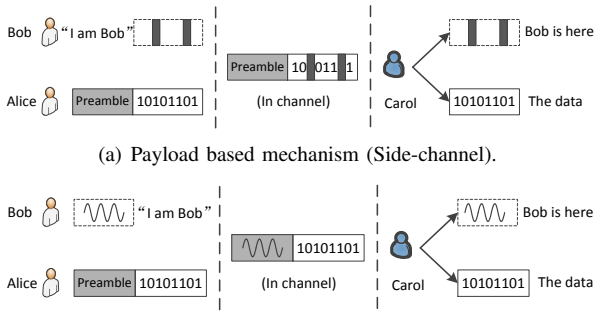
Hitchhike: Riding Control on Preambles

Xiaoyu Ji*, Jiliang Wang†, Mingyan Liu ‡, Yubo Yan§, Panlong Yang§†, Yunhao Liu†
* Department of Computer Science and Engineering, HKUST
† School of Software and TNLIST, Tsinghua University
‡ Department of Electrical Engineering and Computer Science, University of Michigan
§ PLA University of Science and Technology

***Abstract*—Recently, carrying control signals on passing data packets has emerged as a promising direction for efficient control information transmission. With control messages carried on data payload, the extra air time needed for control packets like RTS/CTS is eliminated and thus channel utilization is improved. However, carrying control signals on the data payload of a packet requires the data packet to have a sufficiently large SNR, otherwise both the data packet and the control messages are lost. In this paper, we propose *Hitchhike*, a technique that utilizes the preamble field to carry control messages. Hitchhike completely decouples the control messages from the payload and therefore the superposition of (multiple) control messages has little adverse effect on the operation of the payload decoding. We implement and evaluate Hitchhike in the USRP2 platform with 5 nodes. Evaluation results demonstrate the feasibility and effectiveness of Hitchhike. Compared with the state-of-the-art, e.g., Side-channel in 802.15.4, Hitchhike improves the detection accuracy of control messages by 40% and reduces the data loss caused by control messages by 15%.**



(a) Payload based mechanism (Side-channel).

(b) Hitchhike, a preamble based approach.

Fig. 1. Illustration of Hitchhike compared with payload based mechanism. In Side-channel (Fig. 1(a)), the control message from Bob is carried on the payload of the packet form Alice to Carol. While in Hitchhike (Fig. 1(b)), the control message is carried on the preamble of the packet.

I. Introduction

Due to the scarcity of wireless spectrum, it is a common practice within wireless protocols to send control messages as data messages, see e.g., ZigBee and Wi-Fi. As a prime example, the control messages RTS/CTS used in CSMA/CA-type of protocols are strictly speaking data packets containing control information. While this is certainly a very convenient way of disseminating and exchanging control information, the overhead is significant: these extra data packets occupy a disproportionately large portion of air time compared to the amount of information they carry [1].

Recently, there has emerged a promising new direction for efficient control message delivery, one that tries to build the control plane on top of real data packets, see e.g., [2], [3]. The basic idea is to embed control information in data messages so the two may be transmitted concurrently. This is illustrated in an example shown in Fig. 1(a). Bob wants to transmit a control message to Carol telling her that he is Bob (the equivalent of the "HELLO" control packet in many protocols). Instead of using a separate data packet for this, Bob tries to ride his message on top of a data packet from Alice to Carol as follows. He intentionally interferes a few bits of Alice's packet and uses these bits to convey his control information. Carol receives the (slightly) corrupted data packet and extracts the control information from Bob by calculating the relative distance among those bits. The original data message can also be recovered due to the modulation redundancy [2] in physical layer (PHY) implementation. This so-called in-band control message[1] transmission mechanism eliminates the extra transmission time for control messages and therefore improves channel utilization.

However, there are three problems with the above mechanism. First of all, this mechanism requires high SNR for the data packet. The basic assumption for this mechanism to work is that the payload has a large redundancy, while in reality, due to the existence of noise and interference, the redundancy margin may be very limited. As a result, it is possible to corrupt the data bits in the payload beyond recovery by inserting the control message, especially under low SNR conditions. Second, even if the control message does not adversely affect the decoding of the payload, it may be difficult to correctly decode the control message under low SNR conditions. There may be other "error bits" caused by noise and unintentional interference, which could mislead the receiver to incorrectly decode and interpret the control message. Lastly, this mechanism suffers when multiple users try to transmit control messages via the same data payload. In Fig.1(a), if another user, say David, also transmits his control message on top of Alice's packet, then the two control messages interleave and Carol may not be able to decode either of them.

In this paper, we propose Hitchhike, a novel technique that carries control messages on the preamble rather than data payload, and decodes them using correlation. As shown in Fig. 1(b), the control message from Bob is sent as a unique

[1]We use control for short in the rest of this paper.

signal pattern superposed on the preamble field of the packet form Alice to Carol. Carol then detects both the preamble of the packet and the control signal from Bob by correlation. Compared with payload-based mechanism, preamble-based control plane has the following advantages. First, by carrying the control in the preamble field, the data payload is decoupled from control signaling. As a result, the decoding of control does not interfere with the decoding of data bits. Indeed, detecting a preamble requires far lower SNR than decoding a bit does, thus using this technique control messages can be transmitted at extremely low SNR levels, without affecting their detection accuracy. This also means that, there is no increased SNR requirement on the payload as it is not threatened by interference from the control message. Last but not least, multiple control messages can be transmitted concurrently over a common data packet's preamble field and detected using a bank of correlators.

This technique, however, is not without its own challenges. First of all, the superposition of control signals over a preamble may cause missed detection of the preamble which leads the whole packet to be missed. To address this, we need to carefully design orthogonal codes for control messages that have appropriate lengths with respect to the signal pattern of the preamble. The second challenge lies in the detection and recovery of control information from the mixed control-preamble signal. To eliminate the impact of the preamble in the detection of control, we design a subtraction-detection algorithm in which the preamble signal is subtracted from the mixed signal to provide higher detection accuracy for control signals. Finally, we need to ensure that the control signals can ride exactly on the preambles, i.e., sufficiently synchronized with the preamble. For this we propose to utilize the beginning part of a preamble a synchronization mechanism signaling the incoming of a preamble.

We design and implement Hitchhike on the GNURadio/USRP2 platform, and our experiments involve 5 such nodes. Our evaluation results indicate that placing the specially designed control signals on a preamble has little adverse effect on the normal operation of the preamble. Moreover, the control signals carried by the preamble can also be detected accurately. Compared with the state-of-the-art technique, Side-channel [2], the detection accuracy of Hitchhike is 40% higher and the side effect upon data decoding is 15% lower in average. Our main contributions are summarized as follows:

- We propose a novel concept that utilizes preambles for carrying control information. We contrast this with the payload-based mechanisms and highlight its advantages in being able to operate under lower SNR requirements and in supporting multiple users.
- We design and implement Hitchhike, a mechanism that uses the above concept to deliver control messages in 802.15.4 networks. We design orthogonal codes for control messages and develop the subtraction-detection algorithm for their detection.
- We validate the feasibility of Hitchhike and evaluate its performance with 5 nodes on the GNURadio/USRP2 platform. The results indicate that the preamble and the designed control messages can work in harmony in Hitchhike.

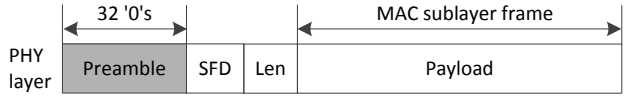


Fig. 2. Format of a packet in the 802.15.4 PHY layer. The preamble field has a repeated signal pattern.

The rest of this paper is organized as follows. Section II details the motivation of this work. In Section III we introduce the design detail of Hitchhike. In Section IV, we present the implementation of Hitchhike, followed by a performance evaluation in Section V. We discuss the potential applications of Hitchhike in Section VI and the related work in Section VII. We conclude the paper in Section VIII.

II. Motivation

In this section we detail the motivation behind the concept and design of Hitchhike. We start with a preliminary on the use of preamble and related physical layer details. We then describe a gap between the required SNRs in detecting the preamble and decoding the data payload. We end by presenting the main challenges in using preambles to carry control messages.

A. A Primer on Preamble

In wireless networks with IEEE 802.15.4 [4] and 802.11 [5] PHY layer, a data packet is mainly composed of four parts: the preamble field, SFD (Start of Frame Delimiter), length and the payload field. As shown in Fig. 2, the preamble is the beginning of the packet followed by SFD and Len, and the rest is the payload. Preamble is primarily used for a receiver to detect the incoming/arrival of a packet and perform synchronization needed for packet reception. The SFD field indicates the start of the MAC sublayer frame and Len specifies the length of the payload.

A preamble has a known repeated pattern so that the receiver can use auto-correlation with the received signal to determine its presence in the air. To ensure detection robustness, a preamble is designed to have a repeated pattern. For example, under the 802.15.4 specification, a preamble contains 32 binary '0's which is in fact 8 repeated '0' symbols. The correlation operation would output a spike if a preamble is present in the signal. At the same time, the receiver synchronizes itself with the sender and calculates parameters like carrier frequency offset (CFO). It begins receiving the payload signal after the SFD and length checking.

B. The SNR Gap: Detection vs. Decoding

As mentioned above, a preamble is detected via correlation, which is different from the decoding operation for a payload. In short, determining an unknown bit sequence is much harder than detecting a known pattern. The reasons are as follows. First, correlation is a coarse decision process and has relatively

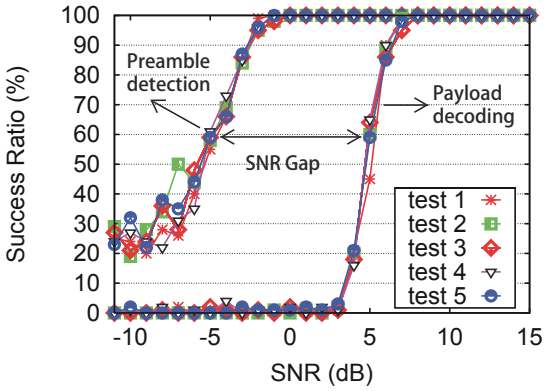


Fig. 3. The SNR gap between decoding the payload and detecting the preamble. The gap can be as large as almost 10 dB.

large tolerance for errors, while decoding a bit entails a high SNR requirement and needs to map the received signal to a constellation diagram accurately to decode a bit. Second, a preamble is composed of repeated signal patterns, which further enhances the detection robustness. By comparison, the redundancy provided by coding and modulation schemes [4], [5] for the data payload is not as much.

To clearly see the gap between what's needed to detect a preamble vs. decode a packet, we conduct an experiment with two USRP2 devices running with 802.15.4 PHY and placed 10 feet apart. One is used as a transmitter and the other a receiver. Packets of length 100 bytes are sent from one to the other with different SNRs, ranging from -10 dB to 15 dB. The received signal trace is collected and decoded using standard decoding block for IEEE 802.15.4 on MATLAB. The experiments are repeated for five times.

Fig. 3 shows the result for the detection of the preamble and for the decoding of the payload, respectively. In each case we see a very clear threshold in acceptable SNR for the corresponding operation to be considered successful. For the decoding of the payload, this threshold is around 7 dB: the payload decoding rate is $> 90\%$ above this level, and falls precipitously to nearly zero when the SNR is below 3 dB. For the detection of preambles, this threshold is around -3 dB, significantly below what's required for the payload. This finding coincides with the report in [1], [6]. Further, we see that the detection of the preamble degrades much more gracefully: it stays above 20% with the SNR falling below -7 dB.

If we set the decoding/detection requirement to above 90%, then this SNR gap is about 10 dB from our experiment. The gap comes from the correlation mechanism and the inherent redundancy in a preamble structure with repeated signal patterns. This gap suggests great potential for preambles to become a carrier of control information. Because of the physical separation, delivering control over preambles does not affect the data bits. What is more, as preambles have low SNR requirement, i.e., the preamble pattern can be recognized with low SNR, there is more robustness in our ability to detect the control messages. We give detailed analysis in the next section.

C. Challenges in Using the SNR Gap

The introduction of control messages on a preamble should not adversely affect the functionality of the preamble. In particular, the preamble is used to detect the beginning of a data packet and perform synchronization between a receiver and a transmitter. If the insertion of control messages affect the operations, then the receiver may miss the entire packet.

Second, the detection accuracy of control messages carried in the preamble needs to be sufficiently high, i.e., with low false positive and false negative probabilities, as control information is extremely important. If the control signal is easily drowned by the preamble signal then this scheme would not be very useful even if it doesn't harm the underlying, original operation of data communication.

We next detail the design of Hitchhike to tackle theses problems.

III. Design of Hitchhike

In this section, we describe the main design of Hitchhike. We first present the code design for control messages to ensure little mutual influence between the preamble and the control messages. Then we describe the design of detection module for the control messages.

A. Design of Control Messages

In order to decouple the control messages from the preamble and detect possibly multiple control messages/signals from the preamble, it is clear that the control signals need to be orthogonal to the preamble signal as much as possible. In later analysis we show that the detection accuracy is proportional to how the control signals are orthogonal to the preamble. Similarly, the control signals need to have as little cross-correlation among themselves as possible. Strong cross-correlation among control signals will result in false detection of a non-signaled message.

In addition, the length of the control signal needs to be carefully designed due to an inherent tradeoff: longer control signals will have a larger adverse effect on the preamble detection, while shorter control signals lower the detection accuracy of the control messages as well as the capacity in carrying control information.

Below we use 802.15.4 as an example to illustrate the design choice of control signals/codes against the preamble. Under the 802.15.4 specification, DSSS (Direct Sequence Spread Spectrum) is used as the modulation scheme. With DSSS, a data symbol containing four digital bits is spread to a 32-chip sequence. The chips are the transmitted signal elements. Therefore, the 8-symbol preamble in 802.15.4 is in fact a sequence of 256 chips. Without loss of generality, we denote the length of the preamble field by l (chips), the length of a control signal by x chips, and by m the number of chips per symbol.

In general, the preamble length l determines its detection accuracy [7]. This is because a preamble contains a repeated

signal pattern, so the larger the value of l, the more redundancy there is in the preamble for the detection to be accurate. For instance, with the cc2420 receiver [8], a commonly used 802.15.4 receiver chip, the preamble length can be configured, with a minimum length as small as 2 symbols, indicating the possibility of transmitting a very short preamble that can be detected. This also means that with an 8-symbol preamble, the remaining 6 symbols serve purely as redundancy, and can potentially be used for other purposes.

Consider now using x out of the l chips to carry a control signal. If the preamble can be detected with probability p from a single symbol (p is naturally a function of the preamble SNR), then taking away x chips to carry control signals leaves $l - x$ chips, or $(l - x)/m$ repeated symbols, and a detection probability of:

$$P(x) = 1 - (1 - p)^{\frac{l-x}{m}} \tag{1}$$

At the same time, the capacity, i.e. the number of orthogonal codes that may be packed in the space of x-chip signals, is proportional to x. Thus we see clearly the tradeoff in selecting a good value of x: larger x benefits the detection of control and the capacity of control information, but decreases the accuracy of preamble detection.

In our design, we wish to use a minimum length of chips to generate a large number of control codes that have low correlation with the preamble and low cross-correlation within the control codes. We implement the control codes with PN sequences in IEEE 802.15.4 [4] and use the length of x = 64.

B. Detecting the Control Message

We next describe how the control messages carried in the preamble field can be detected with the correlation detection technique [6], [9], [10]. In order to be self-contained, we start with a brief introduction to the principle of correlation detection for multiple signals even though this is standard communication basics.

1) Correlation detection of multiple signals: In wireless communication, the received symbols differ from the transmitted symbols due to channel distortion and interference. This is often captured by the following expression, with $y_i[n]$ denoting the nth received symbol from sender i, after attenuation and phase shift to the transmitted symbol $x_i[n]$ caused by the wireless channel [11]:

$$y_i[n] = H_i x_i[n] + w_i[n] \tag{2}$$

where H_i is the channel coefficient between the transmitter and the receiver and $w_i[n]$ a random noise. Suppose that N users transmit their signals simultaneously, the received composite signal can be represented as with $w = \sum_i w_i$ denoting the composite noise signal:

$$y[n] = \sum_{i=1}^{N} H_i x_i[n] + w[n] \tag{3}$$

The correlation-based detection mechanism works as follows. Denote by $C(s, y, q)$ the correlation coefficient between a received signal y and a known pattern s (either the preamble or a specific control signal pattern) of length L at a shifted position q, given by:

$$C(s, y, q) = \sum_{k=1}^{L} s^*[k] y[k + q] \tag{4}$$

where $s^*[k]$ is the complex conjugate of $s[k]$. The value of $C(s, y, q)$ is low when the signal s is not present in y, and even when the signal s is in y, the correlation values has a spike only when $y[k + q]$ aligns well with the beginning of s. When the received signal matches s and they align well, the correlation coefficient of the spike is given by:

$$C(s, y, q) = \sum_{k=1}^{L} s^*[k] y[k + q] = H \sum_{k=1}^{L} |s[k]|^2 \tag{5}$$

When y is a composite signal containing multiple signal patterns of interest, including both the preamble and control messages in our case, we will try to detect the presence of the preamble first, followed by each control messages in serial. This can be done by correlating the received composite signal y with each known signal pattern. Due to the orthogonality between the preamble and the control signals, each correlation operation is enhanced on one signal of interest and suppresses the others, and thus can determine the presence of each signal, one at a time. More precisely, if present in the received composite signal y are the transmitted preamble s_1 from sender 1, as well as control message s_i from sender i, $i = 2, \cdots, N$, then we will have (omitting the noise terms for simplicity):

$$\begin{aligned} C(s_i, y, q) &= \sum_{k=1}^{L} s_i^*[k] \sum_{j=1}^{N} y_j[k + q] \\ &= \sum_{k=1}^{L} s_i^*[k] H_i s_i[k + q] + \sum_{k=1}^{L} \sum_{j \neq i} s_i^*[k] H_j s_j[k + q] \\ &= \sum_{k=1}^{L} s_i^*[k] H_i s_i[k + q] \\ &= H_i \sum_{k=1}^{L} |s_i[k]|^2 \end{aligned} \tag{6}$$

due to the orthogonality between s_i and s_j, $i \neq j$. We normalize the correlation value and detect the presence of a signal by its strength using a threshold. The normalized correlation is as follows:

$$N(s, y, q) = \frac{C(s, y, q)}{\sum_{k=1}^{L} |s[k]|} \tag{7}$$

When s is present in y, the theoretical normalized correlation value is $N(s, y, q) = 1$.

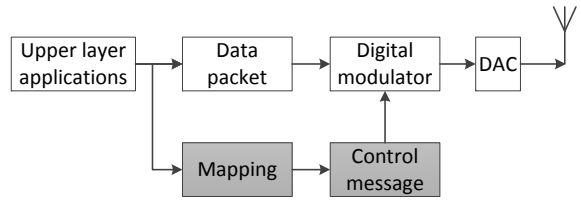
2) The Subtraction-Detection Algorithm: In theory, as shown above, we can use Eq. 7 to detect the control messages. However, in practice, the relative strengths between the preamble signal and control signals and the non-complete orthogonality affect the correlation value because the signals and the channel coefficients cannot be completely independent [6]: the stronger signal dominates the output of the correlation.

To overcome this problem, we adopt a signal subtraction technique, similar to the idea of successive signal cancellation [12]. Specifically, the receiver subtracts the preamble signal

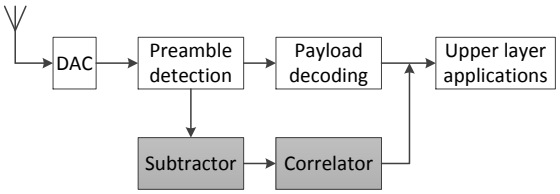
Algorithm 1 The Subtraction-Detection Algorithm

Input: received signal y, known preamble p, controlling messages set S and threshold set T.
Output: The decision set D

```
/* Correlate and subtract preamble from y */
Calculate C(p, y, q) with Eq. 7
if C(p, y, q) > T_p then
    d_p ← 1
    Recover p'
    y ← y − p'
    /* Correlate control signals */
    for all s_i ∈ S do
        Calculate C(s_i, y, q) with Eq. 7
        if C(s_i, y, q) > T_i then
            d_i ← 1
        end if
    end for
    return D
end if
```



(a) The transmitter module.



(b) The receiver module.

Fig. 4. Implementation of transmitter and receiver module in Hitchhike.

from the mixed signal to enhance the correlation values for control messages detection. As we have mentioned, the preamble signal is a sequence of repeated symbol pattern, we can use any individual received symbol and repeat the symbol to recover the entire preamble, considering that the channel condition is relatively static during a short time spans [13]. In our design, we use the symbols that are not overlapped with control messages as a "clean" symbol. Then this clean symbol is subtracted from the mixed signal to obtain the remainder that contains the control signals.

It should be mentioned that the received signal containing the preamble and control needs to be stored. After the preamble is detected and the clean symbol found, this stored signal then goes through the subtraction and correlation operations. Compared to directly operating on the original signal, correlation using the processed signal leads to more accurate detection results. Note that the subtraction operation doesn't require tight synchronization in our case as we do not need to decode the signal and thus coarse subtraction suffices. The subtraction-detection results are presented in Section V.

Combine the correlation and subtraction techniques above, we propose the subtraction-detection algorithm in Alg. 1. Here y denotes the received signal, p is the known preamble pattern, and $S = \{s_1, s_2, ...s_k\}$ represents the set of control signals. The vector $D = \{d_1, d_2, ...d_k\}$ (d_p for the preamble) records the detection results, with $d_i = 1$ denoting the presence of control message s_i and 0 otherwise. As shown, the preamble is detected first by correlating y with p, and then the pure preamble signal p' is obtained by repeating any clean symbol in y. The pure control signal is obtained by subtracting p' from y. For the resulting signal, the control signals are sequentially correlated. Finally, the result D is returned.

C. Other practical implementation issues

1) Frequency offset: Due to manufacturing limitations, the transmitter and the receiver has a center frequency offset δf. This offset will also affect the detection accuracy. However, based on the finding in [6], the frequency offset is relatively static and can be estimated based on history information. Besides, according to our experiment measurement, the offset is usually small in homogeneous networks, e.g., 262.85 Hz in ZigBee networks on average [14]–[16]. Therefore in the current implementation of our detection algorithm, we omit the frequency offset caused by activities within the same protocol.

2) When to get on board: Earlier we described how the redundancy in a preamble (the repeated signal pattern) allows us to use a few symbols to carry control signals. A prerequisite to this is for the transmitters of control messages to be able to quickly detect the presence of a preamble and be sufficiently synchronized to the incoming transmission. Again using the redundancy, the sender does so by detecting the presence of the preamble using only the first few symbols as described earlier.

D. Complexity and Overhead

Hitchhike incurs limited overhead and needs no modification to the protocol and hardware. The extra requirement only lies in the detection, as it costs little to generate the control messages and store the signal. For example, the devices only need to store the signal trace during preamble period, which is of only 256 chips in IEEE 802.15.4 [4]. For signal correlation, with a control messages set which has N control messages, the computational complexity of the correlation process is bounded with $O(N)$.

IV. Implementation

This section describes our implementation of the Hitchhike. We implement the transmitter and receiver components of Hitchhike on GNURadio/USRP2 software radio platform. We use the RFX2400 daughterboards which operate in the 2.4 GHz range. The software for the signal processing blocks is from the open source GNURadio library.

A. Transmitter Implementation

Fig. 4(a) presents the implementation of the transmitter in Hitchhike. A control message generation block is added to the standard transmitter design. Therefore the transmitter is able to transmit both data packet and control commands. A

Fig. 5. Overview of the evaluation environment. 5 USRP2 devices are used in our experiments and the evaluation is conducted in an indoor environment.

control message goes through the same modulation process as data does, while the data packet transmission process is unchanged.

B. Receiver Implementation

The implementation of the receiver is shown in Fig. 4(b). A detector block for detecting the control messages contained in the preambles has been added to the standard receiver design. The detector contains two parts: the subtractor and the correlator. The subtractor module subtracts the preamble signal from the mixed signal with the rebuilt preamble pattern. The correlator block correlates the remained signal with the predefined codes in the control messages set. Finally the detected messages are passed to the upper layer applications.

V. Evaluation

We evaluate Hitchhike in this section. Fig. 5 shows the evaluation environment and devices. We use 5 USRP2 nodes with 4 transmitters and 1 receiver. Among the transmitters, one is transmitting data packets while the rest three transmit control messages. The transmitters and the receiver are connected to a PC respectively via a switch.

We benchmark Hitchhike's performance with the following metrics in our evaluation:

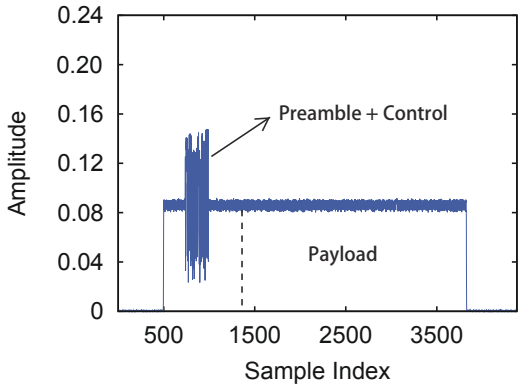
- The detection error introduced by the control signals on the preamble.
- The detection accuracy of control messages and how well the subtraction-detection algorithm works.
- The performance of Hitchhike against the state-of-the-art Side-channel [2] in terms of detection accuracy and loss rate with different SNR.

A. Experiment Settings

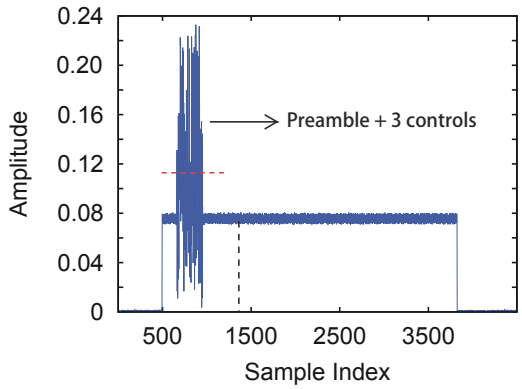
Modulation scheme. Hitchhike takes modulation and demodulation as a black box and works with various modulation schemes. In our implementation, however, we use 802.15.4 DSSS modulation to validate Hitchhike.

Test scenarios. We validate Hitchhike in environments without interference from other protocols like Wi-Fi technology. In the future, we may extend the experiments under the interference from other 2.4 GHz protocol like Wi-Fi and Bluetooth.

Parameter settings. The control messages have length of 64 chips (2 symbols), the data packet have length of 30 bytes.



(a) Only one control on the preamble.



(b) Three controls on the preamble.

Fig. 6. Illustration of received signal traces. Fig. 6(a) depicts the signal trace when only one control signal is added on the preamble. Fig. 6(b) is the case when three control messages are put on the preamble.

The preamble has length of 8 symbols, the same as specified in the IEEE standard [4]. The center frequency is set to 2.432 GHz, which overlaps with the Wi-Fi band. To test the detection accuracy for both the preamble and the control messages, we vary the transmit power of the preamble and the control messages. The powers are configured to have SNR range from 10 dB to 26 dB (by first measuring the noise power and setting the amplitude of the transmitters), therefore the SNR difference between the preamble and the control signal can range from -16 dB to 16 dB. The data transmitters and the control transmitters transmit 100 packets at each SNR level. The detection threshold for the control messages is set to 0.3.

B. Signal Trace Analysis

We begin by showing the representative signal traces when control is superposed in the way described above, in Fig. 6. Each trace is a whole packet and a dotted line delineates the preamble and the payload. Fig. 6(a) shows a single control signal added onto the preamble, resulting in the observed spikes and fluctuation in the preamble field while the payload is unaffected. Fig. 6(b) shows three control signals added onto a preamble. The signal in the overlapping period is enhanced, seen by the power level of the composite signal (the red dashed

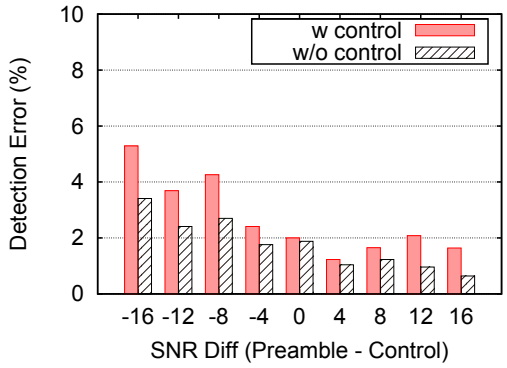


Fig. 7. Detection error of preamble when there is no external interference.

TABLE I
FREQUENCY OFFSET WITHIN ZIGBEE DEVICES.

Estimated CFO (Hz)	*avg.*	*min.*	*max.*
With Control	262.85	181.43	385.81
Without Control	151.26	112.56	181.43

line).

This set of traces also serve to highlight our testing scenarios. We first evaluate the detection accuracy of the preamble after the insertion of the control messages. We then examine the detection of the control messages from the mixed signals with Alg. 1.

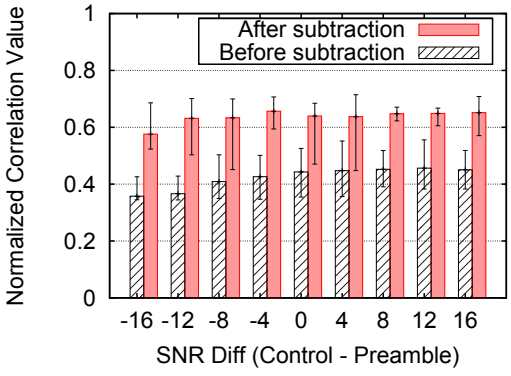
C. Detection Error of the Preamble

We first examine what effect control messages have upon the operation of a preamble by measuring the detection error of a preamble when control messages are added. We will limit our attention to a single control signal in this subsection.

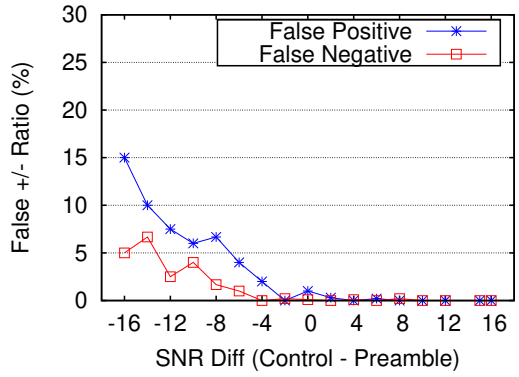
Detection error. Fig. 7 shows the detection error when there is no Wi-Fi interference. As expected, the preamble detection error decreases as the SNR difference between the two (preamble and control) increases. Overall the addition of control messages increases the preamble detection error only slightly, e.g., less than 1% on average. This shows that the orthogonality of the signals, as well as the length of the control signal have been chosen appropriately.

Estimated frequency offset. To further investigate the reason behind the increased error rate, we analyze the estimated frequency offset of the received signal. As mentioned in Section III, a receiver performs frequency offset estimation using FFT and finds the center frequency by looking for the frequency with the strongest power. We measure the estimated frequency offset with the signal traces shown in Fig. 6. When there is no interference from Wi-Fi, the estimated frequency offset is always less than 400 Hz, with and without the control messages, as is shown in Tab. I.

In summary, we conclude that the superposition of control messages with length of 64 chips on the preamble cause little adverse effect (less than 1% additional error rate and no significant frequency offset).



(a) Correlation value of control messages.



(b) Detection false positive and false negative.

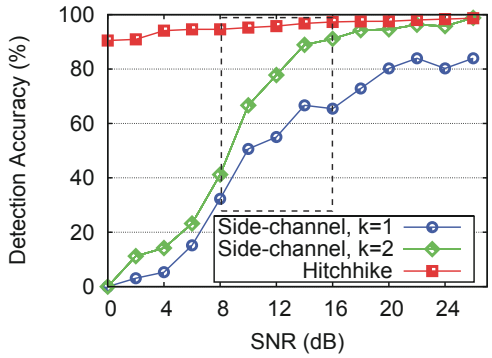
Fig. 8. Detection result for control message in single control user scenario. Fig. 8(a) shows the comparison of correlation value before and after the preamble is subtracted. Fig. 8(b) shows the detection accuracy with Alg. 1.

D. Detection Error of Control Messages

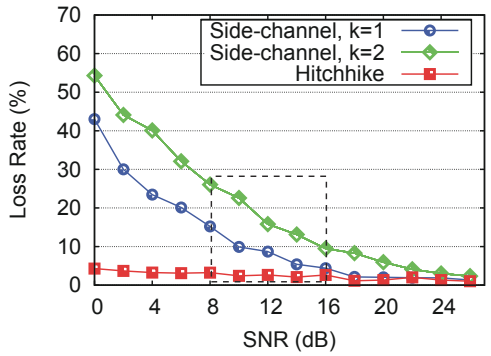
Next we evaluate the detection results of control messages. We first calculate the correlation coefficient of the control signal without subtracting the preamble, and then compare this with that after the adoption of Alg. 1 to verify the effectiveness of the algorithm. We also measure the detection accuracy at various SNR differences.

Correlation coefficient. The bars with mark in Fig. 8(a) show the correlation result for control message without subtracting the preamble signal. We see that the correlation values are only 0.1 above the detection threshold 0.3, when the control SNR is below that of the preamble by more than 8 dB. It is also interesting to see that the correlation coefficient for the control message doesn't depend heavily on the SNR difference. For example, when the SNR difference are -16 dB and 16 dB, respectively, the resulting correlation values differ only by 0.1.

The subtraction algorithm. The bars without marks in Fig. 8(a) show the corresponding correlation value after the preamble signal is subtracted from the mixed signal. Compared with the results before subtraction, the correlation value at each SNR level is improved by at least 0.2, which directly enhances the detection accuracy for control messages. The absolute value for the correlation coefficient is nearly 0.6, 0.3 higher than the threshold, even in the presence of noise.



(a) Detection accuracy of control messages.



(b) Data loss rate for Side-channel and Hitchhike.

Fig. 9. Performance comparison with Side-channel. Fig. 9(a) shows that the detection accuracy of Hitchhike is high even when the SNR of data packet is low. Fig. 9(b) depicts the loss rate of data packets caused by the two approaches.

This evaluation result indicates that our subtraction-detection algorithm works quite well in enhancing the control detection accuracy, especially when the preamble dominates the control signals (SNR difference from -16 dB to 0 dB).

Detection accuracy. Fig. 8(b) presents the detection accuracy using Alg. 1, in the form of false positive and false negative ratios. The former mainly comes from the imperfect cross-correlation among control messages, and the latter is due to interference and noise. In our design, low false negative ratio is more desirable as control messages are important. As shown and to be expected, the false positive and negative ratios are nearly 0 when the SNR of the control signal is larger than that of the preamble. While the errors increase when the preamble gets stronger by comparison, they are both at reasonably low levels, e.g., under 15% for false positive and under 6% for false negative even when control is 16 dB below the preamble.

E. Comparison with Payload-based Mechanism

We end this section with a comparison between Hitchhike and the state-of-the-art protocol Side-channel [2] for 802.15.4 networks. We examine two metrics: the detection accuracy of control messages and the loss rate of data packets caused by control messages. The data packet and the control signal use the same transmitting power and the SNR is set from 0 dB to 26 dB. The control message in Hitchhike has a length of 64 chips and data packet has a length of 30 bytes. We use two methods to encode control messages in Side-channel. The first is to interfere single chips and use 4 chips in each symbol (Side-channel, $k = 1$) and the second is to use two consecutive chips (Side-channel, $k = 2$). The detection accuracy and the extra packet loss rate are obtained by analyzing the received signal trace on receiver side. This evaluation is repeated for 10 rounds at each SNR level.

Fig. 9(a) shows the detection accuracy at different SNR levels of the data packet. We see that the detection accuracy for Side-channel is much lower than that of Hitchhike in low SNR conditions. In particular, it is less than 40% when SNR falls below 8 dB. The reason behind is that noise and external interference cause false "error" bits that destroy the encoded messages. We do see that using two consecutive chips has better accuracy than using a single chip. Lastly, the accuracy of Hitchhike remains high at all SNR levels.

We further analyze the impact control messages have on packet decoding and the result is shown in Fig. 9(b). In all cases data loss decreases with the increase in packet SNR, much as expected. In the case of Side-channel, single-chip performs better than 2-chip interference, so there is a clear tradeoff between high control detection accuracy and low data loss for Side-channel. On the other hand, Hitchhike demonstrates both high detection ratio and low error rate on packet decoding (across all SNR levels).

From this evaluation, we conclude that at high SNRs, both Side-channel and Hitchhike demonstrate high detection accuracy and less impact on packet decoding. However, when SNR decreases, Hitchhike performs better in the two metrics, especially with SNR locating within 8 dB to 16 dB where data packet has limited redundancy margin. When SNR is below 8 dB, where data packets become easy to be corrupted (refer to Fig. 3), Hitchhike can still use these corrupted packets for conveying the important control information.

VI. Discussion

As a PHY layer technique, Hitchhike can be utilized to benefit a variety of upper layer applications. The essential feature of Hitchhike is that by carrying the small control messages on the preambles, the receiver can receive both the data packets and the control messages, with little overhead and performance degradation. Here we demonstrate two applications, priority scheduling and neighbor discovery; the detailed implementation of these applications is left for future work.

Efficient priority scheduling. First, Hitchhike can be used to enhance the efficiency of priority scheduling. In wireless networks, priority scheduling is often required for providing QoS (quality of service), see e.g., [17]–[20]. Hitchhike is useful in scheduling transmitters with different priorities, especially when the schedule is urgent and efficiency is desirable. Different from traditional scheduling mechanisms, a receiver in Hitchhike can schedule transmitters by inspecting the received packet. Specifically, the receiver looks into the

preamble of the received packet and detects the control messages carried on the preamble. As multiple control messages are supported on a single preamble, the receiver is able to learn about all transmitters, their readiness to send packets, and their priorities by interpreting the control messages.

Fast neighbor discovery. Another application is in neighbor discovery [21]. In wireless networks like Wi-Fi, the users around an AP may change dynamically. A critical task for the AP is to learn the new comers as soon as possible [22]. Hitchhike can be applied in this scenario. The newly-joining devices may use a simple control signal to identity itself riding on someone else's packet. Without any extra overhead, the AP can know the newly joining devices while receiving data packets.

VII. Related Work

Recently, there has been a lot of work trying to improve the delivery efficiency of control messages. In general, they can be classified into out-of-band and in-band control. Note that out-of-band control here simply means the control messages are physically separated from data packets in time domain while in-band control is not.

Out-of-band control. Motivated by the fact that control bits in a control packet occupy a small portion of the whole packet, the authors in [1] propose to use short command sequences to replace control packets. The work Gap-sense [23] and Esense [24] code the control messages with signal pulses and energy bursts. However, the out-of-band mechanism cannot eliminate the extra air time of the control messages.

In-band control. Another promising thread is to concurrently transmit the control messages with data packets. μACK [25] uses a portion of channel resources for dedicated control transmission. The work in [26] for the first time analyzes the chip error pattern caused by interference and utilizes this pattern to convey control information in their following work Side-channel [2]. However, due to the overlap of control signal and the data payload, the two interferes with each other, especially in low SNR conditions. Also using correlation based detection, the work [1] asserts that control messages decoding can be done while receiving data. However, this work doesn't carefully consider where to put the control signals.

Motivated by the work for in-band control, Hitchhike builds the control plane on the preamble field of data packets. By decoupling control from the payload, the mutual interference between the control signal and the payload is minimized.

VIII. Conclusion and Future Work

This paper proposes Hitchhike, a novel technique that utilizes preamble for efficient control messages transmission. We carefully design control codes and develop the subtraction-detection algorithm for control messages detection. Evaluation on 5 USRP2 nodes show that control messages carried by preamble can be transmitted with high accuracy whereas the operation of preamble is with limited adverse effect. Several issues are left for future study. First, we shall extend the implementation of Hitchhike in 802.11 Wi-Fi networks, especially in OFDM where the preamble is different from that in 802.15.4. Second, we will implement the applications mentioned in Section VI that can benefit from Hitchhike.

IX. Acknowledgement

This study is supported in part by NSFC under grants of 61202359, 61373166, NSF China Major Program 61190110, NSF under grants of CIF-0910765 and CNS-1217689, NSF China under grants of 61003277, 61232018 and 61272487.

References

[1] E. Magistretti, O. Gurewitz, and E. W. Knightly, "802.11ec: collision avoidance without control messages," in *ACM MobiCom, 2012*.
[2] K. Wu, H. Tan, Y. Liu, J. Zhang, Q. Zhang, and L. Ni, "Side channel: bits over interference," in *ACM MobiCom, 2010*.
[3] A. Cidon, K. Nagaraj, S. Katti, and P. Viswanath, "Flashback: decoupled lightweight wireless control," in *ACM SIGCOMM, 2012*.
[4] IEEE Standard for ZigBee, "Wireless medium access control (mac) and physical layer (phy) specifications," 2006.
[5] IEEE Standard for WiFi, "Wireless lan medium access control (mac) and physical layer (phy) specifications," 2007.
[6] S. Sen, R. Roy Choudhury, and S. Nelakuditi, "Csma/cn: carrier sense multiple access with collision notification," in *ACM MobiCom, 2010*.
[7] B. Radunovic, R. Chandra, and D. Gunawardena, "Adaptive preambles for coexistence," in *MSR-TR-2011-15*.
[8] "cc2420 datasheet," http://www.ti.com/product/cc2420.
[9] E. Chai, J. Lee, S.-J. Lee, R. Etkin, and K. G. Shin, "Building efficient spectrum-agile devices for dummies," in *ACM MobiCom, 2012*.
[10] S. S. Hong and S. R. Katti, "Dof: a local wireless information plane," in *ACM SIGCOMM, 2011*.
[11] H. Meyr, M. Moeneclaey, and S. Fechtel, *Digital Communication Receivers: Synchronization, Channel Estimation, and Signal Processing*. New York, NY, USA: John Wiley & Sons, Inc., 1997.
[12] T. A. D. Halperin and D. Wetherall, "Taking the sting out of carrier sense: interference cancelation for wireless lans," in *ACM MobiCom, 2008*.
[13] K. Srinivasan, P. Dutta, A. Tavakoli, and P. Levis, "An empirical study of low-power wireless," *Sensor Networks, ACM Transactions on*, 2010.
[14] W. Xu, W. Trappe, and Y. Zhang, "Channel surfing: defending wireless sensor networks from interference," in *ACM IPSN, 2007*.
[15] J. Huang, G. Xing, G. Zhou, and R. Zhou, "Beyond co-existence: Exploiting wifi white space for zigbee performance assurance," in *IEEE ICNP, 2010*.
[16] X. Wu and M. Liu, "In-situ soil moisture sensing: Measurement scheduling and estimation using compressive sensing," in *ACM IPSN, 2012*.
[17] A. Saifullah, Y. Xu, C. Lu, and Y. Chen, "End-to-end delay analysis for fixed priority scheduling in wirelesshart networks," in *IEEE RTAS, 2011*.
[18] D. Li, J. Wu, Y. Cui, and J. Liu, "Qos-aware streaming in overlay multicast considering the selfishness in construction action," in *IEEE INFOCOM, 2007*.
[19] M. Li, Z. Li, L. Shangguan, S. Tang, and X. Li, "Understanding multi-task schedulability in duty-cycling sensor networks," *Parallel and Distributed Systems, IEEE Transactions on*, 2013.
[20] X. Wang, X. Lin, Q. Wang, and W. Luan, "Mobility increases the connectivity of wireless networks," *Networking, IEEE/ACM Transactions on*, 2013.
[21] D. Zhang, T. He, F. Ye, R. K. Ganti, and H. Lei, "Eqs: Neighbor discovery and rendezvous maintenance with extended quorum system for mobile sensing applications," in *IEEE ICDCS, 2012*.
[22] Y. Tong, L. Chen, Y. Cheng, and P. S. Yu, "Mining frequent itemsets over uncertain databases," in *VLDB Endowment, 2012*.
[23] X. Zhang and K. G. Shin, "Gap sense: Lightweight coordination of heterogeneous wireless devices," in *IEEE INFOCOM, 2013*.
[24] K. Chebrolu and A. Dhekne, "Esense: communication through energy sensing," in *ACM MobiCom, 2009*.
[25] J. Zhang, H. Shen, K. Tan, R. Chandra, Y. Zhang, and Q. Zhang, "Frame retransmissions considered harmful: improving spectrum efficiency using micro-acks," in *ACM MobiCom, 2012*.
[26] K. Wu, H. Tan, H.-L. Ngan, Y. Liu, and L. Ni, "Chip error pattern analysis in ieee 802.15.4," *Mobile Computing, IEEE Transactions on*, 2012.