Weeble: Enabling Low-Power Nodes to Coexist with High-Power Nodes in White Space Networks

Božidar Radunović
Microsoft Research
Cambridge
bozidar@microsoft.com

Ranveer Chandra
Microsoft Research
Redmond
ranveer@microsoft.com

Dinan Gunawardena
Microsoft Research
Cambridge
dinang@microsoft.com

ABSTRACT

One of the key distinctive requirements of white-space networks is the power asymmetry. Static nodes are allowed to transmit with 15dB-20dB higher power than mobile nodes. This poses significant coexistence problems, as high-power nodes can easily starve low-power nodes. In this paper, we propose Weeble, a novel distributed and state-less MAC protocol that solves the coexistence problem. One of the key building blocks is an adaptive preamble support, an add-on to the PHY layer that allows high-power nodes to detect a low-power transmission even when the difference in transmit power is as high as 20dB. The other key building block is a MAC protocol that exploits the adaptive preambles functionality. It implements a virtual carrier-sensing and automatically adapts the preamble size to optimize network performance. We extensively evaluate our system in a test-bed and in simulations. We show that we can prevent starvation of low-power nodes in almost all existing scenarios and improve the data rates of low-power links several-fold over existing MACs, and as a trade-off we decrease the throughput of the rest of the system by 20%-40%.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Wireless communication

General Terms

Design

Keywords

Coexistence, White-spaces

1. INTRODUCTION

There is recent interest in using the TV white spaces (unoccupied TV channels) for unlicensed communication. The FCC issued an official approval for the US [6]; the UK [28], Canada [2], Brazil and Singapore have made significant progress towards similar decisions. The industry is moving quickly with the development of the white space database [7] and hardware [34]. The IEEE is even developing standards for different applications over this spectrum.

This excitement stems primarily from the excellent propagation characteristics of the TV spectrum – it not only extends the reach of a transmission, it also enables faster transmissions at short distances because of higher SNR. Both these benefits lead to different applications. The former is useful in regional area networks (WRANs) as is enabled by the IEEE 802.22 standard [16]. The latter is useful for in-home media distribution applications, which have been proposed by Dell, Philips, and other companies, and will be enabled by the 802.11af standard [3]. This has led to an important question – can these low power (in-home) and high power (regional-area applications) coexist on the same spectrum?

The FCC has defined the operational parameters of the low and high power nodes. The *high power* nodes, such as base-stations or regional network back-haul nodes, can transmit with a power of up to 4W. Per the rules, these nodes have to be static and query the spectrum database [7] so that they do not create interference on the existing TV channels[1]. The *low power* nodes are limited to 40mW or 100mW of transmit power (depending whether they are adjacent to an active TV channel) and can be mobile. Note that both high and low power nodes are unregulated, except for the spectral mask and the requirements mentioned above. In principle, even consumer devices could use high-power transmissions in any available white-space channel to boost the quality of home network. However, this is unlikely to happen as it would increase the power consumption, heat dissipation, the form factor and the cost of the devices.

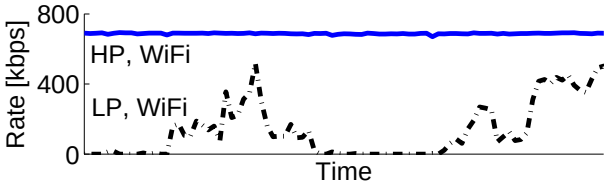


Figure 1: Starvation of a mobile low-power (LP) link in presence of high-power (HP) link.

This large power asymmetry poses a significant challenge to network design. A high-power node might not sense the transmission of a low-power node, and hence, can easily overwhelm the low-power node with interference and interrupt communication (hidden terminal). We observe the starvation in Figure 1, which shows the throughput of a low power and high power node in our white space test-bed (described in Section 5) when using the Wi-Fi MAC.

[1]The database is not used to allocate spectrum to (low or high power) white-space nodes, but only to prevent them from interfering with the incumbents.

CoNEXT'12, December 10–13, 2012, Nice, France.

The IEEE has recognized this problem as well and formed the IEEE 802.19 working group [14] that also addresses this coexistence problem.

A strawman approach to deal with the coexistence problem is to assign disjoint frequencies to high-power and low-power nodes (or frequency division multiplexing, FDM). However, determining the frequency separation is non-trivial because of varying white space availability, different population densities, and mobility of low-power nodes. Moreover, any such frequency assignment needs to be dynamic and global. This makes it extremely difficult to manage. Furthermore, as we show in Section 6, such a technique can also be very inefficient.

In this paper we present Weeble, a novel, fully distributed MAC design for coexistence among high-power and low-power nodes. The main idea behind Weeble is to repair the carrier sense mechanism in a power asymmetric setting and then build a CSMA-like distributed reservation protocol on top of it. We note that our problem is different from the problem of coexistence with legacy devices, i.e. those that cannot be modified [13, 8, 27, 12]. In this paper, our goal is to design a mechanism that can be adapted by both the high and low-power nodes.

A key component of designing Weeble is to define coexistence among low and high-power nodes. In this paper we define coexistence as avoiding starvation of low-power nodes due to high power interference and, more generally, to run the network efficiently and fairly. However, fairness and efficiency are fundamentally conflicting goals [30] and there is no commonly accepted definition of fairness in white-space networks. One can argue that high-power links (such as base-stations and back-haul links) are more important than the low-power links. Thus, our design goals are, in the following order: (a) to avoid starvation of any link, (b) avoid significant performance deterioration of high-power links and (c) increase the total throughput as much as possible.

Weeble achieves the above goals by leveraging two key innovations:

Technique for detecting transmissions at low SNR : First, we design an adaptive preamble detector that allows low power nodes to signal their presence to nodes that receive the signal at very low SNR (even lower than -15 dB). Our design uses two types of preambles to significantly decrease the false positives and it does not require any prior synchronization between the nodes (unlike [12, 9, 33]). By tunning the length of the preamble, one can control the signalling range and the preamble overhead. We build our preamble detector in FPGA as an add-on to an existing OFDM PHY design, and we also show that our implementation is much more efficient than conventional preamble detector designs.

MAC that allows low power and high power nodes to coexist: Second, we design a distributed reservation mechanism for low-power nodes that is based on adaptive preamble signalling (Section 4). In the absence of carrier sense it is not possible to detect the end of a packet transmission. Instead, our low-power reservations are of fixed duration. To make an efficient use of each reservation period, all low-power nodes are allowed to contend and transmit multiple packets when possible during the period. We balance the traffic between the high-power and low-power nodes by prioritizing the access of high-power nodes in between low-power reservation periods (Section 4.1). Finally, we propose an algorithm to adapt the preamble size to maximize the spatial reuse and limit the protocol overhead (Section 4.2).

We have implemented our system on the Lyrtech SDR platform. Using a set of micro-benchmarks we first demonstrate that our adaptive signalling works with 90% accuracy at SNRs below -15dB (Section 6.1). We then evaluate the full Weeble MAC design in a small scale test-bed on several topologies (Section 6.2). In contrast to the recent software-defined radio deployments [33, 36, 9, 8, 12] that do not implement MAC protocols but rely on offline processing of channel traces, we implement and run both PHY and MAC on the deployed nodes. We show that we avoid starvation of low-power flows and we achieve 50% median increase in rates of low-power flows over 802.11 MAC, at the expense of less than 6% median decrease of rates of high-power flows. We further evaluate Weeble on larger topologies using Qualnet simulations with PHY layer parameters as measured in the test-bed (Section 6). We observe up to 10-fold rate improvement of low-power flows over 802.11 MAC and FDM (frequency division multiplexing MAC), with an inevitable trade-off being a 20%-40% rate decrease of high-power flows.

2. WEEBLE OVERVIEW

Carrier sense is a simple and very efficient signaling primitive for sharing the medium in unlicensed wireless networks, such as Wi-Fi. Weeble attempts to provide a similar functionality in white space networks even though not all nodes are in carrier sensing range of each other, for example in networks where different nodes transmit at different transmission powers. To ensure that low power nodes get time to communicate without interference from high power nodes we enhance carrier sense with a new technique called *low-power reservations*.

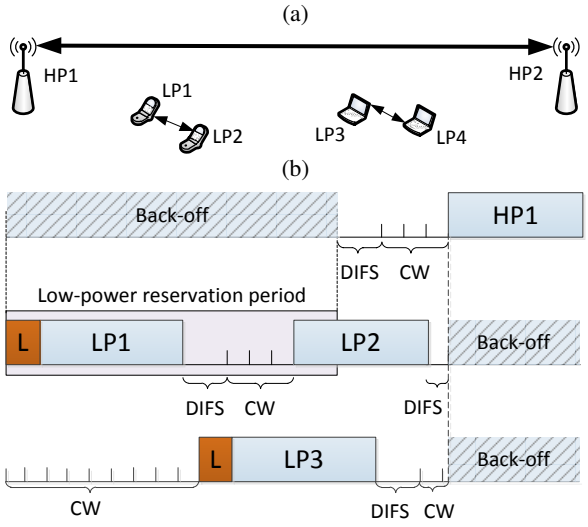


Figure 2: Illustration of the low-power reservation period (HP - high-power transmissions, LP - low-power transmissions, L - adaptive preamble): (a) network topology, (b) sample packet exchange

To signal a start of a low-power reservation to a high power node, we introduce a special preamble. Since this preamble might not always be needed (for example, when the interference does not affect packet reception), a low-power node may decide to prepend this special preamble to its packets only if it experiences interference. All nodes that detect the preamble will refrain from sending for the duration of the reservation (that will be defined later).

To ensure that low-power reservations do not block out a large region, we design a preamble of adaptive size. The longer the preamble, the lower the SNR is at which it can be detected. We choose the preamble size of a low-power link such that it can be detected only by those high-power nodes whose concurrent transmission can cause a packet loss on the respective low-power link. If the preamble is too short, this will lead to packet losses at the low-power

link. If it is too long, it will hamper spatial reuse. We describe the algorithm to determine the preamble length in Section 4.2.

For simplicity, we set the duration of a low-power reservation to be a constant. To efficiently use the reservation period, it is shared by all low-power nodes in the vicinity. The low-power nodes use the conventional carrier sense among themselves and the node that initializes the reservation period (by sending an adaptive preamble) transmits the first packet. It will be followed by other low-power nodes until the reservation expires. Once the reservation has expired, the high-power nodes will resume contending for the access among themselves and the other low-power nodes.

We illustrate the functionality of Weeble MAC in Figure 2. We consider a network depicted in Figure 2 (a) with one high-power and two low-power links. We assume that LP1 and LP2 are sufficiently far from LP3 and LP4 so that they cannot hear each other. Everyone can hear HP1 and HP2 because HP1 and HP2 use much higher transmit power.

One sample execution of Weeble MAC is given in Figure 2 (b). In this example, node LP1 first sends the low-power preamble and starts the low-power reservation period. This causes HP1 and HP2 to back off for the duration of the period. LP3 is not aware of all this, but as it does not sense any transmission it keeps on counting down and eventually starts transmitting during the same low-power reservation period. As LP3 is not aware of LP1, it also includes the low-power preamble in its transmission. Since it is received during an ongoing low-power reservation period, HP1 and HP2 will ignore it. Also note that all low-power nodes are allowed to keep on transmitting once the low-power reservation period has expired, but they are not guaranteed that these transmissions will not be corrupted by a concurrent high-power transmission. We explain the Weeble MAC in more detail in Section 4.1.

3. ADAPTIVE PREAMBLES

Weeble is built upon a new preamble design that (a) allows preamble detection at SNRs below -15dB, (b) has an adaptive detection range, (c) works well, i.e. has few false positives and false negatives, in the presence of interference, and (d) is simple and cheap to implement (in terms of silicon area, and hence also power consumption). We describe our design corresponding to the first three goals in this section – using repetitive preambles for detection at low SNRs, support for adaptive preamble lengths, and a detection mechanism that works in the presence of interference. Although the design of our detector is for an OFDM PHY, which is used in most of the relevant white space standards (802.22, 802.11af), our idea can easily generalize to other PHYs.

A key challenge in the design of our preamble detector is synchronization. State-of-the art implementations of preamble detection [9, 33], are able to detect preambles of size of 1 OFDM symbol at SNRs of down to -15dB, but require synchronization between the nodes [12]. Synchronization is difficult to achieve in our scenario since low-power nodes might never be able to communicate with interfering high-power nodes. We thus decided to use longer preambles, which increases the detector's complexity. We show how our design can be implemented without consuming significantly more silicon.

3.1 Detecting preambles at low SNR

We use the well-known phenomenon that the detection accuracy increases with the length of the preamble [20]. A key question then is how to design such a preamble without increasing the complexity of the detector [35].

The preamble in OFDM is used to synchronize with the start of a packet transmission. Hence the preamble sequence $\mathbf{P}$ is a pseudo-random sequence with as little auto-correlation as possible. In our application, in which the high power node only needs to detect (and not decode) low-power packet transmissions, we are *not interested in the detection with such an accurate timing*. It is sufficient to detect the packet transmission with a timing precision even as high as a few tens of micro-seconds. We leverage this lee-way to simplify the receiver design by using repetitive instead of purely pseudo-random preambles.

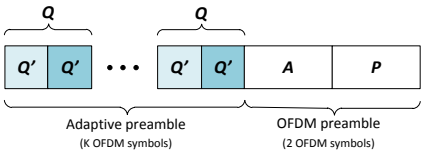


Figure 3: The structure of the PHY header, comprising an adaptive preamble and a standard OFDM preamble (A, P).

We use a symmetric preamble $\mathbf{Q} = (\mathbf{Q}', \mathbf{Q}')$ as the main building block where sequence $\mathbf{Q}' = (Q_0, \cdots, Q_{S/2-1})$ is a pseudo-random sequence. The size of $\mathbf{Q}$ is one OFDM symbol and S is the number of samples per OFDM symbol (typically $S = 64$ or more). Note that this is the same type of preamble used in the state-of-the-art detectors [32, 35]. The key difference in our design is that when we increase the size of the preamble, we do not increase the size of $\mathbf{Q}'$, but we repeat the same basic preamble. Our repetitive preamble $\mathbf{L} = (\mathbf{Q}, \cdots, \mathbf{Q})$ consists of K repetitions of $\mathbf{Q}$ and lasts K OFDM symbols. The preamble is illustrated in Figure 3.

Let $C_n^S = \sum_{i=0}^{S/2-1} (Q_i')^* Y_{n+i}$ be the correlation (multiplication) of the complex input base-band samples Y_n with a complex conjugate of one half preamble $\mathbf{Q}'$. We then calculate the preamble correlation C_n at time n as

$$C_n = \left(\sum_{k=0}^{K-1} C_{n+kS}^S \right) \left(\sum_{k=0}^{K-1} C_{n+S/2+kS}^S \right)^* . \qquad (1)$$

The signal is detected at time n if the correlation C_n is higher than a threshold. We give a schematic representation of our FPGA implementation of the detector in Figure 4.

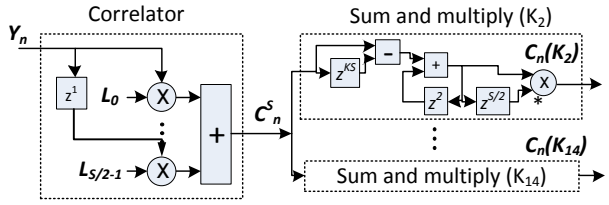


Figure 4: Adaptive detector for different preamble repetitions $K = \{2, 6, 10, 14\}$. Symbol z^n denotes a delay element of n cycles.

In other words, we take the first halves of each of the K repetitions of the preamble (lightly shaded squares in Figure 3), and sum them. We sum the second halves as well (darker shaded squares in Figure 3), we conjugate the sum of the second halves and multiply with the first ones. Intuitively, with (1) we have achieved two important goals: firstly, both factors in (1) grow with K, which allows us to detect the preamble at low SNR; secondly, by multiplying with the complex conjugate of the second, identical copy we get rid of the unknown phase bias, as explained in [35].

Complexity of the detector: We compare the complexity of our detector against the conventional detector [35]. To correlate a preamble of size K OFDM symbols, we do not need to run the full, sample-by-sample correlation of K OFDM symbols ($KS/2$ sam-

ples). Our detector needs $S/2$ complex multipliers and $S/2$ complex adders to calculate C_n^S, and we need one complex multiplier and $2K$ complex adders to calculate C_n. The conventional detector [35] (that does not use a repetitive but a long pseudo-random preamble) performs the same operation, but with $K' = 1$ (no repetitions) and $S' = KS$ (to achieve the same preamble length). For example, for $S = 64, K = 14$ our design requires 416 fewer complex multipliers (93% less) and 388 fewer complex adders (87% less). We evaluate the complexity of our implementation Section 5.

3.2 Adaptive preamble length detector

By choosing the number of the repetitions K, a transmitter can effectively control how far the preamble can be heard. For practical reasons, we limit $K \in \{2, 6, 10, 14\}$. It is the responsibility of the transmitter to choose the appropriate K (we present an adaptive algorithm for choosing K in Section 4.2), and we require that the detector at a receiver does not need to know a priori the number of repetitions K a transmitter has decided to use in the preamble. To that end, we implement four detectors (1) in parallel, for each $K \in \{2, 6, 10, 14\}$. A preamble is *detected* if any of the four detectors exceeds its corresponding threshold.

To illustrate that the detector does not need to know which value of K the transmitter has choosen, consider an example where the transmitter chooses $K = 10$ repetitions. If the correlator $K = 10$ misses the preamble, then it is very likely that the correlator $K = 14$ will also miss the preamble. This is because, in addition to 10 repetitions of the preamble $\mathbf{Q}$, correlator $K = 14$ also correlates 4 other arbitrary OFDM symbols, and the correlation level is lower then expected for a preamble of length 14. Similarly, the correlators $K = 2$ and $K = 6$ are also likely to miss the preamble if $K = 10$ misses it, because the detection accuracy increases with the preamble size. Hence, the performance of the correlation is as good as when only the correlator $K = 10$ is used.

3.3 High-power and low-power preambles

We next discuss the effect of false positives from high-power interferers, which we illustrate using a simple experiment in our test-bed (described in Section 6.1). We place two nodes close to each other, and we put a variable attenuator at the transmit antenna of the transmitting node. In one experiment, the transmitter transmits adaptive preambles $\mathbf{L}$. In the second experiment the transmitter transmits regular OFDM packets without adaptive preambles $\mathbf{L}$ (mimicking high-power nodes' packets). In the third experiment the transmitter is switched off and we only receive the background noise. In all three experiments we measure the maximum correlation value C_n observed in a fixed time interval. We plot each observed C_n against the SNR at the receiver (calculated from the corresponding attenuation values) in Figure 5[2]. False positives have more negative impact on the system performance (as they can potentially starve high-power nodes), hence we plot 90% for confidence intervals for the signal and 99% confidence intervals for the noise.

We see that the correlation against the background noise, when the transmitter is idle, is very low. It visually seems that the signal can be reliably detected versus the background noise for all measured SNRs. However, we also see that the correlation with the OFDM interference can be much higher[3]. For example, the interfering packets at 0dB (the noise level) have 5dB higher correlation than the background noise. It seems from visually inspecting Figure 5 that if we set a correlation threshold of 55dB to reliably detect low-power packets at an SNR of -10dB, we will have a large number of false positives from high-power packets at 0dB and below, which we cannot otherwise detect using energy-based carrier sensing.

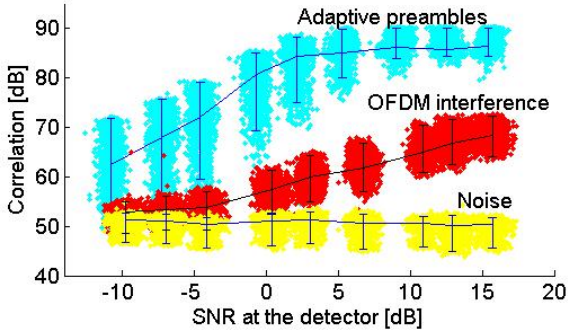


Figure 5: Correlation as a function of SNR for the loopback link and preamble length 10 OFDM symbols ($K = 10$).

To address the problem of false positives, we define a unique short preamble $\mathbf{H}$ for high-power nodes, which is a repetition of $K = 2$ OFDM symbols, different from the ones in the $\mathbf{L}$ preamble. We then use the following detection algorithm

Adaptive detection algorithm

if for any K, $C_n^{L(K)} \geq T^{L(K)}$ **and** $C_n^H < T^H$
and *RSSI < CS threshold*
declare *preamble detected*

where $C_n^{L(K)}$ is the correlation with the preamble $\mathbf{L}$ of length K OFDM symbols and C_n^H is the correlation with the preamble $\mathbf{H}$ at time n, RSSI is the received signal strength indication, and $T^{L(K)}$, T^H and *CS Threshold* are the corresponding thresholds[4].

The main intuition for this algorithm is as follows. The variance in correlation value comes from random attenuation in the system (channel, noise, etc). False positive from high-power interference is more likely at times when the random attenuation is low, hence the interfering signal is high. But at the same time, the correlation with the $\mathbf{H}$ is also likely to be high. So the idea is that, if we detect *both* preambles $\mathbf{L}$ and $\mathbf{H}$ during a short time interval, we can conclude that it is an interfering packet and ignore it. Furthermore, we also ignore correlation if energy is sensed (this implies a regular backoff instead of a low-power reservation).

4. WEEBLE MAC DESIGN

In this section we discuss the Weeble MAC design in detail.

4.1 Low-power Reservations

One of the key challenges in our design, described in Section 2, is how to determine the length of the reservation period. A simple idea would be to use another preamble to signal the end of the period. However, this is unwise for two reasons. Firstly, if the preamble for the end of the period is missed by a high-power node, this node might get starved[5]. Secondly, sending another preamble

[2]The signal correlation stops the increase at high SNRs due to the saturation of the receiver's dynamic range

[3]This is because the background noise and the interfering packets have different statistical properties.

[4]We set the threshold levels to the minimal values that are not exceeded when correlating with the white noise. These thresholds are similar in nature to the CS Threshold, and can be calibrated in the same way in practice.

[5]whereas if the start of the period preamble is missed, this will cause a loss of a single low-power packet.

increases the protocol overhead and the complexity (as we need to detect two different types of preambles).

Instead, we opt to use a reservation period of a fixed duration and we rely on contending low-power nodes to make an efficient use of it. Any node can signal a start of a low-power reservation period. Once a period has started, all high-power nodes that heard the preamble will refrain from transmitting for the fixed period of time. All low-power nodes may contend for the access for the fixed duration of the period using CSMA. Once the packet that initiated the reservation period has ended, other nodes may start transmitting. Also, other low-power nodes in other collision domains will likely transmit in parallel (as already observed in [5] in the context of multi-hop networks). They will not hear the **L** preamble that initiated the reservation period, but they will find the medium idle from high-power transmission and use the opportunity to access it. Both cases are illustrated in Figure 2 in Section 2.

Once a low-power reservation period expires, both high-power and low-power nodes contend for transmission. If a low-power node wins the contention, it sends an adaptive preamble. Then another low-power reservation period starts, and all high-power nodes back off. Otherwise, a high-power transmission will start.

Another challenge is how to prevent performance impairment and potential starvation of the high-power nodes. As illustrated in Figure 2 in Section 2, several low-power nodes that do not hear each other may enchain the reservations and starve a possible multitude of high-power nodes. To avoid this scenario, a high-power node ignores any **L** preambles it detects during a ongoing low-power reservation period.

Another effect, illustrated in Figure 2, gives priority to high-power nodes. If many low-power nodes contend for access, it is very likely that someone will be transmitting at the very end of the a low-power reservation period. However, at this point, high power nodes start contending again, and decreasing the back-off counter. Thus it is more likely that a high-power link will gain access right after a low-power reservation period. Note that the winning high-power transmission (HP in Figure 2) may destroy the last low-power transmission extending beyond the reservation period (the second LP in Figure 2). However, this does not affect the high-power transmission because its SNR is sufficiently high, and hence does not reflect on the efficiency of the network.

To avoid starvation of the low-power nodes, each low-power node contends separately for a low-power reservation period. A low-power period will start whenever any of the low-power nodes wins the medium and sends an adaptive preamble. Hence, the more low-power nodes there are, the more likely they are to gain the access. Clearly, the more low-power nodes there are, the lesser are the chances that a high-power node will get an access, but this is in accordance to the principle of fair medium access.

The duration of the low-power reservation should not be too large, to avoid over-booking the air if there are not too many low-power nodes. It should also not be too small, because each reservation period is preceded by an adaptive preamble, hence short periods incur high overhead. For a network with 802.11a/g PHY, we choose a standard low-power reservation duration of 600 μs. As we illustrate in Section 6.3, this is a good compromise between fairness and efficiency.

Finally, we mention the issue of the hidden terminal problem in our setting. As we have seen in the example from Figure 2, the L preamble of node LP3 is ignored by HP1 and HP2 because it was transmitted during an ongoing reservation period. However, that same preamble could have been respected by some other HP node (say HP3, not shown in Figure 2) who cannot detect LP1 and hence it missed its L preamble, but can detect LP3. Since the two reservation periods are not in sync, node LP3 cannot fully utilize the newly acquired reservation period at HP3. The key thing to observe here is that while LP3 started the reservation period, the same reservation period will be used by other LP nodes obstructed by HP3, hence the efficiency of the network should still be high. This is confirmed by our simulation results, presented in Section 6.3.

We give the pseudo-code of the MAC algorithm below, extending WiFi MAC (HP - true if the node is high-power, LP - true if the node is low-power, L - low-power preamble).

Medium access protocol

if L detected **and** *reservation timer* $>$ *reservation length* **then**
  start *reservation timer*
if HP **and** *reservation timer* $\leq$ *reservation length* **then**
  freeze cw counter
if *carrier sense start* **then** freeze cw counter
if *carrier sense end* **then** count DIFS
if *DIFS count end* **then** unfreeze cw counter
if $cw = 0$ **then**
  if LP **and** *reservation timer* $>$ *reservation length* **then**
    transmit preamble of *preamble_size*
    start *reservation timer*
  transmit packet
  if *no ACK received* **then**
    increase $CW = \min(2\,CW, CW_{max})$
  else $CW = CW_{min}$
  $cw = \text{rand}[0, CW]$

4.2 Adaptation Algorithm

Sending an adaptive preamble incurs overhead. It also prevents any high-power node that detects it from transmitting concurrently. It is thus important to carefully decide when to send adaptive preambles, and how long should they be. We want to use packet losses as a feedback whether to increase or decrease (or switch off) adaptive preambles. However, we need to be able to distinguish losses due to a high-power interferer from other types of losses.

There are three primary reasons why a low-power wireless link will lose packets. The first one is due to interference from concurrent high-power transmissions. The second reason is a MAC level contention. In DCF, as the number of nodes contending for medium access increases, so does the number of collisions that are due to two or more links starting transmitting at the exact same slot (and thus not having enough time to detect each other and avoid collision). If packets are lost due to contention, we do not want to use this as a signal to start using adaptive preambles or increase the preamble length. In fact, longer preambles can only make things worse by introducing additional overhead to an already congested medium. The third reason for wireless losses is the link loss, due to wireless channel changes. We do not want to switch on adaptive preambles or increase the preamble size due to these losses either.

We start by observing that in case of the second and third type of losses, we are not very likely to see many consecutive losses. For example, we measure that the average number of consecutive losses with 16 contending nodes is 2.5. Similarly, most modern rate adaptation algorithms are able to adapt the rate with minimal channel losses (less than 10%-15%), hence wireless losses are unlikely to occur consecutively.

On the contrary, losses due to interference are very likely to occur in a sequence. Namely, if a high-power node has data to send, and if it does not hear a low-power node, it will continuously transmit and kill several subsequent low-power transmissions. Also,

consecutive packet losses are particularly bad for the link performance, as they will exponentially increase the back-off counter, and cause link starvation. If the high-power nodes interfere only sporadically, we will not activate the preamble protection, as this will not create starvation of low-power nodes.

We propose an adaptive preamble tunning algorithm based on the additive-increase, multiplicative-decrease (AIMD) principles. We use the number of consecutive packet losses as a measure of interference. We choose AIMD form of adaptation to be more conservative in blocking HP nodes. The pseudo-code of the algorithm is given below.

Preamble adaptation algorithm

Initialization:
$consecutive \leftarrow 0$;
$counter \leftarrow 0$;
if consecutive loss **then** $consecutive$++;

After every packet transmission:
if consecutive loss **then** $consecutive$++;
else $consecutive \leftarrow 0$;
if $consecutive = 6$ **then** $counter$++; $consecutive$++;
else $counter \leftarrow counter \times 0.9$;
if $counter \leq 2$ **then** $preamble\_size \leftarrow 0$;
else $K \leftarrow (2, 6, 10, 14)$;
$preamble\_size \leftarrow K[\lfloor counter \rfloor - 2]$;

Due to the scale of the power difference between the low-power and the high-power nodes, low-power nodes that contend among themselves are likely to see the same high-power interferers. Although each of them runs the adaptive algorithm on its own, it is very likely that they will use a similar level of protection, regardless of who started a low-power reservation period.

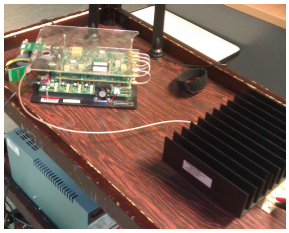
5. IMPLEMENTATION

We have prototyped the MAC and physical layer protocols on the Lyrtech Software Defined Radio SFF-SDR platform [25].

Adaptive preamble detector: We implement the adaptive preambles and the corresponding detection algorithm described in Section 3.2, on the top of an OFDM transceiver in FPGA (Xilinx Virtex-4 SX35). We compare the complexity of our adaptive detector with the complexity of a conventional implementation [35], as described in Section 3.1. The FPGA die consumption for the two implementations is given in Figure 6 (b).

As we can see, the conventional implementation consumes significantly more FPGA slices than the adaptive implementation. For example, the 8 OFDM symbols standard implementation consumes more die than the adaptive implementation that supports correlation across 2, 6, 10 and 14 (repeated) OFDM symbols at the same time. Moreover, we cannot fit a single standard 14 OFDM symbols correlator alone on our FPGA. As a comparison, an entire OFDM transceiver design with a conventional preamble detector fits onto a single Xilinx Virtex-4 SX35 FPGA. This means that a naive detector implementation would take more die (and most likely power as well) than the entire OFDM transceiver design.

Weeble MAC: In contrast to the recent software-defined radio deployments [33, 36, 9, 8, 12] that only use SDRs to collect channel traces and process them offline, we implement the full Weeble MAC in the Lyrtech SDR's DSP using the Colombo SDK [10] for rapid SDR MAC prototyping. This allows us to evaluate if the preambles indeed provide protection from a high-power interferer in different topologies and channel conditions, if the adaptive preamble algorithm correctly chooses preamble sizes to prevent starvations, as well as to assess the effectiveness of Weeble in the real world in presence of multiple high-power transmitters.



Detector length	#slices
Std. (1 symbol)	3,164
Std. (2 symbols)	4,834
Std. (4 symbols)	8,105
Std. (8 symbols)	14,617
Adaptive	13,850

Figure 6: (a) A picture of a HP node in the test-bed; (b) FPGA die consumption for different detectors

The separation of functionality between the FPGA and the DSP, imposed by the limited FPGA size, introduces an overhead. Due to long packet transfer time between the FPGA and the DSP imposed by board's architecture, the minimum MAC slot length is approximately 20 times longer than the Wi-Fi MAC slot length. Consequently, to ensure protocol correctness, we proportionally scale the intra-packet times – SIFS, DIFS and CW. Moreover, to achieve the same ratio between the packet duration and the inter-packet times as in WiFi, we scale the packet transmission times proportionally (our packet transmission lasts 4.5ms, about 20 times longer than it takes to transmit a 1.5 KB packet at 54Mbps). We compare Weeble MAC with WiFi MAC in the same implementation, hence we do not introduce any performance bias.

The only overhead that does not scale is the duration of the adaptive preamble. However, this is a relatively small overhead. It takes 224μs for a 802.11a/g PHY to transmit a 1500B packet at 54 Mbps, which is augmented by a constant MAC overhead of 181μs [26]. The longest preamble takes 56μs, which is 13.8% of the total packet transmission time. These extreme preamble lengths will not be used very often in practice, as illustrated in Figure 9(c), hence the expected overhead from preambles will be even lower. Thus our test-bed results represent a close approximation of the real system's performance.

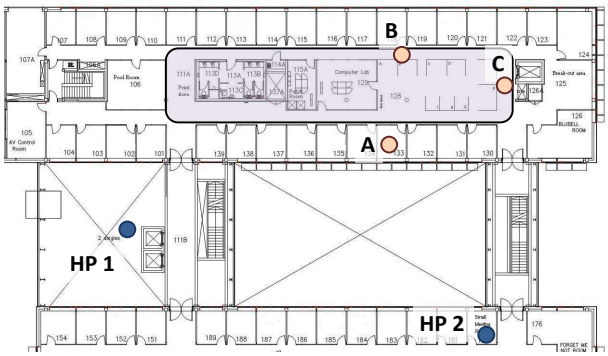


Figure 7: Floorplan of our building with the locations of measurements. Building dimensions are approximately 60m $\times$ 45m. The high power nodes HP 1 and HP 2 are located on the ground floor and second floor respectively and the nodes on the top wing are located on the first floor.

Testbed & Simulation Platform: We set the carrier frequency of the radio transmission to 580 MHz. Our channel bandwidth is 10 MHz. Our indoor test-bed spans both wings and all three floors or our buildings. Its floorplan is given in Figure 7. High-power nodes

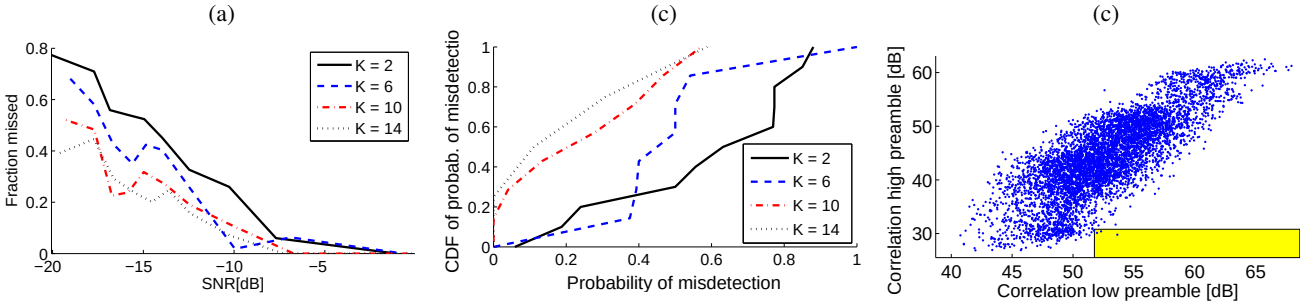


Figure 8: (a): The average number of missed adaptive preambles as a function of SNR and the preamble length K; (b) CDF of probability of misdetection across different channel profiles, for SNR = -16.5 dB; (c) False positives with H and L preambles.

(shown in Figure 6 (a)) have a transmit power of 4W, and low-power nodes transmit at 40mW. To test the scalability of our design in larger networks, we further implement the full Weeble PHY and MAC in QualNet. We use measurements from Section 6.1 to tune the realistic PHY model parameters in QualNet for long preamble detection.

6. PERFORMANCE EVALUATION

In this section we quantify the performance of Weeble in three stages. First, we micro-benchmark the core building block of Weeble – the adaptive preamble technique. Second, we evaluate the benefits of Weeble in our small scale testbed spanning an entire building. Finally, we study the protocol's performance in a large setting using the QualNet simulator.

6.1 Reliability of Adaptive Preambles

We micro-benchmark our adaptive preamble detector in a building-wide deployment by measuring its reliability when detecting preambles at low SNR. We set up a high-power node (HP1, Figure 7) in the atrium of the ground floor and we configure it to send packets, each with a sequence number and the expected length of the preamble to follow. We also set up a low-power node that replies with a preamble of the corresponding length. We move the low-power node, walking slowly, throughout the shaded area on the first floor.

We cannot directly measure the receive SNR at the high-power node because it is below 0dB. Instead, we log the receive (LP) SNR at the low-power node. In various measurements we observed that the channel is approximately symmetric (with only a few dBs of difference) and we record an approximate SNR as the observed LP SNR at the low-power node minus 20dB of transmit power difference.

Detection accuracy: We first evaluate the detection accuracy as a function of the SNR. We divide the range of the recorded SNR into bins, and calculate the aggregate preamble detection rates for the entire duration of the experiment for each bin, for different preamble lengths. This is plotted in Figure 8 (a). Note that the curves are not strictly decreasing in SNR because different channel profiles have different performances for the same SNR. We see that for adaptive preamble lengths $K = 10 - 14$ for SNR $>$ -17 dB, we can have detection probability of 70-80%. Note that, if the SNR $=$ -17dB at the high-power receiver, the interference it creates at the low-power receiver is at most 3 dB (because the power difference is 15 – 20 dB). This is of the same order as a typical carrier sensing threshold [23], so we see that our virtual carrier sensing mechanisms can guarantee the same level of interference protection as the real carrier sense, with 70%-80% success rate even when the power difference is 20dB.

Next, we look at how the probability of detection varies across links with similar SNR values. We divide all the received packets in batches of 20. Since the packets are sent back-to-back, we see that all packets within a batch see the same SNR. For each batch we record a single probability of detection (out of 20 packets in a batch). We then aggregate batches with the similar SNR. We plot the CDF of the probability of misdetection for the lowest SNR of interest in Figure 8 (b). Clearly, different links will take different position on these curves. However, we see that except for the 20% most unfavorable links, we can have more than 70% success detection rates with SNRs as low as -16.5 dB (meaning we are able to prevent a high-power interference on a low-power node stronger than 3.5 dB).

The remaining question is whether a high-power interference will affect the reception of low-power packets in cases when preambles are missed. We defer this discussion to the next section, where we evaluate Weeble MAC in the test-bed.

Avoiding false positives caused by HP interferers: Finally, we evaluate our detection algorithm from Section 3.3 that addresses the problem of false positives. We calculate $C^{L(K)}$ (for $K = 10$) and C^H for about 50,000 packets transmitted by the HP1 node, and received by the LP node at various locations in the shaded area in Figure 7. We present the results in Figure 8 (c). On the x axis we plot the correlation with the **L** preamble $C^{L(K)}$ and on the y axis with the **H** preamble C^H. The lower shaded box is the area of false positives - a packet that detects the low preamble but fails to detect the high preamble. The observed probability of false positive is 10^{-4}.

6.2 Performance of Weeble in a Small Testbed

We now show the performance of Weeble in our building-wide test-bed. We deploy two fixed high power nodes and two mobile low power nodes in the area depicted in Figure 7. The high power nodes do not carrier sense each other, and hence transmit simultaneously. All nodes are backlogged with UDP traffic[6].

Due to the limited size of the FPGAs (a low-end Virtex-4 SX35), we were unable to fit the entire OFDM transceiver and the adaptive detector in the same circuit. Instead, the high-power nodes comprise of a transmitter, an adaptive detector, and the full MAC implementation (where we assume that all transmissions are successfully acknowledged). Consequently, HP1 and HP2 do not need paired receivers in the experiments. Note that this does not greatly affect the experiments, as the nodes always have backlogged packets to transmit, regardless of the results of their previous transmission. Low-power nodes do not require an adaptive correlator and we implement them in full. In this section we compare Weeble's performance to the WiFi protocol (since IEEE 802.11af for white spaces [3] is WiFi-based).

[6]We cannot run TCP traffic as it times out due to our down-clocked MAC implementation.

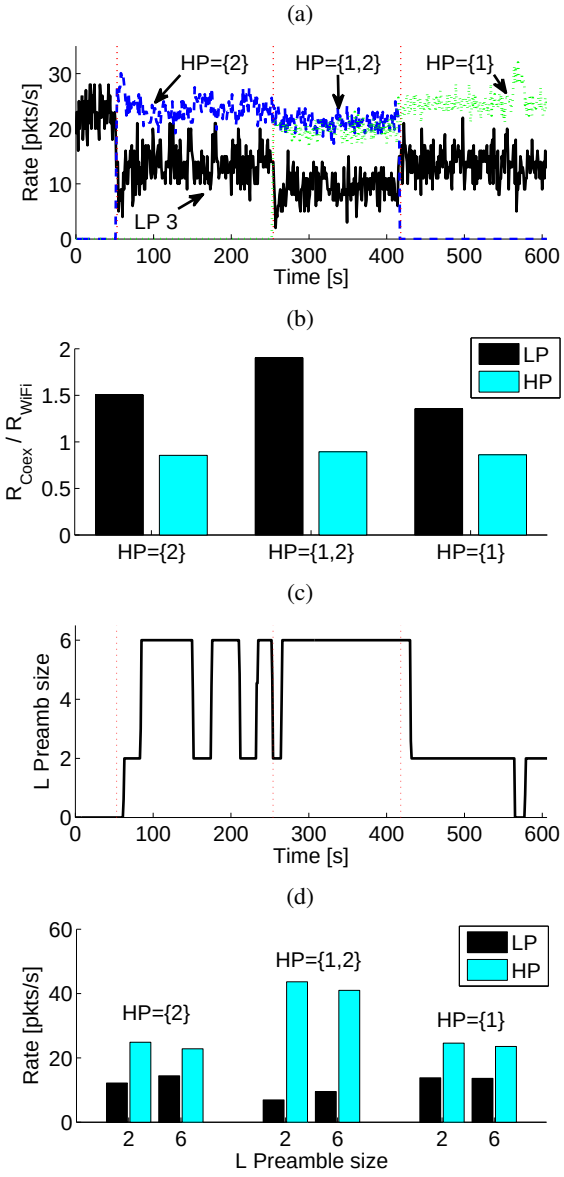


Figure 9: Example scenario with different number of high-power nodes: (a) The rates of flows in time, (b) The relative increase of flow rates when using Weeble MAC instead of WiFi, (c) The evolution of the L preamble size and (d) The average rates of LP and HP flows when using different L preamble sizes.

6.2.1 *Weeble MAC in action*

We first illustrate the dynamics of Weeble in a network where the two low-power nodes are placed at positions A and B in Figure 7. Figure 9 (a) depicts how the link rates change in time, as we switch on and off the interfering HP nodes.

On turning on one HP flow ($t = 50$s), the rate of the LP flow drops by about half. The sending rate of the HP flow drops by about 10%. The total throughput is higher because the medium is more efficiently utilized by two flows than by a single flow. As expected, the HP flow gets a larger share, and the LP flow is not starved.

When both HP flows are turned on ($t = 250$s), the LP flow's rate drops only slightly. This is because the two HP flows do not interfere with each other and the MAC converges to the optimal schedule: simultaneous HP1 and HP2 transmissions alternate with a single LP transmission. The adaptive preamble detection mechanism proves to be successful in presence of two non-coordinated high-power transmissions.

We compare the performance improvement over the WiFi MAC protocol in Figure 9 (b). We see that the rate of the LP flow improves by 50%-90% at the expense of a modest (10%-15%) decrease of the rates of the HP flows. Note that in this example WiFi exhibits slightly lower efficiency than Weeble MAC. This is because the low-power packets in WiFi MAC are frequently lost due to collisions.

Finally, in Figure 9 (c) we see the dynamics of the preamble adaptation algorithm (Section 4.2). HP2 can very accurately detect low-power preambles of length $K = 6$ from the LP node, which yields very low loss rates on the LP link. After a period of a low loss, the low-power link will reduce its preamble size to 2, which will in turn increase the loss and the adaptive algorithm will force the preamble size back to 6. As we see in Figure 9 (a), the adaptation process does not reflect substantially on the short-term flow rates (it does not cause any temporary starvations) because the preamble length is only decreased, and not entirely switched off.

In Figure 9 (d) we see the data rates achieved with different lengths K of low-power preambles. We see that in the first period, when LP and HP2 are active, both flows have similar flow rates for $K = 2$ and $K = 6$, hence the algorithm oscillates between the two values. In the second period, when LP, HP1 and HP2 are active, the LP flow sees a further increase in packet losses and a decrease in rate, hence its switches to using exclusively $K = 6$, and achieving a more fair rate. Finally in the third period, when LP and HP1 are active, $K = 6$ has no visible benefits on protecting LP but slightly decreases both rates due to the overhead. For that reason, the algorithm chooses $K = 2$.

We also see from Figure 9 (a) that it takes a 10s-20s for the MAC algorithm to react to an activation/deactivation of a high-power link. Given that the packet transmission times are $\approx$ 20x longer than in a WiFi network (see Section 5 for the discussion), an implementation of Weeble that uses WiFi data rates would observe sub-second reaction times.

6.2.2 *Performance in different topologies*

Setup: We place one LP node either in location B or C, and the other LP node on a trolley at many different locations, all around the shaded area in Figure 7. We record the data rates at each location pair. We repeat the same set of experiments with different parameters (running traffic from B to C, C to B or both directions, and/or having one or two interfering HP nodes), both with Weeble and WiFi MAC.

Starvation: In Figures 10 (1HP-a) and (2HP-a) we see that Weeble successfully avoids starvation of the LP flows. Using Weeble, no flow has recorded a rate below 4 pkts/s in any measured location. With WiFi, 10% of LP flows had rates of less than 4 pkts/s. Also, the median rate of the LP flows running Weeble are about 50% larger than the median rate of the LP flows running WiFi. Both findings are true when either a single or both HP flows are active.

Total throughput: Figures 10 (1HP-b) and (2HP-b) plot the CDFs of the sum of the rates of all flows in the network. We see that when a single HP node is active, the sum of the rates is almost the same for WiFi and Weeble (and occasionally Weeble achieves a higher sum of the rates, as discussed in Section 6.2.1). In the presence of 2 HP flows, the sum of the rates are slightly smaller with Weeble than with WiFi (on average by about 6%). This is because a single LP transmission competes with two concurrent HP transmissions and any extra rate allocated to the LP flow decreases the rates of both HP flows.

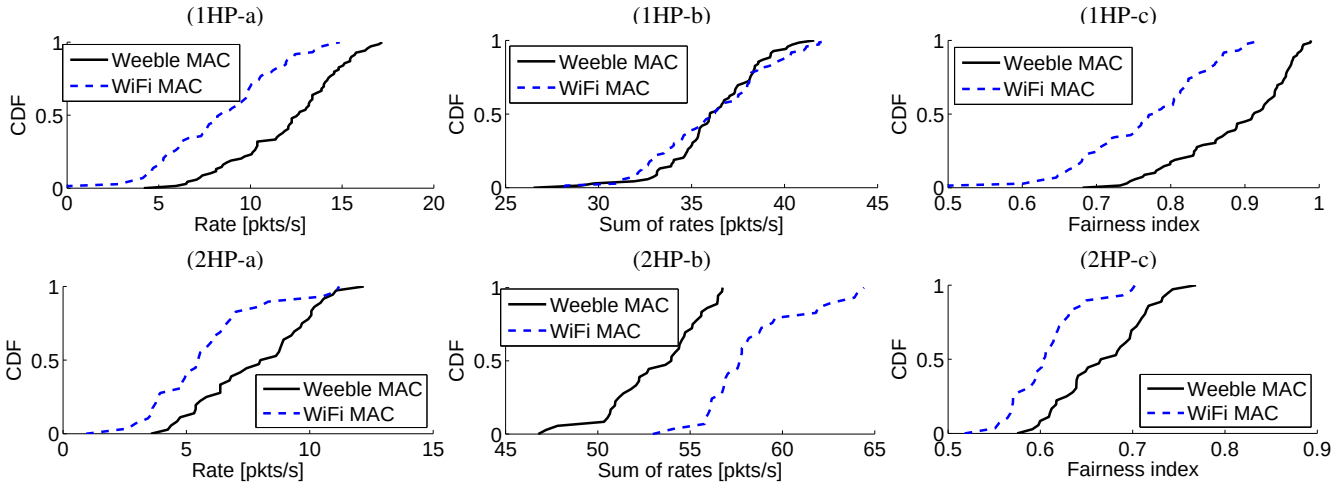


Figure 10: Cumulative distribution functions of the sum of the rates of the LP flows ((1HP-a) and (2HP-a)), the sum of rates of all flows ((1HP-b) and (2HP-b)) and the fairness indices ((1HP-c) and (2HP-c)) for Weeble MAC and WiFi for networks with one and two high-power (HP) nodes.

Fairness: Finally, in Figure 10 (1HP-c) and (2HP-c) we plot the Jain's fairness index [17]. In the design requirements we ask that HP flows get more rate than the LP flows, hence the fairness index is far from one in both cases (and it is smaller when 2 HP flows are active). However, we see that Weeble achieves a median fairness index by 10%-20% higher than WiFi.

Overall, we see that Weeble MAC improves fairness and avoids starvation of low-power flows with a minimal impact on the performance of high-power flows. These observations are consistent through a large number of topologies we evaluated and with one and two active high-power links.

6.3 Detailed Evaluation in Simulations

We now evaluate Weeble in QualNet. There are three main goals of the evaluation in simulations. Firstly, we want to verify that our approach scales to larger networks. Secondly, we want to evaluate the performance with more realistic network parameters, such as TCP traffic and variable data rates. Finally, we want to quantify the overhead of the long preambles with respect to various WiFi data rates.

Setup: We developed a performance model of the preamble detection algorithm from the measurement results obtained in our testbed (Section 6.1), and plugged it into the QualNet simulator. We use autorate as the rate adaptation protocol. In Weeble we select the low-power reservation duration (Section 2) to be 600us.

We compare Weeble to two alternative approaches: WiFi and Frequency Division Multiplexing (FDM) where HP and LP links are allocated different frequencies. In our WiFi implementation, all nodes use the standard WiFi MAC, regardless of the transmit power. In our FDM implementation we divide the available channel into two sub-bands and assign one of the sub-bands to low-power and the other to high-power nodes. The key question when analyzing the FDM's performance is how to split the available bandwidth between the two sub-bands. Clearly, if one knew the exact number of nodes in each band and split the bandwidth proportionally between them, FDM would be the optimal approach. However, this is precisely the difficulty of the problem we are studying, as the network is dynamic, some geographic areas may have more high-power and others more low-power nodes, and there is no centralized coordinator available. We thus assume a topology-oblivious approach in which we assign half the bandwidth to each band. Within each sub-band, nodes compete using WiFi MAC. We double the DCF time-slot in the FDM implementation to keep the overhead proportional to the packet size.

We note that Jain's fairness index is only meaningful for comparing fairness in systems with similar overall rates [17]. In the test-bed evaluation (Section 6.2) we observe that the sum of rates of all the flows are similar in all cases, and we use the Jain's fairness index. This is not the case in the larger networks considered in this section, so instead we use the number of starved flows as the fairness metrics[7]. We also study the rates of the LP flows as a fairness metric, and the sum of the rates of all flows as an efficiency metric.

We evaluate the performance of our MAC on a set of random topologies. We consider a 1km $\times$ 1km square area and a single channel. We randomly place 20 low-power nodes (10 low-power links) and 4 high-power nodes (2 high-power links) in the area and on the same channel. For example, this case could correspond to a real-world scenario with 200 low power links and 40 high power links uniformly spread across 10 available TV channels. High-power links transmit with 4W transmit power and low-power links transmit either with 40 mW or 100 mW (as in the White Space scenario [6]). Link lengths are selected randomly, but in such way that the achievable link rate is at least 12 Mbps. We run a single FTP flow over each link. Flows are single hop. All packet sizes are 1000B.

We select 10 random network topologies. For each topology we execute five runs and each run lasts 20s. We calculate the average rates of flows over all five runs for each topology and we plot the CDFs of different metrics over the 10 random topologies.

Starvation: In Figure 11 (a) we plot the TCP rate of the flow with the smallest rate in the network. We see that the smallest rates of the high-power flows are comparable for all three MAC designs, even in presence of a large number of low-power links. However, Weeble MAC assigns significantly better rates to the smallest low-power flows. Both WiFi and FDM MAC yielded zero throughput to at least one flow in 90% of the cases. This did not happen with Weeble MAC.

We next look at the number of starved flows, that is the number of flows that achieve TCP rate lower than 100 kbps. This is depicted in Figure 11 (b). FDM on average starves around 2 low-power flows

[7]Note that since our test-bed is relatively small there are no starved flows, hence the number of starved flows metric is not applicable to the test-bed evaluation.

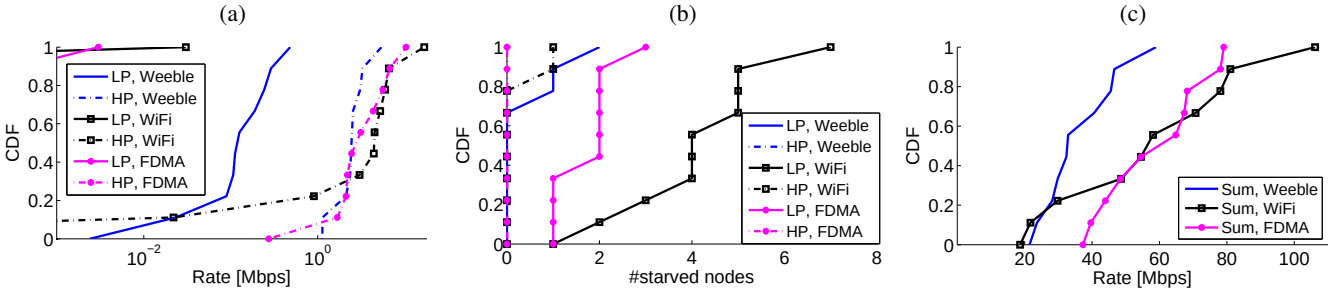


Figure 11: Performance for random scenarios: (a) TCP rate of the flow with the smallest rate; (b) number of starved flows in the network (flows with the TCP rate less than 100 kbps); (c) Sum of TCP rates of all flows in the network.

per network topology (20% of flows) and WiFi around 4 low-power flows per network topology (40% of flows). We see that Weeble starves on average 4% of flows, and in 7 out of 10 topologies it did not starve any flow. We also note that WiFi starves one of the two high-power flows in 20% of the cases. Weeble does not starve high-power flows.

WiFi is particularly bad in starving low-power links because high-power nodes cannot detect them. FDM also starves some low-power links because it reserves only half of the bandwidth for 10 low-power links formed by 20 competing nodes running TCP flows. By design, Weeble MAC is able to dynamically share the capacity among available links, regardless of how many low-power or high-power links are present.

Total throughput: Finally, we look at the total network throughput for the three MAC protocols. We see that the loss in total throughput is 0%-40% when compared to WiFi and 20%-35% when compared to FDM. This inefficiency is due to imperfect carrier sensing. Weeble has to speculatively schedule low-power reservation periods of fixed lengths (particularly bad when a TCP window doesn't have packets to send) whereas with WiFi and FDM the backlogged nodes can resume the countdown whenever the RSSI has gone down. This inevitable trade-off is due to power asymmetry that implies that any signalling between the low-power and high-power nodes comes at a cost. In contrast, WiFi ignores low-power nodes and FDM isolates them in a static sub-band. In both cases there is no signalling between the low and high power nodes. The protocols are non-adaptive, and cause starvations.

Comparison with test-bed results: We see that the effects of interference of high-power flows on low-power flows in simulations are more severe than in the test-bed results, presented in Section 6.2. This is because the simulated network has more links hence there is more contention, and also because the links use TCP, whose performance is more affected by interference. We also see that the total network throughput is more affected than in the test-bed evaluation results. This is again because the simulated network has more low-power links and more medium access time has to be reserved for them.

7. RELATED WORK

Coexistence: Most prior work on how to coexist with an incumbent (a legacy transmitter) or with a high-power node can be divided into two groups. The first group comprises of the designs in which a new technology tries to avoid being starved (avoid interference from incumbents or high-power nodes). One idea is to transmit when the incumbents/high-power node is idle (see e.g. [13]), but it fails when the incumbent is always active. Jung et al [19] propose to use high-power signaling packets to avoid starvation of low-power nodes. This is impossible in our setting due to power and regulatory constraints. Gollakota et al [8] suggest using orthogonal MIMO streams to avoid incumbents. This approach works well with 802.11n against legacy hardware (which mainly uses single antenna for transmission) but it is not suitable for coexistence in white-spaces. In white-spaces, MIMO may be deployed on the high-power nodes; the low-power nodes are assumed portable and MIMO is highly impractical[8]. Both requirements are incompatible with the approach from [8].

The second group are the designs where a new technology tries to avoid starving incumbents/low-power nodes. One example is IEEE 802.11h standard [15] that imposes power control to WiFi nodes to avoid interference with satellite communications. Another example is IEEE 802.22 [16], where secondary users need to sense and back-off when primaries are present. Also, UWB must detect and avoid WiMAX devices in certain regulatory domains [27]. However, in all these cases the detection of primary signals takes long time (as we discuss below) and the packet-level statistical multiplexing is not possible. SWIFT [31] addresses the coexistence problem through dynamic frequency adaptation. However, it does not apply to our asymmetric power setting.

Detection: An important part of the coexistence problem is how can a high-power node detect whether a low-power user is disturbed by its interference. One way of detecting a legacy interferer is cyclo-stationary detection (c.f. [12, 22] and references therein). However, cyclo-stationary is slow at low SNR and one is not able to use it for statistical multiplexing at a packet level. Other feature-based detection techniques, such as [38, 1, 36], also do not work at SNRs below 0dB. The other approach is to detect a known preamble, which is also used as a part of the carrier-sense mechanism [18]. Busy tone approach [11] is a special case of detecting a known preamble. Several recent papers [9, 33] show that it is possible to detect an interferer with 80% accuracy at SNR as low as -14dB, but only if all nodes are coarsely synchronized (see [12, 9, 33] for discussion). Coarse-level synchronization is not possible in our setting. Also, repetitive preambles are used in 802.11 standard for coarse-grained synchronization, but they do not work at low SNR and they are not adaptive. It is worth noting that there are other detector designs for multi-path channels with Gaussian noise, such as [29], but it is not a priori clear how they could be modified to be adaptive, simple and to scale to very low SNR.

Adaptive carrier-sensing: Several papers consider adapting the carrier sensing threshold [37, 21] to optimize the network performance. However, these techniques do not consider power-asymmetric networks and cannot be used when the SNR is below the noise floor.

Antenna design: The use of directional antennas is one approach to tackle the hidden terminal problem and increase network performance [4, 24]. This approach mitigates but it does not solve the coexistence issue, and it increases the complexity. Similarly,

[8]the wavelength of the white-space bands is around $1/2$m

high-gain antennas do not solve the problem as they extend both the reception and the transmission range of high-power nodes.

Existing TV white-space standard activities: IEEE 802.22 [16] is a regional network standard. It introduces a spectrum manager entity at each node, whose goal is to provide the dynamic spectrum access according to constraints obtained from different sources (geo-database, policy, sensing). The time-scale of spectrum allocation is much longer than a packet duration. Another white-space related standard activity is 802.11af [3], which focuses on in-home devices. In order to keep the network simple and leverage on the existing silicon design for 802.11, it extends 802.11 design to address the challenges of white-space networks. Coexistence between high-power and low-power nodes is currently an open problem in the 802.11af standardization process, which we aim to solve with this paper. Our work is also aligned with IEEE 802.19 [14], another ongoing standardization activity that is planning to address similar coexistence problems.

8. DISCUSSION AND FUTURE WORK

Can Weeble MAC ideas be used for different MAC designs? Weeble MAC can be seen as a set of generic signaling and reservation techniques. For example, they could also be applied in conjunction with a spectrum manager proposed in IEEE 802.22 [16] to allow for coexistence among several administrative domains, in case two non-coordinated spectrum managers belonging to two different administrative domains happen to allocate different networks to the same channel. Evaluating such a design remains as a future work.

Can Weeble be generalized to multiple power-level classes? Our protocol naturally extends to multiple power-level classes. Each power-level class i should have has its own $\mathbf{L}_i$ and $\mathbf{H}_i$ preambles. A node at power-level i should be able to detect and back-off on all preambles $\mathbf{L}_j$ coming from nodes at lower power-levels $j < i$. It should also be able to detect and ignore all high-power preambles $\mathbf{H}_k$ for all $k \geq i$. However, this generalization increases the PHY complexity as each node in the network should implement as many correlators as there are power-level classes in the network.

Can one adjust the priorities of high-power and low-power traffic? Our design goal is to avoid starvation of low-power nodes with a minimum impact on the performance of high-power nodes. However, it is easy to extend our design to achieve arbitrary prioritization of different high-power and low-power devices. For example, to guarantee an absolute priority to a low-power wireless mic, one could mandate the high-power nodes to back-off whenever an LP preamble is detected (even within an ongoing low-power reservation periods), and keeping the rest of the MAC the same. An actual implementation of a particular variant of the MAC would crucially depend on the design requirements for different devices and is out of scope of this paper.

Can Weeble MAC work with variable channel widths? Our current preamble detector design assumes that all nodes use the same channel width (e.g. one TV channel). This can be generalized to a scenario where channel bonding is allowed, similarly to the way a green-field 802.11g link can detect a legacy 802.11a/g link. We leave the further exploration for future work.

9. CONCLUSIONS

In this paper we have presented a design, implementation and evaluation of a fully decentralized Weeble MAC for coexistence between low and high-power nodes. It avoids starvation of low-power links, and it only slightly drops the efficiency of the network. The main components of our design are the adaptive preamble mechanism and the low-power reservation protocol. We implement and evaluate our design in a software-designed radio test-bed and in Qualnet simulator and we show that it avoids starvation in all cases with only a slight drop in efficient, unlike the existing distributed MACs for white-spaces.

10. REFERENCES

[1] P. Bahl, R. Chandra, T. Moscibroda, R. Murty, and M. Welsh. White space networking with wi-fi like connectivity. In *Sigcomm*, 2009.

[2] I. Canada. Consultation on a policy and technical framework for the use of non-broadcasting applications in the television broadcasting bands below 698 mhz, Aug. 2011.

[3] H.-S. Chen and W. Gao. Mac and phy proposal for 802.11af, 2010.

[4] R. R. Choudhury, X. Yang, R. Ramanathan, and N. Vaidya. Using directional antennas for medium access control in ad hoc networks. In *Mobicom*, 2002.

[5] M. Durvy and P. Thiran. A packing approach to compare slotted and non-slotted medium access control. In *Infocom*, 2006.

[6] FCC. Second order, FCC 10-174. 2010.

[7] FCC. Order, FCC 11-131. 2011.

[8] S. Gollakota, F. Adib, D. Katabi, and S. Seshan. Clearing the rf smog: Making 802.11 robust to cross-technology interference. In *Sigcomm*, 2011.

[9] S. Gollakota and D. Katabi. Zigzag decoding: combating hidden terminals in wireless networks. In *Sigcomm*, 2008.

[10] D. Gunawardena and B. Radunović. Colombo SDK - simulating the innards of a wireless MAC. In *Mobicom Demo*, 2011.

[11] Z. Haas and J. Deng. Dual busy tone multiple access (DBTMA): A multiple access control scheme for ad hoc networks. *IEEE Trans. on Comm.*, 50(6):975–985, 2002.

[12] S. Hong and S. Katti. DOF: A local wireless information plane. In *Sigcomm*, 2011.

[13] J. Huang, G. Xing, G. Zhou, and R. Zhou. Beyond co-existence: Exploiting wifi white space for zigbee performance assurance. In *ICNP*, 2010.

[14] IEEE. IEEE 802.19 wireless coexistence working group.

[15] IEEE. Amendment 5: Spectrum and transmit power management extensions in the 5 ghz band in europe, 2003.

[16] IEEE. Ieee 802.22 wireless regional area networks, 2010.

[17] R. Jain, D. M. Chiu, and W. Hawe. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. In *DEC TR-301*, 1984.

[18] K. Jamieson, B. Hull, A. Miu, and H. Balakrishnan. Understanding the real-world performance of carrier sense. In *EWIND*, 2005.

[19] E.-S. Jung and N. Vaidya. A Power Control MAC Protocol for Ad Hoc Networks. In *MOBICOM*, 2002.

[20] S. M. Kay. *Fundamentals of Statistical Signal Processing, Volume 2: Detection Theory*. Prentice Hall, 1998.

[21] T.-S. Kim, J. Hou, and H. Lim. Improving spatial reuse through tuning transmit power, carrier sense threshold, and data rate in multihop wireless networks. In *MOBICOM*, 2006.

[22] K. Kyouwoong, I. Akbar, K. Bae, J. Urn, C. Spooner, and J. Reed. Cyclostationary approaches to signal detection and classification in cognitive radio. In *DySpan*, 2007.

[23] LAN/MAN Standards Committee. Part 11: Wireless LAN MAC and PHY specifications, 2003.

[24] X. Liu, A. Sheth, M. Kaminsky, K. Papagiannaki, S. Seshan, and P. Steenkiste. Pushing the envelope of indoor wireless spatial reuse using directional access points and clients. In *Mobicom*, 2010.

[25] Lyrtech. Lyrtech small form factor software defined radio.

[26] E. Magistretti, K. Chintalapudi, B. Radunovic, and R. Ramjee. WiFi-Nano:reclaiming wifi efficiency through 800 ns slots. In *Mobicom*, 2011.

[27] S. M. Mishra, R. W. Brodersen, S. ten Brink, and R. Mahadevappa. Detect and avoid: An ultra-wideband/WiMAX coexistence mechanism. *IEEE Communications Magazine*, 45(6):68–75, 2007.

[28] MocoNews.net. Uk gives green light to fast wireless broadband service, free to all comers, Sept. 2011.

[29] S. Nagaraj, S. Khan, C. Schlegel, and M. Burnashev. Differential preamble detection in packet-based wireless networks. *IEEE Transactions on Wireless Communications*, 8(2):599–607, 2009.

[30] B. Radunovic and J.-Y. Le Boudec. Rate performance objectives of multi-hop wireless networks. In *Infocom*, 2004.

[31] H. Rahul, N. Kushman, D. Katabi, C. Sodini, and F. Edalat. Learning to share: Narrowband-friendly wideband networks. In *Sigcomm*, 2008.

[32] T. Schmidl and D. Cox. Robust frequency and timing synchronization for OFDM. *IEEE Trans. on Comm.*, 45(12), 1997.

[33] S. Sen, R. R. Choudhury, and S. Nelakuditi. CSMA/CN: Carrier sense multiple access with collision notification. In *Mobicom*, 2010.

[34] F. Times. Microsoft trial to use uk tv signals for wifi, June 2011.

[35] F. Tufvesson, O. Edfors, and M. Faulkner. Time and frequency synchronization for OFDM using PN-sequence preambles. In *VTC*, 1999.

[36] L. Yang, W. Hou, C. Cao, B. Zhao, and H. Zheng. Supporting demanding wireless applications with frequency-agile radios. In *USENIX NSDI*, 2010.

[37] X. Yang and N. Vaidya. On the physical carrier sense in wireless ad hoc networks. In *INFOCOM*, 2005.

[38] R. Zhou, Y. Xiong, G. Xing, L. Sun, and J. Ma. Zifi: Wireless lan discovery via zigbee interference signatures. In *MobiCom*, 2010.