Exploring Smart Pilot for Partial Packet Recovery in Super Dense Wireless Networks

Xiaoke Qi*, Lu Wang†, Kaishun Wu†, Jianhua Tao*
*Institute of Automation, Chinese Academy of Sciences
†Department of College of Computer Science and Software Engineering, ShenZhen Univeristy

***Abstract*—Wireless communication is prone to partial packet due to the complex behavior of wireless signal propagation, especially in the dense-deployed wireless networks, where the interference signal is unavoidable. Such packet often contains a few errors, yet has to be retransmitted due to decoding failure. To avoid redundant transmission, partial packet recovery protocols are leveraged to identify and retransmit the corrupted portion instead of the entire packet. However, the information they obtained is limited. In this paper, we investigate two kinds of pilots termed smart pilots to achieve more efficient partial packet recovery. Smart pilot incorporates two novel ideas: (1) Adaptive Hard Pilot that inserts known bits to help recovery the corrupted packet according to the retransmission scheme, and (2) Reliable Soft Pilot that utilizes the confidence information to accurately identify the corrupted portion for retransmission. With these two smart pilots, the receiver is able to recovery more partial packet, and thus the required retransmission portion is greatly reduced. Our experiments show that smart pilot reduces the BER from 10^{-2} to 10^{-4}, and improves the average throughput by 1.8× compared with the existing partial packet recovery protocols.**

I. Introduction

Today, wireless local area networks (WLANs) are facing ever increasing demands for higher transmission rates and more reliable services, e.g., high speed access to the networks, or wireless HD videos delivering. Numerous advances in wireless technology emerge to improve the performance of wireless networks and foster new generations of applications and services [1] [2] [3]. However, the fundamental challenge lies in the fact that wireless links suffer from various impairments, especially in super dense deployed networks with ambient interference [4] [5]. These impairments add noise, attenuate the energy, or otherwise distort the transmitted signal. Thus, the received packets are often corrupted and retransmitted due to decoding failure in the existing 802.11 protocols.

As a promising solution, partial packet recovery (PPR) technology plays an essential role to repair corrupted packets instead of retransmitting them in their entirety [6] [7]. It relies on the observation that despite the errors, partial packet usually contains much useful information. These information can help recover the packet, or at least identify the portion that do not need to be retransmitted. The existing work on partial packet recovery mainly falls into two categories, the error-correcting codes based [8] and block retransmission based [7] [9]. In the error correcting codes based approach, the packets are divided into blocks, each encoded into certain codewords. The received codewords then help to correct the corrupted data bytes according to the error correcting code scheme. While in the block retransmission based approach, the packets are also divided into blocks, each appended with a checksum. The receiver identifies the corrupted blocks through the checksum and only requires the corrupted portion to be retransmitted. Although the target of the above approaches is to leverage the partial packet to reduce the retransmission overhead, the information they can access is limited. Thus, they are not able to exploit more correct information to repair and identify the corrupted portion quickly and accurately.

An ideal partial packet recovery protocol should have the following properties. First, it should contains an error correcting code that can ensure a much reliable transmission. The error correcting codes should also be able to assist corruption localization within a partial packet. Second, it should contains a much precise error estimator, which identifies the corrupted portion with the help of error correcting codes. Third, the retransmission should be efficient and effective with a smallest cost.

Motivated by this, we propose a novel partial packet recovery protocol termed *Smart Pilot* to exploit as many useful information as we can from the partial packet, and leverage it to improve the recovery performance. Our observation is that, through special designed pilots from PHY layer, more information can be investigated to improve the efficiency of partial packet recovery. Smart pilot incorporates two novel ideas: *Adaptive Hard Pilot* and *Reliable Soft Pilot*. Adaptive Hard pilot refers to the bits that are pre-defined and inserted into the information block before encoding. These pilot bits are devised to be pre-known at both sender and receiver side, so the decoder performance can be improved, and the error estimator can obtain more reliable information. Meanwhile, we make use of confidence information from PHY layer and extract the bits with high confidence levels as reliable soft pilot to further improve the decoding performance, which will identify the corrupted portion fast and accurate. With the help of these two pilots, the transmission can be more reliable. If the retransmission is required, the PPR hint will make it more efficient.

To the best of our knowledge, smart pilot is the first work in the literature that leverages two kinds of pilots to exploit as many useful information as we can to improve the recovery performance. It incorporates hard pilot and soft pilot before and during decoding process to provide reliable information for error estimation, and thus greatly reduce the number of retransmitted bits, and increase the overall throughput. We conduct extensive experiments to verify the performance of smart pilot. Our experiments show that compared with the standard partial packet recovery based on 802.11, smart pilot reduces BER from 10^{-2} to 10^{-4} and improves the average

978-1-4673-6305-1/15/$31.00 ©2015 IEEE

throughput by 1.8×, which verifies that smart pilot is capable of exploring reliable messages for error correction, and thus improving the overall throughput.

II. Related Work

Code shortening [10] [11] is a well-known technique to discard some bits for a required rate, which involves the throwing out of codewords and deleting coordinate positions. However, it is essentially different from hard pilot. Code shortening aims at generating different code rates or lengths based on a single check matrix. What it changes is the check matrix. It sets information bits on certain positions to 0 and simply discards the shortened bits after encoding. On the contrary, hard pilot aims at improving the decoding performance by the way of passing the more reliable messages. To achieve this, it inserts an arbitrary number of random bits and encodes based on the code rate rather than the modified check matrix. It does nothing to the check matrix. Furthermore, at the receiver, the inserted pilot bits are utilized to assist the decoding process.

Partial packet recovery has become a hot topic these days. Partial packet refers to the MAC layer packet that has flipped bits after PHY layer decoding. Previous work [9] [3] takes advantage of partial packets to increase the retransmission efficiency and improve the performance of several applications, such as rate adaptation and real-time video streaming. PPR utilizes the confidence level directly. However, the confidence messages are influenced by the channel equalization and demodulation algorithms at the receiver, which are inaccurate. Bit error probability (BEP) and word error probability (WEP) [12] are proposed as reliability-based retransmission criteria. When combining with coding schemes, the algorithm is only required to retransmit estimated bits with high error probability, which reduces the cost of retransmission. Given a threshold of error probability for retransmitting p_t, the probability of false alarm is $1-p_t$. In the paper [13], spectrum efficiency is improved by using a dedicated narrowband as feedback channel to reduce overhead on a finer granularity. However, this dedicated channel may be a waste of throughput.

III. Smart Pilot Design

A. Overview and Design Challenge

In this section, we present the the detailed desgin of smart pilot. It tries to achive a high efficiency partial packet recovery protocol. Fig. 1 illustrates the overall structure of smart pilot for PPR system. The key idea of smart pilot is to extract as many reliable bits as possible to be pilots, and make a nearly optimal partial packet to be retransmitted. Here two types of smart pilots are utilized: hard pilot and soft pilot. Hard pilot is the set of known bits inserted into the information blocks before encoding. With them, the receiver can obtain reliable message information through them for decoding. Soft pilot is the set of bits extracted from the unknown received bits during decoding. They have high PHY confidence level, which makes them reliable for further decoding. After decoding, partial packet retransmission based on pilot exclusion will be adopted,

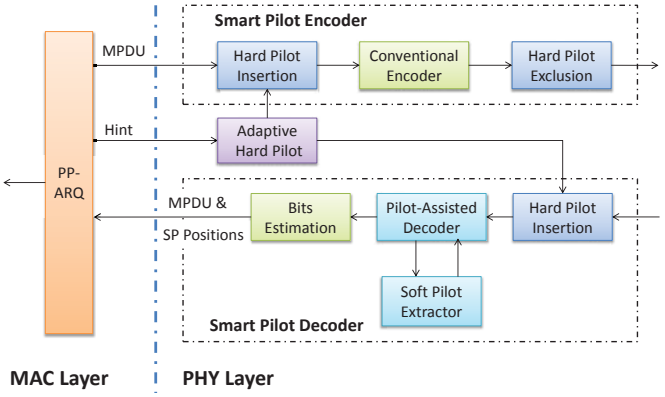


Fig. 1: Block diagram of smart pilot.

which aims to retransmit less fraction of the data bits that can not be corrected.

The idea is simple and efficient, yet there remain several challenges for implementation.

- Hard pilots could improve the decoding performance, while they reduce the efficiency of the packets. Therefore, it makes great concern to choosing the number of hard pilots.
- The threshold of soft pilot extraction should be carefully determined, striking a tradeoff between reliable messages that used as soft pilots and unreliable messages that need to be excluded.
- By leveraging partial packet recovery technique for retransmission, we need to investigate the message bits that cannot be recovered and design a retransmission principle that requires minimum cost.

In the following subsections, we use LDPC codes to illustrate the detailed design of smart pilot, because they have a capability of approaching shannon bound with low decoding complexity. In fact, the idea of hard pilot is totally suitable for the systematic codes, while soft pilot can be used for all ECCs.

B. Adaptive Hard Pilot

In this subsection, we present the design of Adaptive Hard Pilot (AHP). Based on the PPR hints, AHP adaptively chooses the number of hard pilots to enhance the efficiency during data transmission or retransmission. The overall process of AHP consists of three components: HP choosing, encoder and decoder. The number of hard pilots is determined first, which is implemented at the sender in the first transmission and at the receiver if the decoding fails. To do this, we propose logarithmic scaling (LS) based on the relationship between the estimated BER and the code rate. Then, the hard pilots are used in the encoding process. The encoder includes three parts: pilot insertion, standard LDPC encoder and pilot exclusion, as shown in Fig. 1. At the receiver, the decoder is the inverse of the encoder, where pilot re-insertion and LDPC decoding are sequentially implemented.

1) Logarithmic Scaling: Before the PHY layer encoding, the optimal number of hard pilots should be determined. Since

HP could improve the transmission performance, a heuristic idea is to increase the number when the channel is poor at the first transmission, or the PPR hints derived from PP-ARQ show the error probability of the received packets are high.

Two situations require to be considered for the transmission of the current packet. First, if this is the first transmission, the ratio of hard pilots is set as the averaging ratio of the last transmission of previous packet. That is because it is the most expected value for the successful reception. Therefore, assuming the kth packet needs t_k transmissions, the number of hard pilots in the ith code block for the kth packet is calculated as

$$K_p^{(k,1)}(i) = \frac{1}{N_b^{(k-1,t_k)}} \sum_{j=1}^{N_b^{(k-1,t_k)}} K_p^{(k-1,t_k)}(j), \qquad (1)$$

where $N_b^{(k-1,t_k)}$ denotes the number of the blocks for the last transmission of the $(k-1)$th packet.

Second, if this transmission fails, AHP adaptively changes the number of hard pilots based on PPR hints at the next retransmission, in order to obtain potential higher throughput. PPR hints are the probabilities of the successful decoding for all the blocks in a packet at the previous transmission. We obtain PPR hints from PP-ARQ, and the PPR hint is expressed for the ith code block as $\phi(i) = N_{ret}(i)/N$, where $N_{ret}(i)$ denotes the total number of the retransmitted bits for the ith block, and N is the length of a code block. Then, the number of hard pilots is determined using LS for the jth transmission as

$$K_p^{(k,j)}(i) = -\alpha \log(\beta\phi(i))N, \qquad (2)$$

where $0 < \alpha < 1$ implies the relationship between the estimated BER and the ratio of hard pilots in a block. $0 < \beta < 1$ scales the PPR hint to be the real error probability during the retransmitted bits, and the term $\beta\phi(i)$ denotes the estimated BER.

In the next sections, we use K_p to express the number of hard pilots for the current code block for short.

2) AHP Encoder: As shown in Fig. 1, AHP consists of three parts: pilot insertion, standard LDPC encoder and pilot exclusion. Pilot insertion distributes the hard pilots into the whole packet. The position of the hard pilots on the performance will be determined based on the following rules. First, the known bits should be averaged over the cycles, and thereby during the iterative decoding, the performance could be directly improved in each cycle at the aid of the pilot. Second, since the stopping set, which is defined as a set of variables such that all checks connected to that set are connected to at least two different nodes in that set, is related to the performance of LDPC codes under the iterative process [14] [15], it is useful to distribute the pilot bits into each stopping set. Third, if there exist relatively small trapping sets in the Tanner graph, failure events that dominate performance in the error floor region will be increased [16]. To avoid this, usually we design the codes without trapping sets. However, if trapping sets could not be eliminated, a distribution scheme of hard pilot that uniformly poses the pilot on each trapping set will be necessary to prevent the error floor. As a result, for each block K_p hard pilots are inserted into K_b data bits given a position set $\boldsymbol{P}_h$ of hard pilots.

After pilot insertion, LDPC encoder obtains the coded blocks by generating check bits. Further, in order to prevent the data rate reduction induced by inserted pilots, we discard the inserted bits after encoding. The exclusion operation can be implemented for symmetric codes, where the position of pilots in the encoded block is exactly the same as that in the information block. Since symmetric codes have been widely used in many modern standards, such as IEEE802.11, IEEE802.16 and LTE, it is reasonable to assume the encoding type here symmetric.

3) AHP Decoder: AHP decoder and SP decoder are integrated into the total smart pilot decoder, which corrects the distorted message with the assistant of hard and soft pilots, and sends the pilot positions to upper layer as PP-ARQ hints if the decoder fails. In this subsection, AHP decoder is introduced, which mainly completes two things in smart pilot decoding process: pilot re-insertion and LLRs modification.

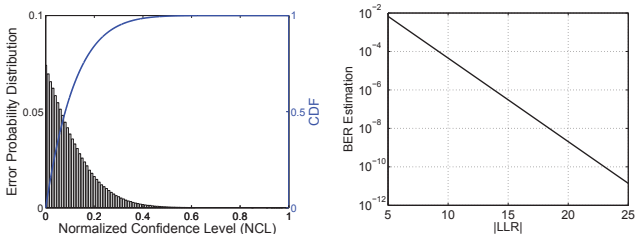
First, hard pilots are re-inserted into packets at the same position as that at the transmitter. Since the receiver knows the PPR hint, the number of hard pilots is inherently clear. Second, since hard pilots provide infinitely high confidence level, LLRs related to them in the initialization and iterative message passing process must be modified to infinity. Notice that since hard pilots increases the reliability of neighboring bits and then propagates to the whole code block, the decoding performance is improved. Moreover, AHP decoder could prevent the potential error propagation, and further enhances the reliability of the transmission, leading to less errors.

C. Reliable Soft Pilot

Reliable soft pilot (RSP) consists of two components: soft pilot extractor and pilot-assisted decoder. The decoder is the other component of the smart pilot decoder. Since various interference always exists in the propagation path, such as noise variation, channel multipath, and collision, the LLRs will be more reliable on some positions, while relatively unreliable on the other positions. The unreliable bits can greatly deteriorate the decoding performance and even make the decoder diverge. To address it, the more reliable LLRs can be effectively utilized to assist decoding, and thus the performance will be expected better. Motivated by this, we propose a soft pilot algorithm to extract reliable bits to assist decoding.

Soft pilot extractor plays a key role for RSP, which extracts the more reliable bits as pilots to assist the next iteration. The algorithm is based on two observations on the normalized confidence level (NCL) and confidence level (CL) shown in Fig. 2, where CL denotes the absolute value of LLR $L_C(i) = |LLR(i)|$, and NCL is defined as the ratio of CL and the maximal CL in a block, i.e.,

$$L_N(i) = \frac{|LLR(i)|}{\max\limits_{j=1,...,N} |LLR(j)|}, \quad i = 1, ..., N. \qquad (3)$$



(a) (b)

Fig. 2: The influence on bit errors by the LLRs. (a) The distribution of error probability in terms of the normalized confidence level (b) BER estimation using LLRs based on softRate.

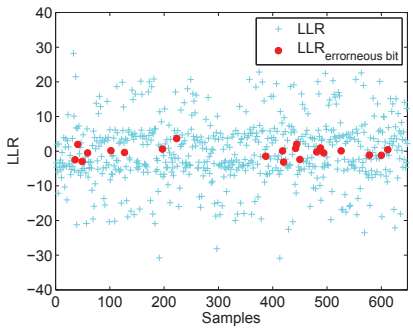


Fig. 3: LLR for all bits versus for erroneous bits.

First, from Fig. 2(a), we observe that lower NCLs cause higher error probability. When NCLs exceed a value, the corresponding bits can be viewed as correct with little error probability. Second, as shown in Fig. 2(b), bits with higher CLs have higher reliability. As the increasing of the CLs, BER could be reduced to a negligible value. Motivated by these, we extract pilots by utilizing the confidence information during decoding.

We divide decoding process into several loops. For each loop, reliable outputs of the decoder are extracted and used for facilitating the decoding as hard pilot in the next loops. To achieve this, two thresholds, i.e., relative threshold T_1 and absolute threshold T_2, are introduced.

Relative threshold T_1 is chosen according to the relationship of NCL and the error probability. From Fig. 2(a), it is shown that the probability of the errors dynamically decreases as the increase of NCL. In particular, the erroneous bits rarely occur when NCL exceeds 0.5, which means that the decoding outputs with NCL higher than 0.5 are more reliable. In our evaluations, we set the relative threshold to a larger value of 0.8 for the sake of more reliability. It means that we will extract the bits with the NCL of larger than 0.8 as pilot candidates. On the other hand, the other threshold T_2 is chosen based on the relationship of CL and the error probability. From the knowledge of SoftRate, the error probability has an approximately exponential decay relationship with CL, as shown in Fig. 2(b), Specifically, BER is lower than 10^{-9} when CL is higher than 20, which can be negligible. Therefore, we set the absolute threshold to 20, and extract bits with CL higher than the threshold as pilots. Then, the pilots extracted by the two thresholds and the previous extracted pilots will be as a prior knowledge for the next decoding process, resulting in more and more reliable outputs.

With these two criterions, we obtain two candidate sets after each loop. Union of them and the hard pilot is then taken as a new pilot set. All of them will assist the decoding process for next loops. In summary, for the qth iteration, the SP extractor assigns reliable bits as pilots based on the overall criterion,

$$\boldsymbol{P}_s^{(0)} = \boldsymbol{P}_h, \tag{4}$$

$$\boldsymbol{P}_s^{(q)} = \arg_{\{i\in[1,N]\&\&i\notin\boldsymbol{P}_s^{(q-1)}\}} \left(L_N(i) > T_1 \parallel L_C(i) > T_2\right) \cup \boldsymbol{P}_s^{(q-1)}. \tag{5}$$

The other part of RSP is pilot-assisted decoder. The decoder obtains more and more reliable information as the iteration processes. Similar to AHP decoder, since the extracted pilots provide infinitely high confidence level, LLRs related to them in the initialization and iterative message passing process must be modified to infinity.

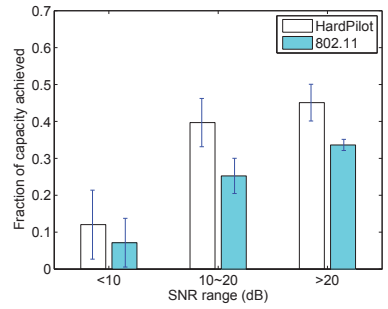
Notice that the smart pilot algorithm can be easily applied for other coding schemes. For example, when considering convolutional codes, the error propagation is introduced if the path is wrongly chosen due to unreliable messages, which may lead to totally wrong estimation in the following data. To address it, first, we could use AHP algorithm to set pilots in the coded sequence to discard the unreliable candidate paths. Second, the SP algorithm also could be applied to take the bits on more reliable paths as pilots, and further improves the decoding performance for the following process.

D. Partial Error Position-based Feedback Algorithm

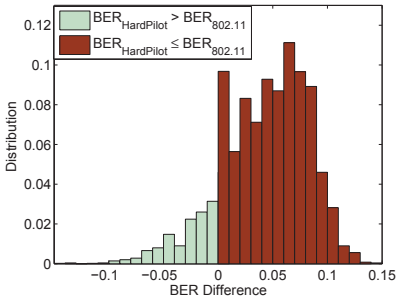
If the decoder fails, the packet requires to be retransmitted. Throughout the all studies on retransmission, PPR is a very heuristic method. By reducing the number of bits transmitted, PPR improves aggregate network throughput significantly. However, it is not enough to only exploit SoftPHY hints which are derived from LLRs after decoding, since not all bits from positions with lower confidence levels are unsuccessfully decoded. For instance, as shown in Fig. 3, PPR chooses all bits below a threshold as "bad" bits, while few of them are really erroneous, resulting in largely unnecessary waste for each retransmission. For finer-grained filtering out bits with higher error probability, we utilize the confidence information from the decoder and the pilots extracted from Section III-C. Therefore, as many bits as possible will be not required to be retransmitted, and thus our algorithm leads to nearly optimal partial packet.

In the proposed partial retransmission algorithm, first we choose the bits with lower confidence levels excluding pilots as the candidate of the retransmitted bits. It is based on the fact that after decoding process, bits with higher correct probability have been extracted as soft pilots. As a result, the leaving positions are with higher error probability. The position of bits which are required to be retransmitted is found as:

$$\boldsymbol{P}_r = \arg_{\{i\in[1,N]\&\&i\notin\boldsymbol{P}_s^{(Q)}\}}(L_C(k) < T_3), \tag{6}$$



(a)



(b)

Fig. 4: Performance comparison of hard pilot scheme and the standard 802.11. (a) Fraction of capacity achieved to Shannon rate, (b) Distribution of BER difference for the case of QPSK, rate 1/2 and SNR of 8dB.

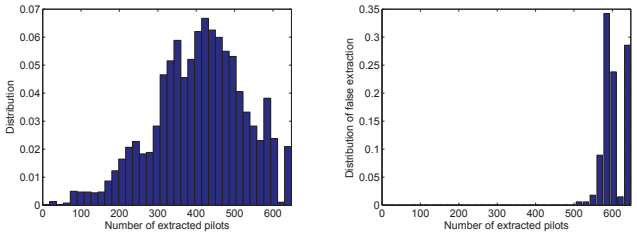
where Q is the real number of iterations for LDPC decoder, and T_3 is a reliability threshold, which is nearly the same as T_2. When combining these criterions, the number of retransmission bits is limited as the channel changes. After that, we acquire to know each boundary of a chunk, which contains a group of bits that requires to be retransmitted. Given the positions of soft pilots, we find the best strategy by using dynamic programming proposed in PPR.

The receiver encodes the feedback set of the chunks and the AHP hint described in Section III-B. At the sender, only the required chunks are transmitted. Moreover, MAC layer also feedbacks the AHP hint to the PHY layer for the decoding of the following transmitted packet.

IV. Evaluation

A. System Implementation

In our trace-driven experiment, we utilize GNU radio testbed and implement smart pilot on Software Defined Radios (SDRs). The Universal Software Radio Peripheral 2 (USRP2) is adopted as RP frontend. Our testbed consists of 8 USRP2 nodes with RFX2450 daughterboards operating on the 802.11 frequency range. We use a channel bandwidth of around 20MHz and split it into 64 subcarriers. These changes are made since we want to make the subcarrier spacing comparable to 802.11 (0.3125MHz) while still maintaining the normal transmission of USRP2. To achieve practical packet retransmission, we collect the traces from the experiments and operate them off-line due to the latency constraint of USRP2.



(a) (b)

Fig. 5: Pilot extraction performance. (a) Distribution of the number of extracted pilots (b) Distribution of the number of wrongly extracted pilots.

The experiments run on the 2.425GHz, and each packet contains 1460 bytes. In the performance evaluation, we use three pairs of USRP2 devices. Furthermore, CSMA/CA as well as hybrid ARQ are applied as the protocols of the data link layer and MAC layer, respectively. we calculate the total number of non-duplicate data packets successfully received by the designated receivers per second as the aggregate throughput.

B. Performance of Smart Pilot

Performance of Hard Pilot: To compare our hard pilot scheme with the standard 802.11, we analyze the information rate (R_b, bits per symbol) and BER reduction, which is the BER difference between 802.11 and our hard pilot scheme. Fig. 4 gives the performance comparison of hard pilot coding scheme and the standard 802.11. For the hard pilot scheme, we set the number of inserted pilots to half the message length. It is observed that the hard pilot scheme significantly outperforms the standard 802.11. Fig.4(a) compares the fraction of capacity achieved, which is calculated by $R_b \log(1 + \text{SNR})$. We show the improved data rate for hard pilot is 80% for SNR range between 10dB and 20dB and 20% for SNR larger than 20dB. Fig. 4(b) shows the statistics of BER difference for 5000 packets, which is calculated by the BER minus of the standard 802.11 and hard pilot. It is shown that the hard pilot scheme reduces BER for more than 90% packets, resulting in the averaging 60% BER reduction. Therefore, the hard pilot coding scheme is more powerful for its ability of effectively preventing error propagation and improving the performance.

Performance of Soft Pilot: Then, we evaluate the performance of soft pilot via BER comparison with setting the parameters of BPSK, 2/3 code rate and 3dB SNR. First of all, the performance of pilot extraction is analyzed, shown in Fig. 5. From Fig. 5(a) we can see that the number of extracted pilots centralizes around 400 bits, i.e., 60% bits can be considered as known, which will greatly improve the decoding performance. Furthermore, as shown in Fig. 5(b), no extracted errors occurs when the number of extracted pilots is smaller than 500, that is, 77.2% bits are extracted correctly. However, when the number of extracted pilots is larger than 500, errors occur among these pilots. The reason is that at the end of the pilot extraction, the error-free bits are already extracted, and the bits except smart pilot have almost erroneous values. Therefore, the false probability of extracted bits will be increased. To account it, a limitation should be added to soft pilot algorithm. We stop

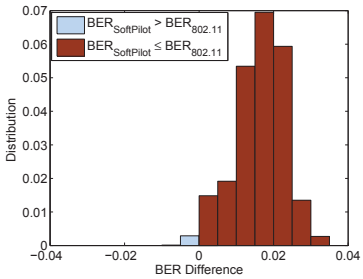


Fig. 6: Distribution of BER difference between the standard 802.11 and soft pilot for BPSK, 2/3 code rate and 3dB.

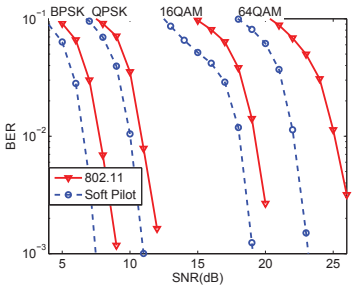


Fig. 7: BER comparison between smart pilot and the standard 802.11 for different modulations at the code rate of $3/4$.

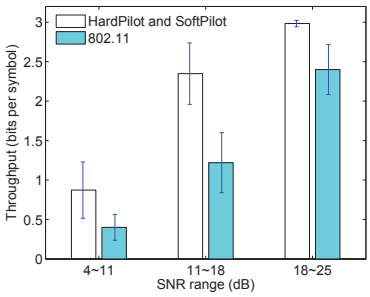


Fig. 8: The overall throughput comparison between smart pilot and the standard 802.11.

pilot extraction once the number of extracted pilots is larger than a pre-defined value. As a result, we can make sure the reliability of pilots with high probability. Fig. 6 illustrates BER difference between the standard 802.11 and soft pilot algorithm. It is seen that soft pilot can reduce BER for more than 99% packets and correct bits with an averaging BER of 0.035. The averaging BER of total 20000 packets significantly reduces from 4.3×10^{-2} to 4.9×10^{-4}, resulting in significant performance improvement.

Furthermore, we also compare the BER between the standard 802.11 and soft pilot for all available modulations at the code rate of $3/4$, as shown in Fig. 7. It is observed that about $1\sim3$dB performance improvement is achieved by soft pilot.

Performance of Smart Pilot: Fig. 8 compares the aggregate throughput of smart pilot with LDPC codes under three equal SNR ranges, e.g., [4, 11]dB, [11, 18]dB and [18, 25]dB. We observe that smart pilot outperforms LDPC codes in all the three ranges, with an average throughput gain of 218%, 193% and 124%. As the channel quality becomes severer (e.g., with a relatively low SNR), smart pilot performs better. This verifies that smart pilot can prevent error propagation when the transmitted coded bits are distorted by the complex behavior of the wireless radio channel, and further improve the decoding performance and throughput performance by leveraging hard pilot and soft pilot. We notice that as the channel condition becomes better, e.g., SNR $>$ 18dB, smart pilot does not have such great performance gain. That is because under a considerable channel condition, the erroneous bits only consist a small portion of the entire packets. Therefore, error propagation is not so severe, and has a relatively small influence on decoding performance. However, smart pilot still achieves a throughput gain of 124%, since less bits are required for retransmission though there are less failed packets.

V. Conclusion

In this paper, we propose smart pilot, a novel partial packet recovery protocol in 802.11 WLANs. Smart pilot aims to leverage as many useful information as we can to improve the efficiency of partial packet recovery. Smart pilot incorporates two components: *Adaptive Hard Pilot* and *Reliable Soft Pilot*. Adaptive Hard pilot are pre-known at both sender and receiver, which aim to improve the decoding and error estimation performance. Soft pilot are PHY layer hint with the help of hard pilot. By exploiting these two kinds of pilot, the transmission can be more reliable, and the error estimator can be much precise. Extensive experiment results show that smart pilot reduces the BER from 10^{-2} to 10^{-4}, and achieves an average throughput gain of $1.8\times$ over the standard 802.11.

References

[1] A. Bhartia, Y. Chen, S. Rallapalli, and L. Qiu, "Harnessing frequency diversity in wi-fi networks," in *ACM MobiCom*, 2011.
[2] L. Li, K. Tan, H. Viswanathan, Y. Xu, and Y. Yang, "Retransmission≠ repeat: simple retransmission permutation can resolve overlapping channel collisions," in *ACM Mobicom*, 2010.
[3] B. Chen, Z. Zhou, Y. Zhao, and H. Yu, "Efficient error estimating coding: feasibility and applications," in *ACM SIGCOMM*, 2010.
[4] G. Hosseinabadi and N. Vaidya, "Concurrent-mac: Increasing concurrent transmissions in multi-ap wireless lans," in *Proceedings of the 19th annual international conference on Mobile computing & networking*. ACM, 2013, pp. 227–230.
[5] V. Gau, C.-W. Huang, and J.-N. Hwang, "Reliable multimedia broadcasting over dense wireless ad-hoc networks," *Journal of Communications*, vol. 4, no. 9, pp. 614–627, 2009.
[6] J. Xie, W. Hu, and Z. Zhang, "Revisiting partial packet recovery in 802.11 wireless lans," in *Proceedings of the 9th international conference on Mobile systems, applications, and services*. ACM, 2011, pp. 281–292.
[7] B. Han, A. Schulman, F. Gringoli, N. Spring, B. Bhattacharjee, L. Nava, L. Ji, S. Lee, and R. R. Miller, "Maranello: Practical partial packet recovery for 802.11." in *NSDI*, 2010, pp. 205–218.
[8] K. C.-J. Lin, N. Kushman, and D. Katabi, "Ziptx: Harnessing partial packets in 802.11 networks," in *Proceedings of the 14th ACM international conference on Mobile computing and networking*. ACM, 2008, pp. 351–362.
[9] K. Jamieson and H. Balakrishnan, "Ppr: Partial packet recovery for wireless networks," in *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4. ACM, 2007, pp. 409–420.
[10] R. H. Morelos-Zaragoza, *The Art Of Error Correcting Coding*, 2nd ed. John Wiley and Sons, 2006.
[11] X. Liu, X. Wu, and C. Zhao, "Shortening for irregular qc-ldpc codes," *Communications Letters, IEEE*, vol. 13, no. 8, pp. 612–614, Aug. 2009.
[12] J. C. Fricke and P. A. Hoeher, "Reliability-based retransmission criteria for hybrid arq," *Communications, IEEE Transactions on*, vol. 57, no. 8, pp. 2181–2184, 2009.
[13] J. S. Zhang, H. C. Shen, and T. K. et. al., "Frame retransmissions considered harmful: improving spectrum efficiency using micro-acks," in *ACM MobiCom*, 2012.
[14] A. Orlitsky, K. Viswanathan, and J. Zhang, "Stopping set distribution of ldpc code ensembles," *Information Theory, IEEE Transactions on*, vol. 51, no. 3, pp. 929–953, 2005.
[15] S. Laendner and O. Milenkovic, "Ldpc codes based on latin squares: Cycle structure, stopping set, and trapping set analysis," *Communications, IEEE Transactions on*, vol. 55, no. 2, pp. 303–312, 2007.
[16] T. Richardson, "Error floors of ldpc codes," in *Proceedings of the annual Allerton conference on communication control and computing*, vol. 41, no. 3. The University; 1998, 2003, pp. 1426–1435.