Throughput Assurance for Multiple Body Sensor Networks

Zhen Ren, Xin Qi, Gang Zhou, *Senior Member, IEEE,*
Haining Wang, *Senior Member, IEEE,* and David T. Nguyen

Abstract—Existing research has demonstrated that inter-body sensor network (inter-BSN) information sharing among coexisting BSNs can enhance applications' performance and save energy. However, how to achieve effective inter-BSN information sharing through wireless communication is a challenging task. On one hand, a BSN should be able to discover neighboring BSNs and establish inter-BSN links with Quality of Service (QoS) assurances. On the other hand, a BSN should be able to prevent the QoS of intra- and inter-BSN links from being degraded by multiple BSNs' mutual interference. In this paper, we propose BuddyQoS, a framework that provides network throughput assurances for coexisting and shared buddy BSNs. In particular, BuddyQoS accurately estimates and adaptively schedules wireless resources to meet the throughput requirements of all inter- and intra-BSN links. Our trace-driven experiment results demonstrate that BuddyQoS outperforms the default CSMA solution in the standard TinyOS-2.x releases in terms of providing throughput assurances.

Index Terms—Body Sensor Network, Protocol Design, Quality of Service, Resource Management

1 Introduction

A Body Sensor Network (BSN) consists of a group of wireless on-body sensor nodes, each of which is equipped with a set of low-power sensors. Data collected from sensors is transmitted to an aggregator (e.g., a PC or a smartphone) and then is either analyzed by the aggregator or reliably delivered to a data center (e.g., a hospital) for future analysis. Being portable, BSN enables a wide range of human-centric applications, including activity recognition [1], smart healthcare [2], assisted living [3], athletic performance evaluation [4], and interactive controls [5].

It has been reported that group activities take a large part of human daily activities [6]. This fact of human behavior indicates a large chance of the BSNs that support human-centric applications coexisting within the communication range of each other. For example, a BSN that supports athletic performance evaluation application always has neighboring BSNs, because the athletes usually train and live together [4]. Also, a BSN that supports smart healthcare and assisted living applications always coexists with other BSNs, because it is not uncommon for old people to live as a group in a retirement community [2][3].

Existing research has demonstrated that when multiple BSNs coexist, inter-BSN information sharing results in both application performance enhancement and energy saving. For example, CoMon [6] demonstrates that sensors in a BSN can be shared with other BSNs and redundant sensors can be turned off to save energy. Moreover, inter-BSN sensing data sharing not only reduces energy overhead, but also enhances application performance such as activity recognition accuracy [7].

In a similar fashion to stand-alone BSNs with human-centric applications performing real-time monitoring, the requirements of stringent network throughput guarantees are also placed on the BSNs that share sensing data. For sensing data that is not shared, the same throughput requirements still hold to ensure application fidelities. For example, the EEG headset used by NeuroPhone [5] generates data at the rate of 64Kbps. Such data needs to be delivered from the wireless EEG headset to its corresponding aggregator with throughput of 64Kbps, so that the online facial expression and neuro-activity can be captured in a timely manner. For shared sensing data, same data is required by more than one BSNs with same throughput requirements, but different communication patterns need to be taken into consideration when the data is delivered.

Considering the aforementioned benefits and stringent throughput requirements, two research questions exist for sharing information among the BSNs participating in the sensors sharing framework (coexisting BSNs). First, how to *accurately* estimate wireless resources to decide whether the throughput requirement of a link can be guaranteed or not. By answering this question, we can set rules to determine whether a link should be established or not. Second, how to *adaptively* allocate wireless resources so as to meet throughput requirements for both inter- and intra-BSN links in the presence of mutual interference from coexisting BSNs. By answering this question, we can provide throughput assurances for the upper applications.

In this paper, we propose BuddyQoS, a framework that provides network throughput assurances among coexisting and shared buddy BSNs. We define a ***buddy BSN*** as the BSN worn by a family member, a friend, or a colleague, who can be

- *Zhen Ren is with Synopsys, Inc., USA.*
  E-mail: zhen.ren@synopsys.com
- *Xin Qi, Gang Zhou, and David T. Nguyen are with the College of William and Mary, USA.*
  E-mail: {xqi, gzhou, dnguyen}@cs.wm.edu
- *Haining Wang is with the University of Delaware, USA.*
  E-mail: hnw@udel.edu

trusted. In this way, we assume buddy BSNs can be trusted and only focus on how to provide network throughput assurances for them. Note that existing privacy research [8] can be also integrated into our solution to protect human privacy if needed.

Some existing efforts [9] [10] have been done for providing user-requested communication QoS in a single BSN. However, how to guarantee communication QoS for multiple BSNs has not yet been studied. Some works, such as [11], support information sharing among sensor networks. Other works, such as BikeNet [12], Bubble-sensing [13], and CaliBree [14], propose new applications that leverage people rendezvous [15] and exploit data collected from different people's devices. However, none of them provides communication QoS assurances for information sharing. Human mobility has also been modeled for analyzing the inter-contact time of different individuals [16] [17] [18] [19] [20]. These efforts aim to predict when devices carried by different people will be in the range of each other for communication. However, they assume perfect communication among the devices without considering wireless interference. Thus, they cannot provide any communication QoS guarantee.

The main contributions of BuddyQoS are summarized as follows:

- As far as we know, BuddyQoS is the first framework that guarantees network throughput among coexisting and shared buddy BSNs, even though some QoS studies exist for individual BSNs in literature.
- BuddyQoS enables neighboring buddy BSNs to discover each other and share sensing data by establishing inter-BSN links. It also adaptively schedules available wireless resources to meet the throughput requirements from the upper applications for both intra- and inter-BSN data communication.
- BuddyQoS is implemented in TinyOS-2.x with nesC. Trace data from TelosB sensor motes is collected and input to TOSSIM for simulation. Our performance evaluation demonstrates that BuddyQoS notably outperforms the default CSMA solution in standard TinyOS-2.x release.

The remainder of the paper is organized as follows. Section 2 reviews related work. Section 3 presents the overview of BuddyQoS, with its individual modules further explained in later Sections: the Hybrid MAC in Section 4, the Admission Controller and the Resource Scheduler in Section 5, and the Buddy Management in Section 6. Section 7 evaluates the performance of BuddyQoS, and finally Section 9 concludes the paper.

2 Related Work

Body sensor networks can enable novel applications. For instance, the authors in [21] explored ways to efficiently support social sensing applications. The work saves sensing power by offloading sensing tasks to nearby fixed sensors. Other researchers [22] attempted to apply BSNs in the field of mobile cloud computing. Their effort saves smartphone energy by sharing neighboring phone data via backend servers. BSNs promise novel uses in healthcare, fitness, and entertainment [23], but research must address various obstacles. In literature, some efforts have been proposed to ensure communication QoS for an isolated BSN. For example, BodyQoS [9] is designed to ensure user-requested throughput for a BSN, while BodyT2 [10] is proposed to ensure both throughput and time delay requirements for a BSN.

Some works are proposed for multiple concurrent applications to share sensors within a sensor network. For example, MetroSense [5] attempts to support multiple concurrent applications to share distributed sensors in urban settings. In contrast to MetroSense that provides sensor sharing within a very large scale sensor network, BuddyQoS supports both intra- and inter-BSN sensor sharing in coexisting and shared buddy BSNs with much smaller scales.

Other works leverage people rendezvous [15] and enforce mobile applications to share data across mobile devices carried by different people. For example, BikeNet [12] is built for a cyclist community, within which cycling-related data is collected to evaluate cyclists' performance and environment. Bubble-sensing [13] is an approach that relies on people rendezvous to distribute sensing tasks among sensors worn by different people and deliver the required data back to the task initiator. CaliBree [14] is a distributed self-calibration approach for sensing devices. A sensing device increases its sensing accuracy by collecting the relative miscalibrations from other nearby sensing devices during opportunistic device rendezvous. In contrast to BuddyQoS, none of these works considers providing communication QoS for information sharing.

There are also existing efforts that model human mobility. For example, Srinivasa et al. proposed CREST [19] to estimate the remaining time for the next people rendezvous using conditional residual time. Rhee et al. proposed truncated Levy walk (TLW) [16] and Lee et al. proposed Self-similar Least Action Walk (SLAW) [17] to produce synthetic walk traces based on human mobility features. With the human mobility models, people rendezvous becomes predictable. All mobile systems sharing information with others can leverage these existing models to enhance system performance. However, these works assume that data can be forwarded successfully once the sensor nodes are close to each other. In this paper, we focus on the communication quality issue, which is not addressed by these previous works.

With a solid number of platforms for sharing sensing data in wireless BSNs, a few researchers turn their attention to the QoS of such networks. In particular, they consider inter-network interference and search for ways to mitigate such interference. Rout et al. [24] indicated the necessity for interference immunity in health monitoring, and introduced an approach that mitigated interference using modified and modulated hermit pulses. Others focus their efforts on ultra-wideband interference mitigation [25][26][27][28]. An interesting Clique-based scheduling algorithm is introduced in [29], where the sensors are clustered into different groups to avoid interference. Zhang et al. [30] presented an intriguing method, which suggested to consider the social nature of Wireless Body Area Networks (WBANs). In particular, they proposed to mitigate the communication interference based

on social interactions of the subjects carrying WBANs. In contrast, we propose a framework that guarantees network throughput among coexisting and shared buddy BSNs. Our solution enables neighboring buddy BSNs to discover each other and share sensing data by establishing inter-BSN links.

Finally, the IEEE 802.15 Task Group 6 has been developing a communication standard named IEEE 802.15.6 to serve a variety of Body Area Network applications. However, the proposal has been long circulating (79 circulations as of now) with mixed approval rates [31]. Additionally, as indicated in several studies [32][33], the technology has a limited support of medical systems, such as patient monitoring systems that require reliable throughput assurance for monitoring patient life functions. Timmons et al. [34] also demonstrated undesirably high power consumption of IEEE 802.15.6 (25.6% - 33.2% higher than their baseline framework). IEEE 802.15.6 adopts the Impulse Radio Ultra-Wideband to physically support the coexistence of multiple BSNs [35]. We believe that once IEEE 802.15.6 becomes a mature standard widely accepted across academia and industry, it will be interesting to develop solutions with flavors of IEEE 802.15.6. We reserve that therefore for our future work.

3 BuddyQoS Overview

In this paper, we propose BuddyQoS, a QoS solution providing throughput assurances for coexisting and shared buddy BSNs. In this section, we present BuddyQoS's components in the top-down order.

BuddyQoS performs sensor sharing coordination to support sensor sharing across coexisting BSNs. Applications in a BSN decide whether to share sensors with or request sensors from neighboring buddy BSNs. When BuddyQoS notifies existing applications of a newly detected neighboring BSN, the applications estimate whether they can benefit or benefit from this new BSN through sharing sensors. Existing methods (such as [6]) of benefit-cost analysis for sensor sharing can be adopted by applications to make such estimations. If the applications decide to share sensors with or use sensors from the new BSN, they send the sharing decisions or requests to the BuddyQoS. BuddyQoS establishes wireless connections with the new BSN for positive sensor sharing decisions and approved sensor sharing requests.

After sensor sharing coordination, applications begin to generate QoS requests with throughput requirements on the usable sensor nodes in local and neighboring buddy BSNs. They assign each QoS request a global priority, which reflects how important the request is. When the available wireless resource is not enough to satisfy all the QoS requests, BuddyQoS uses the priorities to decide which requests are less important and rejects them.

Figure 1 shows the architecture of BuddyQoS, which consists of four main components: Buddy Management, Admission Controller, Resource Scheduler, and Hybrid MAC. Additionally, there are three flows passing through the components. They are application data flow, local BSN management flow and buddy BSN management flow.

The Buddy Management on aggregator handles management messages from neighboring buddy BSNs and maintains their information. With the information of neighboring buddy BSNs, it performs neighbor discovery and sensor sharing coordination. Also, it provides resource schedules from neighboring buddy BSNs for the Admission Controller to make admission decisions.

The Admission Controller on aggregator is responsible for making admission decisions for QoS requests with throughput requirements. When applications from local and neighboring buddy BSNs send throughput requirements on local sensor nodes, it estimates the resource needed to ensure the throughput requirements. In addition, it obtains the resource schedules of neighboring buddy BSNs from the Buddy Management, which allows it to further estimate the total resource needed to satisfy the throughput requirements on the sensor nodes of all BSNs. If the available resource is less than the total resource needed, the Admission Controller rejects some low priority QoS requests. Otherwise, all QoS requests are accepted and maintained in a list, which is then outputted to the Resource Scheduler. The admission decisions are returned to the applications, which send out the QoS requests.

The Resource Scheduler on aggregator collaborates with the Slave Resource Scheduler on sensor nodes to schedule resources for intra- and inter-BSN communication. Particularly, the Resource Scheduler receives a list of admitted QoS requests with different throughput requirements on local sensor nodes from the Admission Controller and other BSNs' management information from the Buddy Management. Then, it computes a TDMA schedule for intra- and inter-BSN communication and enforces the schedule on local aggregator. The Slave Resource Scheduler receives the schedule and enforces the schedule on sensor nodes.

The Hybrid MAC sits on both aggregator and sensor nodes and is above the PHY layer. The Hybrid MAC is responsible for transmitting and receiving both data packets and management messages from the upper layers. Motivated by [36], we design the Hybrid MAC to combine the advantages of TDMA, which is good for resource estimation, and CSMA, which is flexible for scheduling.

4 BuddyQoS Hybrid MAC

We design BuddyQoS to adopt a hybrid MAC protocol that combines the advantages of both CSMA and TDMA for media access with the following two considerations. First, wireless resource scheduling is a natural way to handle the intricate intra- and inter-BSN communication. With TDMA, BuddyQoS can make a communication schedule through estimating the wireless resource usage of each QoS requirement. Second, human mobility results in high dynamics in available wireless resources of coexisting BSNs. Thus, a contention scheme of CSMA is desired for each BSN to make full use of available resources. Although some existing works such as Z-MAC [36] and Funneling-MAC [37] already propose the idea of combining CSMA and TDMA, our hybrid MAC protocol is different because it is a natural design choice that specifically fits the scenario of coexisting and shared BSNs, rather than ad-hoc deployment of multi-hop wireless sensor networks.

In the Hybrid MAC, time is divided into slots. We assume that all aggregators and sensor nodes are synchronized,

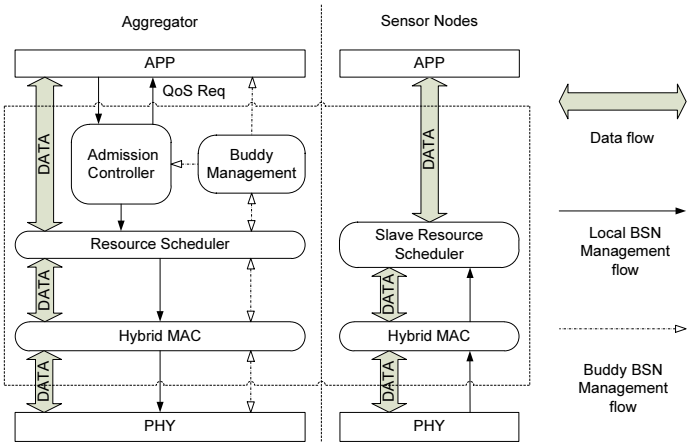


Fig. 1: BuddyQoS Architecture

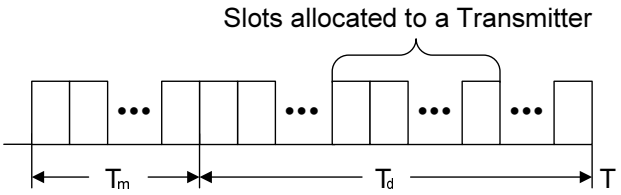


Fig. 2: Interval T Divided into Two Periods

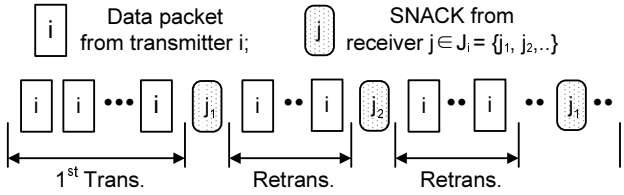


Fig. 3: Transmitter i's Data Transmission with SNACKs.

although our performance evaluation demonstrates that our design is robust to tolerate synchronization errors to some extent. On aggregators and sensor nodes, each packet is sent using a single time slot. When a transmitter has a packet to send, it first backs off some time before sensing the channel. If the channel is clear, the packet is sent out. When multiple transmitters want to access the same time slot, one of them can be assigned as the "owner", which accesses the slot with the minimum backoff $T_{backMin}$. Other transmitters randomly back off between the minimum and maximum backoffs, $(T_{backMin}, T_{backMax}]$. Here the minimum backoff, $T_{backMin}$, can be set to tolerate time synchronize errors, and the maximum backoff, $T_{backMax}$, affects time slot length. This contention scheme allows non-owners to utilize a time slot when the slot owner has no packet to send. It also allows non-owners to set their backoff lengths for slot competition based on their access priorities. For example, a low priority transmitter should get a longer backoff length.

5 Admission Controller and Resource Scheduler

In BuddyQoS, the Admission Controller and Resource Scheduler are responsible for providing throughput assurances for communication in coexisting and shared BSNs. In the rest of this section, we first present a communication paradigm that dampens ACK implosion. Then, we demonstrate how to estimate the wireless resources needed to satisfy throughput requirements within the paradigm. Finally, we describe how the Admission Controller makes admission decisions and how the Resource Scheduler schedules inter- and intra-BSN communication based on the resource usage estimation.

5.1 Communication Paradigm for Shared BSNs

In coexisting and shared BSNs, a data packet from a sensor node in a BSN could have multiple receivers. For example, the receivers may be the local aggregator and some aggregators from neighboring buddy BSNs. A receiver uses an ACK to acknowledge each received packet, so that the transmitter can infer a packet loss by an ACK timer and retransmit the lost packet. When multiple receivers receive the same packets, all of them will send ACKs to the same transmitter, and the large number of ACKs may overwhelm the transmitter. There has already been extensive studies on alleviating the "ACK implosion" problem in building reliable broadcast protocols [38] [39]. To dampen ACK implosion, BuddyQoS adopts the selective NACK (SNACK) mechanism that is similar to the TCP acknowledgment mechanism in [40].

In our communication paradigm, a time interval contains two periods (See Figure 2). The first period contains T_m slots and is used for aggregators to broadcast management messages. The second period contains T_d slots and is used for data delivery. Technically, all receivers (aggregators) are informed of the schedule in advance, and hence know the number of data packets to receive from a transmitter (sensor node) in a time interval. In run-time, each receiver uses a bit vector to compactly store the sequence numbers of lost packets and sends an SNACK that contains the bit vector to notify the transmitter of lost packets.

Figure 3 depicts an example of a transmitter, say i, transmitting data packets and SNACKs during the time slots allocated for it in a time interval. The first several slots are allocated for transmitter i to send its data packets. The following slot is allocated for the receivers to send SNACKs. Since the SNACK slot is not pre-allocated, so any receiver can contend for the slot. After an SNACK from one receiver, say j_1, is successfully

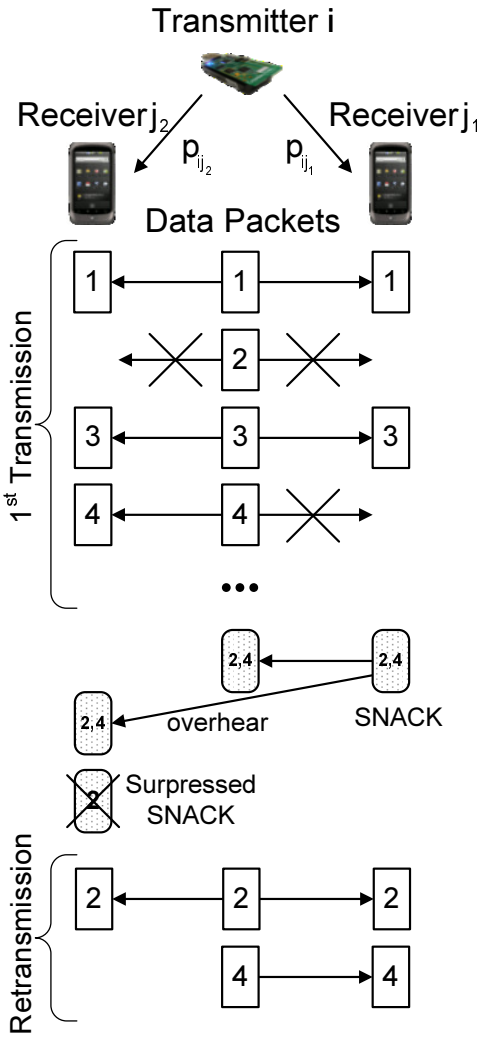


Fig. 4: SNACK Suppression Example

received, the following slots are allocated for the transmitter to retransmit the lost packets. The other receivers overhear the SNACK sent by receiver j_1 and suspend their SNACKs. After retransmissions, if any receiver still has lost packets, it will contend for the next SNACK slot.

In the paradigm, we further introduce SNACK suppression to reduce the number of SNACKs. In SNACK suppression, a receiver can overhear the SNACKs sent from other receivers. To reduce the number of SNACKs, a receiver suppresses its own SNACK when it finds that the overheard SNACK already contains the sequence numbers of its own lost packets. Figure 4 exemplifies an SNACK suppression, where two receivers j_1 and j_2 needs to receive packets from transmitter i. After the first batch of transmissions, receiver j_1 loses the 2^{nd} and 4^{th} packets, and receiver j_2 loses the 2^{nd} packet. Receiver j_2 overhears j_1's SNACK and finds out that the lost packets reported by j_1 already contain its all lost packets. Thus, j_2 suppresses its own SNACK because it receives its all lost packets during the following retransmissions that is triggered by j_1's SNACK. In this example, an SNACK is saved.

To encourage SNACK suppression, a receiver with a higher packet loss rate should send its SNACK before other receivers with lower packet loss rates. In the Hybrid MAC, each receiver sets its backoff time for SNACK slot contention according to its packet loss rate. The higher the packet loss rate, the shorter the backoff time.

5.2 Resource Estimation for Shared BSNs

In the aforementioned communication paradigm, wireless resources are the time slots for sending data packets and SNACKs. In BuddyQoS, the Admission Controller and Resource Scheduler estimate the resource for management information exchange and data delivery among coexisting BSNs.

5.2.1 Resource Estimation for Management Information Exchange

During each T_m period, every aggregator sends one management message that contains its inter- and intra-BSN management information. The number of time slots needed in T_m period depends on the number of neighboring buddy BSNs. At the beginning, T_m is set as T_{init}. Then, in every time interval, T_m is reevaluated with the number of neighboring buddy BSNs, which is obtained from the Buddy Management. If the current T_m minus the number of neighboring buddy BSNs is less than a threshold ΔT, T_m is reset as $T_m+\Delta T$. However, if the current T_m minus the number of neighboring buddy BSNs is larger than $2\Delta T$, T_m is reset as $T_m-\Delta T$. ΔT is a system parameter, which is used to reserve management slots for incoming new buddy BSNs. During system configuration, the value of ΔT is set with respect to the dynamic level of coexisting and shared BSNs.

5.2.2 Resource Estimation for Data Delivery

The BuddyQoS of each BSN estimates the number of time slots for local sensor nodes to deliver data packets, retransmitted data packets, and SNACKs during each T_d period. For a sensor node, the number of data packets to be transmitted depends on the throughput requirement on that node. The number of retransmitted data packets and SNACKs depends on the packet loss rates between the node and its receivers. We list and explain the following denotations for resource estimation:

- b_i: the highest throughput requirement on node i's;
- S_{pkt}: the effective payload size of each data packet;
- q_{ij}: the packet loss rate of receiver j from transmitter i;
- R: the maximum number of (re)transmissions for sending a data packet. After R times of (re)transmissions, failure will be returned to the upper layer and the data packet will be dropped. Otherwise, keeping too much data in the buffer results in buffer overflow. Also, it is meaningless to keep old data for real-time monitoring applications;
- J_i: the IDs of the receivers that listen to node i, $J_i = \{j_1, j_2, \ldots\}$. Thus, the cardinality of set J_i is the number of i's receivers;
- D_i: the number of data packets that needs to be delivered in each interval T from transmitter i to satisfy throughput requirement b_i. Thus, $D_i = \lceil \frac{b_i \times T}{S_{pkt}} \rceil$;
- $E(K)$: K is the number of (re)transmissions for a successful packet delivery to all receivers or a packet delivery failure after R (re)transmissions. $E(K)$ stands for the expected number of (re)transmissions;

- $E(N_j)$: N_j is the number of SNACKs from receiver j. $E(N_j)$ stands for the expected number of SNACKs from j.

Among the above parameters, the throughput requirement b_i is from applications. S_{pkt} and R are set during system configuration. The packet loss rate q_{ij} is the exponentially weighted moving average over the current and history measurements with delay factor α:

$$q_{ij} = \alpha \times q_{ij\_history} + (1-\alpha) \times q_{ij\_current}$$

Then, the number of slots allocated for sensor node i to deliver its data packets equals the number of data packets needs to be delivered multiplying the expected number of transmissions for each data packet, i.e. $D_i \times E(K)$. The number of slots for all receivers in J_i to deliver SNACKs equals the sum of the expected numbers of SNACKs from all receivers, i.e. $\sum_{j \in J_i} E(N_j)$. Details for computing $E(K)$ and $E(N_j)$ are explained as follows.

The expected number of data packet (re)transmissions $E(K)$ is computed using Equations 1∼4. Firstly, according to the definition of $E(K)$ in Equation 1, it is necessary to compute $Pr(K{=}k)$ for each $k \in [1, R]$. Here, $Pr(K{=}k)$ denotes the probability for a successful packet delivery to all receivers after k (re)transmissions or a packet delivery failure after R (re)transmissions.

$$E(K) = \sum_{k=1}^{R} k \times Pr(K{=}k) \tag{1}$$

$Pr(K{=}k)$ is computed using Equation 2, where $Pr(K{\leq}k)$ for each $k \in [1, R{-}1]$ is the probability of a data packet delivery success or failure after less than or equal to k times.

$$Pr(K{=}k) = \begin{cases} Pr(K{\leq}k) - Pr(K{\leq}k{-}1) & , k{<}R \\ 1 - Pr(K{\leq}R{-}1) & , k{=}R \end{cases} \tag{2}$$

To compute $Pr(K{\leq}k)$, we further look into the number of (re)transmissions needed for successfully delivering a packet to each receiver j, which we denote as K_j ($j \in J_i$). Assume that packet loss of each receiver is independent, then $Pr(K{\leq}k)$ can be calculated using $Pr(K_j{\leq}k)$s as shown in Equation 3.

$$Pr(K{\leq}k) = \prod_{j \in J_i} Pr(K_j{\leq}k) \tag{3}$$

After that, for each receiver j, we use its packet loss rate q_{ij} to compute $Pr(K_j{\leq}k)$. It is easy to see that if a data packet has been successfully delivered to receiver j after k (re)transmissions, the first k (re)transmissions must have all failed, with the probability of q_{ij}^k. So Equation 4 is derived.

$$Pr(K_j{\leq}k) = 1 - q_{ij}^k \tag{4}$$

Finally, to deliver data packets from the transmitter i while ensuring throughput b_i, the total number of slots needed equals:

$$D_i \times E(K) = \lceil \frac{b_i \times T}{S_{pkt}} \rceil \times \big(R - \sum_{k=1}^{R-1} \prod_{j \in J_i} (1 - q_{ij}^k)\big) \tag{5}$$

$$E(N_j) = \sum_{n=1}^{R-1} n \times Pr(N_j{=}n) \tag{6}$$

The expected number of SNACKs from receiver j, $E(N_j)$, is defined in Equations 6. Similar to the computation of $E(K)$, we first compute $Pr(N_j{=}n)$, the probability that receiver j has (re)transmitted exactly n SNACKs for successfully receiving all D_i data packets, for each $n \in [1, R{-}1]$ using Equation 7.

$$Pr(N_j{=}n) = \begin{cases} Pr(N_j{\leq}n) - Pr(N_j{\leq}n{-}1) & , n{<}R-1 \\ 1 - Pr(N_j{\leq}R{-}2) & , n{=}R-1 \end{cases} \tag{7}$$

In Equation 7, we calculate $Pr(N_j{\leq}n)$ using receiver j's packet loss rate q_{ij}. It is easy to see that when receiver j sends more than n SNACKs for a data packet, the first n (re)transmissions for the data packet must have failed, with the probability of q_{ij}^k. So the probability that receiver j has sent no more than n SNACKs for one data packet is $1 - q_{ij}^n$. Also, if receiver j has sent out no more than n SNACKs, it means that each of the D_i data packets must have failed for less than n times. We assume that the loss of each packet is independent. So, Equation 8 is derived.

$$Pr(N_j{\leq}n) = \big(1 - q_{ij}^n\big)^{D_i} \tag{8}$$

In order to deliver D_i data packets, the total number of slots needed for sending SNACKs from all receivers to transmitter i equals:

$$\begin{aligned} \sum_{j \in J_i} E(N_j) &= \sum_{j \in J_i} \big(R - \sum_{n=1}^{R-2} (1 - q_{ij}^n)^{D_i}\big) \\ &= R \times |J_i| - \sum_{j \in J_i} \sum_{n=1}^{R-2} (1 - q_{ij}^n)^{D_i} \end{aligned} \tag{9}$$

Note that Equation 8 does not consider the SNACK suppression, because it would require extra information of the correlation between the packet losses of different receivers, which is too costly to obtain in real deployments. We leave it as future work to find out a lightweight real-time measurement of the packet loss correlations. However, the estimation according to Equation 8 is *statistically* larger than or equal to the number of SNACK slots actually used. Therefore, the resource allocated to each node is sufficient to provide the statistical throughput required by the applications. Equation 9 can be noticeably over-estimated only in the extreme case, in which the packet loss ratio is very high and all receivers are losing the packets. However in those cases, the network resources are not sufficient to provide the throughput guarantees even without over-estimation. Finally, the total time slots allocated to transmitter i is the sum of Equations 5 and 9.

5.3 Admission Decisions

To make admission decisions, the Admission Controller in each BSN estimates the number of time slots for management message exchange as described in Section 5.2.1 and time slots to satisfy the throughput requirements on local nodes as described in Section 5.2.2. Meanwhile, the Admission Controller obtains the estimated number of time slots needed by other buddy BSNs from the Buddy Management. Then, the Admission Controller calculates the total number of time slots

needed by all BSNs, including local and neighboring buddy BSNs.

Admission decisions are made based on whether the total number of time slots is larger than T_d or not. If T_d is larger, all local throughput requirements, as well as other buddy BSNs' throughput requirements, can be accepted. Otherwise, each aggregator makes an independent decision to reject some of QoS requests with low priorities. The Admission Controller keeps rejecting QoS requests with the lowest priority until the throughput requirements of the rest QoS requests can be met. Once the Admission Controller finds that any local QoS request need to be rejected, it notifies the corresponding application that the throughput requirement cannot be satisfied. To the application, this means that for one thing, its data stream will be only delivered with best efforts, and for another, it can lower its throughput requirement in order to be admitted. Finally, the set of the accepted QoS requests on local sensor nodes are passed to the Resource Scheduler.

5.4 Resource Scheduler

The Resource Scheduler in each BSN allocates time slots for the aggregator to broadcast its management message in each T_m period and for local sensor nodes to deliver data packets in each T_d period. In addition, the Resource Scheduler on the aggregator and the Slave Resource Scheduler on each local sensor node are responsible for enforcing the schedule.

The Buddy Management in each BSN maintains a buddy list. During each T_m period, the BSN order of sending management messages is the same as the BSN order in the buddy list. With the buddy list from the Buddy Management, the Resource Scheduler in a BSN allocates a time slot for local aggregator to send its management message.

During each T_d period, the BSN order of delivering local data is also the same as the BSN order in the buddy list. To schedule local sensor node to deliver data, the Resource Scheduler in a BSN first looks up the buddy list to find out the schedules of other buddy BSNs. The Resource Scheduler adds up the number of time slots needed by the BSNs before its local BSN in the buddy list and determines at which time slot to start its local data packets delivery. Then, with the list of admitted QoS requests from the Admission Controller, the Resource Scheduler estimates the number of time slots needed by each local node using Equations 5 and 9 and allocates the time slots for local sensor nodes to delivery data packets.

The Resource Scheduler and Slave Resource Scheduler in each BSN enforce the schedules as follows. During each T_m period, all aggregators and sensor nodes listen to all management messages. During each T_d period, each sensor node delivers its data packets during its time slots following the communication paradigm described in Section 5.1. Each aggregator listens to the data packets sent during the time slots allocated to its 'interested' sensor nodes, which could be local or in other buddy BSNs.

6 Buddy Management Design

The Buddy Management maintains the management information of neighboring buddy BSNs in the form of three lists: (1) buddy list, a list of buddy BSNs in the neighborhood; (2) listen list, a list of sensors in other BSNs that local aggregator listens to; and (3) share list, a list of BSNs whose aggregators listen to local sensors. From the other BSNs' management messages received during each T_m period, the Buddy Management extracts the inter-BSN management information to update the lists. The Admission Controller and Resource Scheduler use this information to make admission decisions and resource schedules. Additionally, the Buddy Management uses the management messages to perform neighbor discovery and sensor sharing coordination.

6.1 Neighbor Discovery

During each T_m period, the Buddy Management determines whether a buddy BSN entering and leaving neighborhood and updates buddy list with the help of the Hybrid MAC.

At the very beginning, each BSN has no neighbors but only itself in its buddy list, so it contends for the 1^{st} slot in T_m with some backoff. Here we use the aggregator's ID as its BSN's ID. When a new BSN comes to the neighborhood of BSN j_1, say BSN j_2, both of them contend for the 1^{st} slot. If j_1 wins the 1^{st} slot, then j_2 receives j_1's management message and adds j_1 to its buddy list before itself and tries to access the next management slot. Then, j_1 receives j_2's management message in the later slot and adds j_2 to its buddy list after itself. In this way, the two BSNs discover each other. From then on, the two coexisting BSNs allocate the time slots to send their management messages as described in Section 5.4.

In the Hybrid MAC, a BSN decides its backoff time according to the length of the buddy sublist following itself. To be precise, the more buddy BSNs following the BSN in the buddy list, the shorter the backoff time for it to send its management message. Thus, when a new BSN comes to the neighborhood of multiple BSNs who already know each other, it will fail to access the first several time slots in a T_m period and hence transmit its management message after all existing BSNs. When the aggregator in a BSN hears the management message sent from a new BSN, its Buddy Management adds the new BSN to its buddy list. When an aggregator has not been heard for several consecutive intervals, it is considered leaving the neighborhood and hence removed from all other BSNs' buddy lists.

When a BSN leaves its neighborhood, the neighbor discovery process finds this out in several time intervals. If a shared sensor is in the BSN that has left, no sensing data from that sensor is received during the above time intervals. However after that, another sensor from the existing BSN is selected to replace the leaving sensor. Next, the existing BSNs activate their own sensors, and list them in the broadcasting message. Finally, an application picks a new sharing sensor, and each BSN updates its listening lists to coordinate sensor sharing again.

6.2 Sensor Sharing Coordination

The management message also includes the listen list and the share list from the Buddy Management. Using these two lists, the sensor sharing coordination is performed. During this

process, the listen list acts as a list of sensor sharing requests, and the share list acts as a list of sensor sharing decisions, answering to the sensor sharing requests. The details of the sharing coordination are described as follows:

First, the Buddy Management in a BSN extracts the available sensors of other neighboring buddy BSNs from their schedule in their management messages, and reports them to the applications. Then, the applications decide whether they can benefit from sharing the sensors of other neighboring buddy BSNs or not. If they have sensor share requests, they send them to the Buddy Management. The sensors in the share requests are then added to the listen list and sent out in the management message. When the message is received by the aggregators in other neighboring buddy BSNs, each aggregator checks the listen list, extracts its local sensors from the list, and sends them to the applications to get sharing decisions. If the applications approve the sensor sharing requests, the Buddy Management adds the approved sensors to the share list, which is then sent out in the management message. Otherwise, it adds nothing to the share list. In this way, the sensor sharing requests are answered. When the Buddy Management in the BSN that sends the requests receives the share decisions, it reports the decisions to the corresponding applications.

When the Buddy Management finds a buddy BSN having left the neighborhood, it removes the sharing relations with that BSN from its share list and listen list. In addition, it reports the loss of the corresponding sensors to the applications.

7 Performance Evaluation

We evaluate BuddyQoS using trace-driven simulations. The traces of noise and signal strength used in the simulation are collected from a real office deployment, in which two subjects wear BSNs and work in front of their computers, sitting 2 meters apart and being back to back to each other. Three sensor nodes are attached to the left chest, right wrist, and right ankle of each subject. The aggregator is attached to the left waist of each subject. Each BSN shares a local sensor node with the other BSN. The Received Signal Strength Indicator (RSSI) readings are recorded for 5 minutes when the sensor nodes communicate with the aggregators in both noise and signal scenarios. The noise traces are used to generate noise between each node and its aggregator, following the Closest Pattern Matching (CPM) algorithm [18]. The signal strength traces are used directly in the simulation.

We implement BuddyQoS in TinyOS-2.x with NesC, and simulations are run upon TOSSIM simulator [41]. We compare BuddyQoS with the default CSMA solution in the standard TinyOS-2.x release. Three performance metrics are evaluated: (1) throughput delivery percentage, which equals the delivered throughput over the requested throughput; (2) control overhead, which equals the number of control packets, including SNACKs and management messages, over the number of transmitted data packets; (3) data packet transmission time, which is the time used to transmit a data packet, including the retransmission time and excluding the queuing time.

Table 1 lists the system parameter configurations used in the simulations. The first three parameters are used by the Hybrid MAC. In the default CSMA settings in TOSSIM, the backoff time range is set as $[0.3\text{ms}, 9.78\text{ms}]$. We use a shorter backoff time range of $[0.3\text{ms}, 2.44\text{ms}]$. The short range is feasible since the Hybrid MAC has already avoided most of the collisions through TDMA scheduling. In the simulations, we deliberately introduce $\leq$0.1ms time difference for each node's clock to simulate the synchronization errors. The slot length we set is the sum of the maximum backoff time and the time used to transmit one packet containing 32 bytes data payload. We set T_{init}=5 and ΔT=3, since in our simulation the size of a typical neighborhood is not large at the beginning and it's not very likely that a lot of people join or leave the group at the same time. The decay factor α is set to 0.5, giving equal weights to both the history and current measured packet lose rates.

TABLE 1: System Parameter Configuration

Parameter	Value
Backoff time range $[T_{backMin}, T_{backMax}]$	$[0.3\text{ms}, 2.44\text{ms}]$
Time length of a slot	5ms
Time length of interval T	1000ms
Number of initial slots in T_m period - T_{init}	5 Slots
Number of reserved slots - ΔT	3 Slots
Data packet payload size - S_{pkt}	32 Bytes
Throughput requirement - b_i on node i	1.2kbps
Decay factor - α	0.5

Through the simulations, we first compare the performance between BuddyQoS and the default CSMA solution in TinyOS-2.x in the scenario of two coexisting buddy BSNs. Then, we perform the same performance comparison between the two solutions as the number of buddy BSNs in the neighborhood increases from 2 to 4. The new neighboring buddy BSNs is simulated by the noise and signal strength traces collected from the aforementioned real office environment. Identically, each simulated BSN contains one aggregator and three sensor nodes, and each BSN shares a local sensor node with another buddy BSN. Each aggregator and node can hear all other aggregators and nodes in the neighborhood. The shared nodes are not on the same body position to make the simulation settings realistic.

7.1 Performance Comparison in the Scenario of Two Coexisting Buddy BSNs

First, we measure the throughput delivery percentages of the BuddyQoS and CSMA solutions in the scenario of two coexisting buddy BSNs. The results are demonstrated in Figure 5. For each communication connection between a sensor node and an (local or buddy) aggregator, a group of bar pairs show the average throughput delivery percentages of BuddyQoS and the default CSMA solutions. The error bars show the corresponding standard derivation. For each solution, the simulation is run for 5 minutes. Four groups of bar pairs are plotted for each BSN, with the first 3 bar pairs illustrating the results for the communication connections between the 3 local nodes and local aggregator in a BSN, and the last pair illustrating the results for the communication connection between the local shared node and aggregator in the other buddy BSN. The results prove that our solution ensures 100%

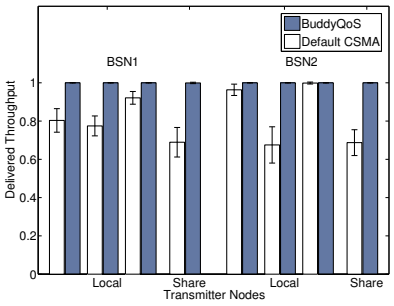


Fig. 5: Delivered Throughput.

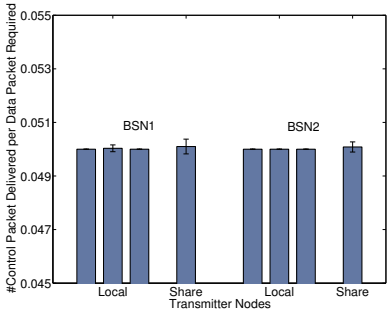


Fig. 6: Control Overhead.

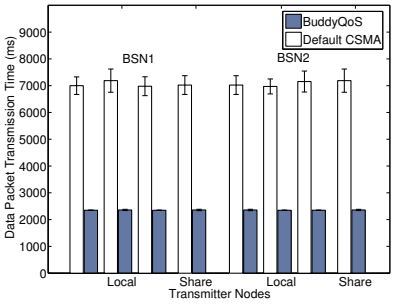
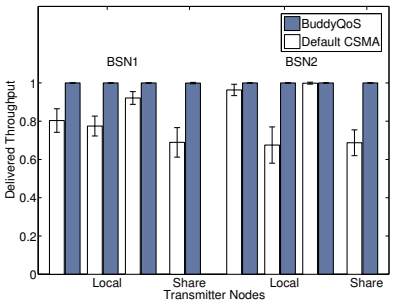
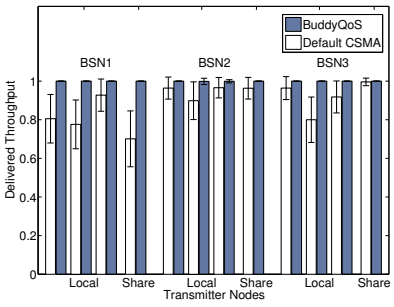


Fig. 7: Data Packet Transmission Time.





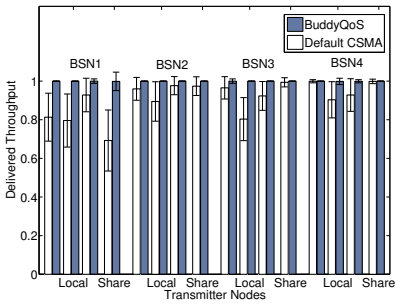


Fig. 8: Percentage of Delivered Throughput with Increasing Number of BSNs

of the required throughput, while the default solution only delivers a portion of the required throughput. Especially, on the shared nodes, the delivered throughput of the default CSMA solution can be as bad as 70% of the required throughput.

Then, we measure the control overhead of BuddyQoS, which is generated by periodically broadcasting management messages and SNACKs. As the aggregator in a BSN sends one management message in each interval to all the local sensor nodes, as well as the sensor nodes it shares from other BSNs, the cost of management messages should be average over these nodes. For example, in the scenario of two coexisting buddy BSNs, one management message is used by 4 nodes (3 local nodes plus 1 shared node). For each sensor node, we compute the control overhead as $\{$(the number of SNACKs it receives + the $1/4$ share of the management message)/the number of delivered data packets$\}$. Figure 6 plots the control overhead for each node in the two neighboring buddy BSNs. The mean values of the control overheads are only around 0.05 with very small standard deviation. The control overhead is low since BuddyQoS uses SNACK and embraces the SNACK suppression mechanism.

Finally, we calculate the time used to transmit a data packet, including the retransmission time and excluding the queuing delay. Figure 7 compares the data packet transmission time of BuddyQoS with that of the default CSMA. We observe that BuddyQoS uses only about $1/3$ of the transmission time of the default CSMA to send a data packet. BuddyQoS has much higher transmission efficiency because BuddyQoS adopts a hybrid MAC that combines the advantages of both CSMA and TDMA. Particularly, BuddyQoS reduces the number of retransmissions caused by collisions. In addition, the adoption of selective NACKs and their suppression also contributes to the lower transmission cost of the paradigm.

7.2 Performance Comparisons in the Scenarios of Multiple Coexisting Buddy BSNs

Figure 8 plots the throughput delivery percentages in the scenarios of 2, 3, and 4 coexisting buddy BSNs. For each scenario, the simulation runs for 5 minutes, and the percentage of delivered throughput is measured every 10 seconds. Then, we plot the mean and standard derivation of the measured results for each node of each BSN (three local sensor nodes and one sensor node shared with another BSN).

Comparing our solution with the default CSMA solution, we evidence that the CSMA solution is not able to ensure the requested data throughput. The percentage of delivered throughput under the CSMA solution varies on different sensor nodes. For some of the sensor nodes, only about 70% of the data can be delivered. The standard derivation of the delivered throughput is also large (nearly 10%), implying that the links are unstable. However, BuddyQoS ensures the applications' throughput requirement on each node, even when the number of buddy BSNs in the neighborhood increases. Notably, BuddyQoS delivers 100% of the required throughput with small standard derivation.

8 Discussion

In this section, we discuss potential limitations of BuddyQoS and other related issues that may matter in practice. In particular, we compare challenges of BSNs and traditional wireless networks. Next, we elaborate interference among co-existing and shared BSNs. We also describe the rationale behind the decision of adopting our proposed sensing sharing strategy. Finally, privacy concerns of BSNs are elaborated.

Media access control (MAC) for traditional wireless cellular networks has been extensively studied in the research community. Many researchers proposed modifications to the original MAC in order to improve system performance and reliability. Such hybrid MAC solutions [42][43][44][45][46] for instance propose co-channel interference models, and devise methods based on MAC to reduce interference. However, BSNs are in many aspects very much different from traditional wireless cellular networks. First, BSN sensor nodes usually operate in a dense body area that introduces many communication challenges [23], such as body shadowing - the body's line-of-sight absorption of RF energy, which, coupled with movement, causes significant and highly variable path loss. Second, such sensor nodes require more careful power management policies due to their low-power nature. Moreover, the communication range of the nodes is much smaller than in traditional networks, and the communication is therefore more prone to interference. In addition to these natural differences of our hybrid MAC protocol tailored for BSNs, our proposed solution also strives to provide throughput assurances for coexisting and shared buddy BSNs. In particular, BuddyQoS coordinates sensor sharing to support common sensor usage among coexisting BSNs.

BSNs often suffer from interference due to their operation in the same vicinity with other BSNs. However, our evaluation demonstrates that BuddyQoS can effectively mitigate this interference. As demonstrated in Section 7, the transmission time is reduced by $2/3$ compared to the default CSMA, which is due to the fact that our framework eliminates collisions that cause many retransmissions. This way BuddyQoS mitigates the interference among co-existing and shared BSNs.

In our BuddyQoS framework, we assume that an application makes sensor sharing decisions based on its needs. The needs are associated with the application level performance, as opposed to the network level performance. In addition, different applications may have different such needs to serve their goals. For example, the CoMon [6] ambience monitoring platform aims to save mobile device energy through sensor sharing, while Remora [7] resource sharing platform targets to improve activity recognition accuracy through sensor sharing. Therefore, it is non-trivial to provide common ground for all applications to carry out such sensor sharing decisions. The goal of BuddyQoS is to give performance guarantees for sensor sharing at the communication level. Our proposed framework achieves its goal through a novel hybrid MAC protocol with interference reduction, and its evaluation demonstrates that the framework outperforms the state-of-the-art solution.

In many application scenarios, sensing data can be shared among trusted parties without any privacy concerns, such as fellow athletes, couples, or friends in a retirement community. Some sensor data, such as data collected from motion and environment sensors, can be shared with neighbors in physical proximity as they are already able to physically see each other in motion. For those application scenarios where privacy could become an issue, a BSN can share only parts of its sensing data and allow users only sharing their non-private information with a limited number of trustworthy parties.

9 Conclusion

When BSN users spend time with family, friends and colleagues, multiple BSNs usually coexist in the communication range of each other. In such scenarios, applications can usually benefit from sensor sharing among BSNs. This paper proposes the first QoS solution, BuddyQoS, to meet application level throughput requirements for both inter- and intra-BSN communications. BuddyQoS accurately estimates wireless resource and adaptively allocates the resource to multiple BSNs in order to achieve throughput assurances. Through trace-driven simulation, we have demonstrated that BuddyQoS notably outperforms the default CSMA solution in standard TinyOS-2.x release with a small cost. In the future, we will study how to provide both throughput and time delay performance assurances for coexisting and shared BSNs.

Acknowledgment

This work was supported in part by U.S. National Science Foundation under grant CNS-1250180 and CNS-1253506 (CAREER).

References

[1] M. Keally, G. Zhou, G. Xing, J. Wu, and A. Pyles, "PBN: Towards Practical Activity Recognition Using Smartphone-Based Body Sensor Networks," in *ACM SenSys*, 2011.

[2] H. Huang, Y. Sun, Q. Yang, F. Zhang, X. Zhang, Y. Liu, J. Ren, and F. Sierra, "Integrating Neuromuscular and Cyber Systems for Neural Control of Artificial Legs," in *ACM/IEEE ICCPS*, 2010.

[3] Q. Li, J. A. Stankovic, M. Hanson, A. Barth, J. Lach, and G. Zhou, "Accurate, Fast Fall Detection Using Gyroscopes and Accelerometer-Derived Posture Information," in *BSN*, 2009.

[4] M. Bächlin, K. Förster, and G. Tröster, "SwimMaster: A Wearable Assistant for Swimmer," in *ACM UbiComp*, 2009.

[5] A. T. Campbell, T. Choudhury, S. Hu, H. Lu, M. K. Mukerjee, M. Rabbi, and R. D. S. Raizada, "Neuro-Phone: Brain-Mobile Phone Interface using a Wireless EEG Headset," in *ACM MobiHeld*, 2010.

[6] Y. Lee, Y. Ju, C. Min, S. Kang, I. Hwang, and J. Song, "CoMon: Cooperative Ambience Monitoring Platform with Continuity and Benefit Awareness," in *ACM MobiSys*, 2012.

[7] M. Keally, G. Zhou, G. Xing, and J. Wu, "Remora: Sensing resource sharing among smartphone-based body sensor networks," in *IEEE IWQoS*, 2013.

[8] C. Karlof, N. Sastry, and D. Wagner, "Tinysec: a link layer security architecture for wireless sensor networks," in *Proceedings of the 2nd international conference on Embedded networked sensor systems*, ser. SenSys '04, 2004.

[9] G. Zhou, J. Lu, C.-Y. Wan, M. D. Yarvis, and J. A. Stankovic, "BodyQoS: Adaptive and Radio-Agnostic QoS for Body Sensor Networks," in *IEEE INFOCOM*, 2008.

[10] Z. Ren, G. Zhou, A. Pyles, M. Keally, W. Mao, and H. Wang, "BodyT2: Throughput and Time Delay Performance Assurance for Heterogeneous BSNs," in *IEEE INFOCOM*, 2011.

[11] A. T. Campbell, S. B. Eisenman, N. D. Lane, E. Miluzzo, and R. A. Peterson, "People-centric urban sensing," in *WICON*, 2006.

[12] S. B. Eisenman, E. Miluzzo, N. D. Lane, R. A. Peterson, G. S. Ahn, and A. T. Campbell, "The BikeNet Mobile Sensing System for Cyclist Experience Mapping," in *ACM SenSys*, 2007.

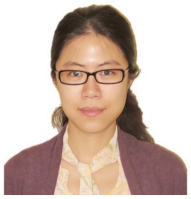
[13] H. Lu, N. D. Lane, S. B. Eisenman, and A. T. Campbell, "Bubble-sensing: Binding sensing tasks to the physical world," *Pervasive and Mobile Computing*, 2010.

[14] E. Miluzzo, N. D. Lane, A. T. Campbell, and R. Olfati-Saber, "CaliBree: A Self-calibration System for Mobile Sensor Networks," in *IEEE DCOSS*, 2008.

[15] R. J. Honicky, "Understanding and Using Rendezvous to Enhance Mobile Crowdsourcing Applications," *Computer*, 2011.

[16] I. Rhee, M. Shin, S. Hong, K. Lee, and S. Chong, "On the Levy-walk Nature of Human Mobility: Do Humans Walk like Monkeys?" in *IEEE INFOCOM*, 2008.

[17] K. Lee, S. Hong, S. J. Kim, I. Rhee, and S. Chong, "Slaw: A New Mobility Model for Human Walks," in *IEEE INFOCOM*, 2009.
[18] K. Lee, Y. Kim, S. Chong, I. Rhee, and Y. Yi, "Delay-Capacity Tradeoffs for Mobile Networks with Lévy Walks and Lévy Flights," in *IEEE INFOCOM*, 2011.
[19] S. Srinivasa and S. Krishnamurthy, "CREST: an Opportunistic Forwarding Protocol Based on Conditional Residual Time," in *IEEE SECON*, 2009.
[20] S. B. Eisenman, N. D. Lane, and A. T. Campbell, "Techniques for Improving Opportunistic Sensor Networking Performance," in *IEEE DCOSS*, 2008.
[21] K. K. Rachuri, C. Efstratiou, I. Leontiadis, C. Mascolo, and P. J. Rentfrow, "Metis: Exploring mobile phone sensing offloading for efficiently supporting social sensing applications," in *Pervasive Computing and Communications (PerCom), 2013 IEEE International Conference on*. IEEE, 2013, pp. 85–93.
[22] S. Yang, Y. Kwon, Y. Cho, H. Yi, D. Kwon, J. Youn, and Y. Paek, "Fast dynamic execution offloading for efficient mobile cloud computing," in *Pervasive Computing and Communications (PerCom), 2013 IEEE International Conference on*. IEEE, 2013, pp. 20–28.
[23] M. A. Hanson, H. C. Powell Jr, A. T. Barth, K. Ringgenberg, B. H. Calhoun, J. H. Aylor, and J. Lach, "Body area sensor networks: Challenges and opportunities," *Computer*, no. 1, pp. 58–65, 2009.
[24] D. K. Rout and S. Das, "Interference mitigation in wireless body area networks using modified and modulated mhp," *Wireless Personal Communications*, pp. 1–19, 2014.
[25] S. Cui, K. Teh, K. Li, Y. Guan, and C. Law, "Narrowband interference suppression in transmitted reference uwb systems with inter-pulse interference," in *Ultra-Wideband, 2007. ICUWB 2007. IEEE International Conference on*. IEEE, 2007, pp. 895–898.
[26] W. Gao, R. Venkatesan, and C. Li, "A pulse shape design method for ultra-wideband communications," in *Wireless communications and networking conference, 2007. WCNC 2007. IEEE*. IEEE, 2007, pp. 2800–2805.
[27] S. M. Ekome, G. Baudoin, M. Villegas, and J. Schwoerer, "Narrowband interference mitigation in uwb communication with energy detector," in *Ultra-Wideband (ICUWB), 2012 IEEE International Conference on*. IEEE, 2012, pp. 67–71.
[28] J. Ibrahim and R. M. Buehrer, "Nbi mitigation for uwb systems using multiple antenna selection diversity," *Vehicular Technology, IEEE Transactions on*, vol. 56, no. 4, pp. 2363–2374, 2007.
[29] Z. Xie, G. Huang, J. He, and Y. Zhang, "A clique-based wban scheduling for mobile wireless body area networks," *Procedia Computer Science*, vol. 31, pp. 1092–1101, 2014.
[30] Z. Zhang, H. Wang, C. Wang, and H. Fang, "Interference mitigation for cyber-physical wireless body area network system using social networks," 2013.
[31] IEEE, "IEEE 802.15 WPAN Task Group 6 (TG6) Body Area Networks," http://ieee802.org/15/pub/TG6.html.
[32] H. Jeong, "An adaptive scheduling algorithm for the patient monitoring system on wbans," in *Internet of Things*. Springer, 2012, pp. 17–24.
[33] J. S. Choi and J. G. Kim, "An improved mac protocol for wban through modified frame structure." *International Journal of Smart Home*, vol. 8, no. 2, 2014.
[34] N. F. Timmons and W. G. Scanlon, "Improving the ultra-low-power performance of ieee 802.15. 6 by adaptive synchronisation," *IET wireless sensor systems*, vol. 1, no. 3, pp. 161–170, 2011.
[35] S. Ullah, M. Mohaisen, and M. A. Alnuem, "A review of ieee 802.15. 6 mac, phy, and security specifications," *International Journal of Distributed Sensor Networks*, vol. 2013, 2013.
[36] I. Rhee, A. Warrier, M. Aia, J. Min, and M. L. Sichitiu, "Z-MAC: A Hybrid MAC for Wireless Sensor Networks," *IEEE/ACM Transactions on Networking*, 2008.
[37] G.-S. Ahn, S. G. Hong, E. Miluzzo, A. T. Campbell, and F. Cuomo, "Funneling-mac: a localized, sink-oriented mac for boosting fidelity in sensor networks," in *Proceedings of the 4th international conference on Embedded networked sensor systems*. ACM, 2006, pp. 293–306.
[38] B. Rajagopalan, "Reliability and scaling issues in multicast communication," in *ACM SIGCOMM Computer Communication Review*, 1992.
[39] F. Stann and J. Heidemann, "RMST: Reliable Data Transport in Sensor Networks," in *The First International Workshop on Sensor Network Protocols and Applications*, 2003.
[40] K. Fall and S. Floyd, "Simulation-Based Comparisons of Tahoe, Reno and SACK TCP," *ACM C2R*, 1996.
[41] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications," in *ACM SenSys*, 2003.
[42] K. Gulati, A. Chopra, B. L. Evans, and K. R. Tinsley, "Statistical modeling of co-channel interference," in *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*. IEEE, 2009, pp. 1–6.
[43] M. Z. Win, P. C. Pinto, and L. A. Shepp, "A mathematical theory of network interference and its applications," *Proceedings of the IEEE*, vol. 97, no. 2, pp. 205–230, 2009.
[44] E. Salbaroli and A. Zanella, "Interference analysis in a poisson field of nodes of finite area," *Vehicular Technology, IEEE Transactions on*, vol. 58, no. 4, pp. 1776–1783, 2009.
[45] X. Yang and A. P. Petropulu, "Co-channel interference modeling and analysis in a poisson field of interferers in wireless communications," in *Classical, Semi-classical and Quantum Noise*. Springer, 2012, pp. 271–282.
[46] X. Ge, K. Huang, C.-X. Wang, X. Hong, and X. Yang, "Capacity analysis of a multi-cell multi-antenna cooperative cellular network with co-channel interference," *Wireless Communications, IEEE Transactions on*, vol. 10, no. 10, pp. 3298–3309, 2011.



Zhen Ren Zhen Ren received her Ph.D. degree in Computer Science from the College of William and Mary in 2012. Her research interest includes wireless communication and networking, sensor networks, especially Body Sensor Networks (BSN), ubiquitous computing, Quality of Service (QoS), and Voice over Internet Protocol (VoIP).

Xin Qi Xin Qi received his BSc degrees in computer science from Nanjing University, China, in 2007 and ME Degree from LIAMA, a joint lab between Chinese Academy of Science and INRIA, in 2010, respectively. He is currently pursuing Ph.D. degree in the Department of Computer Science, The College of William and Mary. His research interests are mainly in Ubiquitous computing and mobile systems.

Gang Zhou Dr. Gang Zhou is an Associate Professor in the Computer Science Department at the College of William and Mary. He received his Ph.D. degree from the University of Virginia in 2007 under Professor John A. Stankovic. He has published more than 60 papers in the areas of sensor networks, ubiquitous computing, mobile computing and wireless networks. The total citations of his papers are more than 4300 according to Google Scholar, among which the MobiSys04 paper has been cited more than 780 times. He also has 13 papers each of which has been cited more than 100 times since 2004. He is an Editor of IEEE Internet of Things Journal. He is also an Editor of Elsevier Computer Networks Journal. Dr. Zhou served as NSF, NIH, and GENI proposal review panelists multiple times. He also received an award for his outstanding service to the IEEE Instrumentation and Measurement Society in 2008. He is a recipient of the Best Paper Award of IEEE ICNP 2010. He received NSF CAREER Award in 2013. He is a Senior Member of IEEE and a Senior Member of ACM.

Haining Wang Haining Wang received his Ph.D. in Computer Science and Engineering from the University of Michigan at Ann Arbor in 2003. He is currently a Professor in the Department of Electrical and Computer Engineering at the University of Delaware, Newark, DE. His research interests lie in the areas of security, networking system, and cloud computing. He is a senior member of IEEE.

David T. Nguyen David has been working on his Ph.D. in Computer Science at the College of William and Mary, USA, since Fall 2011. He is working with Dr. Gang Zhou, and his research interests include mobile computing, ubiquitous computing, and wireless networking. Before coming to W&M (Fall 2011), he was a lecturer at Suffolk University in Boston for 2 years. He was also a lecturer at Christopher Newport University in 2013. In 2014, he worked as a Mobile Hardware Engineer in Facebook's Connectivity Lab, Menlo Park, CA.