# NoPSM: A Concurrent MAC Protocol over Low-data-rate Low-power Wireless Channel without PRR-SINR Model

Haiming Chen, *Member, IEEE,* Zhaoliang Zhang, Li Cui, *Member, IEEE,* Changcheng Huang, *Senior Member, IEEE*

**Abstract**—Concurrent MAC protocols can improve channel usage of wireless sensor networks (WSNs), and provide a high-performance infrastructure for data intensive applications. Most of existing concurrent MAC protocols are based on proactively constructed physical interference models, i.e. PRR-SINR models (PSM). However, it incurs relatively high bandwidth and energy overheads to construct PSM for WSNs. In this paper, we propose NoPSM, which does not take PSM as base to determine transmission concurrency. Instead, the base of NoPSM is reactively constructed interference relationships by passively analyzing overlapping relationships among time logs of block data transmissions and corresponding reception status of each packet in blocks. In this way, NoPSM has two salient features. Firstly, NoPSM is able to construct interference relationships among nodes quickly and accurately along with block data transmissions without needs of network downtime. Secondly, based on the constructed interference relationships, NoPSM can make decisions of transmission concurrency with a comprehensive criterion, which not only estimates quality of any active links after initiating a new link, but also estimates throughput improvement gained from concurrent transmissions. NoPSM has been implemented in Tinyos-2.1 and extensively evaluated in TOSSIM. Experimental results show that NoPSM improves system throughput by up to 60% compared with a traditional CSMA protocol, which cannot exploit potential transmission concurrency. Moreover, NoPSM can gain up to 55% throughput improvement as compared to an existing reactive concurrent MAC.

**Index Terms**—Wireless sensor network, data intensive application, concurrent transmission, interference relationship

## 1 Introduction

Wireless sensor network (WSN) is a self-organized network consisting of a large number of miniature intelligent sensor nodes. Because it can be unobtrusively embedded in the physical environment to get fine-grained observations without human operation, lots of WSNs have been deployed for various applications in recent years. One killer application of WSNs is to collect real-time continuous raw data of physical objects for high-fidelity analysis and precise prediction of disastrous events, such as patients-in-risk health status [1], volcano eruption [2], and algal bloom [3]. Nodes in the WSNs for this kind of application are tasked to sample and report sensor readings with relatively high frequencies and high resolutions, hence they produce intensive data traffic. For instance, each node for health status monitoring is equipped with 3 leads of electrocardiogram (ECG) sensors, each of which can sample physiological data at 250 Hz with 16-bit resolution, so each node can generate data consistently at the rate of 12 kbps. In a WSN for volcano monitoring, echo node is equipped with a seismic sensor and an infrasonic sensor, which are tasked to sample seismic and acoustic data at 100 Hz with 16-bit resolution respectively, so each node can produce data of 400 bytes every second.

So in these WSNs, a number of scattered nodes need to transmit bulk data to a central sink node or gateway via multi-hop relays. When premonitory signs (e.g. irregular heart rhythm [1], ground vibration [2], and featured picture of algae [3]) are detected, nodes may report readings with even higher frequencies. Since the events are usually hard to be predicted, accordingly nodes are driven to generate more intensive data simultaneously and *unpredictably*. As a result, competition over low-data-rate low-power wireless channel and communication interference amongst nodes in the data intensive WSNs are more severe than in the traditional low-duty-cycled WSNs [4]. Hence, it is more challenging to guarantee transmission throughput and delivery latency for the data intensive WSNs than for the traditional low-duty-cycled WSNs. Although some transport layer protocols exploiting block data transfer, like Flush [5] and RBC [6], have been proposed to improve throughput in data intensive WSNs, Media Access Control (MAC) protocols are supposed to be more crucial to address this challenge than the transport layer protocols, because MAC protocols can exploit physical layer information to make more accurate transmission decisions for improving channel utilization.

Recent research on the actual behaviors of the physical layer reveals two effects in wireless networks, namely Capture Effect [7], [8], [9] and Message In Message (MIM) [10]. The revealed effects violate the widely adopted 'collsion as failure' assumption and indicate the potential of concurrent transmissions. In particular, in the presence of these effects, first-arrived stronger signal of interest (SoI) can be decoded successfully despite significant interference from other transmitters. What's more, a receiver can reengage onto a later-arrived stronger SoI from an ongoing transmission or interference. By exploiting these effects, some concurrent MAC protocols like C-MAC [11] and OPC [12] have been proposed to improve channel utilization and network throughput in data intensive WSNs. Differing from the traditional CSMA-based MAC protocols, which reserve channel exclusively for one pair of nodes in their interference ranges, the concurrent MAC protocols allow the interfering nodes to transmit packets concurrently with the on-going nodes, if the on-going transmissions will not be deteriorated to an intolerable degree. So it is a fundamental problem for the concurrent MAC protocols to estimate quality of the interfered links if a new transmission is to be initiated. Because physical interference model characterizes quality of

- *Haiming Chen, Zhaoliang Zhang and Li Cui are with the Institute of Computing Technology, Chinese Academy of Sciences, Beijing China. Email: {chenhaiming, zhangzhaoliang, lcui}@ict.ac.cn*
- *Changcheng Huang is with the Department of Systems and Computer Engineering at Carleton University, Ottawa, Canada. Email: huang@sce.carleton.ca*

*Manuscript received 2015; revised 2015.*

a link when it is interfered by nodes away from different distances, it is regarded as a basis to address the problem. Nowadays the physical interference model is empirically established by pairwise measurements of Packet Reception Ratio (PRR) and Signal-to-Interference-plus-Noise-Ratio (SINR) (a.k.a PRR-SINR model) [13], [14].

However, none of Commercial-Off-The-Shelf low power radio modules (e.g. CC1000 and CC2420 [15]) provides the SINR information directly. Hence, PRR-SINR model cannot be constructed directly either. In C-MAC and OPC, each node infers PRR-SINR model through following operations: (1) any one of its neighboring nodes alone, and all possible combinations of other neighboring nodes are synchronized to transmit packets to the node *in turn*, to measure Received Signal Strength Indicator (RSSI) of received packets; (2) each of its neighboring nodes is synchronized to transmit packets to the node *simultaneously* with all possible combinations of other neighboring nodes, to measure PRR; (3) the measured PRR is paired with the associated SINR value inferred from the RSSI measurements of step (1). Considering that empirical PRR-SINR model has spatial and temporal variations in reality [16], it is necessary for C-MAC and OPC to measure RSSI of each link and update PRR-SINR model periodically, which incurs relatively high bandwidth and energy overheads for resource constrained WSNs. Besides that, the on-purpose RSSI measurements require all the on-going nodes to stop operation (a.k.a network downtime), which may significantly increase buffered data to be transmitted and adversely affect delivery latency of each packet in the data intensive WSNs.

In this paper, we propose a concurrent MAC protocol for data intensive WSNs named NoPSM, meaning that it determines transmission concurrency without proactive construction of PRR-SINR model. Instead it determines transmission concurrency based on explicit interference relationship among nodes, which is reactively constructed through passively analyzing logged time sequence of block data transmission. Briefly speaking, NoPSM transmits data packets in a block, which consists of a batch of fixed-size data packets sent together, as Seda [17] does. The start and end times of each block transmission are recorded by senders and broadcasted to their one-hop neighbors. A bitmap indicating transmission result of each packet in the block is recorded by the receivers and carried back with block-level ACK to the senders. Once the receivers complete collecting time logs, they analyze the time overlapping among transmissions of one-hop neighbors, and infer patterns of interference. Along with time log analysis, they infer PRR of links under corresponding interference from statistically analyzing the bitmap recorded by themselves. The inferred patterns of interference and corresponding PRRs are broadcasted by the receivers to one-hop neighbors, which are in turn used to make decisions of transmission concurrency.

The novelties of our proposed concurrent MAC protocol are embodied in following aspects.

1) Block data transmission is first proposed to be combined with concurrent media access control, not only for improving network throughput [17], but also for shortening period of estimating link quality under different patterns of interference. Because more than one PRR can be statistically calculated in a single round of block data transmission, if there are other nodes transmitting data concurrently, NoPSM is able to construct interference relationships in parallel.

2) Patterns of interference, which is described by different combinations of interfering nodes, is deduced from analyzing logged time sequences of block data transmissions in a distributed way. So both PRR of a link and corresponding interfering nodes, which compose the interference relationships, are passively measured along with block data transmissions. In other words, NoPSM does not require network downtime to construct interference relationships.

3) A new criterion to determine allowable concurrent transmissions is proposed in NoPSM. It checks not only how much deteriorating effect of initiating a new transmission will be on the on-going transmissions, but also how much gain of throughput will be brought in. In other words, even if any pair of active nodes can keep data transmissions above a threshold PRR under interference of a newly initiated transmission, NoPSM will not initiate the concurrent transmission when the increased throughput cannot compensate the decreased one.

4) Performance of NoPSM is extensively evaluated in TOSSIM. Experimental results show that NoPSM improves network throughput by up to $60\%$ compared with a traditional CSMA protocol, which cannot exploit potential transmission concurrency. Moreover, NoPSM can gain up to $55\%$ throughput improvement as compared to an improved version of CMAP [18], which is also a reactive concurrent MAC originally designed for wireless multihop networks.

The rest of the paper is organized as follows. In section 2, we summarize the related work. In section 3, we give an overview of NoPSM, and then elaborate design details of NoPSM in section 4. Evaluation method and experimental results are presented in section 5. In section 6, we make a conclusion of this work.

## 2 Related Work

So far some works have been done to improve performance of MAC protocols for carrying intensive data traffic in wireless sensor networks. These works can be divided into following categories.

**TDMA-based MAC protocols:** TDMA-based MAC protocol is a natural choice to improve channel utilization. WirelessHART [19] is such a standardized solution which bases on 802.15.4 physical layer and uses TDMA-based scheduling to allocate channel resource among nodes in wireless system for industrial automation. However, TDMA-based MAC protocol does not scale very well with number of nodes inherently, because number of time slots in a TDMA frame is bounded by the allowable MAC delay. Although some protocols, like ISO-MAC [20], TreeMAC [21] and i-MAC [22], have been proposed to maximize spatial reuse of time slots, they can be effective only in networks with specific topology (like Tree in [21]) or with repetitive (or predictable) traffic pattern [22]. So a stand-alone TDMA scheme has shortcomings in scalability and flexibility. Z-MAC [23] is a hybrid scheme which combines the strength of TDMA and CSMA for improving channel utilization while achieving good scalability and flexibility. However, because Z-MAC adopts a distributed neighbor discovery and graph coloring algorithm to avoid scheduling conflict in two-hop, it is essentially of high overhead, yet still capable of making binary transmission decision. In contrast, NoPSM aims to address the problem based on a CSMA scheme alone, while making more fine-grained transmission decision.

**Traffic-aware adaptation of low-duty-cycled MAC protocols:** Most of currently widely used MAC protocols are designed for low-duty-cycled WSNs with sparse and predictable data traffic, such as S-MAC [24] and B-MAC [25]. These protocols are mainly designed for reliable data delivery among nodes with asynchronous duty cycles through scheduled listening or low power listening. When traffic load becomes intensive and unpredictable, they can induce high latency and low

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2016.2547867, IEEE Transactions on Mobile Computing



throughput. Some mechanisms have been proposed to make these MAC protocols resilient to load increase, such as X-MAC [26], RI-MAC [27] and pTunes [28]. X-MAC and RI-MAC adopt strobed preamble and low power probing respectively to improve channel utilization. pTunes improves performance of X-MAC and RI-MAC by adapting optimized protocol parameters to topology, link and traffic dynamics. However, their performance is still bounded by the potential capability of traditional CSMA-based scheme. Some other adaptations exploit additional network resources, like multi-channel communication [29], [30] and on-demand wake-up [31], but the adapted MAC protocols still cannot make good use of the limited channel bandwidth, because none of them can resolve heavy interference in data intensive WSNs properly, which leads to growing invalid data transmissions or persistent access backoffs.

**Concurrent MAC protocols based on proactively constructed PRR-SINR model:** C-MAC [11] and OPC [12] were designed to exploit transmission concurrency to improve throughput of data intensive WSNs. The common working process of these two protocols is described briefly as follows. Firstly, both of them build PRR-SINR models in network before data transmission. Then, during data transmission, the node that is going to transmit data predicts SINR of each active link with its interference. Thirdly, based on the predicted SINR and referring to the PRR-SINR model, it determines whether a concurrent transmission can be initiated. As pointed out in the previous session, this kind of concurrent MAC protocols require a network downtime periodically to make RSSI measurement at whole network scale and construct SINR-PRR model for each node in advance, which incur high overheads. Hence, one motivation of our work presented in this paper is to remove the necessity of proactive construction of SINR-PRR model and periodical RSSI measurements in designing concurrent MAC protocols for data intensive WSNs.

**Concurrent MAC protocols based on reactively constructed conflict relationship:** CMAP [18] was designed for wireless ad hoc networks to improve channel usage through addressing the exposed terminal problem. It is essentially a concurrent MAC protocol too, because it exploits potential opportunity for the exposed nodes to transmit data concurrently with the sender. Differing from C-MAC and OPC, it makes decision of transmission concurrency with reactively constructed interference relationships rather than the proactively constructed SINR-PRR model. So it is the most related work to ours. However, they differ in the way to construct interference relationship and to determine transmission concurrency. Specifically speaking, CMAP infers interferer list through extracting the identity of the interfering node from the header or the trailer of the corrupted packet, while NoPSM deduces interference relationship from passively analyzing overlapped transmission periods and corresponding PRR of block data transmissions. Furthermore, CMAP makes binary transmission decision based on the defer patters inferred from the interference list, while NoPSM makes positive decision of transmission concurrency only when it is beneficial for throughput gain of all active nodes in the network.

In addition, it is worth pointing out that PIM [32] can also passively derive interference relationships among nodes by sampling and statistically analyzing the time stamps of data transmission, without generating any measurement packets. However, it is fundamentally different from NoPSM in following aspects. Firstly and foremost, it is to ultimately build PRR-SINR models among nodes for PIM through time analysis, but for NoPSM its purpose of time analysis is to construct explicit interference relationships. Hence, PIM needs to sample RSSI of received data along with time stamps, while NoPSM doesn't need. Secondly, PIM infers interference relationship in packet level, while NoPSM does it in block level. As a result, PIM can infer only one interference relationship in a single round of time analysis, while NoPSM can derive more than one. Thirdly, PIM adopts a centralized tree-based statistics collection algorithm to do time analysis, while NoPSM does it in a distributed way.

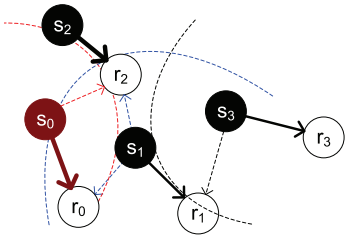



Fig. 1. A potential scenario of concurrent transmissions in WSNs, where $<s_1, r_1>$, $<s_2, r_2>$ and $<s_3, r_3>$ are the currently active links and $<s_0, r_0>$ is the link to be initiated. The solid arrow lines represent data transmissions, the dashed arcs represents proximate interference ranges of the transmitters, and the dashed arrows point to the interfered nodes.

## 3 Overview of NoPSM

In this section, we demonstrate the essence that makes NoPSM differ from traditional CSMA-based and existing concurrent MAC protocols, and the operation sequence and particular protocol architecture of NoPSM.

Fig. 1 demonstrates a potential scenario of concurrent transmissions in WSNs. Originally, $<s_1, r_1>$ is the unique pair of active nodes, and $r_2$ and $s_3$ are in the interference range of $s_1$. If with traditional CSMA-based MAC protocols, both $r_2$ and $s_3$ cannot be involved in any activity until the data transmission between $s_1$ and $r_1$ ends. But with concurrent MAC protocols, $r_2$ is able to reengage in receiving a stronger incoming signal from $s_2$ (represented by a thickened arrow line), and $s_3$ can transmit data to $r_3$ if they will not corrupt data reception process at $r_1$ (represented by a dashed arrow line). So with concurrent MAC protocols $s_1$, $s_2$ and $s_3$ can transmit data concurrently. As a result, they improve spatial reuse of the radio channel and total network throughput.

To assure each of the active links (e.g. $<s_1, r_1>$, $<s_2, r_2>$ and $<s_3, r_3>$ in Fig. 1) of high probability of successful data transmission when multiple interfering nodes transmit concurrently, we need a method to estimate the qualities of interfered links, which are usually indicated as PRR, when initiating a new link. Traditional methods are based on proactively constructed PRR-SINR model, whereas NoPSM is based on reactively and passively constructed explicit interference relationships. The interference relationship is represented by a four-dimensional tuple, i.e. i-vector=(*IID*, *Link*, *PRR*, $N_s$), where *IID* is the set of identities of interferers that transmit concurrently with the sender of the *Link*, and $N_s$ is the number of samples used in estimating the *PRR*. It is named as *interference vector (i-vector)* in this paper. For instance, an i-vector ($\{s_1, s_2\}$, $<s_0, r_0>$, 0.7, 15) means that the *PRR* of link $<s_0, r_0>$ is 0.7 when $s_0$, $s_1$ and $s_2$ transmit concurrently. Moreover, it shows that the PRR is estimated through analyzing transmission results of 15 packets.

Taking the scenario shown in Fig. 1 for example, the procedure to initiate a concurrent data transmission based on the interference relationship is as follows. When a block of data are

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2016.2547867, IEEE Transactions on Mobile Computing



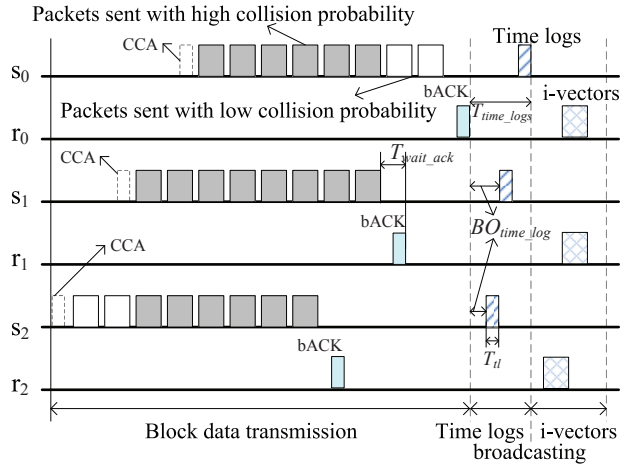



Fig. 2. Protocol operation of NoPSM illustrated by the operation sequence of the 3 pairs of nodes, i.e. $<s_0, r_1>$, $<s_1, r_1>$ and $<s_2, r_2>$ shown in Fig. 1.

pending for transmission at $s_0$, the node $s_0$ snoops the on-going traffic flows and extracts the identity information of source nodes (source ID) from the on-going flows. Since $s_0$ is within the interference range of $s_1$ and $s_2$, it can snoop the traffic flows sourcing from $s_1$ and $s_2$. Then, the node $s_0$ estimates the throughput gain if it transmits concurrently with $<s_1, r_1>$ and $<s_2, r_2>$ based on the *i-vector table*, which consists of a set of i-vectors inferred by itself and received from neighbors. If the throughput gain is not sufficiently large, the node $s_0$ will retry after a delay. Otherwise, the node $s_0$ will transmit the data block immediately.

After block data transmission, the receivers, i.e. $r_0$, $r_1$, $r_2$ and $r_3$ in the above scenario, individually record a *bitmap* to indicate reception status of each packet in a block, and encapsulate the bitmap in a block level ACK (bACK), which is sent back to the corresponding transmitter. For the transmitters, i.e. $s_0$, $s_1$, $s_2$ and $s_3$ in the above scenario, they record the start and end times of each data block transmitted, which are referred to as *time logs*. We denote a time log as a tuple, i.e. $TL=<t_0, t_1>$, where $t_0$ and $t_1$ are the start time and end time of a block transmission respectively. It is worth noting that $t_0$ and $t_1$ are not presented by wall clock time. Based on the assumption that all the nodes are time synchronized, we let each node carry a base time ($\hat{t}_b$). Then each time log can be efficiently represented by relative time, i.e. $t_0 = \hat{t}_0 - \hat{t}_b$ and $t_1 = \hat{t}_1 - \hat{t}_b$, where $\hat{t}_0$ and $\hat{t}_1$ are the wall clock time of transmission start and end respectively. The time granularity of $t_0$ and $t_1$ is millisecond (ms).

After they complete block data transmission, they broadcast time logs to their one-hop neighbors. Taking the node $r_2$ for instance, it can receive time logs from its one-hop neighbors, which are $s_0$, $s_1$, and $s_2$. After $r_2$ collects these time logs, it infers different patterns of interference from the overlapping relationships among these time logs. In particular, a partial overlap between time log from $s_2$ and that from $s_0$ or $s_1$, means interference from $s_0$ or $s_1$ in that period of time. Besides that, $r_2$ can calculate the link quality of $<s_2, r_2>$ in terms of PRR under corresponding interference pattern through statistical analysis of the bitmap recorded by itself, particularly through computing the ratio of number of bit 1 to the total number of bits in the bitmap. In such a way, the node $r_2$ can passively infer several interference relationships (i.e. i-vectors) after a round of block data transmission. The inferred i-vectors are maintained

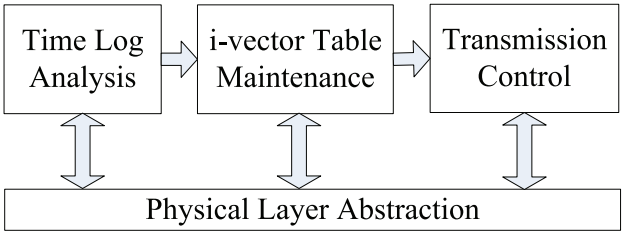



Fig. 3. Components and architecture of NoPSM.

in the i-vector table of $r_2$, and broadcasted to the one-hop neighbors of $r_2$. The nodes that receive the i-vectors, i.e. $s_0$, $s_1$ and $s_2$, update their i-vector tables with these i-vectors, which are used for estimating throughput gain in further block data transmission.

In such a way as $r_2$ does, nodes $r_0$, $r_1$ and $r_3$ infer interference relationships and broadcast the inferred interference relationships to $s_0$, $s_1$ and $s_3$ correspondingly. The above described operation of NoPSM is illustrated by Fig. 2, which includes block data transmission, followed by broadcasting of time logs and i-vectors. It should be noted that initially nodes have not yet self-inferred or received i-vectors, they always make positive decisions on transmission concurrency to establish interference relationships faster. Details about the approach to infer i-vectors will be elaborated in the next section.

Architecture of functional modules composing NoPSM is shown in Fig. 3, from which we can see that NoPSM is built upon a physical layer abstraction, and consists of the following three components: (1) Time Log Analysis that deduces the interference relationship reactively and passively; (2) i-vector Table Maintenance that maintains the interference relationship deduced by the node itself and received from neighbors; (3) Transmission Control that controls transmission opportunity through a two-tiered backoff mechanism and an advantageous transmission decision making approach.

## 4 Design of NoPSM

In this section, we elaborate on the design of NoPSM. We first design a physical layer abstraction for NoPSM to support block data transmission, and then design each component of NoPSM to fulfil functions of concurrent transmission. The design of NoPSM is based on the assumption that all the nodes in WSNs are time synchronized and transmit data packet of equal length. As used by other wireless protocols (e.g. C-MAC [11] and OPC [12], CMAP [18] and TreeMAC [21]), snooping is used in NoPSM for discovering active neighbors. We take it as a natural capability of wireless radio by continually searching for a new preamble or a termination symbol even during packet reception, as illustrated in [7].

### 4.1 Physical Layer Abstraction

The physical layer is abstracted to take responsibility for transmitting data blocks and control packets with an IEEE 802.15.4-compatible radio, like CC2420. Here we present detailed procedure of transmission of data blocks and control packets, along with the specific formats of packets defined for NoPSM.

#### 4.1.1 Block data transmission

As shown in Fig. 2, a block of data is composed of multiple continuously transmitted packets, and all packets in a block have equal length. Except the first packet, all packets in a block

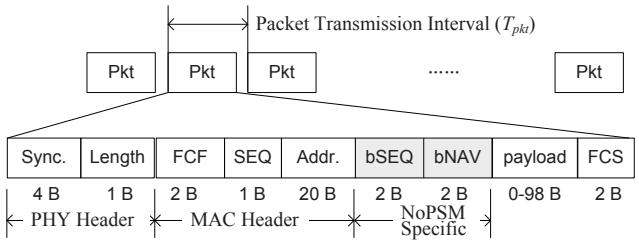



Fig. 4. Illustration of block data transmission and the format of a NoPSM data packet with an IEEE 802.15.4-compatible radio, like CC2420.

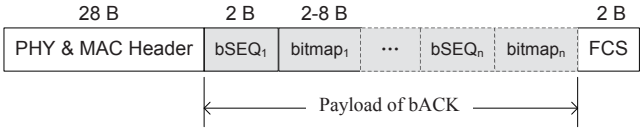



Fig. 5. The format of a block-level ACK. The dashed rectangles indicate optional fields. So at least one pair of bSEQ and bitmap are carried by a block level ACK.

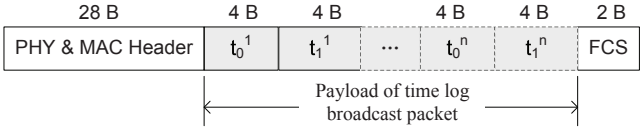



Fig. 6. The format of a time log broadcast packet. The dashed rectangles indicate that more than one time log can be carried by a broadcast packet.

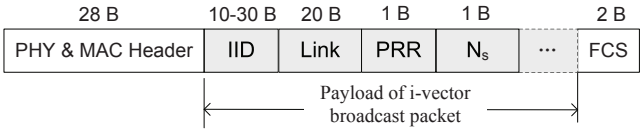



Fig. 7. The format of an i-vector broadcast packet. The dashed rectangles indicate that more than one i-vector can be carried by a broadcast packet.

are transmitted immediately after the previous one without clear channel assessment (CCA). We refer the time interval from transmitting the first bit of a packet to transmitting the first bit of the successive packet in a block as packet transmission interval as shown in Fig. 4. The packet transmission interval consists of packet transmission time and event processing time during packet transmission and reception. Since the physical layer of the radio modules in widely used nodes is usually set with a fixed data rate, e.g. 250Kbps for CC2420, the transmission time of each packet in a block is fixed and in millisecond (ms). Moreover, the total time of processing significant events that occur during the transmission and reception of a packet over a CC2420 radio is also fixed, i.e. about 600 microsecond ($\mu$s) [33]. So in this way, the packet transmission interval can be seen as a constant ($T_{pkt}$).

The format of a NoPSM data packet is shown in Fig. 4. We can see that the first two fields are physical layer headers, which are synchronization preambles and frame length. The Frame Control Field (FCF), Data Sequence Number (SEQ) and Address Information (Addr.) are MAC layer headers defined by IEEE 802.15.4. The payload of NoPSM data packet is prefixed with two specific fields. One is the bSEQ field which carries the sequence number of the block, and the other is the *bNAV* (block Network Allocation Vector) field which indicates the remaining transmission time of the block. Length of these two field are both 2 bytes. The packet is end with a Frame Check Sequence (FCS). Because the field of frame length is 1 byte long, the maximum length of a NoPSM data packet at the MAC layer is 127 bytes. Considering the space occupied by the IEEE 802.15.4 MAC layer header, the NoPSM-specific header and the FCS, payload of a NoPSM data packet can be maximally 98 bytes long.

After transmission of a block, the sender waits for a block-level ACK (bACK) from the receiver for a bACK from the receiver for a period of up to $T_{wait\_ack}$ ms. The format of a bACK is shown in Fig. 5. Each bACK carries *at least* one pair of bSEQ and bitmap to indicate whether the corresponding packet in the previously transmitted blocks is successfully received. Length of a bitmap is 2∼8 bytes, because a block has 16∼64 packets. The reason why a bACK may carry more than one bitmap is that NoPSM allow transient loss of bACK, and let subsequent bACKs feed back the packet reception status of previous block data transmissions. Details about accumulated encapsulation of bitmaps in a bACK will be further explained in Subsection 4.4.1. Based on the bitmaps in the bACK fed back by the receiver, the sender can selectively retransmits the corrupted packets.

#### 4.1.2 Control packet broadcasting

Immediately after block data transmission, NoPSM requires the sender nodes to broadcast *time logs*. The format of a time log broadcast packet is shown in Fig. 6. Specifically speaking, the sender nodes broadcast the just recorded *time logs* after the last one (e.g. $s_0$ in Fig. 2) ends transmitting block data.

In detail, for a sender node, when it finishes transmission, it firstly snoops the channel, and extracts the *bNAV* field from the packet header of each snooped flow, and calculates the end time of all on-going flows by the following equation.

$$T_{last\_end} = max\{bNAV_i | 1 \leq i \leq N_f\} + T_{wait\_ack}, \qquad (1)$$

where $N_f$ is the number of snooped on-going flows, and $T_{wait\_ack}$ is a constant period of time for waiting for a block level ACK. As a special case, if no flow is snooped, i.e. the currently end-sending node is the last one, the above equation is simplified to $T_{last\_end} = T_{wait\_ack}$. Then, the currently end-sending node turns to wait for a block level ACK, and prepares to send time logs $T_{last\_end}$ later.

Because it is highly probable that more than one node will send time logs simultaneously $T_{last\_end}$ later, we design a distributed coordination mechanism to reduce collision probability among them. The principle of the coordination mechanism is to make the nodes back-off for a period of time before sending time logs. In particular, the time duration of back-off for a node ($BO_{time\_log}$) is linearly proportional to the number of on-going flows snooped by it ($N_f$), which is formulated by the following equation.

$$BO_{time\_log} = T_{time\_logs} - N_f \cdot T_{tl}, \qquad (2)$$

where $T_{time\_logs}$ and $T_{tl}$ are two constants for NoPSM. $T_{time\_logs}$ is the duration of the period for broadcasting time logs, and $T_{tl}$ is the duration of transmitting a time log packet. For example, as shown in Fig. 2, node $s_1$ and $s_0$ snoops one and zero on-going flow respectively when they end transmitting a block of data. According to the equation (2), $s_1$ has a shorter back-off period than $s_0$, so $s_1$ broadcasts a time log packet before $s_0$.

For the receiver nodes, once they receive the first arrival time log, they wait for a period of $T_{time\_logs}$ ms. Then they infer i-vectors through analyzing the time logs collected from neighbors and the bitmaps recorded by themselves, and broadcast the inferred i-vectors to one-hop neighbors. The format of an i-vector broadcast packet is shown in Fig. 7. Still taking the

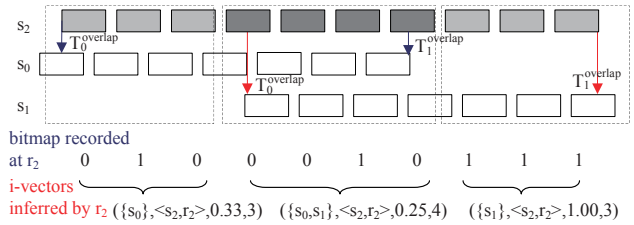



Fig. 8. Illustration of time log analysis at a receiver node $r_2$ based on the time logs received from $s_0$, $s_1$ and $s_2$.

scenario shown in Fig. 1 and corresponding node operations shown in Fig. 2 for example, $r_2$ firstly receives a time log broadcast packet from $s_2$, then from $s_1$ and $s_0$ in sequence. Based on the received time logs, $r_2$ can infer i-vectors indicating interference relationships among $s_2$, $s_1$ and $s_0$. Then $r_2$ broadcasts the inferred i-vectors to $s_2$, $s_1$ and $s_0$.

It should be pointed out that the time logs and i-vectors are not required to be broadcasted immediately after each round of block data transmission. Instead, they are broadcasted every $C_{tl}$ rounds of block data transmission, where $C_{tl}$ is larger than 1, to reduce costs of broadcasting control packets. So we can see that more than one time log are encapsulated into a control packet, as shown in Fig. 6. Besides that, the time logs may be corrupted during broadcasting due to existence of hidden terminals. For example, as shown in Fig. 1, when $s_2$ finishes data transmission, it can only snoop the packets from $s_0$. If $s_1$ ends transmitting later than $s_0$, the time log packets broadcasted by $s_2$ may be corrupted at $r_2$. To improve probability of successful reception of time log packets, we make each control packet carry the time logs which have been sent in the previous $N_{tl}$ broadcasting periods, where $N_{tl}$ is a constant parameter of NoPSM. In addition, the i-vectors can also be corrupted during broadcasting. For example, as shown in Fig. 1 and Fig. 2, $r_0$ and $r_1$ received the first time log broadcast packet from $s_1$ at the same time. So after waiting for a period of $T_{time\_logs}$ ms, they broadcast the inferred i-vectors simultaneously, which incurs packet corruption at $s_1$. Because the i-vectors will be updated reactively and periodically with a short interval (see details in Subsection 4.3), we don't make the i-vectors be broadcasted more than one time.

### 4.2 Time Log Analysis

As mentioned in the section of overview, i-vectors are the basis to make decisions of transmission concurrency. In this section, we elaborate the approach to infer i-vectors through time log analysis.

As shown in Fig. 2, all the sender nodes are required to broadcast time logs after block data transmission. Each receiver node calculates the overlapping time periods between its own and neighbors' block data transmissions by comparing these time logs. Fig. 8 gives an illustration of the procedure of precise time log analysis at the node $r_2$ in the scenario shown in Fig. 1. It shows that $r_2$ receives time log broadcast packets from $s_0$, $s_1$, and $s_2$. By comparing the three time logs, $r_2$ can infer that the first three packets in the block transmitted by $s_2$ is interfered by $s_0$, the intermediate four packets are interfered by $s_0$ and $s_1$, and the last three packet are interfered by $s_1$. The bitmap recorded by $r_2$ is 0100010111, which indicates that half of packets in the block is corrupted during transmission. Through cross analysis of the bitmap and the three patterns of interference, $r_2$ can infer that the PRR of $<s_2,r_2>$ can be 0.33 (i.e. 1/3), 0.25 (i.e. 1/4) and 1.00 (i.e. 3/3) respectively when

**Algorithm 1** Inference of i-vectors from time log analysis: LogA($TL_0$, $TL$, $M$, $bitmap$, $bsize$)

1: **Constant:** $T_{pkt}$, $C_{max}$
2: $interId[0..bsize-1] \leftarrow \emptyset$
3: **for** $i \leftarrow 0$ to $M-1$ **do**
4: $T_0^{overlap} \leftarrow (TL_0.t_0 > TL_i.t_0)\ ?\ TL_0.t_0\ :\ TL_i.t_0$
5: $T_1^{overlap} \leftarrow (TL_0.t_1 < TL_i.t_1)\ ?\ TL_0.t_1\ :\ TL_i.t_1$
6: $j_0 \leftarrow (T_0^{overlap} - TL_0.t_0)/T_{pkt}$
7: $j_1 \leftarrow (T_1^{overlap} - TL_0.t_0)/T_{pkt}$
8: **for** $j \leftarrow j_0$ to $j_1$ **do**
9: $interId[j] \leftarrow interId[j] \cup \{\mathrm{ID}(TL_i)\}$
10: **end for**
11: **end for**
12: $interPtn \leftarrow \emptyset$
13: **for** $i \leftarrow 0$ to $bsize-1$ **do**
14: **if** $(|interId[i]| < C_{max} \wedge |interPtn| < 2^{C_{max}} \wedge interId[i] \notin interPtn)$ **then**
15: $interPtn \leftarrow interPtn \cup \{interId[i]\}$
16: **end if**
17: **end for**
18: $n_1, n \leftarrow 0$
19: **for** $i \leftarrow 0$ to $|interPtn|-1$ **do**
20: **for** $j \leftarrow 0$ to $bsize-1$ **do**
21: **if** $(interId[j] = interPtn_i)$ **then**
22: $n \leftarrow n+1$
23: **if** $(bitmap[j]=1)$ **then**
24: $n_1 \leftarrow n_1+1$
25: **end if**
26: **end if**
27: **end for**
28: update_ivector_table($interPtn_i$, $<s_0,r_0>$, $n_1/n$, $n$, 1)
29: **end for**

interfered by $s_0$ alone, $s_0$ and $s_1$ together, and $s_1$ alone. Hence, three i-vectors can be inferred from the time log analysis, which are ($\{s_0\}$, $<s_2,r_2>$, 0.33, 3), ($\{s_0, s_1\}$, $<s_2,r_2>$, 0.25, 4) and ($\{s_1\}$, $<s_2,r_2>$, 1.00, 3). It is worth noting that $<s_2,r_2>$ may also be slightly interfered by $s_3$ (i.e. far-away interference [34]), but we cannot identify it in the i-vectors. Such unidentifiable far-way interference are seen as aggregated back ground noise in NoPSM, because it can be similar for each node in the data intensive WSNs and needn't be specifically taken into account when making decisions of transmission concurrency.

From the above illustration, we can see that two main steps are included in the procedure of inferring i-vectors through time log analysis. Firstly, the receiver node computes overlapping periods of time logs to infer interference patterns, and then it estimates the PRR of the current link under different interference patterns by statistically analyzing the bitmap. Algorithm 1 gives the pseudocode for inferring i-vector from time log analysis. For describing the procedure clearly, hereafter we refer to the receiver node carrying out the time log analysis as the current node, and its associated block data transmission and link as the current transmission and the current link respectively.

The algorithm has five input parameters. $TL_0$ is the time log of the current transmission. $TL$ is a set of time logs of its neighbors and $M$ is the number of time logs in $TL$. $bitmap$ indicates reception status of each packet in the current transmission. $bsize$ is the number of bits in $bitmap$. The packet transmission interval ($T_{pkt}$) and the maximum cardinality of *IID* in i-vector ($C_{max}$) are taken as two constants in the algorithm. The cardinality of IID is set to be no larger than $C_{max}$ to keep size of i-vector table within a proper range. Line

**Algorithm 2** Maintaining i-vector table with self-inferred or received i-vectors: update_ivector_table(*iid*, *link*, $prr$, $n_s$, *ttl*)

1: Find the i-vector in the table that matches *iid* and *link*
2: **if** (NOT found) **then**
3: add the i-vector (*iid*, *link*, $prr$, $n_s$) to the table
4: **else if** ($ttl > 0$) **then**
5: $PRR$, $N_s \leftarrow$ corresponding elements of the matched i-vector (*iid*, *link*, $PRR$, $N_s$)
6: $PRR \leftarrow (PRR \cdot N_s + prr \cdot n_s)/(N_s + n_s)$
7: $N_s \leftarrow N_s + n_s$
8: update the matched i-vector with the new $PRR$, $N_s$
9: broadcast the updated i-vector (*iid*, *link*, $PRR$, $N_s$, $ttl - 1$)
10: **else**
11: replace the corresponding elements of the matched i-vector (*iid*, *link*, $PRR$, $N_s$) with $prr$, $n_s$
12: **end if**

2∼11 compute the interferers for each packet, and the results are stored in an array named $interId$. In line 9, we use a function named $ID(TL_i)$ to return the source ID of the time log $TL_i$. Line 12∼17 find out the interference patterns, and the results are stored in a set named $interPtn$. Since we limit the cardinality of *IID* in i-vector to be no larger than $C_{max}$, the maximum number of interference patterns in $interPtn$ is $2^{C_{max}}$. Line 18∼29 estimate the PRR of the current link under each pattern of interference and update the i-vector table. The last parameters to call the update_ivector_table function is the time to live (TTL) of the inferred i-vector when it is broadcasted to neighbors. The detailed rules to update the i-vector table will be discussed in Subsection 4.3.

The time complexity of algorithm 1 is determined by Line 3∼11 and Line 19∼29. Through simple analysis, we can get that the time complexity of these two subroutines is $O(M \cdot bsize)$ and $O(2^{C_{max}} \cdot bsize)$ respectively. So the time complexity of algorithm 1 is $O((2^{C_{max}} + M) \cdot bsize)$.

### 4.3 Interference Vector Table Maintenance

Through time log analysis, the current node infers some i-vectors, which are maintained into a data structure, namely i-vector table. At the same time, as shown in Fig. 2, the inferred i-vectors are broadcasted to neighbors. Neighbors who receive the i-vector broadcast packets will also update the i-vector table. So a module named update_ivector_table is called when time log analysis finishes and when i-vectors are received from neighbors.

The detailed process of updating i-vector table is shown in Algorithm 2. For simplicity, we refer to the i-vectors inferred from time log analysis by itself as type I i-vectors, and the i-vectors received from neighbors as type II i-vectors.

Initially, the i-vector table is empty. For a new i-vector (*iid*, *link*, $prr$, $n_s$) (referred to as $V_{new}$ later for short), if there's no i-vector in the table that matches *iid* and *link*, it simply adds the new i-vector to the table. Otherwise, for the type I i-vectors (i.e. $ttl = 1$), suppose the matched existing i-vector is (*iid*, *link*, $PRR$, $N_s$) (referred to as $V_{mat}$ later for short), it updates the existing i-vector according to the rules formulated by Equation (3).

$$PRR = \frac{PRR \cdot N_s + prr \cdot n_s}{N_s + n_s}, N_s = N_s + n_s. \quad (3)$$

Specifically speaking, $PRR$ is updated by making an average over the ones in $V_{new}$ and $V_{mat}$, and $N_s$ is updated by adding the one in $V_{new}$ to $V_{mat}$ correspondingly. After updating i-vector table, the updated i-vector is broadcasted to one-neighbors with $ttl$ decreased to 0. For the type II i-vectors (i.e. $ttl = 0$), $V_{mat}$ is directly replaced with $V_{new}$. Gradually each node accumulates a list of i-vectors for links originated from itself and other nodes, which is used to predict throughput gain for making decisions of transmission concurrency. The i-vectors in the i-vector table are maintained in soft state, which means a timer ($T_{out}$) is set to periodically time out stale i-vectors to accommodate dynamically varied communication environment.

### 4.4 Transmission Control

This module is the core of NoPSM, which controls transmission opportunity of each node to achieve high performance of block data transmission. NoPSM controls transmission opportunity through effective backoff control and advantageous transmission determination. Suppose that a node $s_0$ having data blocks pending to be transmitted, it starts with a backoff period ($T_{bo}$), followed by a period of clear channel assessment (CCA) for $T_{cca}$ ms. During CCA period, the node not only samples the CCA register, but also snoops whether there are some on-going flows. Based on the average of CCA samples and the extracted information from the snooped on-going flows, the node determines whether or not it can start transmission. In this section, we elaborate the design of backoff mechanism and decision making approach.

#### 4.4.1 Transmission backoff

Each node maintains a backoff timer denoted by $T_{bo}$ and a contention window denoted by $[CW_{low}, CW_{up}]$. $T_{bo}$ is set to a random value between $CW_{low}$ and $CW_{up}$, which means nodes backoff for $T_{bo}$ ms before the CCA period of a block transmission. The contention window is [0,0] initially and updated by two backoff policies, namely block backoff and cluster backoff.

1) Block backoff is adopted when the sender receives a bACK. In this case, the sender reset the $CW_{low}$ to 0, and estimates the average PRR, denoted by $PRR'$, of the link based on the bitmaps carried back in the bACK. Then the sender updates the $CW_{up}$ according to the following rules:

$$CW_{up} = \begin{cases} 0, & PRR' > \eta_{cw} \\ CW_{min}, & PRR' \leq \eta_{cw} \ and \ CW_{up} = 0 \\ 2 \cdot CW_{up}, & PRR' \leq \eta_{cw} \ and \ 2 \cdot CW_{up} < CW_{max} \\ CW_{max}, & PRR' \leq \eta_{cw} \ and \ 2 \cdot CW_{up} \geq CW_{max} \end{cases}, \quad (4)$$

in which $\eta_{cw}$ is a constant threshold, $CW_{min}$ and $CW_{max}$ are the minimal and maximal size of the contention window respectively. $CW_{max}$ should not be smaller than the transmission time of a block. The rationale behind the updating rule is that $PRR'$ below a threshold level is taken as a significant indicator of existence of undetected concurrently on-going transmitters. So NoPSM will exponentially increase backoff time to reduce probability of collision with undetected transmitters in the case of $PRR'$ below a threshold level. If $PRR'$ is above the threshold, which means the current decision of transmission concurrency is correct, NoPSM will cancel backoff to improve channel utility and data throughput. So block backoff is mainly used to prevent transient packet losses when hidden terminals exist.

2) Cluster backoff is adopted when the sender has not received any bACK for $N_{uack\_blk}$ continuous blocks. In this case, the sender stops transmitting and sets the contention window to $[CB_{min}, CB_{max}]$, where $CB_{min}$ and $CB_{max}$ are two constant parameters of NoPSM. The rationale behind the updating rule is that unacknowledgement of $N_{uack\_blk}$ continuously blocks is

taken as an indicator of severe interference incurred from intensive data transmission of undetected concurrently on-going transmitters. So $CB_{max}$ should be large enough to complete transmission of $N_{uack\_blk}$ blocks of data.

Hence with these two backoff policies, based on the transmission procedure described in Subsection 4.1, after transmission of a block, the sender waits for a bACK from the receiver. If the bACK is not received within a period of $T_{wait\_ack}$ ms, the sender neither immediately retransmits all packets of the unacknowledged block, nor adjusts the contention window. Instead it transmits the next block in the buffer. In this way, NoPSM prevents wasteful retransmissions and backoffs due to occasional loss of bACK. Correspondingly, the bACK sent by the receiver are accumulative and carries $1 \sim N_{uack\_blk}$ bitmaps for the previous received blocks, where $N_{uack\_blk}$ is the maximum number of blocks allowed to be transmitted continuously without acknowledgement.

#### 4.4.2 Transmission decision

After transmission backoff, the sender node starts a period of CCA for $T_{cca}$ ms. During the CCA period, the sender node samples the CCA register, and searches for outliers in the samplings. The outliers are such points that have channel energy significantly below the noise floor, which is estimated in the same way as B-MAC [25]. If an outlier exists, the channel is assessed to be clear and the node sends the pending blocks immediately. Otherwise, the sender node snoops the on-going flows for extracting useful information, such as the address information in packet header (i.e. *Addr.*) and the remaining transmission time (i.e. $bNAV$). Through the snooped information, the node determines whether to defer or start concurrent block data transmission with the following rules.

1) If more than $C_{max}$ flows are heard, the sender defers its transmission. According to the $bNAV$ fields of the snooped flows, it can know which of the on-going flows ends firstly, then it retries to access the channel when the flow with the earliest end time finishes.

2) If the intended receiver is now involved in any one of the on-going flows, it also defers its transmission and retries when the corresponding flow ends.

3) Otherwise, the node estimates the throughput gain of its transmission and start transmission only when its transmission leads to sufficient throughput improvement. Suppose the node $s_0$ overhears $k$ active links, $<s_i, r_i>$, where $1 \le i \le k$. Let

$$T = \{s_i | 1 \le i \le k\}. \tag{5}$$

$s_0$ can estimate the current *PRR* of each active link $<s_i, r_i>$ under the interference of other nodes in $T$, denoted by

$$PRR(T \backslash \{s_i\}, < s_i, r_i >), \tag{6}$$

by looking up an i-vector ($T\backslash\{s_i\}$, $<s_i, r_i>$, *, *) in the i-vector table. If an i-vector is found, the corresponding *PRR* element in the matched i-vector is returned. If no matched i-vector is found, the result of equation (6) is estimated to be 1. For the latter case, it is probable that the nodes $s_i$ have not yet transmitted concurrently with the other nodes in $T\backslash\{s_i\}$, and thus no i-vector about *PRR* of $<s_i, r_i>$ under the interference of other nodes in $T$ has been established before. However, there is a high chance that $s_i$ starts a concurrent transmission before $s_0$'s decision, and an i-vector with this interference pattern will be built soon. With the estimated result of equation (6), node $s_0$ estimates the aggregated throughput achieved by current set of active nodes to be

$$TH_T = \sum_{s_i \in T} PRR(T \backslash \{s_i\}, < s_i, r_i >) \tag{7}$$

If $s_0$ transmits concurrently with nodes in $T$, the *PRR* of the currently active links will change. We denote the *PRR* of each active link $<s_i, r_i>$ when $s_0$ transmits concurrently with nodes in $T$ by

$$PRR(T \cup \{s_0\} \backslash \{s_i\}, < s_i, r_i >). \tag{8}$$

The *PRR* of active link $<s_i, r_i>$ is also estimated by looking up the i-vector table as described above. When no matched i-vector of $<s_i, r_i>$ is found, the result of equation (8) is also estimated to be 1, in order to encourage unknown concurrent transmission and build the corresponding i-vector soon.

If there exists a node $s_i$ in $T \cup \{s_0\}$ such that

$$PRR(T \cup \{s_0\} \backslash \{s_i\}, < s_i, r_i >) < \eta_{prr}, \tag{9}$$

where $\eta_{prr}$ is a constant parameter of NoPSM, $s_0$ defers its transmission to avoid deteriorating the PRR of the on-going flows too severely.

Otherwise, node $s_0$ estimates the aggregated throughput that will be achieved if it transmits concurrently with nodes in $T$, as

$$TH_{T \cup \{s_0\}} = \sum_{s_i \in T \cup \{s_0\}} PRR(T \cup \{s_0\} \backslash \{s_i\}, < s_i, r_i >). \tag{10}$$

$s_0$ decides to transmit concurrently if

$$TH_{T \cup \{s_0\}} \ge (1 + \alpha) TH_T, \tag{11}$$

in which $\alpha$ ($>0$) is a constant parameter of NoPSM, otherwise it also defers its transmission.

If $s_0$ goes into deferral status, it retries when the on-going flow with the earliest end time finishes.

It is worth noting that the hidden terminal problem cannot be avoided by NoPSM, because the sender can only detect one-hop active links by snooping the channel. Besides that, NoPSM can have a varied version of hidden terminal problem. The hidden terminal exists outside the communication range of the sender and will not interfere with the imminent link, but it can interfere at least one of the on-going flows. Taking the scenario shown in Fig. 1 for example, where there are three active links, namely $<s_i, r_i>$ ($1 \le i \le 3$), $s_0$ is to launch a concurrent link. Under this circumstance, $s_3$ is such a hidden terminal for $s_0$, because $s_0$ only observe that $s_1$ and $s_2$ are active nodes, and estimate $TH_T$ and $TH_{T \cup \{s_0\}}$ by taking $\{s_1, s_2\}$ as $T$. As a result, $TH_T$ and $TH_{T \cup \{s_0\}}$ are both overestimated. The overestimation leads to make false positive decisions of transmission concurrency, and induces failures of data transmission in all active links. We alleviate adverse effect of the hidden terminal problem through introducing random backoffs before block data transmission, which has been elaborated in the Subsection 4.4.1.

## 5 Evaluation

### 5.1 Methodology and Settings

We implement NoPSM in TinyOS-2.1 and evaluate its performance on TOSSIM. TOSSIM 2.1 adopts a physical layer model, named closest-pattern matching model, which is derived from real-world experimental traces on a practically used radio module (i.e. TI CC2420 [15]). This model captures the complex dynamics of noise and interference and simulates the packet delivery based on a measured SINR/PRR curve. Hence, it has been shown that TOSSIM with the trace-driven physical model can approximately simulate low-power wireless communications [35] and packet delivery behaviors of WSNs [36].

In our implementation, we set the constant parameters of NoPSM as shown in table 1. Some settings of these parameters

TABLE 1
Parameter settings of NoPSM

| Protocol module | Parameter | Description | Setting |
|---|---|---|---|
| Physical Layer Abstraction | $T_{cca}$ | Duration of CCA sampling | 12 ms |
| | $T_{pkt}$ | Packet transmission interval | $psize/250+0.6$ ms |
| | $T_{wait\_ack}$ | Time duration to wait for bACK | 4 ms |
| | $T_{time\_logs}$ | Duration of the period for broadcasting time logs | $C_{max} \times T_{tl}$ ms |
| | $T_{tl}$ | Expected time to transmit a time log packet | 1.5 ms |
| | $C_{tl}$ | Cycle of time logs broadcasting period in number of block data transmissions | 5 |
| | $N_{tl}$ | Number of time logs carried by each control packet in number of cycles of previous broadcasting period | 3 |
| Time Log Analysis | $C_{max}$ | Maximal cardinality of IID | 3 |
| i-vector Table Maintenance | $T_{out}$ | Timeout period of i-vectors | 60 s |
| Transmission Control | $\eta_{cw}$ | Minimal PRR of the current link | 0.5 |
| | $CW_{min}$ | Minimal backoff when the link quality is below $\eta_{cw}$ | 4 ms |
| | $CW_{max}$ | Maximal backoff when the link quality is below $\eta_{cw}$ | $bsize \times psize/250$ ms |
| | $N_{uack\_blk}$ | Maximal number of continuously unacknowledged blocks | 4 |
| | $CB_{min}$ | Minimal backoff when $N_{uack\_blk}$ blocks unacknowledged | $CB_{max}/2$ |
| | $CB_{max}$ | Maximal backoff when $N_{uack\_blk}$ blocks unacknowledged | $N_{uack\_blk} \times CW_{max}$ |
| | $\eta_{prr}$ | Minimal PRR of any active link | 0.5 |
| | $\alpha$ | Ratio of throughput improvement | 0.1 |

are based on our analysis of practical use cases, such as $T_{pkt}$, $T_{time\_logs}$, $T_{tl}$, $C_{max}$, $\eta_{prr}$, $\eta_{cw}$, $CW_{min}$, $CW_{max}$, $CB_{min}$, and $CB_{max}$. $T_{pkt}$ and $CW_{max}$ are set in accord with the length of packet ($psize$) and the size of block ($bsize$) in each run of experiment, where nodes transmit data at rate of 250 Kbps. Others are determined by comprehensive preliminary experiments, such as $T_{cca}$, $T_{wait\_ack}$, $C_{tl}$, $N_{tl}$, $T_{out}$, $N_{uack\_blk}$, and $\alpha$. These parameters are set to achieve steady performance of NoPSM.

To evaluate performance of NoPSM with unpredictably intensive traffic load, we define a traffic generation model. It creates 10 bursts of bulk traffic in each flow, and the start time and the source of each flow are randomly assigned. Once a flow of bursty bulk data traffic is generated, the source node transmits data at the maximum rate for 20 seconds unless otherwise mentioned. Number of nodes ($n$) and size of deployment area ($l \times l$) are set depending on flow density ($d$), i.e. $n = 2d$ and $l = \lceil 100\sqrt{n} \rceil$. The transmit power of each node is set to 0 dBm (i.e. transmission range is about $100 \sim 120$m), so as to ensure that each node has at least one direct neighbor. To evaluate performance of NoPSM independent of routing protocol, all the generated flows are destined to a closest one-hop neighbor, and each node is alternatively assigned as sender or receiver in a simulation run, unless otherwise mentioned.

To compare performance of NoPSM with other currently proposed protocols, we choose B-MAC [25] and CMAP [18] as comparison objects. B-MAC is a CSMA-based protocol, which means nodes in the interference range of current transmitters defer to send packets. For data intensive WSNs, the duty cycle of B-MAC is set to 100% to achieve high performance. We refer to the B-MAC with full duty cycle as CSMA. Hence, performance of CSMA is regarded as a baseline to see how much improvement can be taken from concurrent transmission. CMAP is a reactive concurrent MAC based on interference relationship, and is the most related work with ours. The difference between CMAP and NoPSM has been pointed out in Subsection 2. To make comparison results more persuasive, we adapt CMAP to have the capability of block data transmission and identifying conflicting links through retrieving address information not only in the tail but also in the data part of packets. This adaption improves the performance of CMAP by 8% $\sim$15% from our preliminary evaluation. In this paper, we refer to the adapted version of CMAP as CMAP+ to distinguish it from the original one. So both CMAP+ and NoPSM can work with block data transmission, but CSMA transmit only one packet once getting access to the channel. In other words, the $bsize$ of CMAP+ and NoPSM can be set to larger than 1, while the $bsize$ of CSMA is 1 constantly in the following experiments.

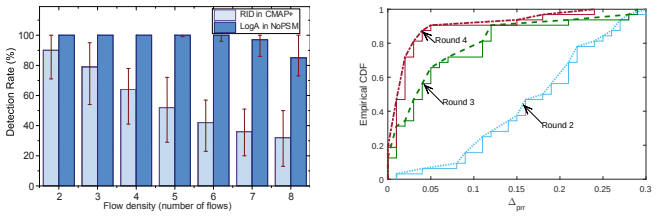
Metrics used in our performance evaluation are *delivery ratio*, *system throughput*, *delivery latency*, and *energy consumption per byte*. The delivery ratio is defined as the average percentage of the number of successfully received packets to the total number of sent packets for each flow. The system throughput is defined as the sum of average data delivery rate of all flows generated in a run of experiment. The delivery latency is defined as the time duration from the time when a packet is put into the sender's transmission buffer to the time when the receiver correctly receives the packet. To measure the energy consumption, we add an energy management module to TOSSIM, which separately accumulate time of each radio state, namely transmitting, receiving and idle. We multiply the accumulated time in each state by the corresponding rated power listed on the CC2420 data sheet [15]. Then, we compute the energy consumption per byte as the result of dividing the total power consumption in the three states by the total number of byte of payload delivered. In addition, we conduct a study on the *fairness* of NoPSM and compare messaging overhead of NoPSM with that of CMAP+. We also discuss the *applicability* of NoPSM in mobile scenarios. It is worth noting that all the random flows are destined to a one-hop neighbor, so as to evaluate performance of MAC protocols independent of routing protocol and without bandwidth constraint from a sink. In such a way, the system throughput can be a good indicator of channel utilization, and delivery latency essentially means the averaged delay for each node to access channel.

### 5.2 Performance of Time Log Analysis

NoPSM and CMAP+ propose different methods to establish interference relationships among nodes. For NoPSM, it is the component of time log analysis (LogA) that is in charge of detecting interference relationships, while for CMAP+ it adopts a Receiver-based Interferer Detection (RID) mechanism. Since interference relationship is the unique basis to determine transmission concurrency, it is crucial for performance of NoPSM and CMAP+. So it is necessary to evaluate performance of

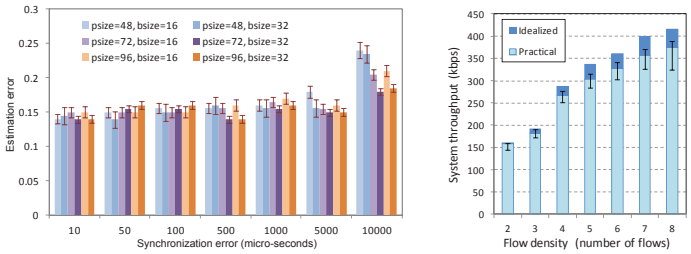
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2016.2547867, IEEE Transactions on Mobile Computing






(a) (b)

Fig. 9. (a) Comparison of detection rates of LogA in NoPSM and RID in CMAP+ with different flow density. (b) The empirical CDF of the absolute difference of PRR ($\Delta_{prr}$) between the current round and the previous round of LogA in NoPSM, when flow density is 8.




(a) (b)

Fig. 10. (a) Estimation error of LogA when flow density is 8. (b) System throughput of NoPSM achieved with *practical* estimation of i-vectors, compared to that achieved with *idealized* estimation of i-vectors, when synchronization error is fixed to 50 $\mu$s, and $psize$ and $bsize$ is fixed to 48 and 32 respectively.

LogA and RID firstly, particularly the *detection rate* of interferers and *estimation accuracy* of link quality under corresponding interference for LogA and RID respectively.

#### 5.2.1 Comparison on detection rate between LogA and RID

We define the *detection rate* as the ratio of the number of successfully estimated i-vectors to the total number of i-vectors that should be estimated. If the detection rate is 100 percent, it means that it detects all the one-hop interferers of a link that has ever transmitted packets concurrently with it. We compare detection rate of LogA and RID with different number of one-hop flows. For each flow density, we run simulations with 20 randomly generated topologies.

Fig. 9.(a) shows the average detection rate of both schemes. We can observe that the detection rate of LogA is very high, which is above $80\%$ for all considered flow density. Moreover, LogA has much higher detection rate than RID especially when flow density is high. This is because RID in CMAP+ can only detect those interferers that are within the communication range of both the sender and the receiver. However, in many cases, the interferer that is one-hop neighbor of the receiver may not be within the transmission range of the sender. For LogA, the cause of missing detection is mainly due to the loss of time log. However, repetitive transmission of time logs in $N_{tl}$ broadcasting periods with dedicated control packets, as described in Subsection 4.1.2, can effectively reduces the chance of the loss of time logs.

The relatively higher detection rate of NoPSM enables it to infer interference patterns more completely. Along with interference detection in each round of time log analysis, PRRs of the link under corresponding interference patterns are deduced from statistically analyzing the bitmap recorded by the receiver side of the link. The inferred interference patterns and corresponding PRRs in each round of time log analysis comprise the interference relationships. To evaluate the convergence of the inference process, we trace the i-vectors constructed in each round of time log analysis. For each i-vector, we calculate the absolute difference of PRR ($\Delta_{prr}$) between the current round and the previous round. The empirical cumulative distribution function (CDF) of $\Delta_{prr}$ when each round of time log analysis completes is shown in Fig. 9.(b). Due to space limitation, here we only show the results when flow density is 8. We can see that after 4 rounds of time log analysis, 92% of i-vectors have small changes of PRR (i.e. $\Delta_{prr} \leq 0.05$), which are seen converged to a steady value of PRR. The rest of i-vectors may be randomly influenced by the undetected interferers, and can fluctuate for a while.

#### 5.2.2 Effect of time synchronization error on estimation accuracy of LogA

Since LogA requires time synchronization of nodes in WSNs, we further evaluate the *estimation accuracy* of LogA in the same scenario as above experiments with the flow density fixed to 8. To gain the ground truth, we first remove the effect of clock drift on simulation, and control all combinations of 2∼4 nodes to transmit packets concurrently and keep other senders silent. The measured PRRs in each receiver under the controlled interference are recorded as the benchmark. Then we restart experiments to construct i-vectors with randomly activated links, with the maximum synchronization error varied from 10 $\mu$s to 10 ms, and compare the estimated PRRs in i-vectors with the benchmark. The absolute estimation error is calculated to indicate estimation accuracy of LogA. For each level of synchronization error, we evaluate estimation accuracy of LogA in 20 randomly generated topologies.

Fig. 10.(a) shows the estimation error with different block size and packet payload size. We can see that LogA achieve quite accurate estimation of PRR under different synchronization errors. In most cases, the estimation errors are around 0.15. Hence, LogA is tolerant to synchronization error. This can be because LogA infers interference relationships through analyzing overlapping periods of time logs and estimating corresponding PRR based on bitmaps in bACK. Synchronization error below 5 ms only cause the overlapping periods of time logs to drift 1 to 2 packets, which will not cause so much effect on the interference patterns. When the synchronization error is increased to 10 ms, the estimation error increases, especially in scenarios with small block size and small packet payload size (e.g. $bsize$ is 16). So in cases with large synchronization error, it is better to transmit packets with longer length or blocks with bigger size. With bigger block size (i.e. $bsize$ is 32) and longer packet payload size (i.e. $psize$ is 96), the estimation error still remains small when the synchronization error is increased to 10 ms. Respecting that the state-of-arts time synchronization protocols for WSNs can achieve error below 10 ms with very small cost [37], [38], so we are confident to say that the performance of NoPSM can be assured with currently available time synchronization protocols.

#### 5.2.3 Effect of estimation accuracy of LogA on performance of NoPSM

With the results of previous experiments, we take a further step to evaluate the effect of estimation accuracy of LogA on performance of NoPSM in terms of *system throughput*. The experimental setting in this subsection is the same as in Subsection 5.2.2, except that the maximum synchronization error is fixed to 50

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2016.2547867, IEEE Transactions on Mobile Computing



$\mu$s, and $psize$ and $bsize$ is fixed to 48 and 32 respectively. We run two groups of experiments. First, we construct i-vector tables for each receiver node through controlled transmission as described in the previous subsection. The constructed i-vector tables are taken as ground truth, and are orderly distributed to their one-hop neighbors respectively. The time logs and i-vectors are broadcasted without effect. The system throughput achieved by NoPSM with different number of flows (from 2 to 8) is evaluated as the idealized performance, which is gained with reliably constructed interference relationships. Second, we restart experiments with time log and i-vector broadcasting periods taking effect to construct i-vectors with randomly activated links, and compared the achieved system throughput with the former group of experiments.

Fig. 10.(b) shows the gap between the practical performance and the idealized performance of NoPSM. We can see that as the number of flows increases, the gap becomes larger, which is mainly caused by reduced detection rate of LogA as shown in Fig. 9. However, even with 100% of detection rate, it is unavoidable that time logs will be corrupted during broadcasting, mainly due to the increasingly severe hidden terminal problem with more flows. As a result, the lost time logs have adverse effects on accuracy of i-vector estimation, which further affects system throughput with NoPSM. However, repetitive transmission of time logs in $N_{tl}$ broadcasting periods with dedicated control packets (Subsection 4.1.2) can effectively improve reliability of time log broadcasting. Moreover, selective retransmissions of corrupted packets (Subsection 4.1.1) and adaptive contention window (Subsection 4.4.1) can partially mitigate the effect of inaccurate estimation of i-vectors on system throughput. So, over all, the system throughput for practical NoPSM is decreased by up to 10%, compared to the idealized NoPSM.

### 5.3 Performance of NoPSM with Different Flow Density

In this subsection, we compare NoPSM with CSMA and CMAP+ in terms of delivery ratio, system throughput, latency, and energy efficiency in networks with different flow density, i.e. the number of flows. The packet payload size ($psize$) of these protocols is all set to 48 bytes and the block size ($bsize$) of NoPSM is set to 64 packets. For each flow density, we generate 20 random topologies and run experiments with the three protocols once in each topology, and compute the average of 20 times of running as the result.

**Delivery ratio and system throughput.** Fig. 11 and Fig. 12 compare the delivery ratio and the system throughput of the three protocols with different flow density. We can see that in all scenarios with different number of flows, NoPSM has delivery ratio of 8∼22% higher than CSMA and 2 ∼11% higher than CMAP+. Its throughput is $30\% \sim 60\%$ higher than CSMA and $11\% \sim 30\%$ than CMAP+. The outperformed delivery ratio of NoPSM is mainly attributed to the selective retransmission mechanism and effective transmission control mechanism. The throughput gain of NoPSM over CSMA is mainly achieved from higher usage of channel by exploiting potential opportunities of transmission concurrency. Besides that, it can also result from reduced transmission time through block data transmission, because transmitting packets in a block saves a lot of time spent in making CCA and backoff for each packet. The throughput gain of NoPSM over CMAP+ is mainly achieved from more chances of concurrent transmissions with effective throughput improvement have been exploited.

With increasing number of flows, delivery ratios of all protocols decrease, while system throughputs of all protocols

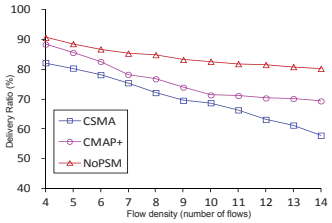



Fig. 11. Delivery ratio with different flow density when $psize$ of these three protocols is 48 and $bsize$ of CMAP+ and NoPSM is 64.

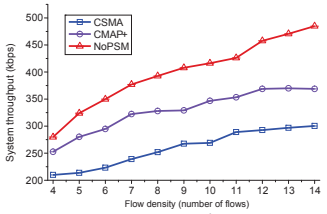



Fig. 12. System throughput with different flow density when $psize$ of these three protocols is 48 and $bsize$ of CMAP+ and NoPSM is 64.

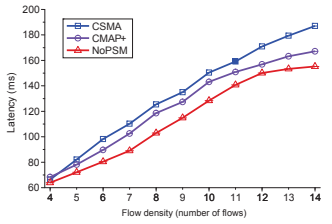



Fig. 13. Delivery latency with different flow density when $psize$ of these three protocols is 48 and $bsize$ of CMAP+ and NoPSM is 64.

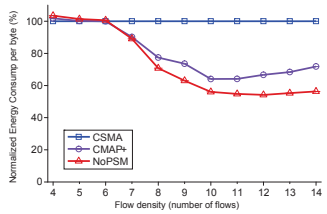



Fig. 14. Ratio of energy consumption per byte to CSMA with different flow density when $psize$ of these three protocols is 48 and $bsize$ of CMAP+ and NoPSM is 64.

increase. This means although each flow has reduced number of successfully received packets on the average, the total number of successfully delivered packets for all the flows still grows. Furthermore, when the flow number is larger than 11, the throughput of CSMA and CMAP+ both stops increasing, while NoPSM still has potential of throughput increase. This is because for CSMA channel usage has been saturated when the flow number is 11. For CMAP+, it has a detection rate less than 35% when the number of flows is larger than 8 as shown in Fig. 9. In other words, the number of practically existing concurrent senders is likely to be underestimated by CMAP+ in networks with dense flows. So in scenarios with dense flows senders with CMAP+ tend to transmit aggressively, which will lead to severe increasing number of corrupted packets. As a result, the throughput of all concurrent transmissions is reduced. However, the detection rate and estimation accuracy of NoPSM are still high even when flow density is increased to 11, so the senders can make correct transmission decisions and ensure effective block data transmissions in most cases. Hence, NoPSM achieves a steady increase of throughput as the number of flows increases.

**Delivery latency.** Fig. 13 shows the delivery latency of the three protocols with different flow density. From this figure, we can see that CSMA, CMAP+ and NoPSM almost have the same latency when the flow density is low, and they all have increasing latency as flows become denser. For all these protocols, the increased latency mainly results from longer time needed to wait for getting access to the channel for each each when the number of flows increases. However, the latency of NoPSM is always lower than that of CSMA and CMPA+. For example, when there are 8 active flows simultaneously existing in the network, the average delivery latency of CSMA and CMAP+ is 125.4 ms and 118.6 ms respectively, while that of NoPSM is 102.9 ms. So in this scenario NoPSM reduces the latency by $13\%$ as compared to CMAP+ and by $17\%$ as compared to CSMA. Moreover, we can also see that the difference of latency achieved by CSMA, CMAP+ and NoPSM becomes larger when

the flow density is more than 11, which can be for the same reason of throughput decrease of CSMA and CMAP+ in these cases, as discussed in the previous paragraph.

**Energy consumption.** Fig. 14 gives the ratio of energy consumption per byte of CMAP+ and NoPSM to that of CSMA. We can see that when the flow density is less than 6, the three protocols lead to almost the same energy consumption. As the number of flows increases, the energy consumption of CMAP+ and NoPSM decreases. Moreover, the energy consumption of NoPSM is less than that of CMAP+. The higher energy consumption of CMAP+ mainly results from more energy wasted in unsuccessful packet transmission, because of the declined rate of interferer detection with increasing number of flows as shown in Fig. 9. For the same reason, especially when the flow density is larger than 11, the energy reduction of NoPSM from CMAP+ becomes wider. On average, NoPSM and CMAP+ reduce the energy consumption by $37\%$ and $28.5\%$ respectively as compared to CSMA, so NoPSM reduce energy consumption by $8.5\%$ as compared to CMAP+.

### 5.4 Performance of NoPSM with Different Block Size and Packet Payload Size

In this subsection, we compare the effects of block size and packet payload size on the performance of CSMA, CMAP+, and NoPSM in scenarios with fixed number of flows to 12 and varied block size from 16 to 64. The settings of transmit power and network size are the same as the previous experiments. For each block size, we first run experiments with packet payload size of 48 bytes, and then with packet payload size of 72 bytes. Setting with a block size and a payload size, we run each experiment with 20 random topologies, and make an average of the delivery ratio, system throughput, delivery latency, and energy consumption per byte respectively as a result shown in Fig. 16 to Fig. 18 respectively. From these figures, we can see that the performance of CSMA does not vary with the setting of block size, because $bsize$ of CSMA is constantly 1, as described in Subsection 5.1.

**Delivery ratio and system throughput.** Fig. 15 and Fig. 16 compare the delivery ratio and the system throughput of the three protocols with increase of block size. From these two figures, we can see that when packet payload size is either 48 or 72 bytes, NoPSM always outperforms CSMA and CMAP+ in terms of both delivery ratio and system throughput. In particular, we can see that NoPSM can keep relatively high delivery ratio (i.e. about 81%) with increase of block size, so larger payload size can always improve throughput of NoPSM. However, the increase of block size can not necessarily improve throughput of CMAP+, because its delivery ratio can decline with increase of block size, as shown in Fig. 15. This can be caused by the following factors. Firstly, larger block size will increase the number of corrupted packets during block transmission for CMAP+, due to its low detection rate when the flow density is larger than 8 as shown in Fig. 9. Secondly, for senders with CMAP+, they adopt the binary criterion to determine transmission concurrency without considering throughput gain comprehensively, which can bring in throughput reduction with deteriorated detection capability of CMAP+ in cases of larger block size.

More specifically, NoPSM can have up to 18% and 11% higher delivery ratio than CSMA and CMAP+ respectively, and the throughput gain of NoPSM over CSMA can be up to 60% when the block size increases to 64. Also, NoPSM achieves about $19\% \sim 49\%$ throughput gain over CMAP+ when the payload size is 48 bytes, and about $22\% \sim 55\%$ when the payload size is 72 bytes.

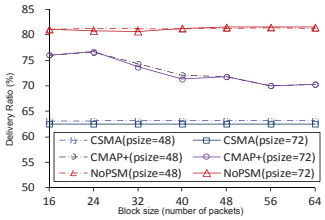



Fig. 15. Delivery ratio with different block size when the flow density is 12.

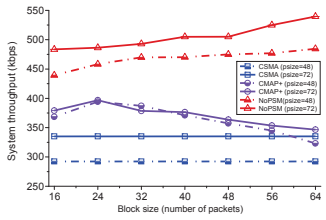



Fig. 16. System throughput with different block size when the flow density is 12.

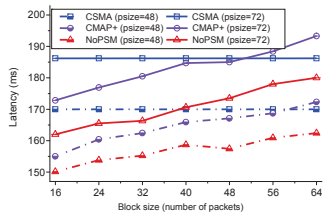



Fig. 17. Delivery latency with different block size when the flow density is 12.

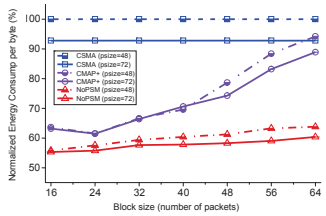



Fig. 18. Ratio of energy consumption per byte to CSMA ($psize = 48$) with different block size when the flow density is 12.

**Delivery latency.** Fig. 17 presents the delivery latency of the three protocols with increase of block sizes. From this figure, we can see that both NoPSM and CMAP+ have increasing delivery latency with increase of block size, because the duration of occupancy for each block increases. However, NoPSM is always lower than CSMA and CMAP+ when the payload size is either 48 or 72 bytes. For instance, when the block size is 32 packets and the payload size is 72 bytes, NoPSM reduces the latency by about $10\%$ as compared to CSMA , and by about $7\%$ as compared to CMAP+. Moreover, larger payload size will increase latency for both NoPSM and CMAP+, since the transmission time of each packet will increase. In particular, when the payload size is 72 bytes, the deliver latency of CMAP+ can be larger than that of CSMA in scenarios with block size larger than 56. This can be because worse channel usages than CSMA have been incurred from concurrent transmissions of CMAP+ with deteriorated detection capability in these cases.

**Energy consumption.** Fig. 18 plots the ratio of energy consumption per byte of the three protocols to that of CSMA with different block sizes. Here, we take the energy consumption per byte of CSMA when payload size is 48 bytes as the baseline. We can see that NoPSM consumes less energy than CSMA and CMAP+, especially when the block size is larger than 32 packets. On average, compared with CMAP+, NoPSM improves energy efficiency by about 18% when the payload size is 48 bytes, and by about 20% when the payload size is 72 bytes. In addition, we can see that the energy consumption of NoPSM increases slowly with increase of block size, but that of CMAP+ rises quickly from the case with packet size of 32 packets, especially when the payload size is 48 bytes. Referring to Fig. 16, we can see that the system throughput of CMAP+ starts to decline when the block size is 32 packets, which means more failed concurrent transmissions occur when the block size is larger than 32 packets. On the contrary, NoPSM keep increasing of throughput in these cases. So we can know that more energy are wasted in unsuccessful block data transmission for CMAP+ compared with NoPSM, when block size increases.

In short, the above experimental results show that NoPSM

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2016.2547867, IEEE Transactions on Mobile Computing



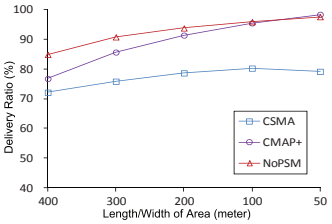



Fig. 19. Delivery ratio with different network area when the flow density is 8, $psize$ is 48 and $bsize$ is 64.

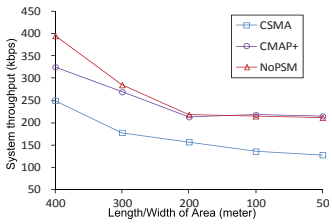



Fig. 20. System throughput with different network area when the flow density is 8, $psize$ is 48 and $bsize$ is 64.

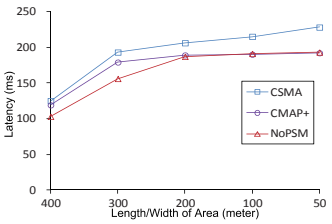



Fig. 21. Delivery latency with different network area when the flow density is 8, $psize$ is 48 and $bsize$ is 64.

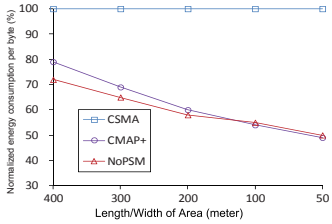



Fig. 22. Ratio of energy consumption per byte to CSMA with different network area when the flow density is 8, $psize$ is 48 and $bsize$ is 64.

can achieve more throughput gain and latency reduction than CMPA+ as flow density and block size increase, and furthermore the throughput gain and latency reduction are not achieved at the cost of more energy consumption than CMAP+.

### 5.5 Performance of NoPSM with Different Network Density

To compare performance of NoPSM with CMAP+ and CSMA in networks of different density, we put 16 nodes uniformly in an area of different length and width, from 400×400 to 50×50 square meter. 8 flows are randomly generated in the same way as described in Subsection 5.1, and the transmit power of each node is set to 0 dBm. The packet payload size ($psize$) of these protocols is all set to 48 bytes, and block size ($bsize$) of NoPSM and CMAP+ is set to 64 packets. For each network area, we generate 20 random topologies and run experiments with the three protocols once in each topology, and compute the average 20 times of running as the result.

**Delivery ratio and system throughput.** Fig. 19 and Fig. 20 compare performance of CSMA, CMAP+ and NoPSM in terms of delivery ratio and system throughput respectively. We can see that denser deployment of nodes can improve delivery ratio for all the evaluated MAC protocols, because the impact of hidden terminal on data delivery becomes weaker with increase of network density. Especially for CMAP+ and NoPSM, when all the nodes located in the same interference area (i.e. length/width of area is about 100 meter), they achieve delivery ratio of more than 90%. However, the system throughput for all the evaluated MAC protocols declines with increase of network density, because all the flows share the same channel to deliver data. The results show that CMAP+ can NoPSM can achieve a maximum of 218 kbps throughput when the length/width of area is less than 100 meter.

**Delivery latency.** Fig. 21 compares the delivery latency of CSMA, CMAP+ and NoPSM with different network density. We can see that CMAP+ and NoPSM have lower latency than CSMA, because they exploit chances of transmission concurrency. Furthermore, the advantage of NoPSM over CMAP+ becomes not so clear when the length/width of area is less than 100 meter, because they almost make the same decisions of concurrent transmission in these scenarios.

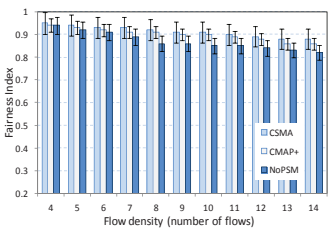



Fig. 23. Fairness index of CSMA, CMAP+ and NoPSM with different flow density.

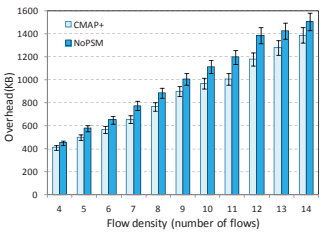



Fig. 24. Messaging overhead of CMAP+ and NoPSM with different flow density.

**Energy consumption.**Fig. 22 shows the ratio of energy consumption per byte of these MAC protocols to that of CSMA with different network density. We can see that NoPSM can outperform CMAP+ in energy efficiency when all the nodes are not located in the same interference area (i.e. length/width of area is more than 200 meter). This mainly results from more energy wasted in unsuccessful packet transmission for CMAP+, because of more false positive decisions of concurrent transmission made by CMAP+ than NoPSM. With increase of network density, their difference in energy efficiency vanishes, because all the nodes located in the same interference area and they can both make accurate decisions of transmission concurrency.

In short, NoPSM has an advantage over CMAP+ in networks with an ordinary node density. With increase of node density, they will have similar performance.

### 5.6 Fairness, Messaging Overhead and Discussion

Although the main goal of NoPSM (i.e. improved system throughput) is achieved as shown above, fairness is a desired property of MAC protocols. We hereby compares the fairness among CSMA, CMAP+ and NoPSM based on experiments with different flow density (see details in Subsection 5.3). The result is shown in Fig 23. The fairness index is defined as $(\sum_f Thr_f)^2/(N \times \sum_f {Thr_f}^2)$ [39], where $Thr_f$ represents a flow $f$'s throughput and $N$ represents the total number of flows. We can see that all these MAC protocols exhibit similar fairness properties when the number of flows increases, because they all suffer from increasing location-dependent contention (e.g. flows in the edge of scenario suffer less contention than that in the center) and hidden terminal problem, and adopt similar random backoff adjustment algorithm. In addition, because NoPSM adjusts a backoff timer based on the estimated PRR of the latest round of block transmission (Subsection 4.4.1), the flows with higher PPR have relatively higher probability to access the channel. So NoPSM has slightly degraded fairness index compared to CSMA and CMAP+, while it is still desirable since it has not yet been specifically designed for fairness. We think that it can be improved by integrating the utility function into backoff scheme as proposed by [40].

In addition, we take a comparative study on the messaging overhead of NoPSM and CMAP+, by statistically counting bytes of broadcasted control messages in experiments with different flow densities (see details in Subsection 5.3). Fig. 24 shows the comparison result. The messaging overhead of NoPSM mainly lies in broadcasting time logs and i-vectors, while that of CMAP+ lies in broadcasting interferer lists. Although the overhead of NoPSM is a little more than CMAP+,

they have the same order of magnitude. Therefore, we can see that the outperformance of NoPSM over CMAP+, which is mainly attributed to the improved detection rate of interferer and refined decision of transmission concurrency, is achieved with equivalently low overhead as CMAP+.

It is worth noting that the messaging overhead of NoPSM includes that for maintaining i-vectors table in soft state. As explained in Subsection 4.3, NoPSM can adapt to slightly varied communication environment (e.g. where all ECG sensors are attached to a single patient who moves alone in a room) in this way. However, NoPSM cannot be directly applied to some situations with higher mobility (e.g. where there are multiple patients moving around and each patient is attached with some ECG sensors), because the interference relationships may become more dynamic and the periodically refreshed i-vectors cannot reflect the current status of communication environment promptly and accurately. We will address the impact of mobility on performance of NoPSM in future.

## 6 Conclusions

In this paper, we present NoPSM, which is a concurrent MAC protocol for data intensive WSNs. NoPSM is featured by its base and criterion to determine transmission concurrency. The base of NoPSM is not proactively constructed PRR-SINR models for each node in network, but reactively constructed interference relationships by passively analyzing overlapping relationships among time logs of block data transmissions and corresponding reception status of each packet in blocks. In this way, the base of the concurrent MAC protocol can be constructed and maintained without network downtime. Based on the constructed interference relationships, nodes make decisions of transmission concurrency with a comprehensive criterion, which not only estimates PRR of any active links after initiating a new link, but also estimates throughput improvement gained from concurrent transmissions. The performance of NoPSM is extensively evaluated in TOSSIM and compare it with CSMA and CMAP+. The experimental result shows that NoPSM outperforms CSMA and CMAP+ in terms of delivery ratio, system throughput, delivery latency, and energy consumption.

## Acknowledgments

Partial work of this paper is supported by the International S&T Cooperation Program of China (ISTCP) under Grant No. 2013DFA10690, the "Strategic Priority Research Program" of the Chinese Academy of Sciences under Grant No. XDA06010403, and the National Science Foundation of China (NSFC) under Grant No. 61100180.

## References

[1] X. Liang and I. Balasingham. Performance analysis of the ieee 802.15.4 based ECG monitoring network. In Proc. IASTED WOC'07, pages 99–104, 2007.

[2] R. Huang, W. Song, M. Xu, N. Peterson, B. Shirazi, and R. LaHusen. Real-world sensor network for long-term volcano monitoring: Design and findings. IEEE Transaction on Parallel and Distributed Systems, 23(2):321–329, 2012.

[3] Z. Wang, Z. Zhao, D. Li, and L. Cui. Data-driven soft sensor modeling for algal blooms monitoring. IEEE Sensors Journal, 15(1):579–590, 2015.

[4] G. Tolle, J. Polastre, R. Szewczyk, D. Culler, et al. A macroscope in the readwoods. In Proc. of ACM Sensys'05, pages 51–63, 2005.

[5] S. Kim, R. Fonseca, P. Dutta, A. Tavakoli, D. Culler, P. Levis, S. Shenker, and I. Stoica. Flush: A reliable bulk transport protocol for multihop wireless networks. In Proc. of ACM SenSys'07, pages 351–365, 2007.

[6] H. Zhang, A. Arora, Y. Choi, and M. Gouda. Reliable bursty convergecast in wireless sensor networks. Computer Communications, 30(13):2560–2576, 2007.

[7] K. Whitehouse, A. Woo, F. Jiang, J. Polastre, and D. Culler. Exploiting the capture effect for collision detection and recovery. In Proc. of IEEE EmNetS-II, pages 45–52, 2005.

[8] J. Lu and K. Whitehouse. Flash flooding: Exploiting the capture effect for rapid flooding in wireless sensor networks. In Proc. of IEEE INFOCOM'09, pages 2491–2499, 2009.

[9] B. Dezfouli, M. Radi, K. Whitehouse, S.A. Razak, and T. Hwee-Pink. CAMA: Efficient modeling of the capture effect for low-power wireless networks. ACM Transactions on Sensor Networks, 11(1):20:1–43, 2014.

[10] J. Manweiler, N. Santhapuri, S. Sen, R.R. Choudhury, S. Nelakuditi, and K. Munagala. Order matters: transmission reordering in wireless networks. In Proc. of ACM Mobicom'09, pages 61–72, 2009.

[11] M. Sha, G. Xing, G. Zhou, S. Liu, and X. Wang. C-mac: Model-driven concurrent medium access control for wireless sensor networks. In Proc. of IEEE INFOCOM'09, pages 1845–1853, 2009.

[12] Q. Ma, K. Liu, X. Miao, and Y. Liu. Opportunistic concurrency: A mac protocol for wireless sensor networks. In Proc. of IEEE DCOSS'11, pages 1–8, 2011.

[13] D. Son, B. Krishnamachari, and J. Heidemann. Experimental study of concurrent transmission in wireless sensor networks. In Proc. of ACM Sensys'06, pages 237–250, 2006.

[14] R. Maheshwari, S. Jain, and S.R. Das. A measurement study of interference modeling and scheduling in low-power wireless networks. In Proc. of ACM Sensys'08, pages 141–154, 2008.

[15] Texas Instruments. Cc2420 datasheet. http://www.ti.com/lit/ds/symlink/cc2420.pdf, 2007.

[16] J. Huang, S. Liu, G. Xing, H. Zhang, J. Wang, and L. Huang. Accuracy-aware interference modeling and measurement in wireless sensor networks. In Proc. of IEEE ICDCS'11, pages 172–181, 2011.

[17] R.K. Ganti, P. Jayachandran, H. Luo, and T.F. Abdelzaher. Datalink streaming in wireless sensor networks. In Proc. of ACM SenSys'06, pages 209–222, 2006.

[18] M. Vutukuru, K. Jamieson, and H. Balakrishnan. Harnessing exposed terminals in wireless networks. In Proc. of USENIX NSDI'08, pages 59–72, 2008.

[19] HART Communication Foundation. Wirelesshart technical notes. http://en.hartcomm.org.

[20] F. Yu, T. Wu, and S. Biswas. Toward in-band self-organization in energy-efficient mac protocols for sensor networks. IEEE Trans. Mobile Computing, (2):156–170, 2008.

[21] W.-Z. Song, R. Huang, B. Shirazi, and R. LaHusen. Treemac: Localized tdma mac protocol for real-time high-data-rate sensor networks. In Proc. of IEEE PerCom'09, pages 1–10, 2009.

[22] K.K. Chintalapudi. I-mac - a mac that learns. In Proc. ACM IPSN'10, pages 315–316, 2010.

[23] I. Rhee, A. Warrier, M. Aia, and J. Min. Z-MAC: a hybrid mac for wireless sensor networks. In Proc. of ACM SenSys'05, pages 90–101, 2005.

[24] W. Ye, J. Heidemann, and D. Estrin. Medium access control with coordinated, adaptive sleeping for wireless sensor networks. IEEE/ACM Trans. on Networking, 12(3):12–25, 2004.

[25] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In Proc. of ACM SenSys'04, pages 95–107, 2004.

[26] M. Buettner, G.V. Yee, E. Anderson, and R. Han. X-MAC: A short preamble mac protocol for duty-cycled wireless sensor networks. In Proc. of ACM SenSys'06, pages 307–320, 2006.

[27] Y. Sun, O. Gurewitz, and D. Johnson. RI-MAC: A receiver-initiated asynchronous duty cycle mac protocol for dynamic traffic loads in wireless sensor networks. In Proc. of ACM SenSys'08, pages 1–14, 2008.

[28] M. Zimmerling, F. Ferrari, L. Mottolay, T. Voigty, and L. Thiele. pTunes: Runtime parameter adaptation for low-power mac protocols. In Proc. of ACM/IEEE IPSN'12, pages 173–184, 2012.

[29] Y. Wu, J. A. Stankovic, T. He, and S. Lin. Realistic and efficient multi-channel communications in wireless sensor networks. In Proc. of IEEE INFOCOM'08, pages 1193–1201, 2008.

[30] B. Raman, K. Chebrolu, S. Bijwe, and V. Gabale. Pip: A connection-oriented, multi-hop, multi-channel tdma-based mac for high throughput bulk transfer. In Proc. of ACM SenSys'10, pages 15–28, 2010.

[31] Y. Sun, S. Du, O. Gurewitz, and D.B. Johnson. DW-MAC: A low latency, energy efficient demand-wakeup mac protocol for wireless sensor networks. In Proc. of ACM MobiHoc'08, pages 53–62, 2008.
[32] S. Liu, G. Xing, H. Zhang, J. Wang, J. Huang, M. Sha, and L. Huang. Passive interference measurement in wireless sensor networks. In Proc. of IEEE ICNP'10, pages 52–61, 2010.
[33] K.K. Chintalapudi and L. Venkatraman. On the design of mac protocols for low-latency hard real-time discrete control applications over 802.15.4 hardware. In Proc. of ACM IPSN'08, pages 356–367, 2008.
[34] D.M. Blough, C. Canali, G. Resta, and P. Santi. On the impact of far-away interference on evaluations of wireless multihop networks. In Proc. of ACM MSWiM'09, pages 90–95, 2009.
[35] B. Dezfoulia, M. Radia, S.A. Razaka, T. Hwee-Pinkb, and K.A. Bakara. Modeling low-power wireless communications. Journal of Network and Computer Application, 51:102–126, May 2015.
[36] H.J. Lee, A. Cerpa, and P. Levis. Improving wireless simulation through noise modeling. In Proc. of ACM/IEEE IPSN'07, pages 21–30, 2007.
[37] Z. Li, W. Chen, C. Li, M. Li, X.-Y. Li, and Y. Liu. Flight: Clock calibration and context recognition using fluorescent lighting. IEEE Transactions on Mobile Computing, 13(7):1495–1508, Jul. 2014.
[38] L. Li, L. Sun, G. Xing, W. Huangfu, R. Zhou, and H. Zhu. Rocs: Exploiting fm radio data system for clock calibration in sensor networks. IEEE Transactions on Mobile Computing, 14(10):2130–2144, Oct. 2015.
[39] X. Yang, N.H. Vaidya, and P. Ravichandran. Split-channel pipelined packet scheduling for wireless networks. IEEE Transactions on Mobile Computing, 5(3):240–257, 2006.
[40] A. Aziz, J. Herzen, R. Merz, S. Shneer, and P. Thiran. Enhance & Explore: An adaptive algorithm to maximize the utility of wireless networks. In Proc. of ACM MobiCom'11, pages 157–188, 2011.

**Haiming Chen** received the BEng and MEng degree in computer engineering from Tianjin University, China, in 2003 and 2006 and the PhD degree in computer science from the Institute of Computing Technology, Chinese Academy of Sciences in 2010. He is now an assistant professor at the Institute of Computing Technology, Chinese Academy of Sciences, where he received the Excellent Researcher award in 2011. He has published over 30 papers in areas of wireless, ad hoc, sensor networks and networked embedded computing systems. He has served as TPC member for several international conferences, such as IEEE ICPADS 2014-2015, and reviewers for several international journals, such as IEEE Internet of Things Journal, Computer Networks, and Ad Hoc Networks. He won the third prize of Beijing Municipal Science and Technology Awards in 2013. One of his co-authored paper was elected as the Best Paper of CCF CWSN 2015. He is a member of IEEE, ACM and CCF (China Computer Federation).

**Zhaoliang Zhang** received the BS degree in computer science from NanChang University, China, in 2006, and the MS degree in computer science from Soochow University, China, in 2009. He received the PhD degree in computer engineering from the Institute of Computing Technology, Chinese Academy of Sciences in 2013. He is now a project manager in China National Electronics Imp.& Exp. Corp.(CEIEC). His research interests include error-resilient communication in wireless sensor networks.

**Li Cui** received her undergraduate degree from Tsinghua University, China, in 1985 and the MSc degree from the Institute of Semiconductors, Chinese Academy of Sciences, in 1988. She received the Ph.D. Degree from the University of Glasgow, UK, in 1999. She was a visiting scholar in the Department of Chemistry at the University of Toronto, Canada, in 1994. In 1995, she worked as a Royal Society Fellow and, from 1996 to 2003, a Research Associate in the Department of Electronics and Electrical Engineering at the University of Glasgow. She received "The 100 Talents Program of CAS" award in 2004 from CAS and is currently a Professor and the leader of the wireless sensor network laboratory at the Institute of Computing Technology, Chinese Academy of Sciences. Her current research is focused on sensor technology and wireless sensor networks. She is an IET fellow (January 2016∼) and a member of IEEE.

**Changcheng Huang** received his B. Eng. in 1985 and M. Eng. in 1988 both in Electronic Engineering from Tsinghua University, Beijing, China. He received a Ph.D. degree in Electrical Engineering from Carleton University, Ottawa, Canada in 1997. From 1996 to 1998, he worked for Nortel Networks, Ottawa, Canada where he was a systems engineering specialist. He was a systems engineer and network architect in the Optical Networking Group of Tellabs, Illinois, USA during the period of 1998 to 2000. Since July 2000, he has been with the Department of Systems and Computer Engineering at Carleton University, Ottawa, Canada where he is currently an associate professor. He won the CFI new opportunity award for building an optical network laboratory in 2001. He was an associate editor of IEEE Communications Letters from 2004 to 2006. He is a senior member of IEEE.