# WiFi-Nano : Reclaiming WiFi Efficiency through 800 ns slots

Eugenio Magistretti *
Rice University
Texas, USA
emagistretti@rice.edu

Krishna Kant Chintalapudi
Microsoft Research India
Bangalore, India
krchinta@microsoft.com

Bozidar Radunovic
Microsoft Research
Cambridge
Cambridge, U.K
bozidar@microsoft.com

Ramachandran Ramjee
Microsoft Research India
Bangalore, India
ramjee@microsoft.com

## ABSTRACT

The increase in WiFi physical layer transmission speeds from 1 Mbps to 1 Gbps has reduced transmission times for a 1500 byte packet from 12 $ms$ to 12 $\mu s$. However, WiFi MAC overheads such as channel access and acks have not seen similar reductions and cumulatively contribute about 150 $\mu s$ on average per packet. Thus, the efficiency of WiFi has deteriorated from over 80% at 1 Mbps to under 10% at 1 Gbps.

In this paper, we propose WiFi-Nano, a system that uses *800 $ns$ slots* to significantly improve WiFi efficiency. Reducing slot time from 9 $\mu s$ to 800 $ns$ makes backoffs efficient, but clear channel assessment can no longer be completed in one slot since preamble detection can now take multiple slots. Instead of waiting for multiple slots for detecting preambles, nodes speculatively transmit preambles as their backoff counters expire, while continuing to detect premables using self-interference cancellation. Upon detection of preambles from other transmitters, nodes simply abort their own preamble transmissions, thereby allowing the earliest transmitter to succeed. Further, receivers speculatively transmit their ack preambles at the end of packet reception, thereby reducing ack overhead. We validate the effectiveness of WiFi-Nano through implementation on an FPGA-based software defined radio platform, and through extensive simulations, demonstrate efficiency gains of up to 100%.

## Categories and Subject Descriptors

C.2.5 [**Computer Systems Organization**]: COMPUTER - COMMUNICATION NETWORKS—*Local and Wide-Area Networks*; C.2.2 [**Computer Systems Organization**]: COMPUTER - COMMUNICATION NETWORKS—*Network Protocols*

*This work was done while the author was an intern at Microsoft Research, India


*MobiCom'11,* September 19–23, 2011, Las Vegas, Nevada, USA.


## General Terms

Design,Experimentation

## 1. INTRODUCTION

While WiFi physical layer (PHY) data rates have increased from 1 Mbps in the original 802.11 to 1 Gbps in the upcoming 802.11ac standard, user level throughputs have not seen a commensurate increase. A key reason for this is the *channel access overhead*. As we show in Section 2, the average channel access delay is 16 $\mu s$ + $9.5 * slottime$. Given slots are 9 $\mu s$ in 802.11a/n [4], the average channel access delay is 101.5 $\mu s$. Thus, while the transmission time for a 1500 byte packet is only 20 $\mu s$ at 600 Mbps, the average channel access overhead is over 500% of the packet transmission time.

Given that slot duration plays a crucial role in WiFi's inefficiency, can we use a slot smaller than 9 $\mu s$? Consider a typical backoff slot. When the backoff counter expires at one node, the node starts transmitting its packet. Preambles in 802.11 are transmitted at the beginning of each packet and contain predefined sequences that help the receiver detect the packet reliably. As we discuss in Section 3, the 9 $\mu s$ slot is designed to accommodate 4 $\mu s$ needed for packet detection/clear channel assessment (CCA) [18] and about 5 $\mu s$ of turnaround time. The turnaround time is designed to accommodate propagation delay, processing and time for switching the radio from receive to transmit. In order to reduce the slot duration to below 9 $\mu s$, one would have to either reduce the time for turnaround or the CCA time. However, in the former case, after CCA determines that the carrier is idle, nodes may be unable to transmit on the next slot given lack of sufficient turnaround time; in the latter case, collisions would increase due to missed detection of busy carrier.

While the above observations argue that the *802.11a/n slot size of 9 $\mu s$ is close to the minimum feasible value*, it is based on one key assumption – preamble transmission and detection is serial, i.e, one device transmits a preamble at any given time while others are performing CCA. Instead, if preamble transmission and detection could be done in parallel, i.e., if devices could detect preambles that are being transmitted from other devices while simultaneously transmitting their own preambles/packets, then 9 $\mu s$ slots are superfluous and slot sizes can be significantly reduced. Based on this crucial observation, our solution to reducing the channel access overhead is a novel PHY/MAC design, *WiFi-Nano, with slots as small as 800 $ns$*.

In WiFi-Nano (Section 3), all transmitters *speculatively transmit their preambles* in the slot where their backoff counters expire. Since preamble detection (CCA) can now take multiple slots, devices continue to detect preambles even while transmitting their own preambles – nodes accomplish this by using analogue self-interference cancellation [14], which allows nodes to remove the effect of their own transmission before performing preamble detection on the received signal. If a device detects a preamble from another device before it finishes transmitting its own preamble, then it aborts its transmission immediately since this implies that the other device had initiated its transmission earlier. Thus, average channel access time can be reduced to 7.6 $\mu s$, an order of magnitude lower than 101.5 $\mu s$ in WiFi.

Apart from channel access overhead, speculative preambles can also be used to reduce the acknowledgement (ack) overhead in WiFi. While an 802.11n ack packet consumes negligible transmission time at 600 Mbps, a 40 $\mu s$ preamble coupled with 16 $\mu s$ SIFS implies that the ack overhead is 2.8X the transmission time of a 1500 byte packet. In WiFi-Nano, instead of waiting for SIFS before transmitting the ack preamble, the receiver speculatively starts transmitting its ack preamble as soon as it finishes reception of the packet. While the preamble is being transmitted, the node finishes decoding the packet. Upon detecting error in reception, the receiver simply aborts its ack transmission. This *speculative ack preamble transmission* allows WiFi-Nano to eliminate SIFS and, thus, reduce the ack overhead.

In order to enable preamble detection when several nodes may be transmitting speculatively, we design and implement a novel preamble detector - the *lattice correlator* which is capable of detecting sub-parts of a pseudo-random preamble. The lattice correlator also allows the receiver to accurately estimate the transmission start time of any detected preamble. This estimate is necessary so that the detecting node can roll back its backoff counter to the estimated start time of the earlier transmission, a crucial requirement for preserving fairness in channel access.

Finally, there are two beneficial side-effects of performing preamble detection in parallel with preamble transmission. First, packet collisions due to identical choice of backoff counters by multiple nodes can be almost eliminated. If two or more devices initiate their preambles in the same slot, these devices will be able to detect the start time of each other's preamble and deduce that collision is imminent. In such a case, they probabilistically abort the transmission of their preambles such that, with high likelihood, only one node remains during the transmission of the payload of the packet. Second, since backoff counter collisions are unlikely in WiFi-Nano, unfairness caused due to capture-effect between near-and-far terminals [8] is also eliminated.

Through real experiments (Section 5) conducted using an FPGA-based software defined radio platform, as well as extensive simulations (Section 6), we demonstrate the effectiveness of WiFi-Nano in improving the efficiency of WiFi.

The realization of WiFi-Nano requires analogue self-interference cancellation [14] and the abilty to transmit and receive simultaneously. The former is an inexpensive noise canceller circuit, while the latter requires an extra oscillator and antenna. Another overhead that WiFi-Nano imposes is the need for a longer preamble detection time compared to WiFi. This is necessary to enable detection of the preamble in the face of interference due to other speculative preamble transmissions. Surprisingly, we find that the preamble detection time for WiFi-Nano turns out to be only about 4 $\mu s$ longer than in WiFi. The reason for this small overhead is due to an avalanche effect that gets triggered during channel access – when several devices speculatively transmit, nodes in their vicinity detect these high SNR preambles and immediately abort their respective transmissions, thereby quickly reducing overall interference. Thus, we believe that the costs of WiFi-Nano are small compared to its benefits.

In summary, we make the following contributions:

- The design and implementation of WiFi-Nano, a novel PHY/MAC design that improves the throughput of WiFi by up to 100% using the following techniques.
- The use of 800 $ns$ slots enabled through the use of speculative preambles in order to reduce channel access overhead.
- The transmission of ack preamble immediately at the end of the received packet to reduce ack overhead.
- The technique for sub-preamble detection and its realization using a lattice correlator to effect appropriate rollback of backoff counters.

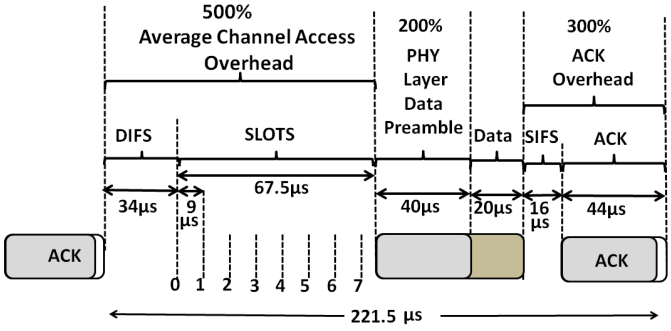


**Figure 1: Overheads in 802.11 at 600 Mbps**

## 2. MOTIVATION

In this section we motivate WiFi-Nano by analyzing the overheads present in 802.11. We start by examining the overheads for a single device transmitting 1500 byte data packets back-to-back using 802.11n at 600 Mbps using the Distributed Coordination Function (DCF) with RTS/ CTS turned off – a common choice in most deployments today. Transmission time for a 1500 byte (maximum MTU allowed by Ethernet) packet at 600 Mbps is 20 $\mu s$. This is accompanied by three key overheads – *channel access*, *data preamble* and *acknowledgement* overheads, as shown in Figure 1.

**Channel Access Overhead :** As dictated by CSMA in 802.11, prior to transmitting its next packet, the device must first sense that the channel is idle for the duration of DIFS. DIFS, which is 34 $\mu s$ long, comprises SIFS (16 $\mu s$) and 2 slots (each 9 $\mu s$). After DIFS, a node must typically defer its transmission for a random number of slots, generated from 0 to CW-1 (contention window size) to allow other devices to share the channel in a fair manner. Given that the minimum value of CW as dictated by 802.11 is 16, the device will, on an average, wait about 7.5 slots before transmission. Thus, the average *overhead for channel access amounts to 16 $\mu s$ + 9.5*slot-time,* i.e., *101.5 $\mu s$, and is independent of data rate (Figure 1). At 600 Mbps, it is about 500% of data transmission time.*

**Data Preamble Overhead :** The transmission of data in every packet is preceded by a physical layer preamble. The preamble is crucial in preparing the receiver for a successful reception. First, it helps the receiver to reliably establish that a packet is being transmitted (*packet detection*) and detect the boundaries of various parts of the packet (*synchronization*) to enable decoding. Second it helps

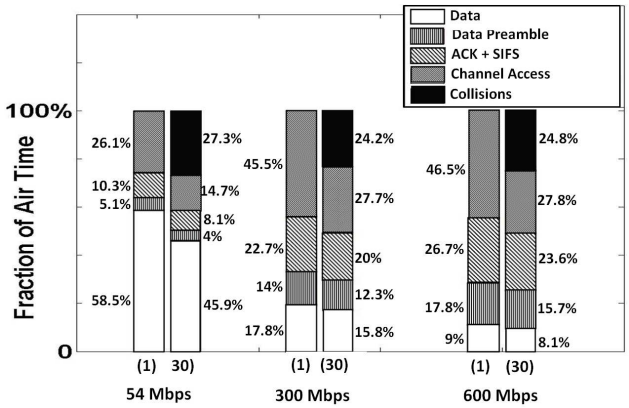


**Figure 2: Overheads at different data rates**

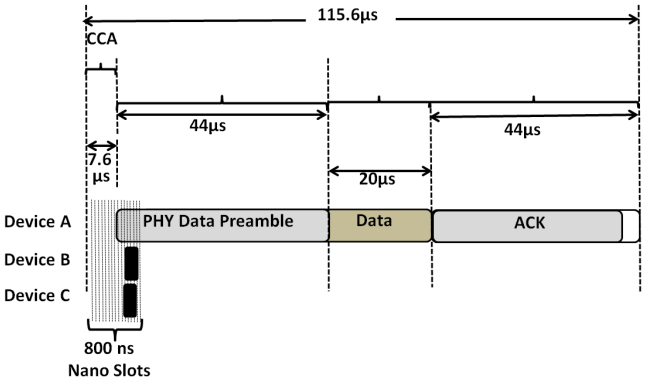


**Figure 3: WiFiNano transmissions at 600Mbps**

in *channel estimation i.e.,* helps combat the vagaries of the wireless environment by providing sufficient information to allow the receiver to estimate and correct for the channel characteristics. Third, for 802.11n MIMO receptions, it helps the receiver to estimate the MIMO parameters required to allow leveraging the spatial orthogonality of multiple streams. Thus, while 802.11a/g preambles are 20 $\mu s$ long (including PLCP header), for 4x4 802.11n, preambles are 40 $\mu s$ long – 200% of packet transmission time (Figure 1).

**Acknowledgement Overhead :** Upon the successful reception of a packet, the receiver responds with an ACK. In order to allow enough time for the receiver to process incoming data and prepare its radio for transmission, nodes must wait for SIFS (16$\mu s$) before transmitting an ACK. The ACK content itself comprises only 14 bytes and should take only 18 $ns$ at 600Mbps. However, since 802.11n uses 4$\mu$ symbols, all packets transmission durations must span multiples of 4$\mu$s. Thus, 802.11 pads the ACK packet with zeros to make it a 4$\mu$s worth of data. Further, the ACK also includes a preamble that is 40 $\mu s$. Thus, as depicted in Figure 1, SIFS and ACK together span 60$\mu$s – 300% of packet transmission time.

So far we have described overheads in the context of a single device. When several devices contend for the channel, packet collisions occur resulting in *collisions overhead.*

**Collision Overhead :** When multiple devices contend, their backoff counters are decremented independently and in parallel. The wait time for accessing the channel is thus determined by the device with the minimum backoff counter value. As a result, overhead (idle time) due to channel access reduces as the number of contending devices increase. However, with increasing contention, the probability that two or more devices may choose to transmit in the same slot increases, leading to increased collisions. Each collision then results in a wasted time overhead equal to the sum of data transmission time, data preamble time and acknowledgement transmission time.

Figure 2 depicts the fraction of air time spent on various overheads for a 1500 byte data transmission at three different data rates in two extreme scenarios[1]. First, with a single transmitter having unequivocal access to the channel and second, with 30 transmitters all randomly located within interference range of each other, representing a dense deployment environment. As data rates increase from 54 Mbps to 600 Mbps, the fraction of data transmission time and hence the efficiency of the system, reduces from about 58% to 9%. The data preamble and ACK overheads together increase form 15% at 54Mbps to 44% at 600Mbps. Finally, channel access and collision overheads together amount to 26% (42% for 30 transmitters) to as high as 45% (52% for 30 transmitters).

Given that preamble and ACK are indispensable, these overheads are not amenable to significant reductions. *The primary focus of this paper is thus to reduce channel access and collision overheads and improve the efficiency of WiFi.*

# 3. WIFI-NANO OVERVIEW

Figure 3 depicts the time-line of three WiFi-Nano transmitters A, B and C operating at 600 Mbps. Comparing Figures 3 (WiFi-Nano) and 1 (WiFi) there are three key differences.

- *800 $ns$ Slots :* Instead of using 9$\mu$sec slots, WiFi-Nano uses 800ns slots – an order of magnitude reduction in slot duration.
- *Speculative Preamble Transmission :* Devices begin speculatively transmitting their preambles when their respective backoff counters expire, even before channel access contention has been resolved. Contention for channel access is carried out simultaneously while preambles are being transmitted, aided by analogue self-interference cancellation [14]. All devices, except the ones whose backoff counters expired the earliest, abort their transmissions mid-way (devices B and C in Figure 3).
- *Speculative ACK :* WiFi-Nano devices eliminate the need for SIFS by speculatively transmitting the preamble even as the received packet is been processed. The ACK transmission is then aborted mid-way upon detecting errors in the received packet.

As a result of these changes, WiFi-Nano *dramatically reduces channel access delays* given that they are a function of SIFS and slot duration (Section 2). Further, as carrier sensing is carried out while transmitting preambles, devices headed for a packet collision detect this condition with high probability and resolve their contention probabilistically as described in Section 4, resulting in a *near zero collision probability*. Finally, speculative acks bring about a reduction in ACK overheads by up to 35%. In the rest of the section we describe the key ideas and innovations in the design of WiFi-Nano. In order to provide the necessary background, we start by describing why WiFi requires 9$\mu$sec slots.

**Why WiFi uses 9$\mu$ slots**

As described in 802.11 standard, the need for using 9$\mu$sec slots stems from four key delays - *time to carrier sense, Rx-Tx switching time, speed of light propagation* and *MAC processing delay*. While 802.11 recommends nominal values for each of these delays, individual manufacturers are free to choose these delays based on their specific hardware capabilities and constraints. However, the sum

[1]These are results obtained using the Qualnet network simulator.

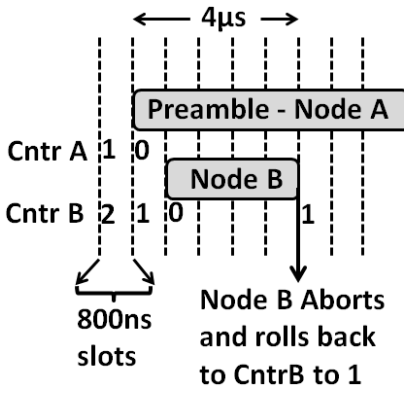


**Figure 4: Making slot duration independent of carrier sensing time**

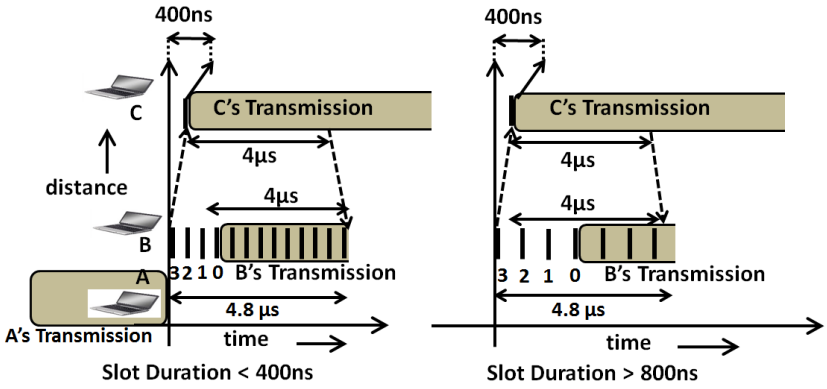


**Figure 5: Design of slot width in WiFi-Nano**

of these delays must ensure a slot duration of 9$\mu$sec.

*Time to Carrier Sense :* Before initiating its own transmission in the current slot, a device must reliably establish that no other transmission was initiated in the past slot. A failure to do so will result in packet collision. As described in Section 4, given the vagaries of noise and interference, the time required to ascertain the presence/absence of an ongoing transmission depends on the signal to interference ratio (SINR) at the receiver. 802.11 recommends 4$\mu$sec to enable reliable carrier sensing from the farthest nodes in the network.

*Rx-Tx Switching Time :* Given that WiFi devices are not required to allow simultaneous transmission and reception, several components *e.g.*,antenna and RF oscillator are shared between the transmission and reception circuits. Thus, devices require time to switch from reception to transmission mode in order to reset these shared components. 802.11 recommends 600ns as the switching time.

*Speed of light propagation :* RF waves travel a distance of 100m in approximately 330ns. Thus, if two devices are 100m apart, then their carrier sensing and notion of slot boundaries may be 330ns apart. Slots must accommodate these delay effects due to propagation delays. 802.11 recommends a value of 800ns to accommodate for speed of light.

*MAC Processing Delay :* Each signal from PHY layer needs to be processed by the MAC layer and then the MAC must issue signals to the PHY. This turn around time depends on specific hardware implementations.

**Speculative Preambles**

WiFi-Nano removes the dependency of slot duration on carrier sense time by requiring that devices should be able to carrier sense while speculatively transmitting data preambles. To illustrate this idea we consider a simple example that uses 800ns slots in Figure 4. Two WiFi-Nano devices A and B contend for the same channel. For the sake of simplicity, in this example, we assume that 4$\mu$sec are required by the devices to detect each other's transmissions. As seen from Figure 4, device A finishes counting down its backoff counter and initiates its transmission before B. Device B finishes its countdown one slot (800ns) after A. Since B requires 4$\mu$sec to detect an ongoing transmission, it is unable detect A's transmission at this time. Instead of waiting however, Node B speculatively initiates its own transmission. Given the capability that B can carrier sense while transmitting, it eventually detects A's transmission four slots later. Upon carrier sensing A's transmission, B realizes that the other node (A) started transmitting earlier than itself, since B started transmitting less than 4$\mu$sec ago (using the lattice correlator described in Section 4). Consequently, B concludes that it cannot be the rightful owner of the medium and aborts its transmission mid-way. Device A, on the other hand, continues its own transmission uninterrupted. *Thus, by enabling carrier sensing while transmitting and using speculative transmission of preambles, channel contention can be performed simultaneously during preamble transmissions, thereby masking channel access overheads under the necessary preamble overheads. Further, slot duration is no longer constrained to be less than the carrier sensing duration.*

In order to enable carrier sensing while transmitting preambles, WiFi-Nano leverages self-interference cancellation [14]. To avoid the device's transmissions from overwhelming its receiver, the transmitted signal from the transmitting antenna is subtracted from the received signal at the receiver antenna, thus mitigating the interference due to the devices' own transmissions. To increase robustness of detection, WiFi-Nano allows a longer carrier sensing time without affecting slot duration, and hence the efficiency, significantly.

**Backoff Counter Roll Back**

Continuing the example depicted in Figure 4, both devices A and B speculatively initiated their transmissions hoping to gain access of the channel after their backoff counters had reached 0. Thus, even though B rightfully aborts its transmission, it is in a position of advantage with respect to its backoff counter value. This is because if B had not experienced any delay in carrier sensing, it would have suspended its backoff counter at 1 as soon as A initiated its transmission. Hence, *in order to preserve fair access, aborting devices in WiFi-Nano roll back their backoff counters to time of initiation of the transmission that wins the contention.* The exact implementation of the roll-back mechanism will be described in greater detail in Section 4.

**How small can WiFi-Nano Slots be?**

Requiring the devices to perform carrier sensing while transmitting necessitates the completely independent operation of the the transmitter and the receiver. This requirement inherently eliminates the need for Rx-Tx switching delay, as the receiver and transmitter are simultaneously operating at any given time. Further, since MAC processing overheads only result in delayed speculative transmissions that may be aborted, their inclusion in slot duration is no longer crucial. Consequently, the only remaining contributor to slot duration is propagation delay.

How does propagation delay affect the duration of a slot? As de-

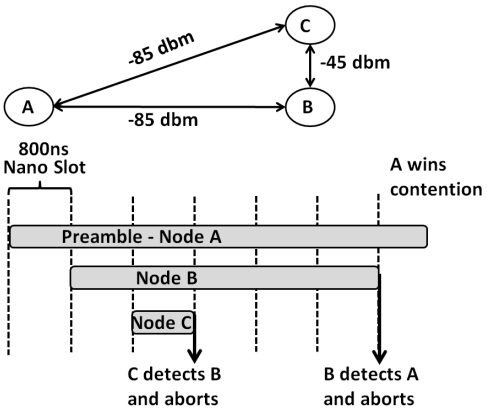


**Figure 6: The Near Far Problem**

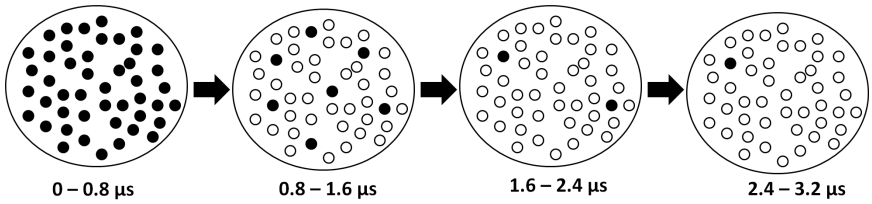


**Figure 7: Chained Contention Resolution Example in WiFi-Nano**

scribed earlier in this section, in order to preserve fairness, devices must roll back their backoff counters to the time of initiation of the transmission that grabs the channel. Propagation delays cause incorrect estimation of this time. *If slot durations are less than twice the maximum propagation delay of the network, these errors lead to incorrect rollbacks, resulting in unfairness.*

We illustrate this using a simple example. As depicted in Figure 5, two of three devices, A and B, are located very close to each other so that the propagation delay from A to B is close to zero. Device C however, is located at the edge of the network with a propagation delay of 400ns from either A or B. Consequently, after A ends its transmission, while B realizes immediately that the channel is idle and starts counting down, C realizes this only after 400ns. Suppose node C has a countdown value of 0 so that it starts to transmit immediately at $t = 400ns$. This transmission will reach B after another 400ns and hence B will start receiving C's transmission at $t = 800$ns. Consequently, when B detects C's transmission it will assume that C started at $t = 800ns$ and roll its counter back to a time corresponding to $t = 800ns$. As depicted in Figure 5, this error has no effect when the slot duration is greater than 800ns *i.e.,* twice the propagation delay. Since we are targeting an indoor AP-based setting with a range of 100 m (propagation delay of 333 ns), we use a slot width of 800 ns in WiFi-Nano. Note that, even with 800 ns slots, WiFi-Nano will continue to work for networks with larger ranges, albeit with some unfairness to nodes farther away.

**Chained Contention Resolution in WiFi-Nano**

Consider an example with three devices A, B and C as depicted in Figure 6. Devices B and C are proximate to each other, however device A is far from both B and C. The RSS of transmissions between links A – B and A – C is -85dBm, while over links B – C it is -45dBm. Further, suppose that device A started to speculatively transmit first, followed by devices B and then C. A's transmission at B will be overwhelmed by the transmission from C and will have an SINR of -40 dB (-85dBm -(-45 dBm)); the same will happen at C. Suppose an SNR of 10dB is required to reliably detect A's transmission. Then, in order for nodes B or C to detect A reliably, the preamble transmitted by A must have processing gain of about 50dB. As described in Section 4, the preamble length for achieving a processing gain of 50dB is 100000, which translates to a preamble that is 5 ms long – an impractically long time. However, such a long preamble from A is necessary only if B and C are continuously transmitting. However, since B and C are very close to each other, aided by self-interference cancellation, C receives B with a very high SINR. Thus, C can detect B in a very short time and abort its transmission. Once C aborts its own transmission, B only needs 10dB of processing gain for detecting A's preamble, and this enables B to detect A and also abort.

In a dense deployment scenario, where several devices contend for channel access, many of those devices will initiate their transmissions speculatively, leading to a high interference environment. However, as proximate devices detect each other and abort, farther devices will become amenable for detection. Thus, as time progresses, more and more devices will abort their transmissions in a chain reaction like manner, expanding in geographical extent. Figure 7 considers an extreme illustrative example starting at some instant when all nodes are speculatively transmitting their preambles. In Figure 7, dark colored nodes are devices that are transmitting preambles. After the end of 800 ns, the closest nodes detect each other's preambles and abort their transmissions. This allows preamble transmissions from farther nodes to be detected during the next 800 ns. Finally, only one node, the node with the earliest transmission start time remains.

As seen from Figure 7 and evaluated in Section 6, the inherent parallelism of chained contention resolution in WiFi-Nano is extremely quick to resolve all contentions in the network. This however, also means that WiFi-Nano may require larger preambles than WiFi.

**Speculative ACK**

WiFi-Nano also eliminates the need for SIFS between data and ack transmissions. According to the 802.11 standard, SIFS of 16 $\mu s$ is designed to accommodate delays such as transferring the received signal from the antenna, MAC processing delay, and time to switch from receive to transmit mode. Since WiFi-Nano nodes have separate transmit and receive paths, the receiver can simply start speculatively transmitting the ack preamble as soon as reception is complete. In parallel, the node decodes the received packet and computes the CRC to check if there are any errors. Upon detection of any error, the node simply aborts the transmission of the ack preamble. Since preamble length at even 802.11a rates is 20 $\mu s$, there is ample time for processing and aborting acks. Finally, note that speculatively transmitting acks does not require full-duplex capability since there is no transmission during reception; we simply overlap the processing of already received data during transmission.

# 4. WIFI-NANO DESIGN DETAILS

In this section we discuss the various components that allowed for the implementation of WiFi-Nano.

As described in Section 3, for chained contention resolution, devices must abort their transmission as soon as possible. To achieve this, WiFi-Nano leverages the fact that the stronger the received transmission, the faster it can be carrier sensed. Consequently, while proximate devices create a strong interference, they also abort quickly, allowing transmissions from weaker devices to be detected. Further, in order to perform a correct backoff roll-back, the aborting device must also be able to accurately estimate the time of initiation of the transmission. WiFi-Nano uses a novel carrier sensing technique - *sub-preamble lattice correlation* - which allows detection of continuous sub-parts of the preamble. In this section, we first start by describing pseudo-random sequence based carrier sensing in order to provide necessary background for the general reader.

**Carrier Sensing Using Pseudo Random Sequences**

Wrongfully concluding an ongoing transmission when there is none, *i.e.,* false alarm, leads to loss of throughput as devices defer transmitting needlessly. On the other hand, missing legitimate transmission from devices that may be far away as their received signal strength (RSS) is weak, may lead to collisions. In order to allow reliable detection, 802.11 standards transmit pseudo-random noise (PN) sequences in the initial part of the preamble. A receiver then detects an ongoing transmission by correlating the received signal with the pseudo-random noise (PN) sequence.

The key advantage of a PN sequence is the sharp distinct peak that it provides exactly when the input signal to the correlator matches the PN sequence. In practice however, even though the transmitter transmits the exact PN sequence expected by the receiver, the received signal is affected by the wireless channel and seldom remains exactly the same. Consequently, the correlation of the PN sequence affected by the wireless channel, $PN_{rec}$ may be poor.

*Schmidl and Cox Correlator [16] :* In order to combat the effects of the wireless channel, a more robust scheme by Schmidl [16] *et al.*transmits two or more copies of the same PN sequence (802.11a/n uses ten copies of the same 800ns sequences). Since each transmitted PN sequence copy is affected by the wireless channel the same way, the individual received copies at the receiver still remain identical. Thus, instead of correlating against the original PN sequence, received copies are correlated with each other. The key disadvantage of this scheme is that correlation peak is not as sharp as the standard PN sequence correlator.

*Tufvesson Correlator [19] :* In our implementation, we make use of a hybrid scheme that combines the advantages of the the standard PN sequence correlation and Schmidtl and Cox correlator. In this scheme, each copy is correlated with the known PN sequence and then outputs of the copies are correlated with each other to obtain a sharp and robust correlation spike. Figure 8 depicts the output of a Tufvesson correlator as a function of time to an incoming 3.2$\mu$s pseudo-random preamble with two identical 1.6$\mu$ copies. As seen in Figure 8, the correlator provides a sharp correlation spike 3.2 $\mu$s after the beginning of the reception of the preamble. *Thus, the time of transmission initiation can also be determined since the spike always occurs at the end of the reception of the preamble.*

**Performance of PN based schemes**

The height of the correlation spike over the noise floor determines the reliability of detection. The height of the spike depends on the Received Signal Strength (RSS) of the preamble and the length of the preamble as $RSS \times \Delta$, where $\Delta$ is called the *processing gain* of the preamble. The processing gain of a preamble increases linearly with the length $L$ of the sequence. For example, at 20MHz since each sample is 50ns long, a 4$\mu$s long pseudo-random sequence will have an effective length of 80 (4000ns/50ns), *i.e.,* a processing gain of $10\log(80) = 19dB$. If the received SNR of the preamble is 0dB, then the correlation spike of the pseudo-random sequence after correlation seen at the receiver will be 19dB over the noise floor, allowing a very reliable detection.

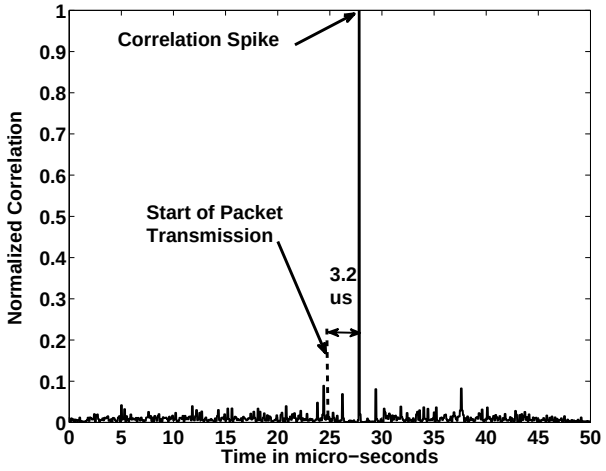


**Figure 8: Carrier Sensing Using Pseudo-Random Preamble**

**The Lattice Correlator**

In order to enable chained contention resolution in WiFiNano, the correlator must provide two functions. First, devices are required to correlate sub-parts of a preamble. For example, in Figure 6, after C aborts, B should be able to correlate only on the part of A's preamble that was received after B aborted. In general when several nodes contend for the channel, the correlation may be required to be performed on different length of contiguous parts of the preamble. Further, detection of sub-parts of the preamble allows the earliest possible detection and hence allows aborting the transmission at the earliest possible time. Second, the need for roll-back requires that the exact position of the correlation be known, since this will help accurately determine the beginning of the packet transmission.

In order to enable both these functionalities, we designed a novel lattice correlator as depicted in Figure 9. Each packet of WiFi-Nano is preceded by a PN sequence comprising several short but distinct 800ns PN sequences PN1, PN2, $\cdots$, PNn. The lattice correlator takes as input the received signal, and generates $\frac{N(N-1)}{2}$, (N is the number of 800ns PN sequences) outputs corresponding to the correlations obtained from each continuous sub-part of the preamble *e.g.,*[PN1, PN2], [PN3, PN4, PN5] *etc.*. Detection of a spike in any of these inputs provides two pieces of information. First, the presence of an ongoing transmission, and second, the start time of the beginning of the reception. The start time of beginning of the packet reception is determined by the position of the last 800ns PN sequence. For example, a spike due to the correlation [PN2, PN3, PN4] indicates that the packet reception started $4 * 800ns = 3.2\mu s$ ago. While stronger transmissions are typically detected in the early stages of the lattice correlator, weaker signals

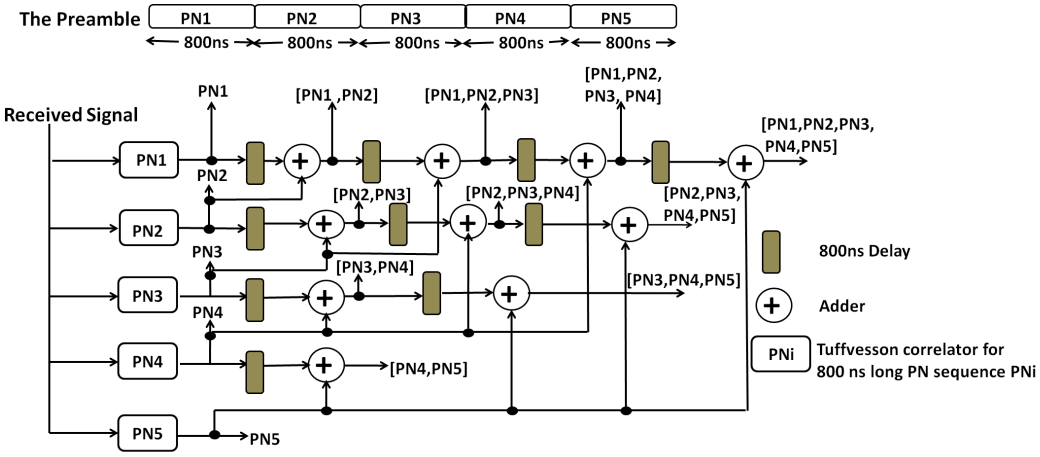


**Figure 9: The Lattice Correlator**

may be detected at later stages.

**Aborting Ongoing Transmissions and Rollback**

Upon receiving a spike from the lattice correlator, the transmitter must first determine whether or not to abort its transmission. For this, the transmitter computes the start time of this reception and aborts its current transmission only if the reception began in an earlier slot or the same slot as itself (in the latter scenario, the abort is executed probabilistically, and is described next). Once the transmitter decides to abort, the transmitter rolls back its backoff counter to a value corresponding to the difference between its own transmission start time and the transmission start time of the received signal. Note that even after a transmitter aborts its transmission, it will still continue to monitor spikes from the lattice correlator since it might have to update its backoff counter to reflect an earlier transmission of a weaker signal that was detected at a later stage of the preamble.

**Probabilistic Collision Resolution**

Since potential collisions can be detected in each 800ns slot, WiFi-Nano uses a novel contention resolution scheme to resolve collisions on the fly. Consider the case of two colliding nodes. Rather than both nodes aborting their transmissions, they could instead continue their transmissions with a probability of 50% after perceiving a potential collision. Thus, in the next 800ns slot, with 50% probability one of B and C wins sole access to the channel. However, with 25% probability, both B and C might abort, in which case, a new contention resolution phase can begin; and with 25% probability, both B and C might decide to continue transmitting. In the latter case, the probabilistic collision resolution process continues again in the next slot. Thus, eventually with a high probability, either both devices abort or only one continues transmitting and obtains sole access to the channel.

Finally, note that when more than two packets collide in a given slot, the number of collisions can be approximately estimated by the number of correlation spikes that occur within a single 800ns slot (this is because the slot boundaries of different nodes are not perfectly aligned due to differences in propagation delays). Upon detecting $k-1$ distinct spikes in a single slot, rather than using 50%, each device continues transmitting with a probability of $\frac{1}{k}$.

Thus, the probabilistic collision resolution mechanism in WiFi-Nano avoids payload collisions with a high probability, thereby significantly reducing the collision overhead seen in Figure 2.

# 5. TESTBED RESULTS

We implemented WiFi-Nano on a DSP/FPGA based software defined radio platform – the SFF SDR from Lyretech Inc. Given that the WiFi-Nano MAC is extremely delay sensitive, requiring operations to be performed at latencies of 100s of ns, the entire MAC layer logic, including the lattice correlator, the logic for aborting transmissions and backoff rollback was implemented on the FPGA. The clock speed of the FPGA was set to 40MHz, giving us a clock cycle of 25ns for performing operations. For self-interference cancellation we used an off-the-shelf RF noise cancellation circuit - the Quellan analogue interference canceler. In this section we evaluate three key aspects of WiFi-Nano – *reliable preamble detection*, *efficiency* and *fairness* through experiments conducted using our implementation.

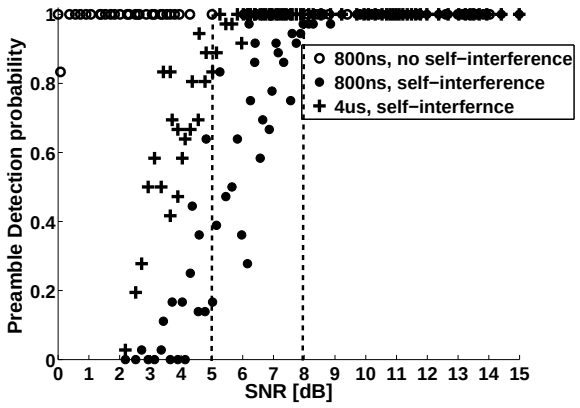


**Figure 10: Preamble detection in WiFi-Nano**

## 5.1 Reliability of Carrier Sensing

Reliable preamble detection is crucial to the performance of WiFi-Nano (Section 3), since missed detections lead to collisions, while false detection alarms (claiming to detect a preamble when there is none) lead to unnecessary backoffs and hence loss of efficiency. In order to evaluate the efficacy of the preamble detection at various SNR values, we conducted the experiment by placing the transmitter at various distances from a receiver. To determine how preamble detection improves with increasing preamble length, at each location we conducted the experiment for preamble lengths of 800ns

and 4$\mu s$. Further, preamble detection must be reliable even when the device uses self-interference cancellation to mitigate interference arising from the transmission of their own preambles. In order to answer the question, "How does self-interference affect preamble detection," the entire experiment was repeated with the receiver transmitting the preambles back-to-back continuously.

Figure 10 depicts the preamble detection probability (PDP) as a function of SNR of received signal. Since noise floor levels at the receiver typically increase when the device is simultaneously transmitting, in order to be fair in comparing the performance with and without self-interference, in each experiment SNR was computed first by setting the receiver to not transmit any packets. Each (PDP,SNR) point in Figure 10 is computed over 4000 packet transmissions. *First, key observation from Figure 10 is by using only 800ns long preambles, the preamble detection probability is close to 100% even at 0dB SNR*. Further, there were no false alarm detections, as the detection threshold was sufficiently high.

If self-interference cancellation were perfect, then preamble detection would not be affected by whether or not the node were transmitting. However, in general this is not true, since self-interference cancellation is never perfect. As seen from Figure 10, PDP of 100% is achieved at around 8dB SNR. Also, as seen from Figure 10, with 4 $\mu s$ preambles, PDP reaches almost 100% at an SNR of 5dB resulting from the increased processing gains. While we believe that other implementations of noise-cancellation such as [1, 14] might provide higher self-interference rejection, we use these experimental results to drive the simulation in Section 6.

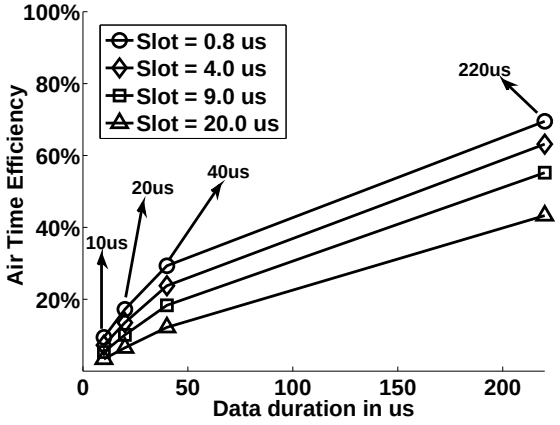


**Figure 11: Airtime efficiency of WiFiNano**

## 5.2 Efficiency of WiFi-Nano

In this experiment we evaluate the air-time efficiency (fraction of time data was transmitted over the air) of WiFi-Nano. The air-time efficiency is a function of two key parameters – i) the fraction of time data is transmitted in each packet and ii) slot duration. Packet aggregation, resulting in a larger packet, increases the fraction of data transmission time in each packet and decreases the impact of preamble and ACK overheads. The usage of smaller slots leads to increased efficiency by reducing channel access overheads. Thus, in this section we quantify the air-time efficiency as a function of these two parameters on our platform as depicted in Figure 11.

In this experiment, a single node with a full packet queue transmitted packets. Data transmission duration in Figure 11 is the time in $\mu s$ that data (excluding preambles and ACKs) is transmitted over the air for each packet. Air-time efficiency is the fraction of time spent in transmitting data. As depicted in Figure 11, for 800ns slots, when 20 $\mu s$ worth of data is transmitted (at 600Mbps this corresponds to a 1500 byte packet), the air-time efficiency is 17.2%. *Note that this is 100% improvement over WiFi*. As the packet sizes are increased to 40$\mu s$ (3kB packet size) and 220$\mu s$ (16kB packet size), the efficiency increases to 30% and 70% respectively.

For larger slot durations, as expected, the efficiency is lower compared to smaller slots. However, even at larger slots such as 9$\mu s$ and 20$\mu s$ (corresponding to WiFi slot durations) performance of WiFi-Nano is marginally better (2% higher) than that of WiFi due to elimination of SIFS.

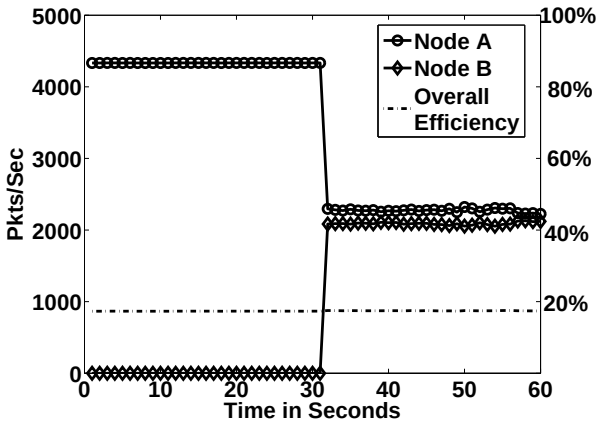


**Figure 12: Fairness in WiFiNano**

## 5.3 Fairness

When multiple devices share the channel, WiFi-Nano must allow fair channel access without sacrificing efficiency. Thus, devices must carrier sense using the lattice correlator, abort their transmissions whenever necessary and rollback their counters appropriately. In order to evaluate fair sharing in WiFi-Nano, we operated two WiFi-Nano devices, Node A and Node B, simultaneously. Throughout the experiments, the queues of both the devices were kept full.

First, only Node A was allowed to transmit in order to establish a baseline. Then, Node B was turned on and both the nodes shared the medium. To emulate the effects of a 600Mbps PHY, for these experiments the packet transmission duration was kept to 20$\mu s$. As seen in Figure 12, when only Node A transmitted, it achieved a rate of about 4300 Pkts/sec corresponding to an air-time efficiency of about 17%, corresponding to WiFi-Nano efficiency as seen in Figure 11. Later, when Node B was enabled, both devices were able to transmit packets about 50% of the time (with an error of about 2.5% from the half-way mark). Moreover, the efficiency mostly remained the same throughout the experiment, indicating that the channel access was performed as intended.

# 6. SIMULATION RESULTS

In this section, we extend our evaluation using simulations to determine the scalability of WiFi-Nano to larger deployments and its robustness to alternative parameter choices. Specifically, *i)* we investigate the choice of the preamble length, and derive values that permit a node to detect any transmitter in the network with high probability; *ii)* we evaluate the benefits of WiFi-Nano in terms of throughput and fairness, as compared to 802.11; *iii)* we investigate the effect of frame aggregation.

## 6.1 Simulation Settings

We implemented WiFi-Nano's preamble detection physical layer in the Qualnet network simulator as an independent module that interfaces below the standard MAC 802.11. The module intercepts packet transmission requests from the MAC layer and performs the preamble contention phase before attempting the transmission. In the simulations, unless specified otherwise, we use a preamble of 8 $\mu s$ for packet detection in WiFi-Nano, compared to 4 $\mu s$ in WiFi. The reason for this choice is explained in further detail in Section 6.2. To simulate the fact that self-interference is not perfect, driven by measurements obtained from our platform, we increased the noise floor by 5 dB whenever a device was transmitting.

We simulate 802.11a and 802.11n modulation rates. While we typically keep the data-rate fixed during a single experiment, we also present results with the auto-rate fallback algorithm for 802.11a. All nodes emit a transmission power of 20 dBm. We choose a path-loss exponent of 2.72 so that the transmission range is 100 m for the 6 Mbps modulation. We simulate a typical LAN setting where multiple nodes communicate with a single access point. In all simulations, nodes are deployed around the access point such that they are able to carrier sense each other (using preamble detection in standard 802.11). Thus, there are no hidden nodes. In the presence of hidden nodes, we believe that WiFi-Nano would suffer packet collisions similar to WiFi; an approach such as CSMA/CN [17] can be used to mitigate the impact of hidden terminals. Unless otherwise specified, the nodes generate fully backlogged CBR traffic, with packet size of 1480 bytes.

By default, WiFi-Nano reacts to preamble collisions using WiFi's binary exponential backoff algorithm. However, we also explore the benefits of using IdleSense [3] for WiFi-Nano when there is high contention. Specifically, we simulate an idealized version of IdleSense with a fixed contention window of 350 when there are 30 active transmitters.

## 6.2 Preamble Length in WiFi-Nano

WiFi recommends about 4 $\mu s$ of its preamble to perform carrier sense. In the rest of this section, we shall refer to this as the *carrier sensing preamble length*. The rest of the WiFi preamble (36 $\mu s$ in 802.11n) is used for performing other functions such as channel estimation, synchronization and MIMO parameter estimation.

As discussed in Section 3, in order to avoid packet collisions while using speculative preamble transmissions, WiFi-Nano devices will require a longer preamble to reliably perform carrier sense compared to WiFi devices. Thus, the carrier sensing preamble length in WiFi-Nano has to be longer than that used in WiFi. *More specifically, the carrier sensing preamble in WiFi-Nano should be long enough to guarantee that all speculatively transmitting devices are able to reliably detect and abort their transmissions, allowing only the earliest device to transmit.* Using a carrier sensing preamble of inadequate length will lead to packet collisions, as more than one device will continue transmitting its packets. Since using a longer preamble adversely affects efficiency, in this section we ask the question, *"What is the shortest possible carrier sensing preamble [2] length that WiFi-Nano can use while ensuring that preambles are correctly detected even in high contention scenarios?"*

[2]Note that the duration of a WiFi-Nano preamble will be the sum of its carrier sensing preamble length and the part of the preamble used in WiFi for other function such as synchronization, channel estimation etc. For example, a WiFi-Nano carrier sensing preamble length of 8$\mu$s would result in a total preamble length of 44$\mu s$ (36$\mu s$ + 8$\mu s$) instead of 40$\mu s$ (36 $\mu s$ + 4$\mu s$) in 802.11n and 24$\mu s$ instead of 20$\mu$s in 802.11a.

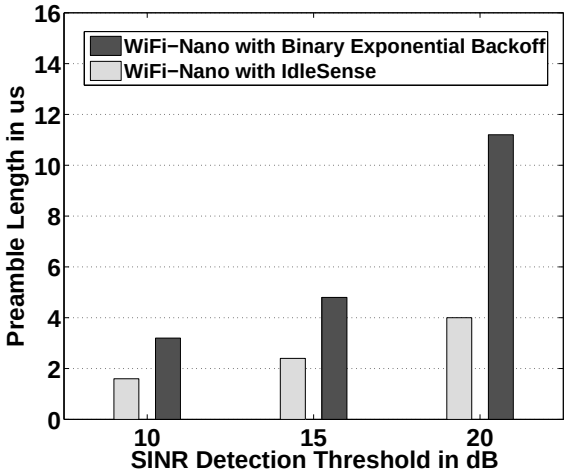


**Figure 13: Preamble length for different detection thresholds**

A key parameter that dictates preamble length is the SINR required for reliable preamble detection without false alarms. While our testbed results indicate that an SINR detection threshold of about 8dB is sufficient (Section 5.1), in noisy environments, higher values of SINR detection threshold may be required. The higher the SINR detection threshold, the longer the carrier sensing preamble length should be. Consequently, in this section we evaluate the carrier sensing preamble length for 10, 15 and 20 dB detection thresholds for considering extremely noisy environments. For each of these detection thresholds, we determine the minimum carrier sensing preamble length that limits the probability of missed preamble detection to under 1%. In our evaluations, in order to be conservative, we consider an extreme scenario for WiFi-Nano – 30 full back-logged transmitters, deployed randomly in a 100 m diameter region. All results were computed as averages over five random topologies, each running for a period of 30s. We tried two backoff schemes – exponential backoff (used by WiFi) and IdleSense.

Figure 13 depicts the minimum carrier sensing preamble length as a function of SINR detection threshold. As seen from Figure 13, when using exponential backoff, a length of 4$\mu s$, the same as that used by WiFi, is sufficient when the SINR detection threshold is 10 dB (corresponding to our testbed results). As SINR detection thresholds are increased to 15dB and 20dB, the required carrier sensing preamble length is 7.2$\mu s$ (an increase of 3.2$\mu s$ over WiFi) and 11.2$\mu s$ (an increase of 7.2 $\mu s$ over WiFi) respectively.

Also notice that IdleSense has a beneficial effect for WiFi-Nano. By limiting the number of nodes contending in a single slot, IdleSense reduces interference and consequently results in a shorter carrier sensing preamble length. In particular, IdleSense is able to maintain the preamble length at 4$\mu$s even in the noisy 20 dB case. In other words, *no change in preamble length will be required in WiFi-Nano while using IdleSense*. For the remaining results, we fix the detection threshold to 15 dB, and conservatively choose a preamble length of 2 OFDM symbols, i.e., 8$\mu s$.

In order to shed better insight as to why the preamble length does not increase dramatically in WiFi-Nano, we investigate the preamble detection process by evaluating how quickly nodes are able to carrier sense and abort their transmissions. For our evaluation, we divide each simulation into epochs – each epoch starts at the end of an ACK transmission for the last packet when the channel becomes idle. In each epoch we count the number of nodes that are transmitting at various increasing time intervals from the start of

the epoch. Average active transmitting nodes as a function of time interval elapsed from the beginning of an epoch is computed by averaging over all epochs in the simulation.

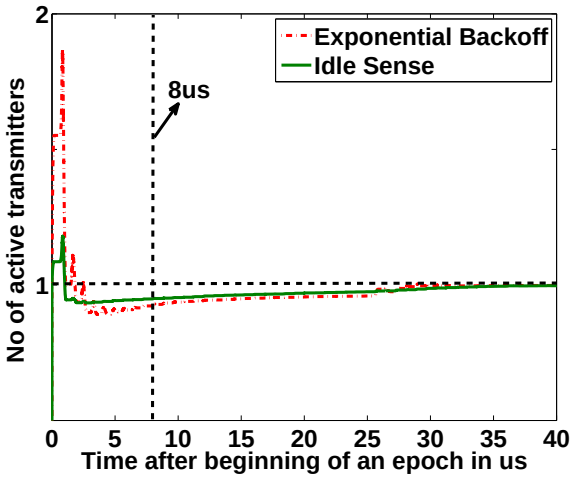


**Figure 14: Number of contending transmitters over time**

Using the same 30 node setting as before, Figure 14 plots the average number of transmitters versus the time displacement after the beginning of the epoch. For instance, a point (1.6,1.8) means that after $1.6\mu$s since the channel became idle, 1.8 nodes on average are transmitting speculative preambles. The channel contention is resolved when the average number of transmitting nodes is equal to 1. As seen in Figure 14, the average number of transmitters decreases rapidly as nodes abort upon detecting other transmitters. After a short period of less than $4\mu$s, the average number of transmitters reduces to slightly below one. The average is below one because colliding transmitters abort probabilistically and, in rare cases, all transmitters may abort.

During the interval between $4\mu$s and $30\mu$s, the curve shifts upward slowly representing a small increase in the average number of transmitters. This occurs because when all colliding transmitters have aborted, some nodes that did not detect these aborted transmissions had their backoff counters expire and thus, they grabbed the idle channel. Finally, the average number of transmitters converges to 1, indicating that only one transmitter grabs the channel and transmits successfully. This indicates that typically not more than $4\mu s$ of carrier sensing preamble may be necessary for WiFi-Nano. Finally, the benefit of IdleSense is evident in this figure, as the number of contenders at the beginning of each epoch is dramatically reduced.

## 6.3 Benefits of WiFi-Nano

WiFi-Nano has three main benefits over 802.11. WiFi-Nano *i)* significantly reduces the overhead of data transmission, thus increasing throughput even when one node is transmitting; *ii)* limits the collision overhead to the length of the preambles and thus improves throughput further when there are many contending nodes; *iii)* improves fairness by eliminating the capture effect, in which nodes closer to an AP, take advantage of the higher SINR with respect to farther nodes and gain higher throughput.

**Throughput.** Figure 15 compares the throughput achieved at different data-rates in two cases, when there is only one transmitter in the network and when there are 30 active transmitters in the cell. Consider the case of a single transmitter. As the data-rate increases,

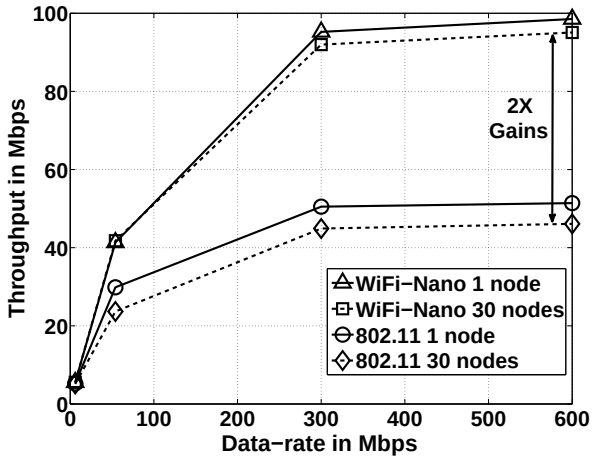


**Figure 15: Throughput**

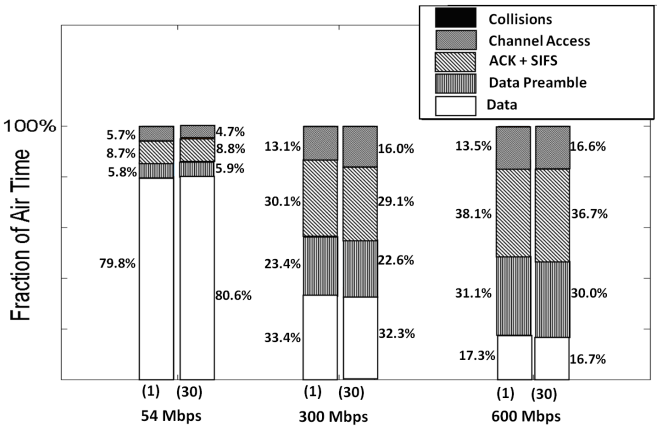


**Figure 16: WiFi-Nano overheads**

the relative improvement of WiFi-Nano over 802.11 increases, due to high channel access overhead of 802.11 at higher rates as described in Section 2. Thus, the throughput gain of WiFi-Nano over 802.11 is $5\%$, $37\%$, $85\%$, $88\%$, at 6 Mbps, 54 Mbps, 300 Mbps, 600 Mbps, respectively. Next, consider the case of 30 transmitters. The gap between WiFi and WiFi-Nano widens further since WiFi-Nano is able to mostly avoid collisions, while the increased number of nodes reduces channel access overhead. On the other hand, the collision overhead in WiFi reduces aggregate throughput compared to the single transmitter case. Thus, at 300 Mbps and 600 Mbps, WiFi-Nano is able to achieve a throughput gain of 117% and 119% over WiFi, respectively.

Similar to Figure 2, we further analyze the performance of WiFi-Nano by decomposing the overhead into Preamble, ACK + SIFS, Channel Access, and Collisions. Figure 16 shows the results for 1 and 30 senders at data-rates of 54 Mbps, 300 Mbps and 600 Mbps. We observe that the time spent in data collisions for the 30-node case is not visible, as collisions accounts for less than $1\%$ of the time. Furthermore, the preamble and ACK overheads cannot be eliminated since they are essential for receiving data at these high rates (the SIFS overhead has been eliminated). Thus, the real overhead that remains in WiFi-Nano is only the channel access overhead which represents between 5.9-16.5% in the Figure.

**Fairness.** Finally, we investigate the benefit of WiFi-Nano in terms

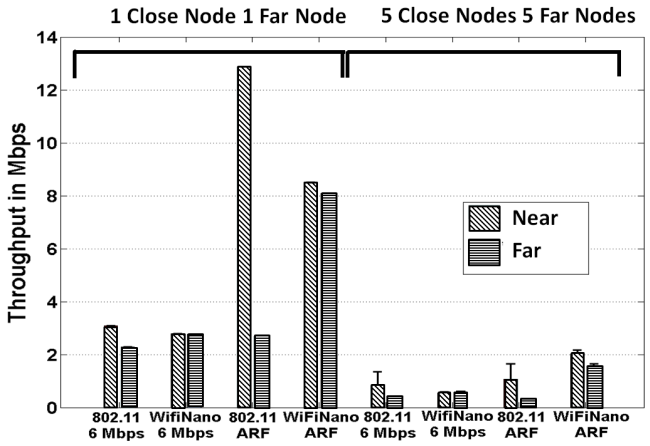


**Figure 17: Fairness**

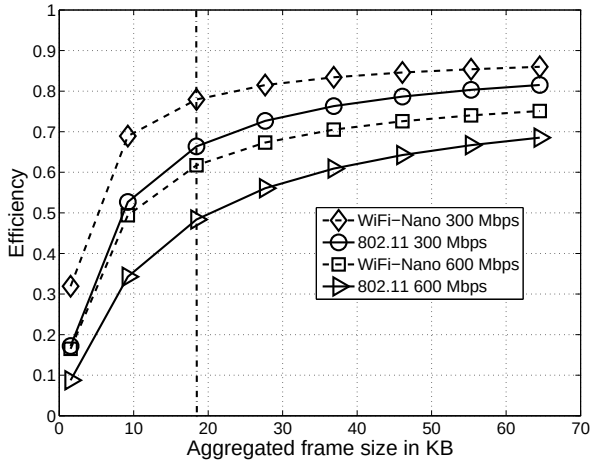


**Figure 18: Frame Aggregation**

of fairness. In 802.11, the backoff counters of two nodes may expire during the same slot, and thus cause a collision; depending on their relative SINR at the receiver, the receiver may be able to decode one of the packets (see capture effect [8]). This unfairly favors the nodes closer to an AP which benefit from a higher SINR. In WiFi-Nano, the preamble detection process terminates with a single node transmitting the data packet in the overwhelming majority of the cases; thus, the role of the capture effect is significantly reduced.

In this experiment, we used two kinds of nodes – near and far nodes. Near nodes are located 10 m from the AP while far nodes at 90m from the AP. We consider both fixed rate transmissions at 6 Mbps as well as auto-rate fallback for 802.11a rates. We conducted two sets of experiments – first, with a single near and far node and second, with 5 near and far nodes. Figure 17 shows the per-node average throughput for WiFi-Nano and 802.11 at different combinations of data-rate and number of nodes. Within each pair of bars, the left bar represents the throughput of closer node(s) while the right bar depicts the throughput of the farther node(s); the error bars correspond to the maximum and minimum throughput of the nodes in each set.

Consider the case of one far and near node. In 802.11, the throughput of the closer node is 35% and 479% higher than the farther nodes when fixed rate and auto-rate fallback, respectively, are used while in the case of WiFi-Nano, the gap is less than 1% and 5%, respectively. Thus, WiFi-Nano is able to re-establish fairness without penalizing the total network throughput.

## 6.4 Frame aggregation

One way of reducing the overhead in WiFi is by transmitting larger packets so that the MAC overhead is reduced to a small fraction of the transmission time. 802.11 standards have incorporated frame aggregation and block acknowledgements to allow frames of up to 64KB to be transmitted after a single channel access. Thus, data transmission time at 600 Mbps can be increased from 12 $\mu s$ for a 1500B packet to 873 $\mu s$ for a 64 KB frame, thereby reducing the channel access overhead from 90+% to under 20%.

In Figure 18, we plot the efficiency (i.e., the ratio of the achieved throughput over the nominal data-rate used) of a single backlogged flow, as the aggregate frame size increases, for 300 and 600 Mbps data rates. We simulate an ideal frame aggregation mechanism with minimal overheads and choose settings so that the access point is able to aggregate frames up to 64 KB.

From the Figure, we see that using 64 KB frames, WiFi efficiency approaches 70% and 80% at 600 Mbps and 300 Mbps, respectively. Even for the fully aggregated 64 KB frame sizes, WiFi-Nano is able to achieve $5-10\%$ throughput gains over WiFi. However, 64 KB average frame sizes are hard to achieve in practice due to the presence of small packets such as TCP acks or buffer contention due to traffic destined for other nodes. At typical average aggregation frame size of 18 KB for large data transfers [18], WiFi-Nano achieves $25\%$ (resp., $17\%$) gains over 802.11 at 600 Mbps (resp., 300 Mbps).

While the above results indicate that frame aggregation can be effective in reducing WiFi inefficiency for large data transfers, it does not help for delay sensitive applications such as voice-over-IP or short HTTP transfers. Thus, we expect higher efficiency gains for WiFi-Nano in typical WiFi settings.

# 7. RELATED WORK

There has been tremendous amount of work targeted towards improving wireless performance. We discuss a few papers that are relevant to WiFi-Nano.

**Performance.** Researchers have proposed several techniques to improve performance in wireless LANs including use of partial packets in recovering from errors [5, 12], using network coding for retransmissions [15], using directional antennas [9, 13], and the use of aggregation and TDMA-like schedule to improve WiFi efficiency in the presence of VoIP traffic [20]. Perhaps closest to WiFi-Nano is Idle Sense [3] and FICA [18].

Idle Sense proposes an alternative to the binary exponential backoff-based WiFi MAC by trying to ensure that hosts in a single wireless LAN use a similar contention window. As we show in our evaluations, the low collision probability of Idle Sense helps WiFi-Nano use a shorter preamble length. FICA tackles the inefficiencies of the WiFi MAC by redesigning both the PHY/MAC using fine-grained *subchannels* and delivering efficiencies of 70% at 600 Mbps. However, the use of subchannels require a synchronous system which makes it difficult for FICA to co-exist with neighboring networks that are not frequency/time synchronized.

**Full-duplex.** Recently, full-duplex single channel wireless communication systems have been proposed [1, 14]. The key challenge in these systems is eliminating the self-interference of the local transmitter, which can be done using a combination of analogue interference cancellation and a nulling antenna. However, perform-

ing full-duplex decoding at MIMO data rates over 20-40MHz bandwidth and at WiFi transmit power is still a challenging, open problem [1]. Finally, while these systems double the capacity of data transmission, they still suffer from the channel access overhead issues of WiFi.

In WiFi-Nano, we only need to correlate with the preamble during transmission and do not expect the transmitter to decode any bits. Thus, the interference cancellation requirement is not as stringent as full-duplex systems. Since WiFi-Nano reduces the channel access overhead, it is complementary to full duplex systems.

**Collisions.** When packet collisions occur in 802.11-based wireless networks, the receiving node may still be able to *capture* one of the transmissions if its signal strength is sufficiently high compared to the interfering signal, thereby reducing the impact of collisions [8]. However, the capture effect can result in unfairness where transmissions from nodes closer to the AP have higher probability of getting captured compared to nodes far away from the AP.

ZigZag decoding [2] is a technique that allows nodes to efficiently recover from collisions due to hidden terminals by exploiting interference-free stretches in the collided packets. CSMA/CN [17] performs collision notification where the receiver can notify the transmitter to abort transmission when capture is not feasible during collisions, thereby reducing the collision overhead. WiFi-Nano also suffers from hidden terminal problems like WiFi and could benefit from CSMA/CN and ZigZag decoding to improve efficiency in the presence of hidden terminals. Further, the collision notification in CSMA/CN is achieved by correlating with a preamble-like sequence while packet transmission is on-going. WiFi-Nano's speculative preamble relies on similar correlation ability at the transmitter except during preamble rather than packet transmission.

Finally, collisions due to overlapping channels is studied in [11] where the authors propose that retransmissions be permuted so that decoding efficiency is improved.

**Measurements.** Carrier sensing and packet detection algorithms are not specified in the standards and is left to the vendor's implementation. Packet detection is accomplished through a combination of correlation with preamble and energy detection [6]. Different vendors have different thresholds for preamble and energy detection [10]. Similar to [10], we also incorporate preamble detection as part of carrier sensing in the Qualnet simulator for our evaluations.

Authors in [7] study the impact of interference on packet reception in 802.11b by carefully controlling the timing of the interferer with respect to the transmitter. They show that packet reception probability increases significantly as soon as the interferer is delayed from the start of the transmitter by as little as 3.2 $\mu s$. While 802.11b preamble is DSSS (unlike the OFDM preambles used in 802.11g/n and in this paper), these results indicate the preamble detection can occur even in the presence of significant interference.

# 8. CONCLUSION

In this paper, we identified WiFi's slot size as a key reason for its inability to deliver the data rate gains achieved at the physical layer to the MAC layer and above. Thus, instead of 9 $\mu s$ slots used in WiFi, we propose WiFi-Nano that uses slot sizes as small as 800 $ns$. The small slot size necessitates that preamble transmission and detection be done in parallel, which is achieved using speculative transmission of preambles and analogue interference cancellation. Furthermore, we design a novel lattice correlator that correlates to parts of the preamble and is able to accurately determine the start time of detected preambles, a key requirement for accurate rollback of speculative preamble transmissions.

Finally, since WiFi-Nano nodes are able to detect collisions during the preamble transmission phase, they abort their transmissions probabilistically so that packet collisions are avoided with high probability. Using testbed experiments and extensive simulations, we show that WiFi-Nano is able to double the MAC throughput of WiFi at 802.11n rates.

# 9. REFERENCES


[1] J. I. Choi, M. Jain, K. Srinivasan, P. Levis, and S. Katti. Achieving single channel, full duplex wireless communication. In *ACM MobiCom*, 2010.

[2] S. Gollakota and D. Katabi. Zigzag decoding: Combating hidden terminals in wireless networks. In *ACM SIGCOMM*, 2008.

[3] M. Heusse, F. Rousseau, R. Guillier, and A. Duda. Idle sense: An optimal access method for high throughput and fairness in rate diverse wireless lans. In *ACM SIGCOMM*, 2005.

[4] IEEE802.11a. IEEE Standard for Telecommunications and Information Exchange Between Systems - LAN/MAN Specific Requirements - Part 11: Wireless Medium Access Control (MAC) and physical layer (PHY) specifications: High Speed Physical Layer in the 5 GHz band, 2000.

[5] K. Jamieson and H. Balakrishnan. Ppr: Partial packet recovery for wireless networks. In *ACM SIGCOMM*, 2007.

[6] K. Jamieson, B. Hull, A. K. Miu, and H. Balakrishnan. Understanding the Real-World Performance of Carrier Sense. In *ACM SIGCOMM Workshop on Experimental Approaches to Wireless Network Design and Analysis (E-WIND)*, Philadelphia, PA, August 2005.

[7] G. Judd and P. Steenkiste. Characterizing 802.11 wireless link behavior. *Wireless Networks*, 16(1), January 2010.

[8] A. Kochut, A. Vasan, A. Shankar, and A. Agrawala. Sniffing out the correct physical layer capture model in 802.11b. In *ICNP*, 2004.

[9] S. Lakshmanan, K. Sundaresan, S. Rangarajan, and R. Sivakumar. The myth of spatial reuse with directional antennas in indoor wireless networks. In *PAM*, 2010.

[10] J. Lee, J. Ryu, S.-J. Lee, and T. Kwon. Improved Modeling of IEEE 802.11a PHY Through Fine-Grained Measurements. *Computer Networks*, 54(4), March 2010.

[11] L. Li, K. Tan, H. Viswanathan, Y. Xu, and Y. R. Yang. Remap decoding: Simple retransmission permutation can resolve overlapping channel collisions. In *ACM MobiCom*, 2010.

[12] K. Line, N. Kushman, and D. Katabi. Ziptx: Harnessing partial packets in 802.11 networks. In *ACM MobiCom*, 2008.

[13] 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 *ACM MobiCom*, 2010.

[14] B. Radunovic, D. Gunawardena, P. Key, A. Proutiere, N. Singh, V. Balan, and G. Dejean. Rethinking indoor wireless mesh design: Low power, low frequency, full-duplex. In *WiMesh*, 2010.

[15] E. Rozner, A. Iyer, Y. M. nd Lili Qiu, and M. Jafry. Er: Efficient retransmission scheme for wireless lans. In *ACM CoNEXT*, 2007.

[16] T. Schmidl and D. Cox. Robust Frequency and Time Synchronization for OFDM. *IEEE Transactions on Communications*, 45(12), December 1997.

[17] S. Sen, R. R. Choudhury, and S. Nelakuditi. CSMA/CN: Carrier Sense Multiple Access with Collision Notification. In *ACM MobiCom*, September 2010.

[18] K. Tan, J. Fang, Y. Zhang, S. Chen, L. Shi, J. Zhang, and Y. Zhang. Fine Grained Channel Access in Wireless LAN. In *ACM SIGCOMM*, August 2010.

[19] F. Tufvesson, M. Faulkner, and O. Edfors. Time and frequency synchronization for ofdm using pn-sequence preambles. In *IEEE VTC*, 1999.

[20] P. Verkaik, Y. Agarwal, R. Gupta, and A. C. Snoeren. Softspeak: Making voip play fair in existing 802.11 deployments. In *NSDI*, 2009.