Compressive Sensing Medium Access Control for Wireless LANs

Tsung-Han Lin and H. T. Kung
School of Engineering and Applied Sciences
Harvard University
Email: {thlin, htk}@eecs.harvard.edu

***Abstract*—We propose a medium access control (MAC) protocol for wireless local area networks (LANs) that leverages the theory of compressive sensing. The proposed compressive sensing MAC (CS-MAC) exploits the sparse property that, at a given time, only a few hosts are expected to request for radio channel access. Under CS-MAC, a central coordinator, such as a wireless access point (AP) can recover a multitude of these requests in one decoding operation, and then schedule multiple hosts accordingly. The coordinator is only required to receive a relatively small number of random projections of host requests, rather than polling individual hosts. This results in an efficient request-grant method. Via a hardware prototype based on a software-defined radio platform, we demonstrate the feasibility of realizing CS-MAC with compressive measurements formed in the air to achieve high efficiency.**

I. Introduction

Compressive sensing is an emerging technology that has drawn considerable attention recently for its ability to acquire and extract critical information efficiently. It has found applications in various fields such as medical imaging, cognitive radio, wireless communication, and sensor networks (see, e.g., [9]). In particular, two features of compressive sensing are worth noting. First, generating compressive measurements is blind to the content of the signal to be compressed and has low computational complexity. Second, it is sufficient to capture the signal with a small number of compressive measurements, which is approximately proportional to its information content, i.e., its sparsity, not its length. Therefore, compressive sensing is attractive in large-scale distributed scenarios where coordination is substantial, and important information is sparse.

Wireless medium access control concerns scheduling of radio channels shared among a network of distributed hosts. The state-of-the-art 802.11 wireless LAN standard adopts a CSMA/CA random access method, whereby contention is resolved by a randomized backoff counter on every host station. The idle channel is won by the host whose backoff counter expires first. This method works well when there are relatively few hosts in the network; however it has limitations due to its distributed control paradigm. For example, collision avoidance relies on local carrier sensing. When carrier sensing cannot function properly, e.g., in the presence of hidden terminals, throughput will greatly degrade (see, e.g., [7]). Quality-of-service (QoS) policies are also difficult to be implemented due to the lack of a central coordination.

While central coordination can potentially provide better scheduling, its efficient implementation however has been a challenge. For example, polling-based MAC protocols [15][1] may suffer from the high communication overhead, proportional to the number of hosts N in the network, or require complex scheduling to improve their efficiency. 802.11 PCF [1] is one example: it is rarely deployed in practice due to its overhead. The inefficiency of polling in part results from the fact that the central coordinator may poll a host that does not have data to send. It is wasteful because in a large network, only a few hosts may be contending for the channel at any given time. This suggests the use of a sparse vector to represent the hosts in the network where the contending hosts are the nonzero components. AP's polling operations can be viewed as constructing the sparse vector by checking its components one by one. In this paper, we propose CS-MAC, a compressive sensing based MAC protocol that allows a coordinator to check all components of the sparse vector at once by receiving only a small number of messages by the hosts.

CS-MAC can realize efficient centralized scheduling for three reasons. 1) Compressive sensing allows CS-MAC to identify contending hosts by receiving only a few compressive measurements with the number of measurements approximately proportional to the number of contending hosts. The communication overhead on the central coordinator thus can be minimized. 2) The compressive measurements of host channel access requests are formed in the air from concurrent transmissions. This analog approach eliminates the need of scheduling request transmissions, and thus enables fast measurement collection. 3) CS-MAC uses a distributed random access protocol to limit the number of contending hosts at a given time when many hosts have data to send. Consequently, the number of required compressive measurements does not need to be adaptive to the network contention level and can be set to a fixed constant. In addition, CS-MAC can scale with the number of hosts. The overhead of CS-MAC grows sublinearly with N, assuming fast decoding algorithms [14] and parallel processing hardware [17] are used.

We summarize the main contributions of this paper as follows: 1) We use compressive sensing to implement a compressive requests/multiple grants MAC for wireless LANs. To our knowledge, we are the first in the literature to design a complete MAC solution based on compressive sensing. 2) We show an analog radio implementation incorporating a suite of low-overhead methods to address key issues such

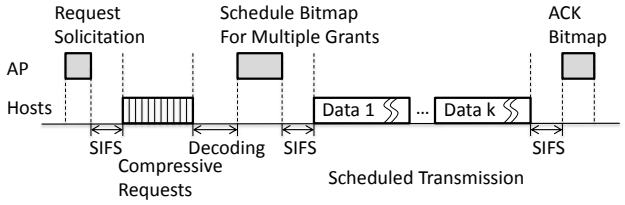

Fig. 1: Overview of CS-MAC operations.

as analog compressive measurements formation and easy-to-implement synchronization. We demonstrate the practicality of host requests recovery on a hardware prototype. 3) We show through software simulation that CS-MAC can offer better performance in both throughput and fairness over two state-of-the-art protocols, 802.11 DCF and Idle Sense [10].

This paper is organized as follows: we first briefly introduce compressive sensing in Section II. We then present the design of CS-MAC in Section III. The implementation of the software-defined radio prototype of CS-MAC and experiments is detailed in Section IV, followed by performance simulation in Section V, and finally conclusions in Section VI.

II. Compressive Sensing

In this section, we provide a brief introduction to compressive sensing. Interested readers are referred to, e.g., [8], for an extensive treatment of the subject. Compressive sensing originates from the observation that natural signals are often *sparse* in some domain, i.e., a signal can be represented by a few significant coefficients, such as natural images in the frequency domain. While traditional compression can exploit this sparsity to reduce signal size, it requires statistical analysis of the entire signal to derive the significant coefficients. Compressive sensing instead attempts to remove this requirement considering an interesting question: *given a sparse signal of sparsity K, i.e., it has only K significant coefficients, can one capture the K coefficients without collecting and analyzing the complete data of some large size N?* Equivalently, given an $M \times N$ sensing matrix $\mathbf{A}$, and the measurement vector $\mathbf{y} = \mathbf{A}\mathbf{x}$, can we recover the signal vector $\mathbf{x}$ with $M < N$? Since this linear system is underdetermined, the problem in general has infinite solutions for $\mathbf{x}$. However, it has been shown that exact recovery is possible with M as small as $O(K \log \frac{N}{K})$, provided that $\mathbf{A}$ satisfies some regulatory conditions such as the restricted isometry property (RIP). Gaussian and Bernoulli random matrices are found to be good candidates for $\mathbf{A}$ that satisfy RIP with high probability. In this case, the measurements can be viewed as *random projections* of the signal to be compressed. Due to their expected high incoherency, the random projections effectively capture all useful information about the signal with high probability when there are enough of them. One way to recover the sparse signal $\mathbf{x}$ is to choose the solution vector with minimum ℓ_1-norm over all feasible solutions. That is, find $\mathbf{x}$ by solving the following minimization problem.

$$\min_{\mathbf{x} \in \mathbb{R}^N} |\mathbf{x}|_{\ell_1} \quad \text{subject to} \quad \mathbf{y} = \mathbf{A}\mathbf{x} \tag{1}$$

and (1) can be solved via linear programming.

In short, compressive sensing enables efficient data collection with a small number of measurements, which is approximately a small constant multiple of the sparsity K when $\log \frac{N}{K}$ is small. CS-MAC exploits this idea for efficient collection of host channel access requests.

III. CS-MAC Design

In contrast to 802.11 DCF, CS-MAC takes a centralized approach to schedule radio channel access. In other words, a central coordinator is used to first learn distributed hosts' needs for channel access and then schedule transmission slots accordingly. For clarity, in this paper we assume a single wireless access point (AP) serving as the central coordinator, and all hosts in the wireless LAN are associated with the AP.

CS-MAC has two key ideas, namely the *compressive requests* and *multiple grants*. In CS-MAC, to acquire transmission opportunities, distributed hosts send channel access requests *concurrently*; thus multiple requests are combined in the air, which we call the compressive requests. The AP can then use the compressive requests to identify the contending hosts and grant transmission opportunities. As noted in Section II, the number of measurements required for recovery is approximately proportional to the number of hosts requesting for channel access at the moment, rather than the total number of hosts in the network. CS-MAC thus can scale to networks with a large number of hosts.

Although the AP can efficiently learn the hosts' needs for channel access through compressive requests, given a fixed number of measurements, only a limited number of K hosts can request concurrently due to the sparsity constraint. To control the number of concurrent requesting hosts, CS-MAC uses a randomized scheme under which the hosts send requests with some probability. Noting that the AP can resolve up to K requesting hosts at a time and grant transmission opportunities to multiple hosts (called multiple grants). This contention thus is a *multi-winner* contention. We will show that in multi-winner contention, collisions are much less likely to occur when K is sufficiently large, and thus the efficiency of CS-MAC is not hampered by the randomized scheme.

The basic operation of CS-MAC is outlined in Figure 1. CS-MAC begins with the AP initiating a request solicitation. Upon receiving the solicitation, hosts wish to access the channel may reply with requests, depending on the outcome of a local coin toss. The concurrent request transmissions then are combined in the air forming compressive requests. The AP next decodes the received compressive requests to identify contending hosts and schedules data transmissions accordingly. The schedule is then broadcast using a schedule bitmap packet. Finally, after the scheduled hosts finish transmitting data packets, the AP broadcasts an acknowledgement bitmap packet to acknowledge received packets.

A. Analog compressive requests: random linear combining in the air

As noted earlier, random projections of a sparse vector can preserve sufficient information with high probability for

recovery. CS-MAC generates such random projections via concurrently transmitting random sequences as the requests from multiple hosts. The concurrent transmissions can be formulated as:

$$\mathbf{y} = \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & ... & \mathbf{a}_N \end{bmatrix} \begin{bmatrix} h_1 x_1 \\ h_2 x_2 \\ \vdots \\ h_N x_N \end{bmatrix} + \mathbf{n} \qquad (2)$$

where $\mathbf{a}_i$ is the random sequence of host i and h_i is the channel coefficient from host i to the AP. x_i is a binary {0,1} variable indicating whether host i sends its request and $\mathbf{n}$ denotes the noise vector. Assuming that $\mathbf{a}_i$ has length M and the channel is coherent over the transmission period, $\mathbf{y}$ is the received signal of length M at the AP.

For simplicity, the random sequence is generated from Bernoulli distribution of {1,-1}. Assuming there are at most K hosts requesting at any given time (we will justify this assumption later in Section III-B), (2) can be viewed as a sparse recovery problem that has only K nonzero $h_i x_i$ in the unknown vector. Then M can be as small as $cK \ll N$ for exact recovery where c is a small constant. Empirically the value of c around 3 to 4 is sufficient for recovery provided that $\mathbf{n}$ is relatively small. Note that in identifying requesting hosts, the compressive sensing decoding process yields the solution of $h_i x_i$ without having to estimate the channel state information. Since $h_i x_i$ is 0 when host i does not request for channel access, we can use a threshold to distinguish between zero and nonzero x_i.

Finally, we note that the random sequence is assigned to each host by the AP during association. Therefore the AP knows the random sequence associated with each host and thus can solve (2) by using a proper sensing matrix in decoding. In addition, unlike many other centralized approaches, CS-MAC requires no sophisticated membership management at the AP. If a host leaves without notice, it is equivalent to the host not requesting for channel access. Since compressive sensing almost only concerns the number of nonzero components, a small increase in the total number of unknowns N resulting from loose membership management will almost not change the required number of measurements M.

B. Multi-winner contention for multiple grants

For exact recovery, the number of measurements M needs to be set based on the sparsity K, the number of requesting hosts. In the worst case where the network is very busy, K can be as high as N when all hosts need channel access. Therefore, without an adaptation scheme, a fixed large M would be required to support a potentially large K, resulting in inefficiency when in reality K is expected to be small. Furthermore, a larger M would not be practical if the longer measurement period exceeds the channel coherence time.

We propose to use a distributed control scheme to limit the sparsity K that CS-MAC needs to handle at any time, allowing CS-MAC to use a small fixed M. The basic idea is to use a random access protocol which stipulates the hosts to send requests with some probability p. p can be adjusted based on the network contention level: if collisions occur when more than K hosts send the requests, p will be reduced to avoid future collisions; otherwise p is increased for the hosts to take advantage of the unsaturated channel. The classic additive-increase-multiplicative-decrease (AIMD) principle can be employed to adjust p to ensure fairness among hosts. Note that the collisions can be detected by the AP when the decoding of compressive requests fails. The AP then can notify the hosts of the collision using the schedule bitmap so that they can adjust the requesting probability accordingly. We set the AIMD parameters similar to those in Idle Sense [10]: the probability is increased by 0.001 and decreased by $\frac{1}{1.2}$ for every adjustment.

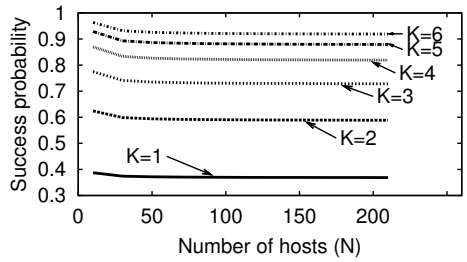

Fig. 2: Advantages of multi-winner contention. The probability of successful resolution of contention increases dramatically when K grows from 1 to 5.

It is possible that such a random access scheme suffers from excessive collisions and delivers low MAC efficiency. For example, slotted ALOHA only achieves at best approximately 36% efficiency due to collisions. CS-MAC, however, does not suffer from the issue because multiple requesting hosts can be resolved for each compressive requests. To see this, we can derive the probability P_s that the requesting hosts are identified successfully as:

$$P_s = \sum_{i=1}^{K} \binom{N}{i} p^i (1-p)^{(N-i)} \qquad (3)$$

Figure 2 shows the maximal P_s under different sparsity limit K and different number of hosts N. We can make the following observations. First, P_s remains almost constant under a varying number of hosts. Second, when $K = 1$ (corresponding to slotted ALOHA), the maximal P_s is only 0.36 as expected. However, when $K = 5$ the maximal P_s is dramatically increased to 0.9. This suggests that collisions are a lot less likely to occur when more than one winner is chosen in a contention. Also, a small number of winners is sufficient to mitigate the inefficiency significantly, and thus K needs not to be large. This can assure that the transmission time of compressive requests will be under channel coherence time. In our CS-MAC implementation, we set M=20 for K=5.

C. Synchronizing concurrent transmissions

To form compressive measurements from concurrent transmissions, all hosts need to be synchronized to the symbol-level. Fine-grained synchronization between hosts in such

a distributed setting is generally a difficult task. CS-MAC instead only requires loose synchronization between hosts. It uses the request solicitation message as a reference signal. Assume a 300m radio range. Given this propagation delay, timing misalignments between transmitted symbols will be capped at $2\mu s$ [16].

To overcome the $2\mu s$ misalignment, we use a long symbol length of $5.12\mu s$ duration to ensure that the transmitted symbols always overlap, and the AP can safely take compressive measurements. Note that although the symbol is much longer than that in a perfectly synchronized scenario (e.g., in this case each symbol can be only 50ns long with a 20MHz bandwidth), the incurred overhead is still small due to the small number of symbols required for compressive requests. Given that CS-MAC only needs $M = 20$ symbols for compressive requests, the compressive requests span approximately $100\mu s$ duration. Since $K = 5$ hosts can be resolved and scheduled for data transmission after the compressive requests, CS-MAC on average only adds $20\mu s$ overhead to each data transmission, which corresponds to only two backoff slots in 802.11. Furthermore, the $100\mu s$ duration of the compressive requests is well below the 10-20ms channel coherence time, and thus (2) still holds.

D. Protocol overhead

Table I lists the parameter values of CS-MAC. Detailed descriptions of individual control packets are omitted due to space limitation. Compressive sensing decoding may incur a significant overhead. For example, when linear programming is used to perform ℓ_1-norm minimization, the computational complexity is roughly $O(N^3)$ or higher. Reducing the decoding complexity has been a subject of intensive research in recent years [6]. Currently the best state-of-the-art algorithm can lower the decoding complexity down to $O(N \log \frac{N}{K})$ [5] at the expense of a relatively weak error guarantee for the recovered solution. Here we use $O(N^2)$ to approximate decoding complexity. This means the decoding time will be about $20\mu s$ with a 2GHz CPU when N=200.

Based on Table I, and assuming the underlying physical layer runs 802.11g (54Mbps), the overhead for CS-MAC to send a data packet is $67.1\mu s$ excluding the overhead of random backoff and collisions in multi-winner contention. For 802.11 DCF, the overhead is $80.1\mu s$ without including the cost of random backoff and collisions. If the RTS-CTS mechanism is turned on, the overhead further goes up to $145.1\mu s$. We can see that while CS-MAC is a centralized approach for medium access control, its overhead is still comparable to basic 802.11 DCF, which only permits distributed random access.

TABLE I: Parameter values of CS-MAC

Request solicitation	14 bytes	Decoding	20 μs
Compressive request	102.4 μs	Schedule bitmap	37 bytes
ACK bitmap	37 bytes	PHY header	20 μs

E. Performance gains of CS-MAC and system considerations

CS-MAC is a centralized MAC protocol, and thus has important gains over conventional CSMA-based protocols that are distributed in nature.

Hidden terminals. The classic hidden terminal problem arises when two hosts associated with the same AP cannot hear each other. As a result, they cannot detect ongoing transmissions and will interfere with each other. Throughput drops significantly when hidden terminals are present [7]. Current 802.11 protocol adopts the RTS-CTS exchange to avoid collisions, however its overhead is so high that RTS-CTS is often turned off. The hidden terminal problem arises because of a lack of global information at each host. Thus it is naturally solvable using a central scheduling approach such as CS-MAC. The scheduler guarantees a dedicated time slot for a host to send packets without interference. In an environment with multiple APs, there could be more complex hidden terminal scenarios where RTS-CTS cannot even function correctly [11], or exposed terminal scenarios which cause channel underutilization [7]. These problems can be solved by running CS-MAC on a central scheduler coordinating among the APs.

Short-term fairness. It is well-known that the binary exponential backoff scheme in 802.11 DCF delivers poor short-term fairness [10]. In general, to achieve good short-term fairness, one needs to estimate the network traffic load over small time intervals to prevent a single host from taking an unproportionally large portion of the channel. In a basic scenario where every host can hear each other, Idle Sense [10] enforces the fairness by maintaining equal transmission probability at every host in a distributed fashion. The network load is estimated by observing the number of idle slots between transmissions. However, this estimation becomes difficult when hidden terminals exist. In this case, one may need to introduce additional control mechanisms to propagate load information [11]. CS-MAC takes a similar approach as Idle Sense that the hosts request for the channel with some probability. The network load then can be estimated by observing the number of requesting hosts, and the AP can simply use a broadcast to regulate host request probability. As a result, CS-MAC can achieve good short-term fairness regardless of hidden terminals in the sense of 802.11 DCF.

QoS. Centralized approaches can ease the implementation of quality of service (QoS) policies. For example, DOCSIS [4], the protocol for cable Internet access, is known for its ability to perform QoS scheduling for multimedia applications. We expect CS-MAC to deliver similar QoS capability since it takes a similar request/grant method. Take EDCF [3] in 802.11e as an example: EDCF provides differentiated service support by prioritizing flows in the network through adjustments to the contention window size for each flow. However, configuring priority-level parameters to achieve QoS is not obvious. In [18], proportional differentiation is proposed such that the ratio of channel sharing between different priority levels are to be set. To implement this policy, CS-MAC simply grants more transmission slots to higher priority flows.

Coexisting with 802.11. CS-MAC may coexist with 802.11. When conventional 802.11 DCF hosts join the network, they

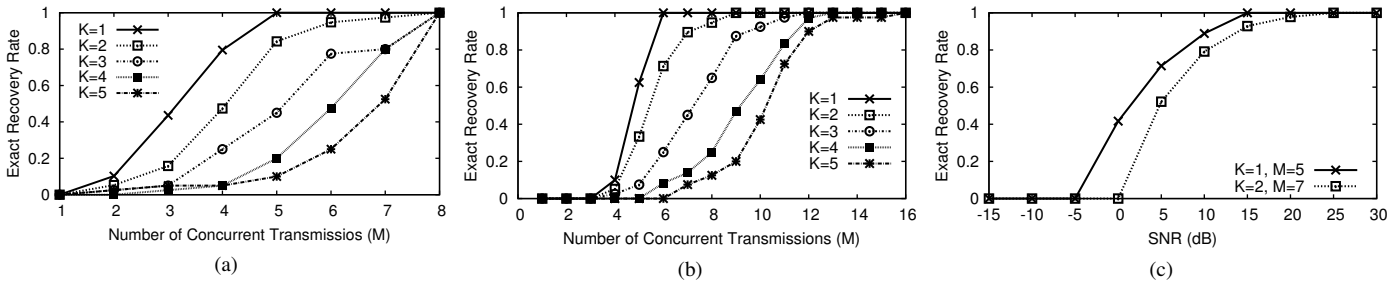

Fig. 3: Experiment results on hardware prototype. (a) Recovery performance for compressive requests in the 8-node scenario, and that in (b) the 16-node scenario. (c) Recovery performance under different SNR in the 8-node scenario.

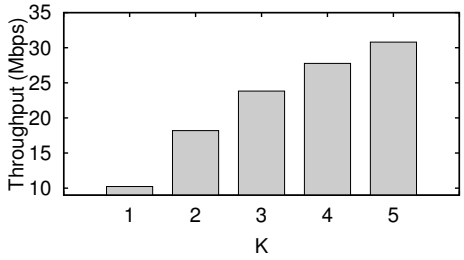

Fig. 4: Impact of multi-winner contention. The aggregated throughput of CS-MAC in a 40-node scenario can reach 30Mbps when K increases from 1 to 5.

will not be granted any channel access under CS-MAC as they will not participate in CS-MAC requesting. We can solve this problem by having the CS-MAC AP also perform CSMA before sending the request solicitation under CS-MAC. In other words, the AP contends with 802.11 hosts for the channel, and thus gives room for 802.11 hosts.

IV. Compressive sensing recovery on hardware prototype

To demonstrate the practicality of analog compressive requests combined over air and their recovery by compressive sensing decoding, we implement these functions on software-defined radios. Our testbed has 9 USRP-N200 nodes distributed in an area of 2m x 2m, equipped with the WBX daughterboards. The nodes operate with a 0.78MHz bandwidth center at 916MHz. For fast prototyping, we calibrated the nodes before the experiments to ensure that there is no frequency offset. In practice, the AP's frequency can be used as a reference and the calibration can be done after receiving the request solicitation.

Among the 9 nodes, one node serves as the AP and the other 8 nodes serve as the hosts. Being near the AP, the hosts have a 20-30dB SNR. We randomly pick K hosts to request for channel access, and see if we can use compressive sensing decoding to recover the requests from the received signals. We use the `l1-MAGIC` package [2] to perform sparse recovery. The detection threshold for requests is set to 10dB to avoid false positive detection. Optimal threshold setting involves error and false positive/negative analysis, and needs further study.

Figure 3(a) shows the results of the experiment. When K=1, we can recover the requests exactly with M=5 concurrent transmissions or more. This is consistent with the $M{\sim}4K$ estimate on the required measurements stated earlier. For $2{<}K{\leq}5$, we need 8 measurements for 100% recovery rate, but when fewer measurements are used, as M increases, we still can observe the increase in recovery rate. Next, we want to see how the recovery performs in a larger network setting. Due to limited hardware availability, we use the 8 USRP-N200 nodes to emulate a 16-node scenario. A single physical node will act as two different virtual nodes. The radio signals transmitted by the two virtual nodes are assumed to be perfectly combined without any noise. The virtually combined signal is then transmitted by the physical node. The results are shown in Figure 3(b). When K=2 and 3, requests are always recovered with 9 and 12 measurements, respectively.

To check the performance under varying SNRs, we vary the transmission power in the 8-node scenario. Using K=1, we set M=5, the minimum measurements required for exact recovery as observed in Figure 3(a). Figure 3(c) shows the results. Compressive requests achieves good performance when SNR is higher than 15dB, a reasonable SNR requirement in wireless LANs. Similar performance is observed when K=2, M=7. Note that this parameter setting cannot guarantee 100% recovery as indicated by Figure 3(a), and thus shows a slightly worse performance.

V. Performance Simulation

To test a larger set of conditions, we implement CS-MAC with an event-driven software simulator. In the simulations, we assume the physical layer runs 802.11g (54Mbps), and all hosts always have data to send. We first show the impact of multi-winner contention in a 40-host scenario. Figure 4 shows the aggregated network throughput of CS-MAC with different K. The aggregated throughput of CS-MAC when K=1 is only 10Mbps due to the high collision probability. In contrast, the throughput is increased to 30Mbps by setting K=5 when collisions become less likely to occur. For all other simulations, K is set to 5 as described in Section III-B.

Next we compare CS-MAC with Idle Sense [10] and 802.11 DCF in scalability to number of hosts, fairness, and QoS.

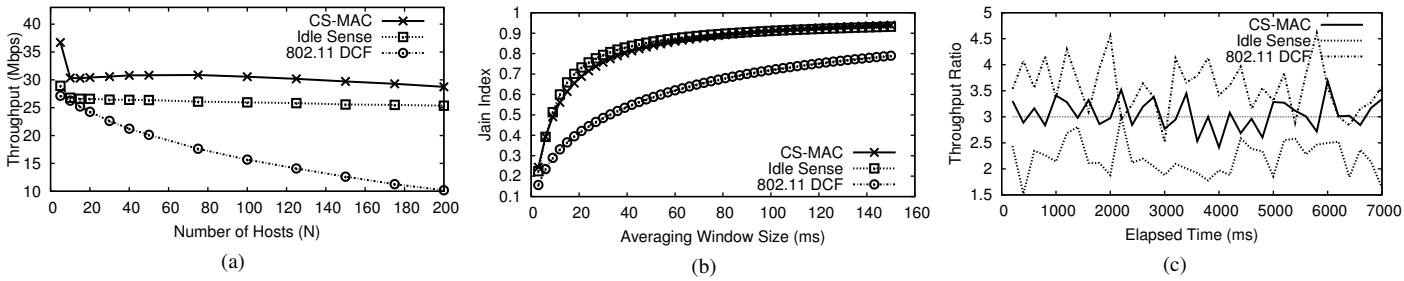

Fig. 5: Software simulation results. (a) CS-MAC aggregated throughput with a varying number of hosts. (b) Short-term fairness [12]. (c) Proportional differentiated service for QoS.

Figure 5(a) shows the aggregated throughput of CS-MAC with a varying number of hosts. CS-MAC can scale to a network with 200 hosts without losing much of its efficiency, while 802.11 DCF loses its efficiency due to the increased number of collisions. Figure 5(b) shows the short-term fairness when N=40. CS-MAC achieves similar performance with Idle Sense. We note that when hidden terminals are introduced in the simulations, the throughput of both Idle Sense and 802.11 DCF drops significantly. If we enable RTS-CTS, they still suffer from poor short-term fairness due to the lack of correct network load estimation. (Performance plots of are not shown here due to space limitation.)

Lastly, Figure 5(c) shows the performance results of a simple implementation of a proportional differentiated service QoS policy. We want to have a particular host in the 10-host collection to have throughput three times higher than the rest. For 802.11 DCF and Idle Sense, due to the lack of central coordination, we implement the policy by setting the contention window of the host to be three times smaller than other; however this implementation fails to achieve the correct ratio for both of the protocols. In contrast, CS-MAC can realize the ratio easily by AP scheduling.

VI. Conclusion

CS-MAC of this paper uses a central controller to schedule hosts. It is easy to see that a centralized approach can conveniently avoid hidden terminal problems, assure QoS and enhance fairness. However, it can be difficult to devise an efficient implementation for a centralized scheme due to the need of gathering global information from all hosts. In this paper, we note that compressive sensing can change the equation. Since host requests are expected to be sparse, they can now be recovered with far fewer measurements than before. This can fundamentally lead to a shift towards centralized MAC approaches for wireless LANs.

While in this paper we have developed the basic concepts of CS-MAC and demonstrated its working under lab settings, we recognize that much further work is needed. In particular, a better understanding of the robustness of CS-MAC for networks with larger propagation delays and larger variation in host SNR is necessary. We also need to develop models for optimal choices of K and M for the contention period and for optimal design of the measurement matrix of (2) to accommodate more users. Finally, we need to devise advanced synchronization mechanisms to allow shortened symbol time for the contention period. See MIMO/CON on using compressive sensing to weaken the synchronization requirement [13].

Acknowledgement

This material is in part based on research sponsored by Air Force Research Laboratory under agreement number FA8750-10-2-0180. The U.S. Government is authorized to reproduce and distributed reprints for Governmental purposes not withstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies of endorsements, either expressed or implied, of Air Force Research Laboratory or the U.S. Government.

References

[1] IEEE standard 802.11-2007.
[2] l1-MAGIC. Available at www.l1-magic.org.
[3] IEEE 802.11e/D5.0. 2003.
[4] Cable television laboratories, inc. data-over-cable service interface specifications, in radio frequency interface specification. 2004.
[5] R. Berinde, P. Indyk, and M. Ruzic. Practical near-optimal sparse recovery in the l1 norm. In *Allerton 2008*.
[6] R. Berinde et al. Combining geometry and combinatorics: A unified approach to sparse signal recovery. In *Allerton08*.
[7] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang. MACAW: a media access protocol for wireless LANs. In *SIGCOMM 1994*.
[8] E. Candès. Compressive sampling. In *Proceedings of the International Congress of Mathematicians*, volume 3, pages 1433–1452, 2006.
[9] M. Davenport, M. Duarte, Y. Eldar, and G. Kutyniok. Introduction to compressed sensing. *Electrical Engineering*, pages 1–68, 2011.
[10] M. Heusse. Idle sense: an optimal access method for high throughput and fairness in rate diverse wireless LANs. In *SIGCOMM 2005*.
[11] Y. Jian and S. Chen. Can CSMA/CA networks be made fair? In *MobiCom 2008*.
[12] C. Koksal, H. Kassab, and H. Balakrishnan. An analysis of short-term fairness in wireless media access protocols. In *SIGMETRICS 2000*.
[13] T.-H. Lin and H. T. Kung. Concurrent channel access and estimation for scalable multiuser MIMO networking. *Harvard Univ.*, [online] 2012, http://nrs.harvard.edu/urn-3:HUL.InstRepos:9299797.
[14] D. Needell and J. Tropp. Cosamp: Iterative signal recovery from incomplete and inaccurate samples. *Applied and Computational Harmonic Analysis*, 26(3):301–321, 2009.
[15] O. Sharon and E. Altman. An efficient polling MAC for wireless LANs. *IEEE/ACM TON*, 9(4):439–451, 2001.
[16] K. Tan, J. Fang, Y. Zhang, S. Chen, L. Shi, J. Zhang, and Y. Zhang. Fine-grained channel access in wireless LAN. In *SIGCOMM 2010*.
[17] S. Tarsa et al. Performance gains in conjugate gradient computation with linearly connected gpu multiprocessors. In *USENIX HotPar'12*.
[18] Q. Xue and A. Ganz. Proportional service differentiation in wireless LANs using spacing-based channel occupancy regulation. In *MM 2004*.