CFO distortion to ensure that it cancels exactly the signal that interfered in the time domain. Applying CFO back is a matter of multiplying the time domain samples by a factor of $e^{j2\pi\delta f_i n}$.

3.4 Speculative ACKing

AutoMAC sends multiple packets in every time-slot from different clients to the AP. The channel between each client and the AP will be different, i.e. some will be stronger than the others. Consequently, packets from different clients can be decoded at different times. The exact time is a function of two factors: the order of decoding, and the relative SNRs of the different client-AP channels that are involved in the concurrent transmission.

However, as described, the above PHY and MAC protocol will keep sending rateless transmissions for packets even though they could have been decoded with far fewer transmissions. The reason is of course that the AP does not ACK a packet until it has actually decoded that packet. For example the packet that is decoded last after interference contributions from all other transmissions are subtracted out is ACKed only after the sender has sent at least as many transmissions as the first packet that has to be decoded *while considering all other packets as interference*. Clearly decoding the first packet requires far more rateless transmissions since its effective SINR is low due to interference from the other packets, while the last decoded packet requires far fewer since it sees a much better SNR.

AutoMAC designs a novel speculative ACKing technique to combat the above waste. The key insight is an algorithm that accurately estimates how many rateless transmissions any packet will take to be decoded. Prior work on Strider [6] has shown that there is a fixed relationship between the number of rateless transmission required to decode a packets and the SINR at which the transmisisons are received. Hence, if a node can estimate the effective SINR for any packet, it can accurately estimate how many rateless transmissions it will take to decode that packet.

The challenge then is to estimate the effective SINR for any packet. This is a function of both the raw channel from the specific client, the position at which that client's packet is decoded, and the raw channels from the other clients. To keep track, the receiver keeps track of the total SINR we have accumulated for a specific packet p as follows

1. For every transmission, the node estimates the channel for all the packets that are part of the transmission. Note that this is possible because of the MAC protocol that allows nodes to send training blocks before each transmission one after the other.
2. For packet p, the receiver keeps track of total power received as $T_p = \sum_i^K h_i P$ where h_i is the channel gain for p's i'th rateless transmission and P is the transmit power. It does the same for all the other packets that are received in any transmission.
3. The total interference I_p for decoding p is computed as the sum of the powers for all the other packets that are concurrently transmitted with p and act as interference for p's decoding. In other words they get decoded after p does.
4. The effective SINR for p after receiving M transmissions that contain it is then computed as $T_p/(I_p + M * N)$.

To estimate if M transmissions are enough, the receiver consults the SINR vs number of transmissions graph for Strider. If M is equal to or greater than the number required, then the receiver speculatively ACKs packet p.

Note that when the receiver speculatively ACKs, p will likely not actually be decoded yet. That happens when all the other packets that are supposed to be decoded before p get decoded. Of course, there will be cases where the speculative ACK might be optimistic, i.e we may need more rateless transmissions to decode p. In that case, the receiver can send an ACK that specifically requests more rateless transmissions for packet p from the appropriate sender. Fig. 3 demonstrates an example of AutoMAC's MAC protocol including speculative ACKing for a three node scenario with Alice, Bob and Charlie transmitting to the AP.

3.5 Exploiting Broadcast on the Downlink

AutoMAC uniquely allows the AP to use the same encoding technique on the downlink. Assuming the AP wishes to transmit n packets on the downlink to n different nodes, it uses the following procedure

1. Encode each packet using Strider's rateless code to create a stream of rateless frames, without yet passing them through the OFDM PHY.
2. Take the n rateless encoded frames, add them up in the complex domain to generate one frame.
3. Next it prepends the appropriate headers which include information about all the packets being sent together. Finally, the frame is passed through the OFDM PHY and transmitted.

3.5.1 Decoding at the Clients

To decode the above rateless encoded frames, the client uses a similar procedure as the uplink. Lets say the client has accumulated m such rateless frames each having a packet intended for it among the n packets that are encoded together. In effect, except for its own packet, everything else acts as interference. The client can use a technique similar to successive interference cancellation to decode its own packet.

Remember from the discussion in Sec. 2 that in the downlink case, the client with the strongest channel should decode its packet after having decoded and cancelled all the other packets. The next strongest channel node should decode his second-last and so on. How can each client know the relative strengths of the SNRs of all the other clients? Assuming such channel knowledge is untenable for AutoMAC since its main goal is to eliminate the need for such coordination.

AutoMAC's key insight is to let every client pretend that it is decoding its own packet last, i.e. every client believes that it has the best channel initially. Eventually, the node with the actual strongest channel will accumulate enough transmissions to ensure that the effective SINR is sufficient to decode the packet intended for it. As soon as that happens, it will speculatively ACK just like in the collision case. The next speculative ACK will come from the node with the second strongest channel and so on. Thus, in effect the AP will get an ordering of the channel strengths. The AP uses this information to signal to all the clients the order in which they should decode the transmissions to recover their own packets. Each client now applies SIC in the specified decoding order to decode their own packet.

Finally, note that speculative ACKs serve the same purpose on the downlink as the uplink. After a node speculatively ACKs, a new slot opens up for the AP to transmit a packet to a different node in the downlink. The AP picks one of the outstanding packets, according to the fairness policy discussed next.

3.6 Fairness

The MAC protocol as described does not ensure fairness. To see why consider the uplink case where nodes are concurrently transmitting. The node whose packet is decoded while treating all other packets as interference experiences very low SINR, while the node whose packet is decoded last achieves the best SNR. Thus even though both nodes use the same power and get the same channel time, the node decoded last will achieve much better performance than the node decoded first. The same effect exists in the downlink since the exact same SIC algorithm is used in there too.

AutoMAC attempts to ensure the following notion of fairness. A node should not achieve a lower throughput with AutoMAC than that it would achieve with a conventional MAC protocol that allocates equal channel time to all nodes. The reasoning is that AutoMAC should never be less fair than what a well designed conventional MAC protocol would have achieved in current networks.

Ensuring Uplink Fairness: In order to ensure uplink fairness as described above, AutoMAC requires the AP to estimate the maximum possible throughput that each contending node could achieve using a conventional PHY/MAC scheme. This estimate is arrived at and subsequently updated based on the signal strength observed whenever the corresponding node transmits a packet. The actual throughput that the node would achieve in a conventional setting with time based fairness can then be calculated by simply dividing this estimate by the number of contending nodes. This throughput value is deemed to be the node's *fair share*.

The key idea is whenever a node that has been under-served, wins the contention, the AP should allow it to transmit multiple packets without contending again. By under-served, we mean the node's actual throughput is significantly lower than its fair share. To achieve this, all nodes are required to set one bit in their transmitted packets based on whether or not they have more data to send. When the AP decodes an under-served node's packet and observes this bit to be set, it asks the corresponding node to continue transmitting the next batch without going into contention.

Ensuring Downlink Fairness: AutoMAC aims to provide the same time-based notion of fairness on the downlink, i.e. traffic to each client gets an equal amount of channel time on the downlink. Similar to the uplink, the client decoding his packet last experiences an interference free channel, as if the AP was transmitting only to that client. Thus as long as the AP ensures that every node gets equal opportunity to decode its packet last, it will ensure the required fairness.

However, the challenge is that in the downlink the AP does not have the same luxury of choosing random packet decoding orders, the order is fixed by the relative channel strengths of the clients. Hence, the probability of a weak channel node being the last to decode is lower than that of a strong channel node. Consequently, the AP uses a slightly different algorithm. For each client, it keeps track of how many times it has been asked to decode last. When the next opportunity arises to pack multiple packets in a downlink transmission, it picks the client whose packets have been decoded last the least number of times. It then searches for a client with outstanding packets which it knows in the past has had a weaker channel than the first client (because it was asked to decode before the first client). That client's transmission is added to the broadcast transmission. The process is repeated re-

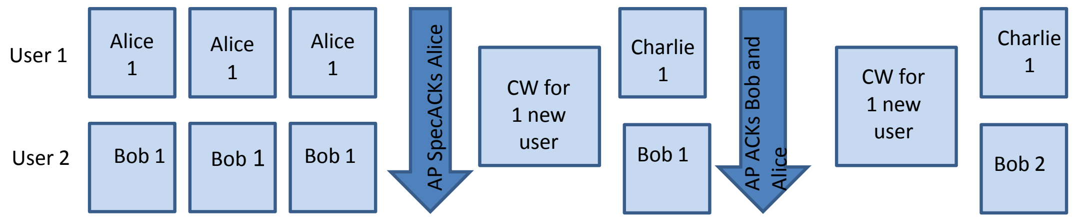

Figure 3: AutoMAC: MAC protocol with Speculative ACKing

cursively until no other clients can be found to fit the above condition. Note that with the above algorithm, sometimes we may find no other client's transmission to send along with the first client's transmission, and thus lose the throughput gains we could get from multiplexing concurrent transmissions. However, AutoMAC consciously makes this tradeoff to ensure fairness. Over the long run, the above algorithm ensures that each client's packets gets equal amount of time being decoded last in the downlink broadcast transmissions, ensuring the desired fairness.

4. EVALUATION

We evaluate AutoMAC on an indoor testbed of 15 USRP2s and trace driven simulations. For evaluating the AutoMAC's PHY layer gains we compare with **omniscient scheme**. This scheme has perfect advance knowledge of the channel strength, and picks the maximum possible bitrate that can be decoded error free. The bitrate choices are from the 8 different bitrates available in the 802.11 standard. For the evaluation of AutoMAC's MAC protocol, we compare AutoMAC with the **802.11 protocol doing omniscient rate adaptation**. This scheme also has perfect advance channel knowledge, and picks the best possible Wifi bitrate. However, it suffers from the inefficiencies of the 802.11 MAC protocol. This is done in order to exclude the benefits of Rate Adaptation, achieved due to Strider's implementation [6] in our analysis.

As observed in [6] we achieve a throughput that is roughly 30% higher than the omniscient scheme on the uplink. The rest of our findings are summarized below:

- In our testbed experiments, AutoMAC outperforms the omniscient scheme by 35% on the downlink too!
- AutoMAC's decoding algorithm can decode all constituent packets within collisions with upto 3 packets. AutoMAC thus eliminates hidden terminals in our testbed. Further, on the downlink it can multiplex upto 3 packets together.
- AutoMAC's design accurately estimates channels, frequency and sampling offsets even under collisions.
- In networks with contention, AutoMAC provides a throughput gain of 60% over 802.11 MAC doing omniscient rate adaptation on the uplink and 50% over an 802.11 style MAC ensuring time based fairness.

5. INDOOR TESTBED EXPERIMENTS

In this set of experiments, we evaluate AutoMAC using experiments in our indoor testbed of USRP2s. We compare with the omniscient scheme which has perfect advance channel knowledge and picks the best bitrate which can be decoded. This is done in order to exclude the benefits of rate adaptation via Strider and focus on the gains due to exploiting interference.

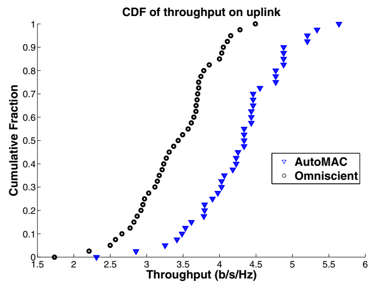

Figure 4: AutoMAC: CDF of throughput gains on uplink

5.1 AutoMAC: Exploiting Interference on the Uplink

Method: In this experiment, we statically place one USRP node which acts as the AP. Next, we randomly place 10 client nodes in the testbed so that they have different channels to the AP. For each client-AP link, we transmit 1000 packets between the two nodes using all the different bitrates, and pick the one which achieves the maximum throughput. Next, we connect all the client nodes together to a common clock as described in prior work [13] to ensure that they transmit one after the other according to an omniscient schedule that allocates equal channel time to all nodes and avoids collisions. For AutoMAC, we use the same clock setup, but allow a random set of two nodes to concurrently transmit.

Recall from Sec. 3.3 that in order to accurately estimate the channel under interference, we require the training blocks to arrive in different time slots. In a commerical deployment we envision the clients would synchronize based on the reception of the conntention advertisement from the AP which essentially acts like a clock. However, in our setup we need to rely on an external common clock for the synchronization. This is the distinguishing factor from the collision experiments done in [6], which were asynchronous.We repeat this experiment 10 times for the same location of the nodes and take the average throughput for either scheme, expressed in terms of bits/second/Hz. We then change the locations of the client nodes to get a different SNR and repeat the above procedure. We plot the CDF of throughputs achieved by the two schemes in Fig. 4.

5.1.1 *Accuracy of Channel and CFO Estimation*

In this experiment, we evaluate the accuracy of AutoMAC's channel and CFO estimation techniques. We select 3 client nodes and allow them to transmit to the AP using AutoMAC. Next, we immediately make them transmit to the AP separately one after the other without collisions using the standard WiFi OFDM PHY . Note that while doing Successive Interference Cancellation, any error in channel estimates of the stronger client, leads to increased interference for all the

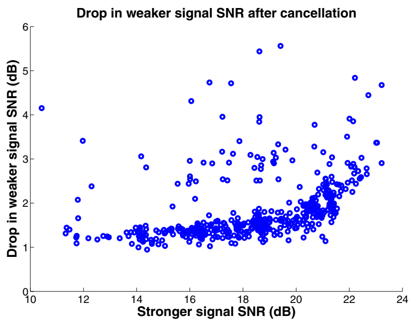

Figure 5: Reduction in Channel SINR post cancellation due to errors in Channel and CFO Estimation

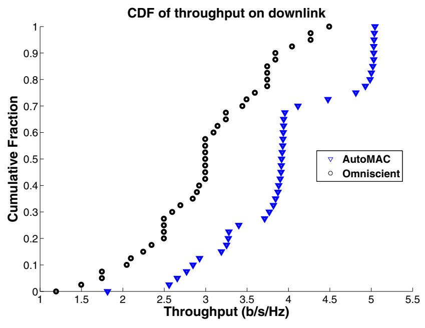

Figure 6: AutoMAC: CDF of throughput gains on downlink

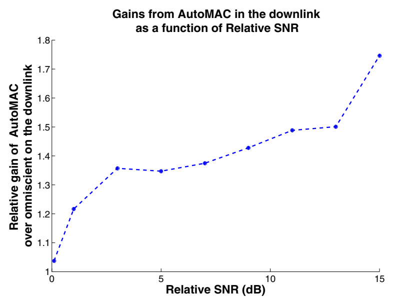

Figure 7: Gains from AutoMAC in the downlink as a function of Relative SNR

clients that need to be decoded subsequently. We plot the corresponding reduction in channel SINR for the weaker client as a function of the signal strength of the stronger client in Fig. 5.

5.1.2 *Accuracy of Speculative ACKing*

In this experiment, we evaluate the accuracy of AutoMAC's speculative ACKing technique. We check when the AP speculatively ACKs in anticipation of being able to decode a packet after SIC, what fraction of the time it actually is succesful. Using the results of the experiment, we observe that the AP is totally accurate in atleast 80% of the scenarios. In 10% of the scenarios, the AP sends a speculative ACK before it is actually capable of decoding. We believe, these errors do not cost much in terms of the system throughput, since the AP can request more rateless transmissions from the appropriate sender. These additional transmissions contribute constructively towards the packet decoding and hence are not wasteful. However, in 10% of the scenarios, the AP fails to send the speculative ACK on time. This results in wasteful transmissions (typically just 1 extra transmission) which could have been avoided. Overall, the inaccuracies of speculative ACKing have negligible effect on the system throughput $(2-3\%)$.

5.2 AutoMAC: Exploiting Interference on the Downlink

Method: In this experiment, we statically place one USRP node which acts as the AP. Next, we randomly place client nodes in the testbed so that they have different channels to the AP. For each client-AP link, we transmit 1000 packets between the two nodes using all the different bitrates, and pick the one which achieves the maximum throughput. Finally, we repeat the experiment with AutoMAC's downlink technique. We plot the CDF of throughputs achieved by AutoMAC and the conventional PHY in Fig. 6.

Analysis: AutoMAC provides a median throughput gain of 35% over the optimal conventional downlink scheme. The reason is of course higher concurrency, a AutoMAC AP manages to deliver packets to multiple destinations in every transmission. The gains vary and as in the uplink, are a function of the relative SNRs between the clients. We plot the relative gains of AutoMAC as compared to the omniscient scheme as a function of the relative SNRs in Fig. 7. The figure shows that the throughput of the omniscient scheme is almost as good as AutoMAC when the SNRs of the clients are similar, but the gains increase as the relative SNR increases. This is consistent from the theoretical insight about broadcast channels as discussed in Sec. 2 and in [1].

6. TRACE DRIVEN EMULATION

Similar to prior work [22, 10] we turn to trace driven emulation to evaluate AutoMAC's performance in a larger network setting with varying traffic. We compare against a state-of-the-art 802.11 MAC protocol which performs optimal rate adaptation. We also compare against an 802.11 style MAC protocol with CSMA and exponential backoffs which also ensures equal channel time fairness. However, since these techniques require fast turnaround times to send ACKs and synchronized feedback etc, which the USRP2s are not equipped to do. Additionally, it is hard to generate controllable high-contention and varying traffic in experimental settings.

Trace: We collect real channel information for the simulations via a high precision channel sounder [2]. The channel sounder is an equipment designed for high precision channel measurement, and provides almost continuous channel state information over the entire measurement period, and can measure channel SNRs as low as -3dB. Our experiments are conducted at night on the band between 2.426 and 2.448GHz which corresponds to WiFi channel 6, and include some interference from the building's WiFi infrastructure which operates on the same channel. The transmit node of the channel sounder is placed at ten different locations in our testbed, and its channel to the receive node is measured over a period of 100 seconds where the receive node records and estimates detailed channel state information for all frequencies in the 20Mhz channel. We collect around 100000 measurements over a 100 second period for 10 different locations of the transmit node.

Emulator: We feed this trace to a custom emulator written in MATLAB with the AutoMAC's implementation. AutoMAC allows for 2 concurrent transmissions on the uplink in each time slot. The emulator implements a 802.11 style MAC

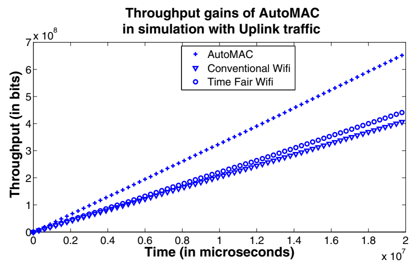

Figure 8: AutoMAC: Throughput gains of AutoMAC in simulations with Uplink Traffic

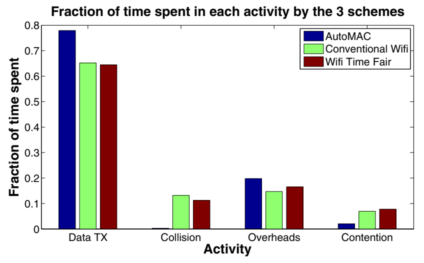

Figure 9: Fraction of time spent by clients in different states across the 3 schemes

with ACKs, CSMA and exponential backoff with the default parameters. This scheme is allowed perfect knowledge of the channel strength and can pick the optimal conventional bitrate. This is done in order to ensure that the gains of rate adaptation [6] do not show up in our analysis.

To better judge the source of the gains, the emulator also implements an 802.11 style MAC with CSMA and exponential backoffs, which ensures a notion of time based fairness. That is, it ensures all contending nodes get roughly the same channel time. This is done by allocating a fixed time window to each node whenever it wins the contention. The user can only transmit on the channel for this fixed period of time irrespective of its channel strength.

6.1 Performance

Method: We compare AutoMAC with the 802.11 MAC protocol which does omniscient rate adaptation and with an 802.11 style MAC which ensures time based fairness. In these experiments, we simulate the network with 8 nodes connected to an AP. We plot the evolution of throughput on the uplink with time in the three scenarios in Fig. 8.

Analysis: AutoMAC provides a 60% gain on the uplink over the conventional 802.11 MAC protocol with omniscient rate adaptation, and a 50% gain over the 802.11 style MAC protocol ensuring time based fairness. The gain due to imposing time based fairness on 802.11 MAC is consistent with the observations in [20].

In Fig. 9, we plot the fraction of time spent by the network in different activities (i.e. useful data transmission, collisions, Overheads (SIFS, ACK, packet header) and contention) for the three different MAC protocols.

A close observation of Fig. 9 leads us to a better understanding of the source of the gains.We observe the following:

- The breakup of times spent looks similar for the Conventional Wifi and the Time fair Wifi. The main difference between the two protocols shows up only when we analyse the individual nodes seperately.

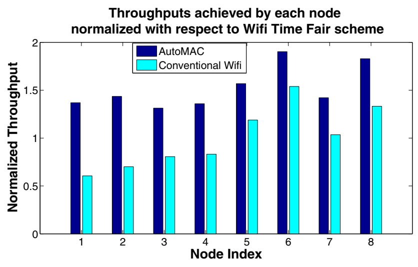

Figure 10: AutoMAC guarantees atleast as much fairness as the optimal conventional scheme

- Note that the nodes in other MAC protocols waste a significant portion of their channel access oppurtunities due to lack of coordination and inability to decode from collisions. The centralized MAC protocol of AutoMAC reduces collisions to a minimum. Here, collision is defined as observing more than 2 simultaneous transmissions on the channel.

- The contention time is also comparatively lower in AutoMAC, due to the frequency domain backoff technique [17].

- The comparison of overheads is slightly more involved. Overheads here include the time spent by nodes in interframe spaces,(i.e. SIFS, DIFS), while receiving ACK and while transmitting packet headers and training symbols. The time spent in receiving ACK is comparatively lower in AutoMAC, since, AutoMAC requires an ACK to be sent only once every batch. However, the interframe spaces and packet headers are significantly higher in AutoMAC due to the smaller packet size. Overall, the time spent in overheads is comparable across the 3 schemes.

- Due to AutoMAC's efficient MAC protocol, the fraction of time spent in sending useful data is 20% higher as compared to the conventional MAC protocols.

Keep in mind that beyond these gains, the protocol also gains from being able to exploit the uplink interference in the PHY layer.

6.2 Fairness

Fig. 10 plots the node-wise throughputs of AutoMAC and traditional Wifi normalized by the node-wise throughputs of the time fair Wifi scheme. Note that AutoMAC achieves atleast as much throughput for all the individual nodes as a Wifi scheme with time based fairness. Conventional Wifi is worse for certain nodes since the MAC cannot perfectly schedule all the nodes to ensure fairness. As the figure shows, AutoMAC provides fairness and in fact higher throughput due to factors discussed earlier.

7. RELATED WORK

AutoMAC is related to prior work in rateless codes and hybrid ARQ. Rateless codes such as LT [14] and Raptor codes [19] allow one to automatically achieve the capacity of an erasure channel without knowing the packet loss probability in advance. However, these techniques require whatever packets are received to be correctly decoded, and do not work in wireless channels where packets are corrupted. AutoMAC builds on recent work on rateless coding for Gaussian channels [6, 15]. AutoMAC's key contribution is to design a MAC protocol that exploits these rateless codes to systematically encourage and take adavntage of interference.

AutoMAC is related to prior work on interference cancellation [9, 4, 11, 12, 8]. However, all prior techniques require that the colliding packets be encoded at the correct bitrate to enable them to decode collisions. For example in SIC, if the colliding packets have been encoded at a bitrate corresponding to the idle channel (which will happen because the colliding hidden terminal senders cannot know in advance that they will collide), SIC will fail to work [4, 18]. Zigzag has a similar but less acute problem, since it also needs correct decoding of its interference free chunks, which requires the packets to be encoded at the correct bitrate. Further Zigzag needs the same set of packets to collide across successive collisions. AutoMAC does not have any of these problems, since its rateless property automatically adjusts the effective bitrate to enable it to decode collisions, and it can decode even if collisions are between different sets of packets.

8. CONCLUSION

AutoMAC provides a rateless MAC design that systematically exploits interference and consistently achieves very good performance across both uplink and downlink scenarios. AutoMAC suggests a number of avenues for future work including extending it to 802.11n MIMO scenarios as well as OFDMA.

9. ACKNOWLEDGEMENT

The first author was supported by Stanford Graduate Fellowship during the course of this research. The authors acknowledge the financial support from NSF POMI grant. We are also indebted to the reviewers of this paper for their detailed feedback and to our colleagues at Stanford University whose inputs have been invaluable to this paper.

10. REFERENCES

[1] T. Cover and J. Thomas. *Elements of Information Theory.* Wiley, 2006.

[2] G. Czink, B. Bandemer, and A. Paulraj. Stanford July 2008 radio channel measurement campaign. In *COST 2010*, October 2008.

[3] Free Software Foundation. Gnuradio. http://gnuradio.org.

[4] S. Gollakota and D. Katabi. ZigZag decoding: combating hidden terminals in wireless networks. In *SIGCOMM '08: Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, pages 159–170, New York, NY, USA, 2008. ACM.

[5] A. Gudipati and S. Katti. Automatic rate adaptation. In *ACM HotNets*, 2010.

[6] A. Gudipati and S. Katti. Strider: Automatic rate adaptation and collision handling. In *ACM SIGCOMM*, 2011.

[7] R. Gummadi, R. Patra, H. Balakrishnan, and E. Brewer. Interference Avoidance and Control. In *7th ACM Workshop on Hot Topics in Networks (Hotnets-VII)*, Calgary, Canada, October 2008.

[8] D. Halperin, J. Ammer, T. Anderson, and D. Wetherall. Interference cancellation: Better receivers for a new wireless mac. *Hotnets*, 2007.

[9] D. Halperin, T. Anderson, and D. Wetherall. Taking the sting out of carrier sense: interference cancellation for wireless lans. In *MobiCom '08: Proceedings of the 14th ACM international conference on Mobile computing and networking*, pages 339–350, New York, NY, USA, 2008. ACM.

[10] D. Halperin, A. Sheth, W. Hu, and D. Wetherall. Predictable 802.11 packet delivery from wireless channel measurements. In *ACM SIGCOMM*, 2010.

[11] S. Katti, S. Gollakota, and D. Katabi. Embracing wireless interference: analog network coding. In *SIGCOMM '07: Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 397–408, New York, NY, USA, 2007. ACM.

[12] L. Li, K. Tan, Y. Xu, H. Vishwanathan, and Y. Yang. Remap decoding: Simple retransmission permutation can resolve channel collisions. In *ACM MOBICOM*, Sept. 2010.

[13] S. K. Lin and D. Katabi. Random access heterogenous mimo networks. In *ACM SIGCOMM*, 2011.

[14] M. Luby. Lt codes. In *Proc. of FOCS 2002*, 2002.

[15] D. J. Perry and H. Balakrishnan. Rateless spinal codes. In *ACM HotNets*, 2011.

[16] T. Schmidl and D. Cox. Robust frequency and timing synchronization for ofdm. *IEEE Transactions on Communications*, Dec. 1997.

[17] R. S. Sen and S. Nelakuditi. No time to countdown: Backing off in the frequency domain. In *ACM MobiCom*, 2011.

[18] S. Sen, N. Santhapuri, R. R. Choudhury, and S. Nelakuditi. Successive interference cancellation: a back-of-the-envelope perspective. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, Hotnets-IX, pages 17:1–17:6, New York, NY, USA, 2010. ACM.

[19] A. Shokrollahi. Raptor codes. *IEEE/ACM Trans. Netw.*, 14(SI):2551–2567, 2006.

[20] G. Tan and J. Guttag. Time-based fairness improves performance in multi-rate wlans. In *Usenix Annual Technical Conference*, 2004.

[21] D. Tse and P. Vishwanath. *Fundamentals of Wireless Communications.* Cambridge University Press, 2005.

[22] M. Vutukuru, H. Balakrishnan, and K. Jamieson. Cross-layer wireless bit rate adaptation. In *ACM SIGCOMM, Barcelona, Spain*, August 2009.

[23] M. Vutukuru, K. Jamieson, and H. Balakrishnan. Harnessing Exposed Terminals in Wireless Networks. In *5th USENIX Symposium on Networked Systems Design and Implementation*, San Francisco, CA, April 2008.