Energy-based Rate Adaptation for 802.11n

Chi-Yu Li[1] Chunyi Peng[1] Songwu Lu[1] Xinbing Wang[2]

[1] University of California, Los Angeles, CA 90095, USA [2] Shanghai Jiao Tong University, China

{lichiyu, chunyip, slu}@cs.ucla.edu, xwang8@sjtu.edu.cn

ABSTRACT

Rate adaptation (RA) has been used to achieve high goodput. In this work, we explore to use RA for energy efficiency in 802.11n NICs. We show that current MIMO RA algorithms are not energy efficient for NICs despite ensuring high throughput. The fundamental problem is that, the high-throughput setting is not equivalent to the energy-efficient one. Marginal throughput gain may be realized at high energy cost. We propose EERA, an energy-based RA solution that trades off goodput for energy savings at NICs. Our experiments have confirmed its energy savings at NICs while keeping the cost at the device level and across clients acceptable.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication*

General Terms

Algorithms, Design, Experimentation, Performance

Keywords

Energy Efficiency, IEEE 802.11n, MIMO, Rate Adaptation

1. INTRODUCTION

Rate adaptation (RA) is a popular mechanism [6,8,11,14,19,21–23] to improve the performance of wireless network interface card (NIC). It dynamically selects the best physical-layer configuration (e.g., various modulation and coding schemes) depending on time-varying channel conditions. The traditional goal of RA is to achieve high goodput (i.e., effective throughput). In this work, we explore to use RA to ensure energy efficiency on recent 802.11n NICs.

Our research is motivated by two factors. First, 802.11n devices are increasingly popular. The shipment reached 5.9M in the second quarter of 2010. It is expected to accelerate at an annual rate of 15% in upcoming years [5]. Battery-powered smartphones and tablets have become the next target for 802.11n [3]. Second, an 802.11n device consumes much more power than its legacy 802.11a/b/g NIC. Our measurements show that, an 802.11n 3x3 MIMO receiver consumes about twice the power of 802.11a during active transmission, and 1.5 times power when idle. Therefore, energy efficiency becomes a critical issue for 802.11n NIC operations.

Existing RA solutions are effective to ensure high goodput but not energy savings. We observe that, two popular 802.11n RA algorithms ARA [22] and MiRA [19] incur per-bit energy waste at an NIC as large as 54.5% and 52.9%, respectively. The energy waste still exists with/without the power-saving mechanisms of SMPS and PSMP. The root cause is that, current RAs obtain high goodput at whatever energy cost. Marginal goodput gain is realized by powering on more antennas, more streams, and higher MCS rates. The fundamental problem is that, conflicts arise between high-goodput settings and low-energy settings in 802.11n NICs.

We propose EERA, a new RA algorithm that trades off goodput for energy savings at an 802.11n client NIC. EERA searches for the MIMO setting consuming less per-bit energy, rather than achieving higher goodput. It thus slows down communication to save energy. However, this slowdown is contained by two conditions: EERA must accommodate its data source rate and not affect other clients using traditional RAs. EERA supports both single-client and multi-client operations. In the former case, it abstracts the problem as multi-dimensional search, and exploits ternary search and MIMO characteristics to speed up its runtime convergence. The latter case builds on top of single-client design. Each client is periodically allocated a fair share of airtime, and can only use up this airtime share but no more. Fair sharing of extra airtime protects each client through isolation, thus enabling coexistence of EERA and other MIMO RA clients. Moreover, EERA is configurable. Each client specifies a tuning knob, which controls multi-client interference, and energy balance at NIC and other device components. EERA reverts to traditional goodput-optimizing RA if needed.

In all test scenarios, EERA consistently outperforms other RA algorithms in terms of NIC energy efficiency. It saves about 30% energy compared with ARA and MiRA in all scenarios. It saves 6-36% in static settings and 20-24% in mobility and field tests, compared with another energy-saving proposal MRES [20]. At the device level, EERA does not incur nonnegligible energy increase for Web, VoIP and video applications, but incurs 1.77% waste in FTP compared with ARA in our tested scenario. This is because FTP application program stops consuming more energy once a file transfer completes; faster transmission helps to reduce energy consumption on FTP application. EERA also compares well with ARA in multi-client scenarios. It increases packet delay by about 0.07-0.19ms in tested cases. For contention, it increases retries by at most 3.4% (0.009 retries/ms), but avoids up to 34% retries (0.08 retries/ms) when the traffic source rate is low.

The rest of the paper is structured as follows. Section 2 introduces the background on 802.11n MIMO and its power-saving schemes. Section 3 uses a case study to examine the limitations of 802.11n RA in NIC energy efficiency, and Section 4 models the 802.11n NIC energy. Sections 5, 6, and 7 describe the design, implementation, and evaluation of EERA, respectively. Section 8 discusses the related work and Section 9 concludes the paper.

2. BACKGROUND

In a nutshell, rate adaptation (RA) offers an effective mechanism to exploit the multi-rate, adaptive modulation capability at the physical layer. The design of RA is more complex in 802.11n than in the legacy 802.11a/b/g systems. Given the wireless channel condition, it has to select the appropriate configuration in three dimensions, the modulation and coding scheme (MCS), the chain setting (i.e., the chosen numbers of transmit and receive antennas at the sender and the receiver), and the number of spatial streams in the 802.11n scenario.

The 802.11n specification defines a large parameter space, thus posing *scaling* issues for RA design. The MCS rates span from 6Mbps to 600Mbps. Each sender/receiver can *activate* one to four transmit/receive antennas. The standard also supports multiple-stream operations, i.e., single-stream (SS), double-stream (DS), triple-stream (TS), and quadruple-stream (QS). The streams are bounded by the smaller number of transmit and receive antennas. Each setting is denoted by $N_t \times N_r / rateStream$, with N_t and N_r being transmit and receive antennas, $rate$ being the MCS rate, and $Stream$ being the number of streams. For example, 3×3/13.5SS defines the setting using a single stream on three transmit/receive antennas each, with the MCS rate being 13.5 Mbps.

Power Saving Mechanisms in 802.11n: Since our interest is on energy savings, we briefly introduce two power-saving mechanisms defined by the 802.11n standard, i.e., Spatial Multiplexing Power Save (SMPS) and Power Save Multi-Poll (PSMP). Both reduce power consumption during the non-active (i.e., *idle* or *sleep*) period without data transmission. SMPS reduces the idle power consumption at the receiver by activating only one receive chain. The standard supports two modes. In the *Static* mode, the client statically retains a single receive chain. In the *Dynamic* mode, the receiver switches to multiple receive chains during data transmission, but shifts back to one chain afterwards. PSMP allows for a receiver to sleep during its non-active period (e.g., when AP transmits to other clients). The 802.11n standard also supports two modes: Scheduled PSMP (S-PSMP) and Unscheduled PSMP (U-PSMP). In S-PSMP, AP periodically initiates a PSMP sequence to schedule the transmission. In U-PSMP, AP starts an unscheduled sequence and delivers to those wakeup clients. Both SMPS and PSMP complement our RA scheme. Our design primarily handles the *active* period for data transfer and strikes balance between active and non-active energy consumption to ensure NIC energy efficiency.

3. 802.11N RA LIMITATION: AN ENERGY EFFICIENCY PERSPECTIVE

We now use a simple case study to examine the limitations of current 802.11n RA algorithms in terms of energy efficiency. We show that, current solutions are effective to achieve high goodput, but may not ensure energy efficiency. There exists fundamental conflicts between the best goodput setting and the most energy-efficient setting for 802.11n NICs.

3.1 Experimental Setting

Our goal is to quantify the energy consumption of 802.11n NICs under various RA algorithms. We select two existing 802.11n RA algorithms for comparison. Atheros RA (ARA) algorithm [22] is used by Atheros 802.11n NICs, while MiRA [19] is a new proposal for 802.11n radios. Both ARA and MiRA apply *sequential* search to probe different settings and locate the best setting eventually. ARA probes from the medium setting (i.e., the highest MCS rate in the DS mode). If the probe succeeds, it switches to the TS mode; otherwise, it goes down through MCS rates in DS and SS modes. Its implementation excludes half of rates during the probing process. In contrast, MiRA uses zigzag probing and starts from the highest MCS in the TS mode, and then switches to DS and SS until it succeeds. Both algorithms can be implemented in the current platforms using available 802.11n chipsets.

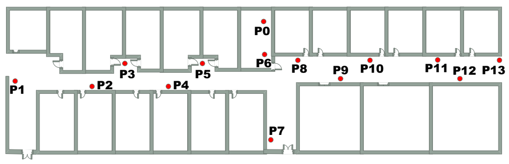


Figure 1: Experimental floorplan.

	EE	ARA	MiRA
E_b (nJ/bit)	19.2	29.7	29.4
Gap (%)	–	54.5%	52.9%
Goodput (Mbps)	35.4	52.4	52.5
#Bits (Mbits)	3598	3576	3598
Energy (J)	69.0	106.2	105.6

Table 1: Per-bit energy consumption, goodput, number of bits, and energy consumption for ARA, MiRA and the optimal EE setting at Location P1. The UDP source rate is 30 Mbps.

We conduct our experiments in a controlled laboratory environment over the 5GHz band; no external interference is observed on the used channel. Both AP and clients operate in an office building (see Figure 1 for the floor plan); Spots P1 to P13 represent different locations for the client, whereas AP is always at P0. We consider the infrastructure mode only, the dominant deployment in reality. Both the AP and clients are programmable 802.11n devices, which use Atheros AR9380 2.4/5 GHz MIMO chipset and support three transmit/receive antennas. The platform supports SS, DS, and TS modes, with transmission rates up to 450Mbps over 40MHz bands. We also use Intel 5300 wireless NIC on the client side. We plot results for downlink transmissions, with the client being the receiver. In each test, we send the UDP traffic generated by *iperf* at constant source rate (say, 30 Mbps for the tests in Table 1) for 120 seconds and collect measurement results over five runs.

We use power meter Agilent 34401A to record the consumed power. For PSMP, we collect traces and use the ideal doze power to simulate its energy value, since PSMP is not implemented in current drivers. Energy saving from PSMP is overestimated without counting its processing overhead. To quantify the energy efficiency of RA algorithms, we use *per-bit energy consumption* E_b as the evaluation metric, defined as the consumed energy when exchanging each bit given a setting. This metric represents the energy consumption when transferring each bit, including consumed energy during both active and non-active periods [20].

3.2 Case Findings

Our experiments show that, both ARA and MiRA incur large energy waste, compared with the most energy-efficient fixed setting. Table 1 gives the *per-bit energy consumption* by both ARA and MiRA, as well as the best fixed setting (called "EE" in the table) achieving highest energy efficiency among all settings at P1 (see Figure 1). The results show that, ARA and MiRA incur per-bit energy waste as large as 54.5% and 52.9%, respectively, when

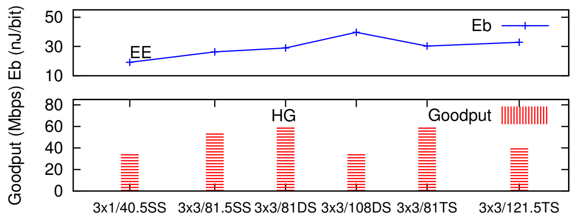


Figure 2: Goodput and per-bit energy of various settings at P1.

Setting	3x1/ 40.5SS	3x3/ 81SS	3x3/ 81DS	3x3/ 108DS	3x3/ 81TS	3x3/ 121.5TS
Active Power (mW)	580.6	812.3	975.0	982.5	1046.4	1063.4
Idle Power (mW)	541.2	765.6				

Table 2: Power consumption of the most energy-efficient setting and those used by MiRA and ARA at P1.

compared with the best setting. Interestingly, we also observe that MiRA and ARA ensure higher goodput during active data transmissions, about 52.5Mbps and 52.4Mbps, respectively, compared with the EE setting that yields only 35.4Mbps. Note that all can sustain the 30Mbps UDP data source.

We next find why current algorithms incur energy waste for NICs. It turns out that, both ARA and MiRA are able to achieve high goodput during the active period, but these settings are not among the most energy-efficient ones. To this end, we first compute the goodput, as well as the *per-bit energy consumption* for those selected settings, and plot them in Figure 2. The setting yielding highest goodput (marked with "HG" in the figure) is 3x3/81DS. However, this HG setting is not the most energy-efficient one (i.e., 3x1/40.5SS, marked with "EE"). The gap in *per-bit energy consumption* between these two settings reaches 11.1 nJ/bit, incurring energy waste as large as 57.8% when using the HG setting. Figure 3(a) further plots the rate distribution (in percentage) of ARA and MiRA. We see that both algorithms are effective in reaching high goodput. This observation is consistent with the primary goal of their design. ARA mainly selects two settings, 3x3/81DS and 3x3/108DS, whereas the selection of MiRA spreads over 5 settings. ARA chooses fewer settings because it considers only half of the rate settings. However, these high-goodput settings consume more energy per bit, as large as 39.7 nJ/bit (3x3/108DS), about twice the *per-bit energy consumption* of the EE setting 3x1/40.5SS. They hence make ARA and MiRA deviate from the EE setting by as large as 54.5% and 52.9%, respectively, in Table 1.

We further examine why the highest-goodput settings cannot ensure best energy efficiency. It turns out that, at these high-goodput settings, the per-bit energy cost to obtain the marginal goodput gain is pretty high. To achieve higher goodput, we have to activate more antennas and more streams no matter how much extra power could be consumed. Here, the "HG" setting (3x3/81DS) consumes additional 394.4 mW and 224.4 mW in active and non-active periods, compared with the "EE" setting (3x1/40.5SS) (shown in Table 2). Figure 3(b) plots goodput and *per-bit energy consumption* for various settings at P1. It shows that, E_b does not monotonically decrease as the goodput increases; Several dips appear in these settings. The marginal goodput gain at the cost of extra energy consumption becomes smaller or even negative for some high-rate settings. As a result, it becomes less energy efficient to chase for higher goodput.

Moreover, our study reveals that current RA algorithms may not have fast convergence when locating the best setting. Both ARA and MiRA apply sequential search to locate the best setting by sequentially probing feasible settings. These sequential search operations may result in slower convergence. This can be illustrated by

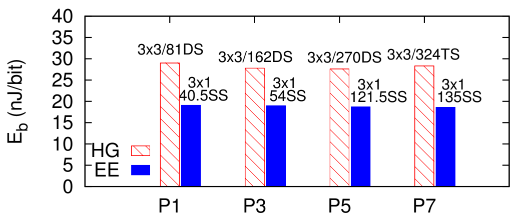


Figure 4: Per-bit energy by RA and EE settings at various locations with 3 transmit antennas and 30 Mbps source.

the detailed probing process of ARA and MiRA shown in Figure 3(c). The sequential search in ARA and MiRA further faces the scaling issue when the number of settings becomes larger. Assume three antennas at the AP. The number of receive antennas can be one (SS), two (SS/DS), and three (SS/DS/TS). The client thus supports $1+2+3 = 6$ (i.e., $N_r(N_r+1)/2$, where N_r is the maximum number of receive antennas) modes, and each mode allows for multiple MCSes (8 MCSes for 802.11n). The total number of settings can reach $6 \times 8 = 48$. The scaling issue becomes more prominent when the number of antenna grows to eight and the number of MCSes per mode reaches ten in the upcoming 802.11ac [4]. This leads to the overall search space of 360 ($= 10 \times 8(8+1)/2$) choices.

3.3 Dynamics of Energy-Efficient Settings

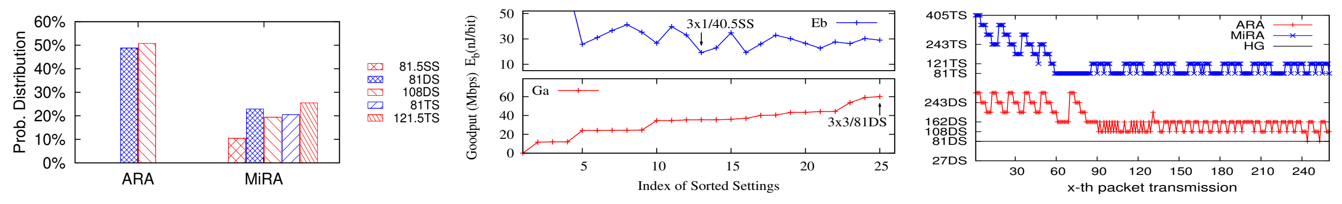
We next show that, the most energy-efficient (EE) setting varies with several factors, including location, data source, power-saving schemes, and the number of activated AP antennas.

Location dependence: We now show that the energy waste by current RA schemes is location dependent. The fundamental reason is that, the EE setting varies with locations, but it is not the same as the HG one in general. Figure 4 shows *per-bit energy consumption* for HG and EE settings at various locations; the HG setting is where current RA schemes would stay at. The data source is set as 30 Mbps and AP uses three transmit antennas. The HG setting underperforms the EE setting in terms of energy efficiency. Specifically, HG consumes 51.8%, 46.3%, 47.6%, and 52.2% extra energy compared with EE at P1, P3, P5, and P7 respectively.

Effect of power-saving schemes: Current RA schemes still incur energy waste, whether or not we use the power-saving schemes. However, power-saving schemes may reduce the waste percentage because of reduced power during the non-active (idle/sleep) state. The root cause is that, the EE setting varies with the use of power-saving schemes, while the HG setting remains invariant as long as the channel condition remains unchanged. Assume three antennas at AP. When we use different power-saving schemes (i.e., SMPS and PSMP), the EE setting turns into 3x1/40.5SS and 3x1/54SS, respectively, still different from the HG setting. Note that, the *per-bit energy consumption* of the HG and EE settings indeed decreases due to smaller energy consumption at idle/sleep states. The difference thus becomes smaller. However, the gap is still as large as 31.8% at these two locations as shown in Figure 5(a).

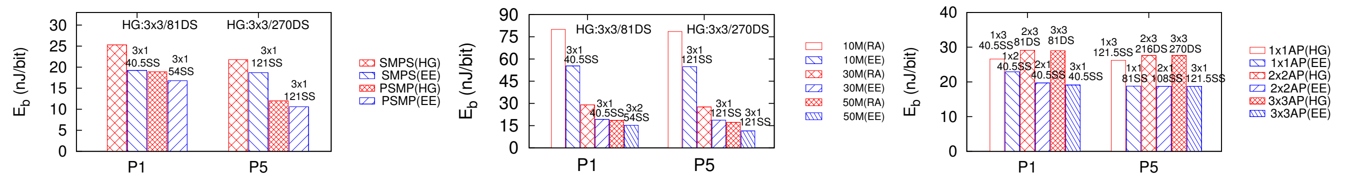
Effect of data source rates: We also study the impact of source rates on energy efficiency. We note that the energy inefficiency of current RAs also varies with data source. The reason is that, the EE setting varies with source rates, while the HG setting remains unchanged. For instance, when the source rate increases from 10 Mbps to 50 Mbps, the EE setting at P1 changes from 3x1/54SS to 3x2/54SS. The energy waste by HG still reaches 44.7%, 51% and 21.7%, for three source rates, as shown in Figure 5(b).

Effect of activated antennas: The energy efficiency of current RA schemes also changes with the number of activated antennas.



(a) Selected rate distribution (b) Goodput vs. per-bit energy (c) RA Probing process

Figure 3: Performance of ARA and MiRA at P1. All the settings in (a) use three receive antennas.



(a) Power saving schemes (b) Source rates (c) Number of AP antennas

Figure 5: Per-bit energy consumption of the HG and EE settings at P1 and P5.

When the number of AP antennas reduces from 3 to 1, the HG setting at P5 changes from 270DS to 216DS, and finally to 121.5SS using three receive antennas, with less per-bit energy consumption. However, Figure 5(c) shows that, the difference between HG and EE is still as large as 47.6%, 47.6% and 39.4%, respectively. In general, the fewer the number of AP antennas, the smaller the gap between HG and EE settings. It depends on how much more receive antennas contribute to goodput improvement and power increase. When the receiver is closer to AP, the marginal goodput gain is small when activating an extra receive antenna; it is ineffective to use more receive antennas in terms of energy efficiency. However, an extra receive antenna may bring higher marginal gain when the client is far away from AP.

3.4 Impact on Device-Level Energy Efficiency

While the most energy-efficient setting saves energy at the 802.11n NIC, it may have negative impact on the device. Since it deviates from the HG setting, it slows down data transmission and prolongs the delivery time. For instance, the portion of active period increases from 56% to 85%, when the EE setting 3x1/40.5SS, but not the HG setting 3x3/81DS, is used in Section 3.2. This slowdown might negatively increase energy consumption of other components (e.g., display, CPU, disk, memory), since they may have to stay active longer for data transmission over NICs.

To assess the impact on device-level energy efficiency, we study two issues: (1) Can energy consumption of other components be decoupled from the NIC status? (2) If not, how much is the incurred energy increase at these components? We focus on display and CPU, since they contribute to the dominant portion of energy consumption at laptops and smartphones [7, 18]. In our experiments, we measure the energy consumption over the same duration when EE or HG is used. The duration always lasts until the slower transmission by EE has completed. The device enters its default power-saving mode after HG completes its transmission first. We use ASUS F8S laptop with Intel Core2 Duo T8300 processor in our measurements. Its display power is from 2.7 W to 9.4 W, much bigger than that of smartphones and other mobile devices [7].

We find out that the display energy consumption is mostly independent of the NIC status. It is mainly decided by the degree of brightness and how long users want to interact. We measure the display energy with different brightness (from 0% to 100%) in four 802.11n NIC states: off, idle, high-rate reception (e.g., 3x3/81DS at P1), and low-rate reception (e.g., 3x1/40.5SS at P1). We have

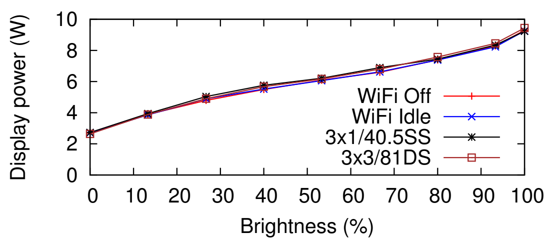


Figure 6: Display energy varies with degree of brightness.

CPU State	C0	C1	C2	C3
EE (3x1/40.5SS)	5.8%	0%	26.0%	66.4%
HG (3x3/81DS)	5.5%	0%	42%	52.0%
Power@800MHz (W) (from [2])	16.8 – 21.3	16.8 – 21.3	10.3 – 13.0	9.8 – 12.4

Table 3: CPU status statistics using HG and EE at P1.

20 runs for each NIC state. Figure 6 plots the average display power consumption over the same measurement period; the standard derivation of each point is within 0.15. The display energy remains invariant at different 802.11n NIC states (including the OFF state), as long as the brightness and duration are fixed. This is also consistent with other studies on display energy [7, 18]. Moreover, it concurs with user experience. The power-saving operations and active duration of the display are mainly determined by user interaction (e.g., screen off after 15-second idle period) but not NIC or other components; it can power off at any time even when NIC is still transmitting. We also tested more cases with various source rates, locations and wireless rates. The results are similar.

We also discover that, the CPU status can be slightly changed due to slower wireless transmissions. Given the same measurement duration, the incurred power increase is roughly in order of tens of milliwatts. We use Intel PowerTop [1] to record CPU status when HG and EE settings are used in our experiments. The CPU has five power states: C0 (normal), C1 (normal), C2 (stopgrant, i.e., sleep), C3 (deep sleep) and C4/C6 (deeper sleep) [2]. In our traces, no C4/C6 state is observed. Table 3 shows an example trace of the CPU status. We observe that, the EE setting slightly increases the CPU duration at C0 (by 0.3%) when the transmission time increases from 56% to 85%. In our experiments, we observe that the CPU mainly work at 800 MHz or in its idle mode. We also list the CPU power consumption for different states at 800 MHz using the data sheet of [2]. The incurred CPU power is no more than 60 mW (at most 21.3 W $\times$ 0.3%) over the measure interval.

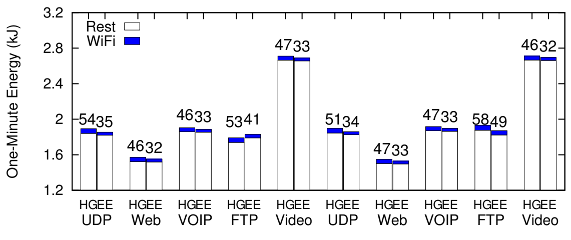


Figure 7: Device and NIC energy consumption in various application scenarios.

In fact, it could be smaller, since less energy is consumed at C2 and C3 states and CPU works at a lower frequency during idle. We also test more cases and obtain similar results.

We further quantify the device energy consumption, as well as NIC, with five applications. They include (1) UDP: start 30 Mbps downlink flow; (2) Web: fetch a 3.8 MB webpage five times within a minute; (3) VoIP: chat for two minutes; (4) FTP: download a 721.9 MB file; and (5) Video streaming: play a 10-minute 1080p HD video. In our experiments, VoIP includes both uplink and downlink traffic, its average rate is 163 Kbps, and the average packet size is 162B. The average source rate of Video streaming is 3.9 Mbps, and the average packet size is 1277 Bytes, while the client side buffering (5-second buffer) is used. Figure 7 plots the average device energy consumption over one minute, where the number above each bar being the NIC energy consumption in Joul. We make two observations. First, the 802.11n HG/EE settings have negligible impact on the energy consumption of other components at the device except in the FTP case. Second, NIC energy saving is the major source to device-level energy efficiency. Though we only observe a small percentage of device-level energy savings, the savings are about 42–54% for NIC alone except in the FTP case (29% and 18% at P1 and P7, respectively). Since NIC consumes only 520–900 mW but the laptop consumes about 30W, the saving percentage at the device level is not large. The gain percentage will be much larger for smartphones [7].

In the FTP case, the HG setting downloads this file faster (104 seconds), compared with 148.2 seconds using EE at P1. Therefore, the FTP application stops early and consumes 4.4 KJ, smaller than the consumed energy of 4.5 KJ in EE over 148.2 seconds. For other applications, there is no significant change even though wireless transmission finishes early, thus imposing little impact on the energy consumption of other components. Note that, device energy consumption is also location dependent in the FTP scenario; device-level energy saving is still achieved at P7. In all cases, EE ensures energy savings for 802.11n NICs.

4. MODELING 802.11N NIC ENERGY

We now model the power consumption of an 802.11n NIC.

Per-Bit Energy: The NIC energy efficiency is quantified by *per-bit energy consumption* E_b, i.e., the consumed energy when transmitting/receiving each bit. Assume that the data source can be accommodated by the setting. Given the time interval of interest T, the per-bit energy is calculated as

$$E_b = Energy/N_{Bits} = (P \times T)/(S \times T) = P/S, \quad (1)$$

where P represents the average power consumption and S represents the data source rate over the entire period T. When data source rate S is smaller than the achievable goodput G, i.e., $S \leq G$, the client experiences both active and non-active (i.e., idle or sleep) modes. Then we have, $P \cdot (T_a + T_{na}) = P_a \cdot T_a + P_{na} \cdot T_{na}$,

Platform	α_1	α_2	α_3	$f(N_{ss})$			P_f	i_1	i_2
				SS	DS	TS	(mW)		
Atheros 9380	2.31	19.8	0.3	0.6	4.6	7	429.0	2.31	19.8
Intel 5300	2.95	195	0.33	3.3	4.1	4.3	496.8	2.9	195

Table 4: Receive power model for Atheros 9380 and Intel 5300.

where P_a and P_{na} are the power consumption by the 802.11n receiver during the active period T_a and non-active period T_{na}, respectively. Since data delivery occurs during the active period, we have $G \cdot T_a = S \cdot T$. Hence, E_b can be computed as

$$E_b = \frac{P_a \times T_a + P_{na} \times T_{na}}{S \times T} = \frac{P_a - P_{na}}{G} + \frac{P_{na}}{S}. \quad (2)$$

Therefore, the data source S is determined by higher-layer applications, while the goodput G is decided by rate settings and wireless channel. We next model the active and non-active power consumptions, which depend on the NIC state including both rate settings and power-saving modes.

Power Model of an 802.11n NIC: The power consumption of an 802.11n receiver can be decoupled as $P_{rx} = P_{rc} + P_{rb}$, where P_{rc} is the power consumption of MIMO circuitry, and P_{rb} is that of baseband signal processing [9]. P_{rc} includes power consumption of all circuit paths, each of which contains all the circuits from RF to analog to digital converter (ADC), e.g., frequency synthesizer, low/band-pass filter, mixer, low-noise amplifier and variable gain amplifier. Based on the MIMO power model of [9, 10], the ADC power can be estimated as a linear function of bandwidth whereas the remaining circuits consume constant power. The baseband power consumption P_{rb} scales with bandwidth and also depends on the number of receive chains. The decoder power correlates with the number of streams and rate settings; P_{rb} can be approximated as a linear function of the number of receive antennas, channel width, and rate. All these models have been validated by our measurements [17]; the details are omitted due to space limit.

Now we model the receiver power at three states: *active*, *idle* and *sleep*. The receiver active power P_{ra} includes both P_{rc} and P_{rb} for circuit and baseband processing. It can be modeled as:

$$P_{ra} = (\alpha_1 \cdot N_r + f(N_{ss})) \times BW + \alpha_2 \cdot N_r + \alpha_3 \cdot r + P_f,$$

where N_r is the number of receive antennas, N_{ss} is the number of streams, BW is the channel width (MHz), r is the rate setting (Mbps). P_f is a constant in mW, and $\alpha_1, \alpha_2, \alpha_3$ are model coefficients. These parameters are platform dependent. Table 4 list the estimated coefficients for Atheros 9380 and Intel 5300 from our measurements [17]. The active power lies in $550-1200$ mW. Note that, power consumption increases with N_{ss} and N_r. For example, the power for 3x1/81SS, 3x2/81SS and 3x3/81SS is 588.7 mW, 700.5 mW and 812.3 mW, respectively. However, given the same N_r and N_{ss}, the gap using different MCSes is negligible. For instance, 3x3/13.5SS and 3x3/135SS consume 798.8 mW and 823.1 mW, respectively, close to 812.3 mW for 3x3/81SS.

The idle power P_{ri} roughly equals to the circuitry power (P_{rc}) because of almost no processing during idle. It is thus estimated as

$$P_{ri} = i_1 \cdot N_r \times BW + i_2 \cdot N_r + P_f,$$

where i_1 and i_2 are idle power coefficients. The idle power depends on the number of antennas and channel width, but not the number of streams. For example, Atheros 9380 consumes about 541.2 mW, 653.4 mW and 765.6 mW when using 1, 2, and 3 receive antennas during idle (BW = 40MHz).

In sleep mode, we discover that few components remain active. Both Atheros 9380 and Intel 5300 NICs consume constant power (158.4 mW and 166.5 mW), saving at least $2/3$ of idle energy.

The 802.11n transmit power mainly varies with the number of transmit antennas (N_t) and channel width. In fact, the power consumption for power amplifier dominates transmit power and it is in proportion to N_t. The measured transmit power for Atheros 9380 is 1.16 W, 1.88 W and 2.64 W in case of one, two, and three transmit antennas used. The transmit power for Intel 5300 is given by [13].

5. EERA DESIGN

EERA trades off goodput for energy savings at an 802.11n client NIC. It seeks to find the MIMO setting that consumes less per-bit energy at NIC, rather than that achieves higher goodput. EERA runs at AP in its default operation mode, which transmits downlink data and reduces per-bit energy at each client receiver. Note that EERA does not offer a holistic solution that minimizes entire device energy; It only saves NIC energy from the RA perspective.

EERA is a configurable RA algorithm. Each client i specifies a threshold parameter $R_{c,i}$, which defines the minimum goodput that EERA cannot go below when selecting settings. It specifies how much a client is willing to slow down to save its NIC energy. The parameter is set in the percentage (say, 90%) of the highest goodput to the client. When $R_{c,i}$ is chosen as 100%, EERA reverts to traditional goodput-optimizing RA.

The per-client parameter $R_{c,i}$ serves as a tuning knob for EERA. It offers flexible tradeoffs between goodput and energy savings along multiple dimensions. It may help to balance energy budgets between NIC and other components of the mobile device due to communication slowdown at each client. It also facilitates to mitigate cross-client interference. Since it limits on how much an EERA client may slow down, a client can configure this parameter to reduce effect on other traditional RA clients. Moreover, this parameter may take into account application requirements (e.g., minimum throughput needed by video streaming service).

The overall idea of EERA is to let each client select the most energy-efficient setting from its feasible candidates when slowing down. However, this slowdown is contained by two factors: it must accommodate its data source rate, and not affect other clients when they were to choose their highest-goodput settings. EERA supports both single-client and multi-client operations. In the single-client case, it abstracts the problem as multi-dimensional search, and exploits ternary search and MIMO characteristics to speed up its runtime convergence. The multi-client case builds on top of single-client design, but requires additional operations. Note that each client cannot slow down too much to affect others. This is done by setting a limit, expressed in airtime share, for each client. The client cannot select a setting such that its extra communication time (due to slowdown) exceeds its airtime share, no matter how energy efficient this setting can be. To this end, AP calculates the temporal fair share (defined in airtime) for each EERA client. Assume that each client uses its highest-goodput rate and extra air time is available thereafter. The extra air time is then fairly allocated among all active clients. Each client uses EERA to minimize energy based on its fair airtime share. This way, an EERA client cannot be arbitrarily slow to hurt others. We now present detailed designs on both single-client and multi-client cases.

5.1 Single-Client Case

We first consider the simple, single-client scenario. In this case, EERA formulates the energy-efficient RA as a multi-dimensional search problem that locates the low-energy MIMO setting. It organizes settings into a multi-level tree, and then applies the ternary search scheme over each branch. At each setting, EERA uses probing to obtain the per-bit energy. The probing is "in band" by using multiple data frames sent from the AP to the client. By further exploiting the MIMO communication features, EERA can simultaneously prune multiple branches at runtime, thus eliminating those probings deemed unnecessary. Its runtime efficiency is even better than ternary search. The solution also works with/without complementary power-saving schemes (e.g., SMPS and PSMP). There are three key issues: (a) How to organize the settings into a search graph for fast lookup? (b) How to prune branches and reduce probing at runtime? and (c) How to estimate the per-bit energy for each setting? We next elaborate on these details.

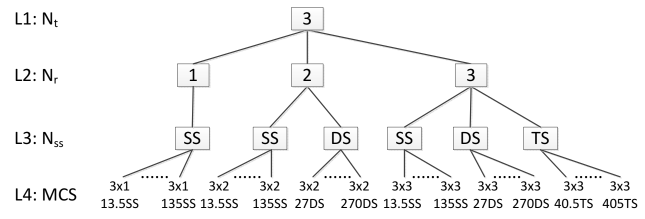


Figure 8: An example of the MIMO search tree

5.1.1 RA as Multidimensional Search

In addition to MCS rates, MIMO RA in 802.11n has to consider more dimensions: the number of transmit and receive antennas activated, and the number of data streams used. We thus abstract the problem of RA as multidimensional search. The goal is to find the setting given pre-specified optimality criteria. The traditional objective for a RA is high goodput, whereas the goal for EERA is reduced energy consumption. Since low-energy settings also depend on the data source (shown in Section 3.3), we pose RA as the following problem. Given the data source, EERA searches for lower-energy settings that can sustain the source along four dimensions: the number of transmit antennas N_t, the number of receive antennas N_r, the number of data streams N_{ss}, and the various MCS options N_{MCS}. The devised algorithm needs be efficient in its runtime complexity, as well as incurring low probing overhead. The search has to scale to large space. The search space for 802.11n, which supports four antennas, can have 80 settings. It doubles for the upcoming 802.11ac standard, which supports eight antennas.

We organize the search graph as a four-level tree, where each node denotes a setting with its estimated per-bit energy. As shown in Figure 8, the hierarchy of the tree is built following the order of N_t, N_r, N_{ss}, and N_{MCS}. Specifically, the first level is organized using the number of transmit antennas N_t, with the second level being the number of receive antennas N_r. Since the number of data streams N_{ss} is the minimum of N_t and N_r, we use it as the third level of the tree. The bottom level is the MCS options, which typically have the largest number of choices.

As the first heuristic, the AP uses the maximum number of antennas. This eliminates the top level and reduces to a three-level tree. The rationale is as follows. Our goal is to reduce energy consumption at the client with full collaboration from the AP. Therefore, it is easy to show that, given the maximum number of antennas activated at AP, the client has the largest number of choices, thus leading to better search results on energy savings.

The search tree-based abstraction also illustrates how traditional RAs work. They typically follow sequential search (e.g., MiRA and ARA) or randomized search (e.g., the MIMO version of SampleRate algorithm [19]). Consequently, these algorithms have the complexity of $O(N_t N_r N_{ss} N_{MCS})$. Take MiRA as an example at P1. The number of search steps is about 35. Assume 30Mbps data source and no PS mode with 3x3 AP at P1. The per-bit energy of all the settings is shown in Figure 3(b). MiRA will go all steps shown in Table 5 to reach the low-energy setting 3x1/40.5SS. Given each

Branch	Probing Sequence: MCS (E_b nJ/bit)	Steps
*3x1/SS	135SS(∞) $\cdots \rightarrow$ **40.5SS(19.2)** $\rightarrow$ 27SS(25.7)	7
3x2/DS	270DS(∞) $\cdots \rightarrow$ **54DS(27.6)** $\rightarrow$ 27DS(36.6)	8
3x2/SS	108SS(∞) $\cdots \rightarrow$ **54SS(22.7)** $\rightarrow$ 40.5SS(22.9)	4
3x3/TS	405TS(∞) $\cdots \rightarrow$ **81TS(30.3)** $\rightarrow$ 40.5TS(33.0)	8
3x3/DS	162DS(∞) $\cdots \rightarrow$ **81DS(29)** $\rightarrow$ 54DS(30.2)	4
3x3/SS	121.5SS(∞) $\cdots \rightarrow$ **81SS(26.4)** $\rightarrow$ 54SS(26.5)	4
Total		35

Table 5: Search steps of MiRA sequential search at P1.

Branch	$[l, r]$	Steps	# Pruned Settings
3x3/SS	$[0, 7]$	4	23
3x3/DS	$[2, 4]$	3	2
3x3/TS	$[2, 3]$	2	0
3x2/SS	$[2, 4]$	3	2
3x2/DS	$[2, 3]$	2	0
3x1/SS	$[2, 4]$	3	0
Total		17	27

Table 6: Ternary search steps with simultaneous pruning at P1.

branch with the same N_r and N_{ss}, sequential search needs to keep on probing until reaching the setting after the optimal one.

5.1.2 Ternary Search in Each Branch

Given the multidimensional search, EERA uses a novel solution technique, called ternary search with simultaneous pruning, to locate low-energy setting in EERA. The resulting algorithm is more efficient than sequential or randomized search. We start the search at the lowest-level, i.e., all the MCS rates given fixed receive antennas and the number of streams. The proposed ternary search uses the following property (its proof is in [17]):

Property I: The per-bit energy E_b is a unimodal function with respect to the MCS rate, given fixed number of chains and fixed number of streams.

The above property makes case for ternary search. Note that binary search cannot be applied since E_b is not a monotonic function with respect to MCS rates. We sort the MCS rates in the increasing order based on their indices, say, $[l, r]$, and find the MCS rate that yields lower per-bit energy. In ternary search, we select two intermediate points that partition the interval into three equal segments, i.e., $m_1 = l + (r - l)/3$; $m_2 = r - (r - l)/3$, as shown in Figure 9. There are three cases: (1) if $f(m_1) < f(m_2)$, then the minimum cannot be on the right side $[m_1 + 1, r]$. We then search only the left side $[l, m_1]$; (2) if $f(m_1) > f(m_2)$, then the situation is similar. The minimum cannot be on the left $[l, m_2 - 1]$, so we go to the right side - $[m_2; r]$; (3) If $f(m_1) = f(m_2)$, then the search should be conducted in $[m_1, m_2]$. It can be solved recursively by referring to the first two cases.

An illustrative example is given in Figure 10. We apply ternary search on each branch. Take the branch 3x1/SS as an example. Initially, the indices of the MCS rates are $[0, 7]$, and we choose two intermediate rate settings, 40.5SS and 108SS, to partition the branch into three segments, i.e., $m_1 = 2$; $m_2 = 5$. The former setting can achieve 19.2 nJ/bit, whereas the latter gets high per-bit energy due to high loss. Therefore, the minimum of per-bit energy is located in the interval $[0, 4]$. Then, the next intermediate points picked for probing are 54SS and 27SS. Finally, we can locate the optimal setting 40.5SS over this branch. After each branch is traversed, the best setting, 3x1/40.5SS, can be reached. The number of total search steps is 25.

5.1.3 Simultaneous Pruning of Branches

It turns out that EERA can be more efficient than the above tree-based scheme, which uses ternary search over each lowest level. The technique is to simultaneously prune the search space and reduce probing at each step. Consequently, it not only eliminates some branches for further lookup based on runtime search results, but also reduces the range for the ternary search (i.e., l or r). The technique exploits the MIMO communication characteristics. It has two concrete cases at each search step, depending on whether the MCS rate used by the setting has exceeded the channel capacity.

Low-loss probing The first case is when the probing of the rate at the current search step does not result in high packet loss, i.e., it yields reasonable goodput. In such a case, we can apply the first rule to eliminate some lower MCS rates from further ternary search, given the fixed numbers of chains and streams. It uses the following property:

Property II: The lower bound of a setting's per-bit energy can be estimated from its loss-free goodput.

Specifically, whenever current probing finds a new setting with lower per-bit energy, we use the above rule to eliminate those lower-MCS-rate settings, which cannot have the same per-bit energy even in the loss-free case. The rationale is that, when a setting achieves its maximum goodput (in the loss-free case), it obtains the lowest per-bit energy. When such bounds cannot beat the current setting, these MCS rates can be removed from the search process.

High-loss probing The second case is when the probe of the rate incurs high packet loss (say, larger than a threshold such as 90%), thus giving very low goodput. This tells us that the current MCS rate exceeds the channel capacity. We then eliminate some settings from further search based on the following property:

Property III: Loss monotonically increases with (1) MCS rate, given the same N_r, N_{ss}; (2) decreasing N_r, given the same MCS rate and N_{ss}; (3) N_{ss}, given the same MCS rate and N_r.

Since the probe at the current MCS m fails, then we can eliminate two more scenarios, both of which would yield higher loss (i.e., further exceeding the channel capacity). The first is the settings with the MCS higher than or equal to m and the number of chains lower than N_r. These settings would also fail in probing, given the same N_{ss}. The second scenario concerns those settings with MCS higher than or equal to m, the number of streams higher than N_{ss}, and the number of chains lower than or equal to N_r. These settings would also fail.

With both pruning heuristics, we further reduce the search steps to 17 by excluding settings at runtime, as shown in Table 6. During the search of the 3x3/SS branch, we prune up to 23 settings at other branches. For example, high-loss probing at 3x3/108SS triggers the following 15 settings to be pruned based on *Property III*: those higher than or equal to 3x3/216DS, 3x3/324TS, 3x2/108SS, 3x2/216DS, and 3x1/108SS. Moreover, based on *Property II*, the per-bit energy of 3x3/81SS (i.e., 26.4 nJ/bit) helps to remove the following 8 settings, with their loss-free goodput lower than 3x3/81SS: the ones lower than or equal to 3x3/54DS, 3x3/81TS, 3x2/27SS, 3x2/DS, and 3x1/13.5SS. The pruning incurred by the 3x3/SS branch results in smaller search space, $[2, 4]$, at 3x3/DS. The continuous pruning reduces the remaining search space at each branch to only 2 to 3 MCS rates.

5.1.4 Estimation at Each Setting

We need to calculate two metrics: the per-bit energy for each setting, and the data source rate. For a given setting, we need to estimate its consumed per-bit energy E_b. In EERA, we compute instantaneous E_b upon receiving every aggregate frame at the receiver using Equation (2). The active and non-active power at a given setting can be measured *a priori*, since they do not change at runtime. The goodput is computed upon the arrival of an aggre-

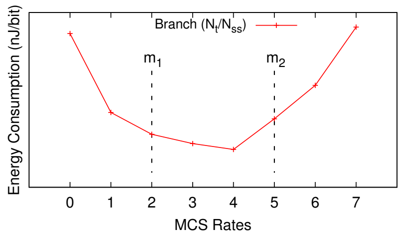


Figure 9: Ternary search.

Branch	Probing Sequence: MCS (E_b nJ/bit)	Steps
*3x1/SS	**40.5SS(19.2)** $\rightarrow$ 108SS(∞) $\rightarrow$ 54SS(19.3) $\rightarrow$ 27SS(25.7)	4
3x2/DS	**81DS(22.9)** $\rightarrow$ 216DS(∞) $\rightarrow$ 108DS(∞) $\rightarrow$ 54DS(27.6)	4
3x2/SS	40.5SS(22.9) $\rightarrow$ 108SS(∞) $\rightarrow$ **54SS(22.7)** $\rightarrow$ 81SS(26.0)	4
3x3/TS	121.5TS(32.9) $\rightarrow$ 324TS(∞) $\rightarrow$ 162TS(∞) $\rightarrow$ **81TS(30.3)** $\rightarrow$ 40.5TS(33.0)	5
3x3/DS	**81DS(29)** $\rightarrow$ 216DS(∞) $\rightarrow$ 108DS(39.7) $\rightarrow$ 54DS(30.2)	4
3x3/SS	40.5SS(26.6) $\rightarrow$ 108SS(∞) $\rightarrow$ 54SS(26.5) $\rightarrow$ **81SS(26.4)**	4
Total		25

Figure 10: Search steps of EERA ternary search at P1.

gate frame. The source rate is also estimated following the procedure described next. Once we obtain the instantaneous E_b for each probe frame, we estimate the moving average E_b at time t, denoted by $\overline{E_b}(t)$, using the instantaneous per-bit energy $E_b(t)$ and the standard procedure $\overline{E_b}(t) = (1-\alpha)\cdot\overline{E_b}(t-1) + \alpha\cdot E_b(t)$, where $\alpha = \frac{1}{8}$ is the weighting factor. This can smoothen out transient variations while tracking the evolving trend.

We also estimate the data source rate, which affects the energy-efficient setting. The estimation is implemented at the transmitter buffer. Upon each frame arrival or departure at the buffer, the instantaneous source rate can be estimated as $S(t) = G(t) + \Delta Q(t)$, where $G(t)$ is the outgoing goodput, and $\Delta Q(t)$ is the buffer change at t. We then compute the moving average of $S(t)$, using procedures similar to $\overline{E_b}(t)$. This way, we estimate the source rate by monitoring the change of data buffer and outgoing goodput.

We further make a somewhat counter-intuitive observation. In the 802.11n context, buffering packets at the source and transmitting them in a batch might not always help in energy savings in EERA. Note that, EERA selects the proper energy-efficient setting to afford the traffic source. Amortizing transmissions in batches tends to over-estimate the source rate, typically resulting in another higher-goodput setting that is less energy efficient. Consider that EERA uses a setting 3×1/13.5SS for a 5Mbps source. When we buffer this flow for 90% time, it becomes a flow of 50 Mbps for 10% time and idle for the rest 90% time. Consequently, EERA has to select a higher-goodput setting 3×3/81DS to accommodate this source. Though this setting transmits faster, the overall energy during both active and non-active periods may still be larger than the one at the setting of 3×1/13.5SS. The formal statement of this results in a theorem format is available in [17]. Note that this is different from frame aggregation in 802.11n, which EERA always uses to amortize transmission overhead.

It should be noted that EERA can work with or without the other power-saving schemes, such as SMPS and PSMP. These schemes only change the per-bit energy at a given setting by having smaller non-active power P_{na}. They work together with EERA since these two complement with each other by primarily managing the active and idle periods, respectively. EERA also has mobility and interference handling mechanisms, similar to the design in the literature [19]. We omit the details due to space limit. Finally, EERA has nice runtime complexity (its proof is in [17]):

THEOREM 1. *(Search Complexity) Assume that the increase of power consumption at the MIMO receiver with the MCS rate is negligible. EERA has a worst-case search complexity no worse than* $O(\log N_{MCS}\cdot N_r\cdot N_{ss})$.

5.2 Multi-Client Case

In the multi-client scenario, EERA uses additional mechanisms to prevent its clients from hurting others (running EERA or traditional RA schemes such as ARA/MiRA). An EERA client selects lower-goodput (but more energy-efficient) settings only if it does not affect other clients' transmissions when they were to use their highest-goodput rates. Specifically, a client is given certain amount of extra airtime it can use to slow down for energy savings. The extra airtime is allocated through a temporal fair share, which helps to isolate one client from another during transmission. Each client can only use up its fair share of airtime to slow down for energy savings, but cannot spend more time designated for other clients.

Specifically, EERA runs over regular time intervals (called epoch, and its duration is T_{ep}) periodically. During each epoch, it has three phases of operations. In the first phase, AP probes each client for its highest-goodput setting. This can be done via a traditional MIMO RA algorithm such as ARA or MiRA. We can refine ARA and MiRA by eliminating their sequential search. We abstract the problem as multi-dimensional search, and apply binary search over each tree branch, similar to Sections 5.1.1 and 5.1.2, but for goodput instead of energy. During the second phase, AP calculates the temporal fair share for each client. The fair airtime share stipulates how much extra time each client may spend when slowing down to save energy. During the third phase, EERA selects the most energy-efficient setting given the constraints set by the airtime share and pre-configured threshold $R_{c,i}$ for client i.

The fair share calculation is as follows. Assume every client uses its highest-goodput rate setting during epoch k. Given the highest goodput $G_{c,i}$ and the source rate $S_{k,i}$ for client i, its used airtime percentage is given by $\frac{S_{k,i}}{G_{c,i}}$. The unused airtime (in percentage) by all n clients during epoch k is thus obtained as $1-\sum_{i=1}^{n}\frac{S_{k,i}}{G_{c,i}}$. EERA equally allocates this extra airtime among all n clients. Therefore, during epoch k with duration T_{cp}, each client i is allocated airtime share $F_{k,i}$ as $(1-\sum_{i=1}^{n}\frac{S_{k,i}}{G_{c,i}})\cdot\frac{T_{ep}}{n}$. If client i cannot use up its airtime share (say, limited by its parameter $R_{c,i}$), we then allocate fair share based on the celebrated max-min fairness [12]. Note that other fairness index (e.g., proportional fairness) may also be used to allocate the extra airtime in EERA.

Once each client is allocated its airtime share, it can effectively apply operations during each epoch, similar to the single-client case of Section 5.1. The minor difference is that, tree branches can be further pruned by both parameters of pre-configured threshold $R_{c,i}$ and fair share $F_{k,i}$. The rule is that the selected setting cannot exceed the airtime share, nor yields goodput lower than $R_{c,i}$ percent, compared with its highest-goodput setting during current epoch.

5.3 Other Issues

There are a number of additional issues to consider in EERA.

Coexistence of EERA and other MIMO RA clients EERA may coexist with clients running other traditional 802.11n RA algorithms. Its client ensures that other clients can always use their highest-goodput rates. Only when extra airtime is available (e.g., sources of other clients are not fast enough), does it slow down to save NIC energy. When clients running ARA/MiRA are greedy, EERA reverts to goodput-optimizing RA mode.

Greedy clients Greedy clients (e.g., those running TCP flows) always demand highest goodput. For these clients, we set the pre-configured parameter $R_{c,i}$ as a fixed percentage (say, 90%). The value $1 - R_{c,i}$ is interpreted as the percentage of goodput the client is willing to give up for NIC energy savings during each epoch.

Uplink case EERA can be extended to support the uplink case. Using client transmit power model, EERA minimizes $\frac{P_a(tx)-P_i}{G_{UL}}$, where $P_a(tx)$ is the active transmit power, G_{UL} is the uplink goodput, and P_i, is the idle power. Transmit power is dominated by the power amplifier, while baseband processing is much smaller. Hence, at each setting, power consumption of different MCS rates varies slightly, less than 5% based on our measurements. The mixed uplink and downlink case can also be supported similarly. AP acts as the coordinator by collecting the required information to calculate fair share for each uplink/downlink client. It then notifies each uplink client its airtime share.

Ad-hoc mode EERA currently does not support ad-hoc operations. There are two associated challenges: (1) How to allocate fair share of airtime in the multihop setting? (2) How to coordinate RA operations among multiple clients in a fully distributed manner? We plan to study these issues in the future.

6. IMPLEMENTATION

We implement EERA in the open-source driver, ath9k, for Atheros 802.11n WiFi chipsets. EERA resides at the transmitter. To save energy at the client, AP coordinates clients to configure their receive RF chains for downlink transmissions, whereas each client configures its transmit chains for uplink traffic. During the association phase, a client enables EERA at AP by issuing a request with mandatory parameters, including the maximal number of receive antennas, power parameters, as well as its non-active power parameters under different power-saving schemes. All such messages are exchanged by a new 802.11n management frame.

Two technical issues arise in EERA operations. First, how to work with other power-saving schemes? Once the client changes its power-saving scheme (e.g., enabling SMPS), it notifies its AP (e.g., via sending a SMPS frame). Upon receiving this notification, the EERA module at AP automatically updates the client's power-saving mode and estimates its per-bit energy. Second, transient loss may occur during the switching process of the client's receive chains due to the inconsistent views of the chains between both sides. For example, after the client switches its receive chain setting from three to two, EERA at AP may still use the TS rates before the acknowledgement from the client is received. Consequently, the client cannot decode these packets with only two receive chains. During chain switching, EERA uses rates accepted by both settings.

7. PERFORMANCE EVALUATION

We conduct extensive experiments to assess EERA performance. The main criterion is energy savings at the client NIC. In addition to ARA and MiRA algorithms, we also compare EERA with MRES [20], a recent proposal to improve 802.11n energy efficiency by adjusting the number of RF chains. The proposed MiRA and MRES work with maximum two receive chains and DS modes. We extend both to support three chains in TS mode. We conduct extensive tests in static office environment (see Figure 1), with various factors of client location, wireless configuration and traffic pattern. We also examine EERA in more scenarios of mobility, interference, uplink traffic, multiple-client settings, and field trials. In our experiment, both AP and the client support three antennas, working on 40MHz channel over 5GHz band. The default setting is to use UDP-based downlink transmission to a client without enabling any power-saving mode.

	ARA	MiRA	MRES
Static UDP	(13.4-35.6) %	(14.3-36.1) %	(5.8-26.8) %
Static TCP	(5.1-20.5) %	(10.4-32.3) %	(7.3-23.8) %
Application	(26.5-33.9) %	(26.6-35.2) %	(6.7-36.5) %
Mobility	27.8 %	30.1 %	20.3 %
Field Trials	31.7 %	33.1 %	24.1 %

Table 7: NIC energy savings of EERA over other designs.

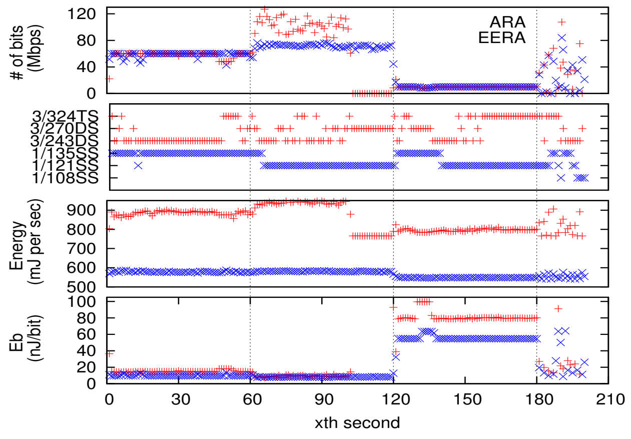


Figure 11: Trace of EERA and ARA in hybrid-traffic example.

A quick summary of EERA performance is as follows. In all test scenarios, EERA consistently outperforms other algorithms in terms of NIC energy efficiency. Table 7 summarizes its energy-saving percentage in major test settings, which also includes field trials in an office building. In general, EERA saves about 30% energy compared with ARA/MiRA in all scenarios (equivalent to energy waste of 43% by ARA/MiRA). Compared with MRES, EERA saves about 6-36% NIC energy in static settings and 20-24% in mobility and field tests.

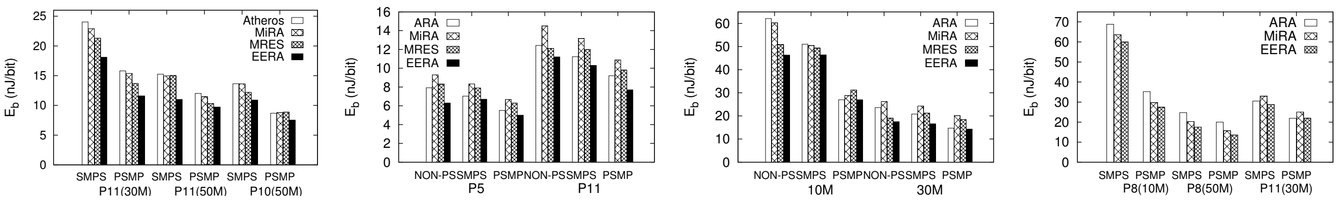
7.1 An Example of EERA Performance

We first test EERA at a static location P5 with hybrid traffic patterns. It runs 210 seconds, including the first minute with 60 Mbp UDP traffic, the second minute with 500 MB TCP-based file downloading, the third minute with 10 Mbps UDP traffic, and the last 30 seconds for ten small (<10MB) file downloading. Figure 11 plots the traces of the delivered bits, the selected major rate settings, energy consumption, and per-bit energy over time for both ARA and EERA. The NIC energy saving of EERA over ARA and MiRA ranges between 29.2–35.1%, whereas the gain over MRES is 6.8–20.8% (6.8% for 10M UDP). Therefore, EERA outperforms all three other algorithms in terms of NIC energy consumption. This is because EERA selects a more energy-efficient setting (typically a lower-rate setting, 3x1/135SS, 3x1/121SS, or 3x1/108SS), which balances energy and goodput. EERA outperforms MRES because it quickly locates the energy-efficient setting, while MRES incurs more overhead since it runs on top of conventional RA algorithms.

7.2 Single Client Under Various Factors

We now evaluate energy efficiency of EERA in various single-client scenarios, including at different locations, with different wireless configurations (e.g., the number of AP antennas, frame aggregation, and power-saving modes), under a variety of traffic sources (e.g., source rates, packet size, applications), and in settings with mobility, interference and uplink transmissions. The detailed results are reported in [17]. We next summarize our main findings.

Client locations: We vary the client location (from P1 to P13) to examine how EERA performs under various wireless channels



(a) Power saving modes (b) TCP (c) Interference (d) Uplink UDP

Figure 12: Per-bit energy consumption of a single client under various factors.

with 30 Mbps source. EERA consistently outperforms other algorithms, with more than 30% energy savings over both ARA and MiRA, and 8.6–22.2% savings over MRES. The trace analysis shows that, EERA always chooses and stays at the energy-efficient setting adaptive to wireless channels. Note that the gain becomes smaller at far locations. It is because the wireless link becomes too weak, leaving little room to trade off goodput for energy savings.

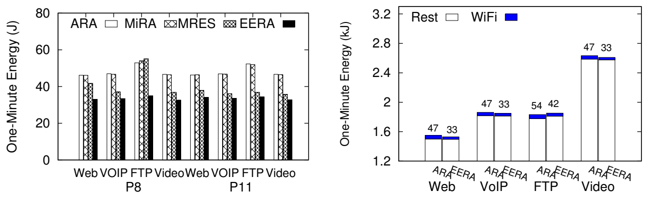
Number of AP antennas: We change the number of AP antennas from one to three, and observe its impact on EERA. Our results indicate that, EERA always outperforms other algorithms. For instance, for the client with a 30 Mbps source at P3, the gain is 18.5%, 33%, 33% over ARA and MiRA, and 16.5%, 25.2% and 9.6% over MRES, when one, two, and three AP antennas are activated, respectively. It confirms our finding in Section 3.3, which states that, fewer AP antennas may reduce the per-bit energy gap between the highest-goodput and most-energy-efficient settings.

Frame aggregation: We study the impact of frame aggregation on EERA by limiting the maximum aggregation level (from 0% to 100%). Frame aggregation has little effect on EERA energy efficiency unless the EE setting chosen by EERA cannot sustain its traffic source at lower aggregation. A rare example is observed when the maximum aggregation decreases from 40% to 20%, given a 30 Mbps source at far location P11. In this case, EERA has to pick up a higher-rate setting to sustain its traffic source since frame aggregation is low. Its energy savings drop from 32.8% to 12.1% subsequently. In general, frame aggregation is beneficial to EERA since it helps to reduce transmission overhead.

Power-saving modes: We also evaluate EERA when the client enables different power-saving modes of SMPS and PSMP. Figure 12(a) plots the per-bit energy at two locations P10 and P11 under two source rates. Compared with ARA, MiRA and MRES, EERA still yields energy savings from 13.37% to 28%, from 14.31% to 26.6%, from 5.96% to 26.8%, respectively. When compared with the case without power-saving modes, the reduction in energy savings is attributed to decreased power at the idle/sleep state. Smaller non-active power favors faster transmission so that the client enters into idle/sleep mode early. However, we note that, racing to sleep cannot ensure highest energy efficiency at NIC. If the active rate is not selected properly, energy waste during active period cannot be offset by the reduction in idle/sleep energy consumption.

Traffic sources: We now assess EERA under various traffic patterns. We vary UDP source rates (from 10 Mbps to 80 Mbps) and packet sizes (from 100 B to 1400 B) at different locations. EERA outperforms all other algorithms. The source rate change does affect the per-bit energy consumption (the slower source, the larger E_b), but not much on the saving gain; EERA achieves about 30% energy savings over both ARA and MiRA, and 5.1–12.4% over MRES at near and far clients (i.e., at P8 and P11, respectively). Regarding packet size, we observe that they impose negligible impact on per-bit energy of all RA algorithms, because frame aggregation minimizes the potential overhead due to small packets.

We also conduct experiments with TCP flows. Figure 12(b) demonstrates that, EERA consistently outperforms others no matter whether any power-saving mode is enabled or not. EERA yields energy savings at NIC from 5.1% to 20.5% over ARA, more than 19% over MiRA, and from 7.3% to 23.8% over MRES. This shows that, EERA estimates dynamic TCP source rate well and locates the EE setting quickly.



(a) NIC (b) Device at P10

Figure 13: Energy consumption under various applications.

Applications: We further gauge EERA for four popular applications: Web, VoIP, FTP and Video streaming. Their settings are described in Section 3.4. Figure 13(a) plots the per-bit energy of clients at P8 and P11 for these applications. EERA outperforms all three algorithms, since it handles a variety of traffic patterns in terms of source rate, traffic dynamics and different packet sizes. Its saving percentage at NIC ranges between 26.5–33.9%, 26.6–35.2%, and 6.7–36.5% over ARA, MiRA and MRES, respectively.

We also measure device-level energy consumption under these applications. The results are shown in Figure 13(b). At the device level, all components (except NIC) consume 4.1J, 1.9J, and 9.8J less energy when EERA runs for Web, VoIP and Video streaming applications, compared with ARA. These correspond to the energy gap of 0.27%, 0.11% and 0.39%, respectively. For FTP, EERA wastes 1.77% energy compared with ARA, i.e., about 31.5J during one minute. This is because FTP application program stops consuming additional energy once a file transfer completes. In contrast, Web, Video and VoIP applications need to remain active after data transfer for user interaction. For NIC only, EERA outperforms ARA for all four applications of Web, VoIP, Video, and FTP, with savings of 0.95%, 0.75%, 0.54%, 0.66%, respectively. It correspondingly saves 14.3J, 13.6J, 14.0J, 11.8J during a minute.

Mobility: In order to measure the efficiency of EERA and its probing effectiveness, we move a client from P6 to P1 through P4 and P2, and then go back to P6 at approximately constant, pedestrian speed of 1 m/s. AP sends 30Mbps UDP source to the client. In this mobility case, EERA outperforms ARA, MiRA, MRES with NIC energy savings of 27.8%, 30.1%, and 20.3%, respectively. Moreover, Table 8 lists the major rate settings selected by all RA algorithms, at each location. It shows that EERA is still able to locate the energy-efficient settings during mobility.

We further study its probing cost. EERA, as well as MiRA and MRES, uses a single aggregate frame to probe each setting. Per setting, there are up to four transmissions until they succeed. Our trace analysis shows that, EERA is able to exclude most less-energy-efficient settings with simultaneous pruning. For example,

	P6 → P4 → P2 → P1 → P2 → P4 →P6	E_b
EERA	3x1/108SS → 3x1/108SS → 3x1/81SS → 3x2/54SS → 3x1/81SS → 3x1/108SS → 3x1/108SS	19.7
MRES	3x1/135SS → 3x1/121.5SS → 3x2/108DS → 3x2/54SS → 3x1/81SS → 3x1/121.5SS → 3x1/135SS	24.7
ARA	3x3/324TS → 3x3/243DS → 3x3/162DS → 3x3/108DS → 3x3/108DS → 3x3/216DS → 3x3/324TS	27.3
MiRA	3x3/324TS → 3x3/243DS → 3x3/162DS → 3x3/108DS → 3x3/162DS → 3x3/243DS → 3x3/324TS	28.2

Table 8: Selected rate settings over locations during mobility.

EERA needs to probe four settings to locate the optimal 3x1/108SS at P4, whereas MiRA and MRES probe five and ten settings, respectively. Moreover, EERA starts from lower-rate settings with a higher chance to success. In fact, only 7 frame transmissions are needed in EERA, whereas 15 and 29 frame transmissions are for MiRA and MRES, respectively. The lower probing overhead also contributes to energy efficiency of EERA compared with MRES.

Interference: We gauge the performance of EERA in the interference scenario by placing the client into the crowded 2.4GHz band (Channel 11), on which we sniff more than 10 APs. The channel width is switched from 40 MHz to 20 MHz. We evaluate this case for different traffic sources and power-saving schemes at P12. Results are shown in Figure 12(c). Although external interference reduces the gain of EERA due to more packet losses and collisions, the NIC energy savings still reach up to 25.8%, 33.3% and 22.1%, compared with ARA, MiRA and MRES, respectively.

Uplink traffic: We also evaluate EERA for uplink traffic in the power-saving modes of SMPS and PSMP. In this experiment, the clients at P8 and P11 send UDP traffic to the AP. Figure 12(d) shows that, EERA outperforms others with energy savings up to 32%. Large gain is observed at close location P8. It is because EERA mainly uses one transmit chain for uplink transmission, whereas ARA and MiRA always activate three transmit chains.

7.3 Multi-Client Scenarios

We now evaluate EERA in multi-client scenarios. We consider three cases: (1) multiple EERA clients associated with the same AP, (2) EERA and non-EERA (ARA is used here) clients coexisting within an AP, and (3) EERA and ARA clients coexisting with two co-located APs. We test with two clients and three clients in these these cases. The detailed configuration is shown in Table 9. The benchmark scenario is when all clients use the same ARA algorithm. The goal is to see whether and how an EERA client affects others when slowing down its transmission. We are particularly interested in assessing how well the fair share mechanism in EERA works. To this end, we evaluate two versions of EERA, with and without the fair share mechanism (called F-EERA and Naive EERA (N-EERA), respectively).

Settings	C_1	C_2
2C-Bench	ARA	ARA
2C-EERA	EERA	EERA
2C-Coexist	ARA	EERA

Settings	C_1	C_2	C_3
3C-Bench	ARA	ARA	ARA
3C-EERA	EERA	EERA	EERA
3C-Coexist	ARA	EERA	EERA

Table 9: Experimental settings for multiple clients (Left: two clients; Right: three clients).

Multiple EERA clients: We examine the overhead incurred by EERA in terms of packet delay D_p, and then evaluate its energy efficiency. In the experiment, clients C_1, C_2 and C_3 are placed at P6, P8 and P11, respectively; the traffic source at C_2 varies its rate from 10 Mbps to 70 Mbps; the source of other clients is 10 Mbps. Figures 14(a) and 14(b) plot the average packet delay in 2C-EERA and 3C-EERA scenarios, compared with their ARA benchmark settings. The result shows that, F-EERA is able to achieve comparable packet delivery latency as ARA does. The latency increase in F-EERA is within 0.2 ms per packet in the 2C-EERA case, and 0.13 ms per packet in the 3C-EERA scenario. As the source rate of client C_2 increases, F-EERA has to accommodate multi-client traffic demand and allocate extra air time among all clients in a fair manner. Client C_1 is consequently allocated with smaller airtime share, and F-EERA forces it to select a higher-goodput setting compared with the case using N-EERA. It thus adapts to multi-client traffic load with reasonable overhead. In contrast, N-EERA further increases packet latency when client C_1 slows down for its NIC energy efficiency. It also hurts other clients in packet latency under high traffic demand. Figure 14(c) plots the per-bit energy consumption for both clients in the two-client scenario. As the source rate increases, energy-saving gain also decreases. F-EERA saves 30.5% and 29.7% for C_1 and C_2, respectively, given a 10 Mbps data source. The saving reduces to 5% for C_1 when data source increases to 70 Mbps, in which case the aggregate traffic demand is close to the link capacity. Similar results are also observed in the 3C-EERA case. In summary, F-EERA chases for energy savings when the aggregate traffic demand is low, and behaves similarly to conventional RA algorithms when the traffic demand is high.

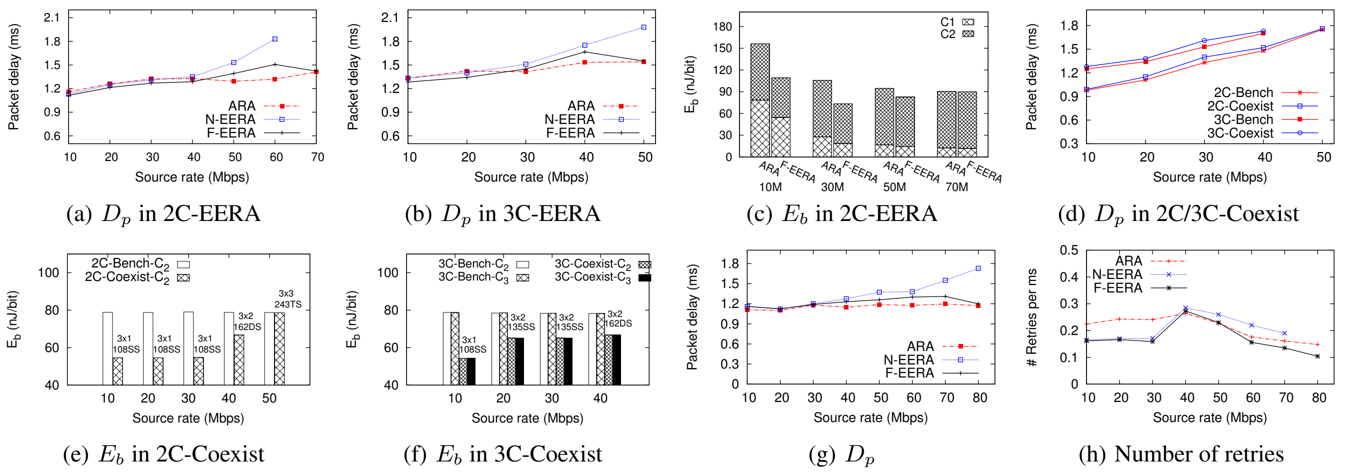
Coexistence of EERA and ARA clients: Now EERA and ARA clients coexist with the same AP. The settings are identical to above experiments, except that client C_1 runs ARA. We seek to examine whether EERA clients affect other conventional RA clients. We also explore how the energy efficiency of EERA clients is affected. Figure 14(d) plots the average packet delay at C_1 when other clients run F-EERA. The delay gap between benchmark and coexistence cases is below 0.08 ms per packet, thus negligible. Figures 14(e) and 14(f) plot the per-bit energy for F-EERA clients (clients C_2 and C_3) in the 2C-Coexist and 3C-Coexist settings. Similar to Figure 14(c), energy saving decreases as the aggregate traffic demand grows. For instance, energy-saving gain reduces from 30.8% to 0.1% in the 2C-Coexist case, and decreases from 31.0% to 14.6% in the 3C-Coexist case. .

Coexistence of EERA/ARA clients with two APs: We further study how an EERA client coexists with other RA (ARA is used here) client in the two-AP scenarios. In the test settings, client C_1 always interacts with an AP running ARA. However, client C_2 receives data from another AP running ARA, N-EERA, and F-EERA, respectively, at P5. These two APs are co-located in spatial proximity and can hear each other. Both contend for the same channel. Figures 14(g) and 14(h) plot the average packet delay and the number of retries per millisecond at client C_1. The number of retries serves as an indicator for channel contention between two AP-client pairs. The results show that, F-EERA incurs comparable overhead as ARA, with the packet-delay gap being at most 0.12 ms. As for the number of retries, F-EERA performs even better than ARA. We gauge that it is because more active transmission time reduces the likelihood of contentions.

In summary, EERA compares well with ARA in multi-client scenarios. In case of multiple EERA clients, it increases per-packet delay by at most 14.2% (0.19ms) and 8.7% (0.13ms), but saves NIC energy by 30.8% and 31%, in 2-client and 3-client cases, respectively. For co-located EERA and ARA clients, it increases packet delay by at most 5.3% (0.07ms) and 5.2% (0.08ms) in 2-client and 3-client coexistence settings, respectively. For contention, EERA increases at most 3.4% retries (0.009 retries/ms), but avoids up to 34% retries (0.08 retries/ms) when the traffic source rate is low.

8. RELATED WORK

Numerous RA algorithms [6, 8, 11, 14, 19, 21–23] have been proposed in the literature. All aim to achieve high goodput, rather than energy efficiency. Moreover, these proposals typically use sequential or randomized search; the search does not scale well to 802.11n/ac where search space is bigger. Other MIMO RA propos-



(a) D_p in 2C-EERA (b) D_p in 3C-EERA (c) E_b in 2C-EERA (d) D_p in 2C/3C-Coexist

(e) E_b in 2C-Coexist (f) E_b in 3C-Coexist (g) D_p (h) Number of retries

Figure 14: Energy efficiency and packet delay in multi-client scenarios.

als (e.g., ESNR [14] and Soft-Rate [21]) cannot be implemented in current commodity 802.11n platforms.

Recent theoretical studies on energy-efficient MIMO systems seek to find the crossover point, which trades off MIMO gains at the cost of increased power consumption [9, 16]. Both cannot be used on commodity platforms. Several research studies [10, 13, 15, 20] have focused on energy savings in MIMO systems. [10] seeks to find the most energy-efficient settings only for transmission period, using their MIMO-OFDM based software-defined radio; it is not 802.11n standard compliant. [13] identifies factors that affect energy consumption on 802.11n commodity hardware. MRES [20] examines the strength and limitation of SMPS, and proposes a energy-saving solution through dynamically adjusting chain settings. Snooze [15] schedules client sleep time, and configures chains for energy savings. All these efforts do not address the problem from RA perspective. Moreover, EERA also complements and works with power-saving mechanisms such as SMPS and PSMP.

9. CONCLUSION

Rate adaptation for 802.11n devices is more complex than that in legacy 802.11a/b/g systems, since it has to adjust over multi-dimension PHY parameter space. Various proposals [6, 8, 11, 14, 19, 21–23] have so far focused on improving goodput. However, we show that, this is possibly achieved at higher energy cost at NICs. In the race for higher speed in wireless technologies (e.g., 802.11n and 802.11ac WLAN, and 4G LTE WWAN to name a few), we believe that energy efficiency is equally important. The technology has to balance between energy and speed. EERA reports our effort on adapting RA to improve NIC energy efficiency.

10. ACKNOWLEDGMENT

We highly appreciate the constructive and insightful comments by our TPC shepherd and the anonymous reviewers, and the professional help from TPC co-chair Dr. Alex C. Snoeren during the shepherding process.

11. REFERENCES

[1] Intel PowerTop 2.0. `https://01.org/powertop/`.
[2] Intel Core 2 Duo Processor Datasheet, Jan. 2008.
[3] Atheros, Marvell to Push Fast 11n Into Phones, Feb. 2011. `http://www.pcworld.com/article/219395/atheros_marvell_to_push_fast_11n_into_phones.html`.
[4] IEEE P802.11 Wireless LANs Proposed TGac Draft Amendment, Jan 2011.
[5] ABI Research. Demand for 802.11n Spurs Wi-Fi Equipment Market to 18 Million Units in 2Q 2010, Nov. 2010.
[6] P. A. Acharya, A. Sharma, E. M. Belding, K. C. Almeroth, and K. Papagiannaki. Rate Adaptation in Congested Wireless Networks through Real-time Measurements. *IEEE Transactions on Mobile Computing*, 9(11), Nov. 2010.
[7] A. Carroll and G. Heiser. An Analysis of Power Consumption in a Smartphone. In *USENIX Annual Technical Conference*, 2010.
[8] X. Chen, D. Qiao, J. Yu, and S. Choi. Probabilistic-Based Rate Adaptation for IEEE 802.11 WLANs. In *IEEE Globecom*, 2007.
[9] S. Cui, A. J. Goldsmith, and A. Bahai. Energy-constarined Modulation Optimization. *IEEE Trans. Wireless Communications*, 4(5), 2005.
[10] W. Gabran and B. Daneshrad. Hardware and Physical Layer Adaptation for a Power Constrained MIMO OFDM System. In *IEEE ICC*, 2011.
[11] A. Gudipati and S. Katti. Strider: Automatic Rate Adaptation and Collision Handling. In *SIGCOMM*, 2011.
[12] E. L. Hahne. Round-Robin Scheduling for Max-Min Fairness in Data Networks. *IEEE Journal on Selected Areas in Communications*, 9:1024–1039, 1991.
[13] D. Halperin, B. Greenstein, A. Sheth, and D. Wetherall. Demystifying 802.11n Power Consumption. In *HotPower*, 2010.
[14] D. Halperin, W. Hu, A. Sheth, and D. Wetherall. Predictable 802.11 Packet Delivery from Wireless Channel Measurements. In *SIGCOMM*, 2010.
[15] K.-Y. Jang, S. Hao, A. Sheth, and R. Govindan. Snooze: Energy Management in 802.11n WLANs. In *ACM CoNEXT*, 2011.
[16] H. Kim, C.-B. Chae, G. De Veciana, and R. W. Heath. A Cross-layer Approach to Energy Efficiency for Adaptive MIMO Systems Exploiting Spare Capacity. *IEEE Trans. on Wireless Communication*, 8:4264–4275, Aug. 2009.
[17] C. Li, C. Peng, and S. Lu. Achieving 802.11n NIC Energy Efficiency through Rate Adaptation. Technical report, UCLA Computer Science, 2012.
[18] A. Mahesri and V. Vardhan. Power Consumption Breakdown on a Modern Laptop. In *PACS*, 2004.
[19] I. Pefkianakis, Y. Hu, S. H. Wong, H. Yang, and S. Lu. MIMO Rate Adaptation in 802.11n Wireless Networks. In *MOBICOM*, 2010.
[20] I. Pefkianakis, C.-Y. Li, and S. Lu. What is Wrong/Right with IEEE 802.11n Spatial Multiplexing Power Save Feature? In *ICNP*, 2011.
[21] M. Vutukuru, H. Balakrishnan, and K. Jamieson. Cross-layer Wireless Bit Rate Adaptation. In *SIGCOMM*, 2009.
[22] M. Wong, J. M. Gilbert, and C. H. Barratt. Wireless LAN using RSSI and BER Parameters for Transmission Rate Adaptation, 2008. US patent, 7,369,510.
[23] S. H. Y. Wong, H. Yang, S. Lu, and V. Bharghavan. Robust Rate Adaptation for 802.11 Wireless Networks. In *MOBICOM*, 2006.