BodyT2: Throughput and Time Delay Performance Assurance for Heterogeneous BSNs

Zhen Ren, Gang Zhou, Andrew Pyles, Matthew Keally, Weizhen Mao, Haining Wang
Computer Science Department, College of William and Mary

***Abstract*—Body sensor networks (BSNs) have been developed for a set of performance-critical applications, including smart healthcare, assisted living, emergency response, athletic performance evaluation, and interactive controls. Many of these applications require stringent performance assurance in terms of communication throughput and bounded time delay. While solutions exist in literature for providing joint throughput and time delay assurance by proposing specific MAC protocols or extensions, we provide this joint assurance in a novel radio-agnostic manner. In our approach, the underlying MAC and PHY layers can be heterogeneous and their details do not need to be known to upper layers like the resource management. Such a radio-agnostic performance assurance is critical because a range of radio platforms are adopted for practical body sensor usage. Our approach is based on a group-polling scheme that is essential for radio-agnostic BSN design. Through theoretical analysis, we prove that with the group-polling scheme, achieving joint throughput and time delay assurance is an NP-hard problem. For practical system deployment, we propose the BodyT2 framework that assures throughput and time delay performance in a heterogeneous BSN. Through both TelosB mote lab tests and real body experiments in an Android phone-centric BSN, we demonstrate that BodyT2 achieves superior performance over existing solutions.**

I. Introduction

A Body sensor network (BSN) consists of a group of wireless sensors, which are either wearable on or implanted into a human body to monitor vital physiological parameters and body movements. The data collected by body sensors are transmitted to an aggregator (e.g., a cell phone) and then is reliably delivered to a data center (e.g., a hospital) in real-time for analysis. BSNs have attracted significant interest from a wide range of applications, including smart healthcare [1], assisted living [2], emergency response [3], athletic performance evaluation [4], and interactive controls [5]. Many of these applications are performance-critical, requiring stringent throughput and time delay performance assurance. For example, in the NeuroPhone application [5], which uses a wireless EEG headset (16 channels in total and 4Kbps per channel) for detecting the neural signals of a human brain to control iPhone applications, throughput and time delay should be guaranteed to deliver the neural signals from the EEG sensors to the iPhone for interactive controls.

To provide joint throughput and time delay performance assurance within BSNs, two research challenges need to be addressed: irregular BSN link quality and heterogenous BSN radio platforms. In [6], the general low power wireless sensor communication is reported to be notoriously irregular. In [7] [8], the link quality in a BSN is reported to be highly dynamic and even harder to predict than in a general wireless sensor network due to interference from environment [9], body activities [2], and body fading [10]. In order to ensure the requested performance in the presence of such irregular BSN link quality, available resources must be adaptively rescheduled according to efficiency and cost. Also, existing body sensor devices, especially medical sensor devices, often use heterogeneous radio platforms, such as CC1000, ZigBee/CC2420, and Bluetooth. It is indispensable to achieve the performance assurance in a radio-agnostic manner to support platform portability.

In literature, many existing works propose specific MAC protocols or extensions to specific MAC protocols and radio platforms for providing statistical throughput and/or time delay performance assurance. Representative works are [11], [12], [13], [14], [15], [16], and [17]. Some other works, even though radio-agnostic is discussed, do not provide any performance assurance but instead provide best effort solutions for enhancing throughput and/or reducing time delay. Representative works are [18], [19], [20], [21], and [22]. Another group of works provide either throughput or time delay performance assurance, but not both. Representative works are [23], [24], [25], [26], and [8]. In [27], a solution is presented for multiple BSN data streams that can guarantee different throughputs but with only a single time delay bound. However, this work does not meet our goal of allowing different data streams to request both different throughputs and time delays. Moreover, [27] is based on an individual-polling scheme, in which each data packet transmission from a sensor mote is preceded by a polling message from the central aggregator (details will be given in section II), rather than the more effective group-polling scheme, in which multiple data packet transmissions are allowed after a single polling message. Consequently, [27] is not appropriate for radio agnostic performance assurance and also introduces a minimum of 50% communication overhead.

In this paper, we propose a novel and efficient radio agnostic solution for heterogeneous BSNs. Our solution allows different data streams to request different throughput and time delay performance assurances with reduced communication overhead. We use both theoretical analysis and practical system development to achieve this goal. In particular, we theoretically prove that the joint throughput and time delay performance assurance with a group-polling scheme is NP-

This work was supported in part by NSF grants ECCS-0901437 and CNS-0916994.

hard while the throughput performance assurance is solvable in polynomial time. Meanwhile, we develop BodyT2, a practical solution for joint throughput and time delay performance assurance in heterogeneous BSNs. Through both TelosB mote lab tests as well as real body experiments in an Android phone-centric BSN, we demonstrate that BodyT2 greatly outperforms existing solutions.

The rest of this paper is organized as follows. In Section II, we formulate the problem of joint throughput and time delay performance assurance and analyze its complexity. We present the BodyT2 design in Section III and its performance evaluation in Section IV. We present conclusions in Section V.

II. Problem Definition and Analysis

In this section, we theoretically analyze BSN resource scheduling in order to meet requested performance assurance. We first explain the asymmetric BSN architecture and compare two BSN scheduling schemes: group-polling and individual-polling. Then, based on the more effective group-polling scheme, we prove that scheduling for the throughput performance assurance is a P problem, while the joint throughput and time delay assurance is NP-hard.

A. Group-Polling v.s. Individual-Polling

An asymmetric architecture is desired for BSNs in which a comparatively more powerful aggregator polls less powerful sensor motes for data communication [8]. Two scheduling schemes have been proposed based on this asymmetric BSN architecture. In the *individual-polling* [27] scheme, each data packet transmission from a mote is preceded by a polling packet from the aggregator that specifies which mote is polled. Since this scheme adds in a minimum of 50% communication overhead, it is not appropriate for practical radio-agnostic system deployment. A more effective and energy efficient *group-polling* scheme is introduced in [8], in which multiple data packet transmissions are allowed from a mote following a single polling packet from the aggregator. The series of packets sent after a polling packet, which can be more than one packet, is called a *packet train*. Group-polling is strongly preferred over individual-polling mainly for the following two reasons:

• *Efficiency.* Compared with individual-polling, group-polling requires much fewer polling packets to deliver the same amount of data packets, greatly saving communication bandwidth ($\leq$250Kbps in popular sensor motes like TelosB) and energy (sensor motes are usually powered by AA batteries). The saved communication bandwidth can be used to serve more data streams in a BSN, enhancing the BSN capacity. By listening to more sparsely transmitted polling messages, sensor motes have more sleeping time and hence the system lifetime is extended.

• *Catering to Radio-Agnostic BSN Designs.* Since heterogeneous radio platforms are widely adopted in the commercial market, radio-agnostic performance assurance is needed in BSNs. Group-polling better caters to this demand than individual-polling since it operates on a virtual MAC (VMAC) abstraction [8]. For throughput performance assurance, VMAC abstracts common MAC behaviors with time-domain parameters: T_{minPkt} and T_{maxPkt}. These are respectively the lower and upper bound of the time that the underlying MAC uses for handling a packet transmission request. When the channel is clear, the radio control is returned to VMAC within T_{minPkt}; when suffering interference, the underlying MAC may return the radio control within T_{maxPkt} and report giving up after exceeding the maximum number of backoffs and/or retransmissions. During runtime, VMAC also measures the average MAC response time T_k for each mote k in a BSN, which reflects the average communication cost of a specific mote for a single data packet communication. So, $T_k \in [T_{minPkt}, T_{maxPkt}]$.

Without knowledge of the underlying MAC implementation, the aggregator using individual-polling has to reserve the maximum time T_{maxPkt} for a single data packet transmission. In most cases, the data packet can be successfully transmitted with time much less than T_{maxPkt}, so the rest of the reserved time is wasted. However, with group-polling the aggregator can efficiently estimate the time needed to transmit a packet train as $T_k \times NumofPkt$. Even though the underlying MAC is only allowed to send a data packet when the remaining reserved time is no less than T_{maxPkt} (otherwise, we risk losing control of the underlying radio), this packet's real transmission time T_k^* is usually much less than T_{maxpkt}. The difference ($T_{maxPkt} - T_k^*$) can be salvaged and merged to the time reserved for sending the next packet. In this way, fluctuation of the transmission time is absorbed and tolerated.

B. Throughput Assurance

In BodyQoS [8], throughput performance assurance is provided with the group-polling scheme. Each data stream i specifies its throughput requirement b_i and the scheduling algorithm determines the resource, specifically the time resource, for the data stream.

Definition 1 (BodyQoS Scheduling problem): Suppose group-polling is used in a BSN. Given a fixed-length time interval $T_{interval}$ and N data streams in the BSN with throughput requirements $\{b_i\}$, the problem is to decide the time schedule for each data stream, such that in $T_{interval}$ the delivered throughput is no less than the requested throughput.

In order to solve this problem, BodyQoS first computes the required bandwidth for each data stream when the channel is clear, which is called the ideal bandwidth. Also, the time to send one packet is T_{minPkt} when there is no interference, and the number of data packets to be delivered within $T_{interval}$ is $\lceil \frac{b_i \times T_{interval}}{8 \times S_{pkt}} \rceil$, where S_{pkt} is the affective payload size of a single data packet in bytes. Then at run time, the effective bandwidth is measured. With the ratio of the ideal bandwidth to the moving average result of the effective bandwidth, BodyQoS dynamically recomputes the average packet sending time and the number of data packets, the product of them is the time needed for delivering stream i's data packets. The time for sending one polling message is estimated as T_{maxPkt}, and BodyQoS adopts a constant number (1 is default) of

polling messages within $T_{interval}$ for each data stream, which is configured as a system-wide parameter.

Admission decisions are made based on the total required throughput of all QoS streams, and the scheduling algorithm computes the time schedule for each stream. Since it needs constant time complexity to compute the time of both data communication and polling for individual data streams, computing the required time schedule for all motes in the network is a P problem. In summary, with only the throughput requirement in the group-polling scheme, the BodyQoS scheduling problem is solvable in polynomial time.

C. Joint Assurance of Throughput and Time Delay

For time delay performance assurance, $d_{k,i}$ is introduced to denote the requested time delay bound for data stream i on sensor mote k. The complete performance assurance requirement is denoted as $(b_{k,i}, d_{k,i}, p_{k,i})$ where $b_{k,i}$ specifies the throughput requirement and $p_{k,i}$ denotes the priority. Instead of scheduling polling messages for individual data stream as in [8], here the aggregator aggregates polling messages for all data streams on the same mote. To put it another way, the aggregator does not specify how much time each stream on a mote uses but only allocates enough time to satisfy the total throughput requirement of all streams on the same mote. Thus, a packet train sent from a mote can contain data packets from different data streams.

Now, the scheduling problem is more complicated with the added time delay requirement since it needs to ensure that individual data packets are delivered within $d_{k,i}$. This is equivalent to ensuring that the gap between any two consecutively scheduled packet trains for mote k is bounded by $d_{k,i}$ minus the time of transmitting one polling packet and one data packet. So, if data arrive just after the end of a packet train, the data can be timely transmitted in the next packet train. When multiple data streams are on the same mote k, the aggregator considers the minimum delay requirement $\min_i\{d_{k,i}\}$. For convenience of presentation, we introduce two intermediate symbols:

- $B_k = \sum_i b_{k,i}/S_{pkt}$ is the number of packets required to be sent for all streams on mote k in a unit time.
- $G_k = \min_i\{d_{k,i}\} - T_{maxPkt} - T_k$ is the maximum gap allowed between consecutive packet trains of mote k.

The packet train schedule can be represented as $\{(st_{k,j}, et_{k,j})\}$, where $st_{k,j}$ is the start time for the aggregator to send the polling message of the packet train j of mote k and $et_{k,j}$ is the latest time a data packet from this packet train is allowed to be received at the aggregator. The BodyQoS Scheduling Problem in Def. 1 can be extended to the following BodyT2 Scheduling problem.

Definition 2 (BodyT2 Scheduling problem Π*):* Suppose group-polling is used in a BSN. Given N motes in the BSN with performance requirements (B_k, G_k), the problem is to decide the time schedule $\{(st_{k,j}, et_{k,j})\}$ such that for all $k \in [1, N]$, $j \in$ N, the following constraints are satisfied:

- *Length Constraint.* $\forall k, j,\ et_{k,j} - st_{k,j} = T_{maxPkt} + T_k \times \lceil (et_{k,j} - et_{k,j-1}) \times B_k \rceil$. It ensures that the allocated time is enough to transmit both the data and polling packets for all streams on mote k based on the throughput requirements.
- *Gap Constraint.* $\forall k, j, st_{k,j} - et_{k,j-1} \le G_k$. It ensures that the gap between any two consecutively allocated packet trains of mote k is bounded by the minimum time delay requirement of all streams on mote k.
- *Disjoint Constraint:* $\forall k_1, k_2, j_1, j_2$, if $k_1 \neq k_2$ or $j_1 \neq j_2$, then $st_{k_1,j_1} \neq st_{k_2,j_2}$; if $st_{k_1,j_1} < st_{k_2,j_2}$, then $et_{k_1,j_1} \le st_{k_2,j_2}$. It ensures that time periods allocated to different packet trains do not overlap, i.e., no internal interference.

Lemma 1: The BodyT2 Scheduling problem Π is NP-hard.

With the following three steps, we demonstrate that a known NP-complete problem, the Partition problem (Π'), is polynomially reducible to our BodyT2 Scheduling problem Π. Let π' and π refer to any instances of problems Π' and Π, respectively. We construct a polynomial reduction f that converts any instance π' of the Partition problem to some instance $\pi = f(\pi')$ of our BodyT2 Scheduling problem such that π' has a solution if and only if $\pi = f(\pi')$ has a solution.

Step 1: Construct the polynomial reduction f from π' to π.

Definition 3 (Partition problem Π'*):* Given a finite set A of numbers, is there $A' \subseteq A$, such that $\sum_{a_k \in A'} a_k = \sum_{a_{k'} \in A - A'} a_{k'}$?

For any partition problem instance π' with set $A = \{a_1, \ldots, a_n\}$ of n integers, we choose a constant c such that $c \times a_k \ge 2$ for all $k \in [1, n]$. We construct the following instance $\pi = f(\pi')$ of the BodyT2 Scheduling problem with $n+1$ motes. We let $T_k = T_{maxPkt} = 1, \forall k$, and let $T = c \times (\sum_{a_k \in A} a_k + 2 \times a_{n+1})$, where $a_{n+1} \ge 2/c$ (so $c \times a_{n+1} \ge 2$). We define (B_k, G_k) as:

$$(B_k, G_k) = \begin{cases} (\frac{c \times a_k - 1}{T}, & T - c \times a_k), & k \in [1, n] \\ (\frac{c \times a_{n+1} - 1}{T/2}, & T/2 - c \times a_{n+1}), & k = n+1 \end{cases}$$

This reduction can clearly be done in polynomial time.

Step 2: Prove that if π' has a solution, then $f(\pi')$ has a solution.

For any partition problem π', assume there is a solution such that $A' = \{a_{k'_1}, \ldots, a_{k'_{n_2}}\}$, $A - A' = \{a_{k_1}, \ldots, a_{k_{n_1}}\}$, $n_1 + n_2 = n$, and $\sum_{a \in A'} a = \sum_{a \in A - A'} a = T/2c - a_{n+1}$. We have the following scheduling $f(\pi')$ which repeats with a cycle of length T.

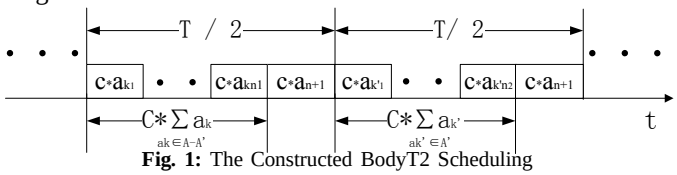


Fig. 1: The Constructed BodyT2 Scheduling

As shown in Fig. 1, packet trains of mote $k_1, \ldots, k_{n_1}$ are scheduled in the first half of T and packet trains of mote $k'_1, \ldots, k'_{n_2}$ are scheduled in the second half. We then have: for mote $k = k_1, \ldots, k_{n_1}$,

$$\begin{aligned} st_{k,1} &= 0, \quad st_{k_2,1} = et_{k_1,1}, \quad \ldots \\ st_{k,j} &= st_{k,j-1} + T, j > 1 \\ et_{k,j} &= st_{k,j} + c \times a_k, \end{aligned}$$

for mote $k = k'_1, \ldots, k'_{n_2}$,

$$\begin{aligned} st_{k'_1,1} &= T/2, \quad st_{k'_2,1} = et_{k'_1,1}, \quad \ldots \\ st_{k,j} &= st_{k,j-1} + T, j > 1 \\ et_{k,j} &= st_{k,j} + c \times a_{k'}, \end{aligned}$$

for mote $n+1$,

$$\begin{aligned} st_{n+1,j} &= \begin{cases} T/2 - c \times a_{n+1}, & j = 1 \\ st_{n+1,j-1} + T/2, & j > 1 \end{cases} \\ et_{n+1,j} &= st_{n+1,j} + c \times a_{n+1}. \end{aligned}$$

Now, we check whether the three constraints in Def. 2 are satisfied. First, we check for mote $k \in [1, n]$. Since the right side of the Length Constraint equals $1+T\times B_k = 1+T\times \frac{c\times a_k - 1}{T} = c\times a_k$ and the left side of it equals $et_{k,j}-st_{k,j} = c\times a_k$, the Length Constraint is satisfied. The Gap Constraint also stands as $st_{k,j}-et_{k,j-1} = T-c\times a_k = G_k$. In any interval T, $st_{k_1,j} < \ldots < st_{k_{n_1},j}$, $et_{k_1,j} = st_{k_1,j}+c\times a_{k_1} = st_{k_1,1}+(j-1)\times T+c\times a_{k_1} = st_{k_2,1}+(j-1)\times T = st_{k_2,j}$, so the packet trains of mote $k \in [k_1, k_{n_1}]$ do not overlap, i.e., the Disjoint Constraint stands. In a similar way, we can also prove that the Disjoint Constraint stands for mote $k \in [k'_1, k'_{n_2}]$.

Second, we check for mote $n+1$. The Length Constraint is satisfied as its right side equals to $1+T\times B_{n+1} = 1+\frac{T}{2}\times \frac{c\times a_{n+1}-1}{T/2} = c\times a_{n+1} = et_{n+1,j}-st_{n+1,j}$, which equals to its left side. Since $st_{n+1,j}-et_{n+1,j-1} = T/2-c\times a_{n+1} = G_{n+1}$, the Gap Constraint also holds. In the same period, $et_{k_{n_1},j} = st_{k_1,j}+\sum_{k=k_1,\ldots,k_{n_1}}(et_{k,j}-st_{k_j}) = st_{k_1,1}+(j-1)\times T+c\times \sum_{a_k \in A-A} a_k = (j-1)\times T+T/2-c\times a_{n+1} = st_{n+1,2j-1}$. In a similar way, $et_{k'_{n_2},j} = st_{n+1,2j}$. So, the packet trains of mote $n+1$ do not overlap with those of other motes and the Disjoint Constraint stands. Therefore, the schedule in Fig. 1 is feasible.

Step 3: Prove that if $f(\pi')$ has a solution, then the corresponding π' has a solution.

Assume that $f(\pi')$ has a schedule $\{(st_{k,j}, et_{k,j})\}$ that satisfies the three constraints in Definition 2. We need to construct a solution for the corresponding π'.

First, in the schedule $\{(st_{k,j}, et_{k,j})\}$, we can prove that there must exist a period T that satisfies:

- $\forall k \in [1, n]$, $\exists$ exactly one j, such that $(st_{k,j}, et_{k,j}) \subseteq T$ (abusing the denotation T a little bit) and

$$\begin{cases} et_{k,j} - st_{k,j} = c \times a_k; \\ st_{k,j} - et_{k,j-1} = G_k; \end{cases}$$

- For mote $n+1$, $\exists$ exactly one j, such that $(st_{n+1,j}, et_{n+1,j}) \subseteq T$, $(st_{n+1,j+1}, et_{n+1,j+1}) \subseteq T$ and

$$\begin{cases} et_{n+1,j} - st_{n+1,j} = et_{n+1,j+1} - st_{n+1,j+1} = c \times a_{n+1}; \\ st_{n+1,j} - et_{n_1,j-1} = st_{n+1,j+1} - et_{n_1,j} = G_{n+1}; \end{cases}$$

- $T = (et_{n+1,j-1}, et_{n+1,j+1})$. This can be proven by contradiction. But, due to space limitations, the detailed proof is not presented here.

Second, we construct a subset of motes $\{k'_1, \ldots, k'_{n_2}\}$ such that during time period T,

$$(st_{k,j}, et_{k,j}) \subseteq \begin{cases} (et_{n+1,j}, st_{n+1,j+1}), & k \in \{k'_1, \ldots, k'_{n_2}\} \\ (et_{n+1,j-1}, st_{n+1,j}), & k \in \{k_1, \ldots, k_{n_1}\} \end{cases}$$

$$\text{where } \{k_1, \ldots, k_{n_1}\} = \{1, \ldots, n\} - \{k'_1, \ldots, k'_{n_2}\}$$

With the Disjoint Constraint, we can derive

$$\sum_{k \in \{k'_1, \ldots k'_{n_2}\}} (et_{k,j} - st_{k,j}) = c \times \sum_{k \in \{k'_1, \ldots k'_{n_2}\}} a_k \le G_{n+1}$$

$$\sum_{k \in \{k_1, \ldots k_{n_1}\}} (et_{k,j} - st_{k,j}) = c \times \sum_{k \in \{k_1, \ldots k_{n_1}\}} a_k \le G_{n+1}$$

Since $\sum_{k \in [1,n]} (et_{k,j} - st_{k,j}) = 2 \times G_{n+1}$, we have $c \times \sum_{k \in \{k_1, \ldots k_{n_1}\}} a_k = c \times \sum_{k \in \{k'_1, \ldots k'_{n_2}\}} a_{k'}$. So, the partition problem π' has a solution $A' = \{a_{k'_1}, \ldots, a_{k'_{n_2}}\}$.

Therefore, with steps 1∼3, we prove Lemma 1, i.e., our BodyT2 Scheduling problem is NP-hard.

III. BodyT2 Design

Since the BodyT2 Scheduling problem for joint throughput and time delay assurance is NP-hard, it is nontrivial to obtain the optimal solution. In this section, we propose an empirical solution for practical system deployment. We present the necessary/sufficient conditions for admission control and also the algorithms for admission control and time resource scheduling. We also extend the existing VMAC [8] for enforcing the time resource scheduling result to meet the time delay performance requirements in addition to the throughput performance requirements.

A. Admission Control

The admission controller examines the performance assurance requests $\{(b_{k,i}, d_{k,i}, p_{k,i})\}$, $k \in [1, n]$ and makes ACCEPT/REJECT decisions. In time period T, the admission controller computes the total required time for satisfying all streams' requests when interference is captured and reflected by T_k. This includes both data and polling packets. The total number of data packets mote k needs to transmit is $D(k, T) = \lceil B_k \times T \rceil$ (B_k as defined in Section II-C). The total number of polling packets for mote k, defined as $P(k, T)$, equals the number of packet trains scheduled for that mote. In BodyQoS [8] which only provides throughput assurance, $P(k, T)$ is simply fixed as 1 for each T, but when the time delay assurance is jointly considered it is more difficult to determine. The total required time for both data and polling packets can be computed as $D(k, T)\times T_k + P(k, T)\times T_{maxPkt}$ which needs to be no more than the total available time T.

1) The Necessary and Sufficient Admission Conditions: If mote k is scheduled to send $P(k, T)$ packet trains during T, the sum of gaps between its packet trains plus the time for sending the $P(k, T)$ polling packets is $T-D(k, T)\times T_k$. Also, the gap between any two consecutive packet trains of mote k should be bounded by G_k (defined in Section II-C). So, $T-D(k, T)\times T_k \le P(k, T)\times (G_k+T_{maxPkt})$. When the gap decreases, the number of packet trains increases. Since G_k is the maximum gap allowed, the minimum number of packet trains is:

$$P_{min}(k, T) = \frac{T - D(k, T) \times T_k}{G_k + T_{maxPkt}}. \quad (1)$$

So, the minimum required time for sending data and polling packets for mote k is:

$$S_{min}(k, T) = D(k, T) \times T_k + P_{min}(k, T) \times T_{maxPkt}. \quad (2)$$

Therefore, the necessary condition of admission control is:

$$\sum_k S_{min}(k,1) \leq 1. \quad (3)$$

To derive a sufficient admission condition, assume a round-robin schedule in which all motes within T receive the same number of polling messages from the aggregator. The number of polling messages is estimated as the maximum value of $P_{min}(k,T)$ for all k. In this way, a sufficient condition for admission control can be derived as:

$$\sum_k (D(k,1) \times T_k + \max_k\{P_{min}(k,1)\} \times T_{maxPkt}) \leq 1. \quad (4)$$

Algorithm 1 Admission Control

Input: performance requests $\{(b_{k,i}, d_{k,i}, p_{k,i})\}$ for data stream $i \in$ N on mote $k \in [1..n]$, the average packet transmission time $\{T_k\}$ for mote k
Output: ACCEPT or REJECT decision
repeat
  if the necessary condition in Inequ. (3) is broken **then**
    REJECT and remove the request with the lowest $p_{k,i}$ from $\{(b_{k,i}, d_{k,i}, p_{k,i})\}$; continue;
  end if
  if the sufficient condition in Inequ. (4) stands **then**
    return ACCEPT;
  end if
  $t_c = 0$; $\forall$ remaining k, let $et_{k,j-1} = 0$ and $R_k = 0$;
  loop
    call Alg. 2 with input ($\{(b_{k,i}, d_{k,i}, p_{k,i})\}$, t_c, $\{et_{k,j-1}\}$, $\{R_k\}$) and get output ($(st_{k,j}, et_{k,j})$ or FAILURE);
    if Alg. 2 returns FAILURE **then**
      REJECT and remove the request with the lowest $p_{k,i}$ from $\{(b_{k,i}, d_{k,i}, p_{k,i})\}$; break;
    else
      $t_c = et_{k,j}$; $et_{k,j-1} = et_{k,j}$;
    end if
    if at least one packet train is allocated to each mote **then**
      return ACCEPT;
    end if
  end loop
until $\{(b_{k,i}, d_{k,i}, p_{k,i})\} = \emptyset$
return ACCEPT;

2) The Admission Control Algorithm: With the necessary and sufficient conditions, the admission controller can make preliminary decisions: if the necessary condition fails, a REJECT decision is made; if the sufficient condition holds, an ACCEPT decision is made; otherwise, if the sufficient condition fails but the necessary condition holds, it is hard to tell whether an appropriate schedule can be obtained for the requested data streams. As we have proven in Section II-C, this is actually an NP-hard problem. Therefore, we integrate an empirical solution into our admission control Alg. 1. With the help of Alg. 2 (to be explained later), Alg. 1 tries to make an appropriate schedule, i.e., determining the start and end time of packet trains for all motes to meet the joint throughput and time delay constraints. If a schedule is found, an ACCEPT decision is made; otherwise, a REJECT decision is made. When a REJECT decision is made, the data stream with the lowest priority is removed and the admission controller tries to make ACCEPT/REJECT decisions again with the remaining data streams. This process repeats until either an ACCEPT decision is made or all data streams are finally rejected and removed. The later case happens when interference is so strong that no packets can be timely delivered.

3) Algorithm for Scheduling the Next Packet Train: Alg. 2 presents details of scheduling the next packet train. It is used in both the admission control Alg. 1 and the time resource scheduling Alg. 3 that we will discuss later. In Alg. 2, we introduce R_k to denote the number of expected but unsent packets from mote k. So, by the end of a packet train $et_{k,j}$, even though the aggregator expects to receive $D(k, et_{k,j} - et_{k,j-1})$ ($D(k,t)$ as defined in Section III-A) packets from mote k based on the throughput requirement, it may actually receive $D(k, et_{k,j} - et_{k,j-1}) - R_k$ packets. A negative R_k value means that the aggregator receives more packets than expected from mote k, so it allocates less time for mote k's next packet train. When sensor data sampling and packet arrival are uniformly distributed, R_k provides flexibility to time resource scheduling. Since R_k is measured and can only have a nonzero value at runtime, R_k is set to zero in admission control. Jointly considering R_k and Def. 1. Length Constraint, we have:

$$et_{k,j} - st_{k,j} = (D(k, et_{k,j} - et_{k,j-1}) + R_k) \times T_k + T_{maxPkt}. \quad (5)$$

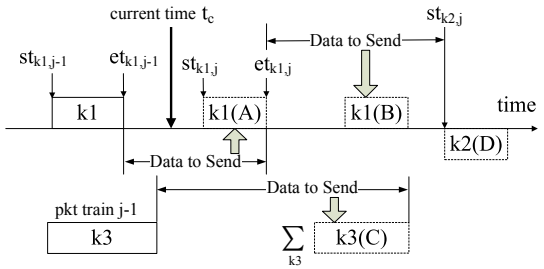


Fig. 2: Scheduling the Next Packet Train

Suppose the most recently scheduled packet train, say packet train $j-1$ for mote k, has the schedule of $(st_{k,j-1}, et_{k,j-1})$, then the latest start time of mote k's next packet train j should be $et_{k,j-1}+G_k$. In this algorithm, we try to schedule the next packet train j for the mote that has the minimum $et_{k,j-1}+G_k$ value, say mote k_1, which is similar to the earliest deadline first policy. An empirical rule we use here is: we give mote k_1's packet train j a schedule if and only if we can foresee that any other mote, say k_2 as in Alg. 2 and Fig. 2, can also have its packet train j scheduled.

As shown in Fig. 2, k_1 is the mote that has the earliest start time $st_{k,j} = et_{k,j-1}+G_k$. k_2 is another arbitrary mote that has a later start time $st_{k_2,j}$. k_3 is another arbitrary mote with its start time $st_{k_3,j}$ in between those of k_1 and k_2. Suppose k_3's most recent packet train schedule is $(st_{k_3,j-1}, et_{k_3,j-1})$. Then, during $(t_c, st_{k_2,j}]$, k_3 desires to send at least one packet train (C in Fig. 2). The total time that all such k_3 motes require is $\sum_{k_3} S_{min}(k_3, st_{k_2,j} - et_{k_3,j-1})$ which can be computed according to Eqn. (2). Also, during $(et_{k_1,j}, st_{k_2,j}]$, mote k_1 requires time $S_{min}(k_1, st_{k_2,j} - et_{k_1,j})$ to send packet train B. The time between packet trains A and D should be long enough to schedule packet trains B and C, that is,

$$\sum_{k_3}(S_{min}(k_3, st_{k_2,j} - et_{k_3,j-1}) + R_{k_3} \times T_{k_3}) + S_{min}(k_1, st_{k_2,j} - et_{k_1,j}) + R_{k_1} \times T_{k_1} \leq st_{k_2,j-1} - et_{k_1,j}. \quad (6)$$

Here, $st_{k_2,j} = et_{k_2,j-1} + G_k$ which is the latest possible start time of mote k_2's next packet train j.

With Inequ. (7), we make sure that there is enough room to schedule packet train A. Also, with Inequ. (8), we make sure that the distance between packet train A and mote k_1's previous packet train $j-1$ is bounded by G_{k_1}.

$$(D(k_1, et_{k_1,j} - et_{k_1,j-1}) + R_{k_1}) \times T_{k_1} + T_{maxPkt} \leq et_{k_1,j} - t_c \quad (7)$$

$$et_{k_1,j} - et_{k_1,j-1} - (D(k_1, et_{k_1,j} - et_{k_1,j-1}) + R_{k_1}) \times T_{k_1} - T_{maxPkt} \leq G_{k_1} \quad (8)$$

Finally, $et_{k_1,j}$ is computed as the largest value that satisfies Inequ. (6)$\sim$ (8) and $st_{k_1,j}$ is computed with Eqn. (5).

Algorithm 2 Scheduling the Next Packet Train

Input: performance requirements $\{(b_{k,i}, d_{k,i}, p_{k,i})\}$, the current time t_c, the end time of the most recently scheduled packet trains for all motes $\{et_{k,j-1}\}$, $\{R_k\}$
Output: the next packet train schedule $(st_{k,j}, et_{k,j})$ or (FAILURE)
$\forall k$, compute the G_k value based on its definition in Section II-C
get $\min_k\{et_{k,j-1} + G_k\}$ and assume it is $et_{k_1,j-1} + G_{k_1}$
for any k_2 ($k_2 \neq k_1$) **do**
  /*Check if the period $(st_{k_1,j}, st_{k_2,j}]$ is long enough for packet trains of all other motes (say k_3 as an arbitrary one)*/
  for any k_3 ($st_{k_1,j} \leq st_{k_3,j} \leq st_{k_2,j}$, $k_3 \neq k_1$, $k_3 \neq k_2$) **do**
    With Eqn. (2), estimate $S_{min}(k_3, st_{k_2,j} - et_{k_3,j-1})$ which is the time that mote k_3 needs in $(t_c, st_{k_2,j}]$
  end for
  compute $\sum_{k_3} S_{min}(k_3, st_{k_2,j} - et_{k_3,j-1})$
  estimate the largest $et_{k_1,j}$ that satisfies Inequ. (6), (7), and (8)
  if $\nexists$ such $et_{k_1,j}$ **then**
    return (FAILURE)
  end if
end for
$et_{k_1,j}$ = the minimum $et_{k_1,j}$ value computed above for all k_3
compute $st_{k_1,j}$ with Eqn. (5)
return $(st_{k_1,j}, et_{k_1,j})$

B. Time Resource Scheduling

In time resource scheduling, the aggregator sequentially computes the time allocated to each packet train. More specifically, the time resource scheduling Alg. 3 calls Alg. 2 to compute a schedule $(st_{k,j}, et_{k,j})$ for the next packet train as well as a schedule $(st_{k',j}, et_{k',j})$ for the packet train after the next. BodyT2 communication supports two kinds of data: the QoS data that requires throughput and time delay guarantee, and the best effort data that does not. If enough time ($\geq 2 \times T_{maxPkt}$) is available before starting the next packet train, VMAC is called to poll for best effort data. Then, when time proceeds to $st_{k,j}$, VMAC is called to poll mote k to enforce schedule $(st_{k,j}, et_{k,j})$. The time resource scheduling waits while mote k transmits QoS data packets. The execution of current schedule ends when either an early termination of this packet train is received from mote k due to lack of data or the time proceeds to $st_{k',j}$. After that, parameters T_k and R_k are updated to assist scheduling the next packet train while the process repeats.

Algorithm 3 Time Resource Scheduling

Input: performance requirements $\{(b_{k,i}, d_{k,i}, p_{k,i})\}$, $\{R_k\}$
Output: function calls to VMAC
$\forall k$, $et_{k,j-1} = 0$; $R_k = 0$
loop
  call Alg. 2 with input ($\{(b_{k,i}, d_{k,i}, p_{k,i})\}$, t_c = the current time, $\{et_{k,j-1}\}$, $\{R_k\}$) and get output ($(st_{k,j}, et_{k,j})$ or FAILURE)
  if Alg. 2 returns FAILURE **then**
    /* this only happens when the interference level largely increases after the admission control*/
    execute the admission control Alg. 1 again to remove low priority streams;continue;
  end if
  $et_{k,j-1} = et_{k,j}$; $R_k = 0$;
  if $st_{k,j} \geq$ the current time $+ 2 \times T_{maxPkt}$ **then**
    call VMAC to poll for best effort data
  end if
  wait until the time proceeds to $st_{k,j}$;
  call Alg. 2 with input ($\{(b_{k,i}, d_{k,i}, p_{k,i})\}$, $t_c = et_{k,j}$, $\{et_{k,j-1}\}$, $\{R_k\}$) and get output ($(st_{k',j}, et_{k',j})$ or FAILURE);
  if Alg. 2 returns FAILURE **then**
    for the same reason above, execute the admission control Alg. 1 again to remove low priority streams;continue;
  end if
  call VMAC to poll mote k for QoS data;
  wait until the time proceeds to $st_{k',j}$ or mote k terminates the packet train early; then, update the values of $et_{k,j}$, T_k, and R_k with runtime measurements and let $et_{k,j-1} = et_{k,j}$;
end loop

C. Enforcing Time Schedule on VMAC

VMAC is located on both the aggregator and motes for enforcing the time resource scheduling result computed by Alg. 3. We extend the existing VMAC [8] to enforce the newly added time delay requirement in additional to the throughput requirement. The extended VMAC not only checks the remaining allocated time but also the specified time delay constraint for each packet transmission. It also notifies the aggregator to terminate the packet train if there is no packet to send.

On the aggregator, VMAC receives calls from the above scheduler and calls the underlying real MAC functions. For a packet train schedule $(st_{k,j}, et_{k,j})$, VMAC sends a polling message to mote k with the allocated time length $PL_{k,j} = et_{k,j} - st_{k,j} - T_{maxPkt} + (st_{k',j} - et_{k,j})$. Here, $st_{k',j} - et_{k,j}$ is the gap between mote k's packet train j and mote k''s packet train j. Since this gap immediately follows the scheduled time period $et_{k,j} - st_{k,j} - T_{maxPkt}$ and is also not scheduled to any other packet train, it is allocated to extend the length of packet train j for mote k. When VMAC is called to poll for best effort data before a packet train schedule $(st_{k,j}, et_{k,j})$, it broadcasts a message, indicating that the following time period ($st_{k,j} -$ current time $- T_{maxPkt}$) is open for all motes' best effort communication. During this period, potential collision resolution among different motes' transmissions is handled by the underlying specific MAC protocols.

When a mote, say mote k, receives a polling message, VMAC enforces the time resource scheduling result by feeding QoS or best effort data to the aggregator within the allocated time periods. When polled for QoS packets with length $PL_{k,j}$ (computed in the previous paragraph), VMAC on mote k computes the amount of data that each stream i on mote k requests to send since the end of mote k's previous packet train. Then, VMAC organizes the data into a packet train in which the packets with earlier deadlines, including those for retransmissions, are put ahead of those with later deadlines. Before sending each data packet, VMAC conducts the following checks:

- If the remaining allocated time is less than T_{maxPkt}, VMAC does not send the data packet and the packet train terminates. This ensures that the control of the underlying radio is returned to the upper layers before the allocated time expires. Again, it is worthy to repeat that in most cases it takes less time than T_{maxPkt} to deliver this data packet. However, VMAC is able to salvage the unused time of this data packet to send the next data packet.
- If the deadline of the data packet is earlier than the current time plus T_k, it is immediately dropped since we may otherwise waste time on a packet that finally misses its deadline.
- If the current data packet is the only QoS data packet remaining in the mote, VMAC sets the $NoMoreData$ bit in the replied packet's header which informs the aggregator of the early termination of the packet train.

IV. Performance Evaluation

BodyT2 is implemented in TinyOS 2.x with NesC, and evaluated through both TelosB mote lab tests and real body experiments in an Android phone-centric BSN. BodyT2 is compared with the state-of-the-art BodyQoS [8] as well as the default best effort solution in the standard TinyOS 2.x release. Three performance metrics are used: (i) the percentage of delivered throughput, i.e., the timely delivered data throughput over the requested data throughput; (ii) the data packet deadline miss ratio, which is computed as the number of data packets that miss their deadlines divided by the number of data packets requested to be sent from motes; and (iii) the average energy consumed to timely deliver one application data byte to the aggregator. Detailed evaluation settings are given below:

TelosB mote lab tests. A data stream with performance requirement (5kbps throughput, 200ms time delay) is admitted into BodyT2 to report data from source to the aggregator in the lab experiments. Besides the existing interference from the lab environment like WiFi and Zigbee [9], a TelosB node is also introduced to generate explicit interference (see Tab. I).

Real body experiments in an Android phone-centric BSN. We also develop an Android phone-centric BSN to demonstrate the effectiveness and efficiency of BodyT2 and present the prototype BSN in Fig. 3. The aggregator of the BSN is zoomed to Fig. 4 in which one TelosB is plugged in the USB hub to directly communicate with the Android phone. Multiple sensor motes can also be plugged in the USB hub and operate on different frequencies for improving the aggregator throughput. Additional sensor motes can be attached on the human body and wirelessly communicates to the aggregator. Our main technical contributions for developing such a BSN lies in four aspects: Android OS kernel support, hardware support, TinyoS support, and application support. Due to space limitations, more technical details are not presented here but available in our technical report [28].

TABLE I: Interference Settings

Interference Level	Interference Strength	Interference Period
Level 0	Lab background noise	0s∼120s
Level 1	Lab background noise + 1 noise packet every 30ms	120s∼180s
Level 2	Lab background noise + 1 noise packet every 25ms	180s∼240s
Level 3	Lab background noise + 1 noise packet every 20ms	240s∼300s

Fig. 3: A Phone-centric BSN

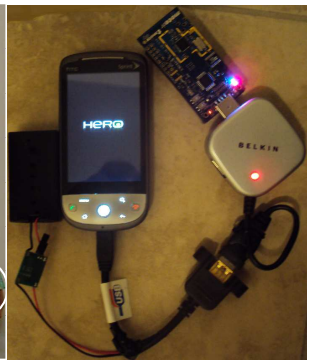


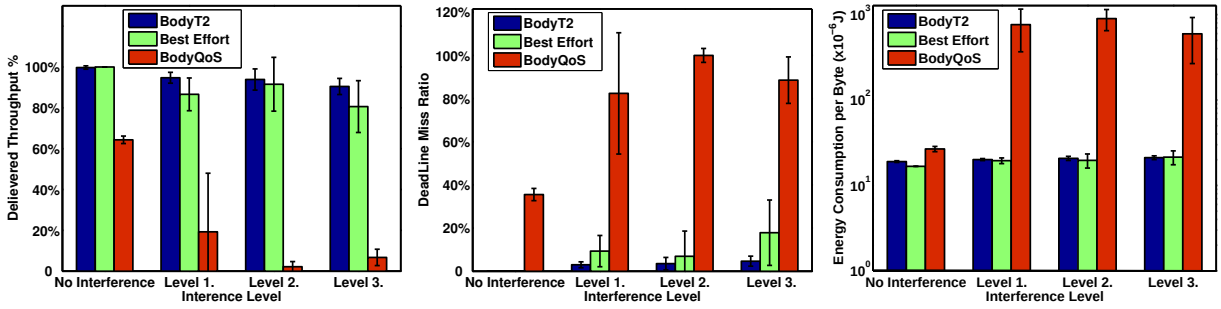
Fig. 4: The Aggregator

In our real body experiments, TelosB devices are attached to a human body as shown in Fig. 3: a TelosB is attached to the left chest that generates a data stream with the performance requirement (4kbps throughput, 500ms time delay) and requests BodyT2 service; a TelosB is attached to the left wrist that generates a data stream with the performance requirement (2kbps throughput, 1000ms time delay bound) and also requests BodyT2 service; a TelosB mote is attached slightly above the right hip that generates a data stream with the performance requirement (4kbps bandwidth, 500ms time delay) but requests best effort service; the same aggregator as shown in Fig. 4 is put inside the bottom left pocket of the jacket for data collection and analysis.

All experiments described above are repeated multiple times and similar results are observed. In the following subsections, we present two groups of representative results which demonstrate that BodyT2 largely outperforms the existing BodyQoS and best effort solutions.

A. Performance Results of TelosB Mote Lab Tests

Fig. 5 (a) plots the mean and standard deviation of the percentage of timely delivered throughput when different interference levels are present in the lab experiment. We first observe that BodyT2 achieves a higher timely delivered throughput ratio than those of best effort and BodyQoS. In fact, BodyT2 achieves up to 10% higher throughput ratio than best effort and 91% higher throughput ratio than BodyQoS. Second,



(a) Delivered Throughput% (b) Deadline Miss Ratio (c) Energy Consumption Per Delivered Byte

Fig. 5: Performance Comparison of BodyT2 with BodyQoS and Best Effort Through TelosB Mote Lab Tests

we observe that BodyT2 achieves a more stable throughput delivery ratio than those of best effort and BodyQoS. As shown in the figure, the largest standard deviation for BodyT2 is 5.2% under interference level 2, while best effort has the largest standard deviation of 13.2% under interference level 2 and BodyQoS has the largest standard deviation 28.7% under interference level 1. Third, we observe that the performance gain of BodyT2 over best effort and BodyQoS increases when interference increases. For instance, throughout the 4 interference periods, BodyT2 has a less obvious decrease of the throughput delivery ratio than those of best effort and BodyQoS. BodyT2 achieves superior performance than existing approaches because its design addresses the joint throughput and time delay requirements, while the existing approaches do not. We are also aware that BodyQoS performs much better than best effort in [8] when only the throughput requirement is considered, but it performs worse than best effort when the time delay requirement is jointly considered here. This is because BodyQoS is not designed to address the time delay requirement and hence data packets can be held too long to be timely delivered. Due to uncertainty of the lab background noise, the interference intensity may fluctuate with time. So, packets that were scheduled to be sent out but actually unsent in the previous time period, when the interference is comparatively strong, may be able to be sent out in the current time period, when the interference is comparatively weak, to fullfill the throughput requirement. This is why sometimes the percentage of delivered throughput exceeds 100%.

Fig. 5 (b) presents the data packet deadline miss ratio. First, we see that BodyT2 achieves an extremely low deadline miss ratio ($< 5\%$) under all 4 interference levels, while best effort has 17.9% packets missing deadlines under interference level 3 and BodyQoS misses all deadlines under interference level 2. Second, we see that the deadline miss ratios for best effort and BodyQoS largely increase when interference increases. For example, best effort's deadline miss ratio raises 11% from interference level 2 to 3. Meanwhile, BodyT2's deadline miss ratio remains almost constantly low. For similar reasons, BodyQoS performs the worst among the three. BodyQoS misses all deadlines under interference level 2 but has nonzero throughput delivery ratio under interference level 2, because data packets not delivered in the previous time period, i.e., under interference level 1, are sent out here.

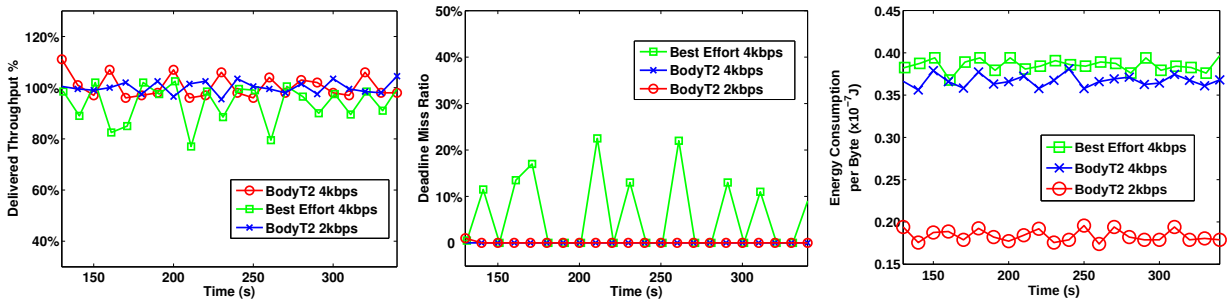
Fig. 5 (C) shows the energy consumption per timely delivered application data byte, measured in Joules (J). As the number of timely delivered data byte for BodyQoS drops to zero, we may have division by zero. So, we assign a very large energy consumption value $1\times10^{-3}J$ in such cases. Since the y-axis value for BodyQoS is much larger than that of BodyT2 and best effort, we plot the y-axis with a log scale. From Fig. 5 (C), we observe that BodyT2 uses similar energy as that of best effort. We also observe that when interference increases BodyT2's energy consumption per timely delivered data byte remains stable, but best effort's energy consumption per timely delivered data byte fluctuates and becomes less stable. This is because fewer data bytes are timely delivered in best effort than BodyT2 when interference increases even though best effort dose not waste more energy retransmitting packets that finally miss deadlines.

B. Performance Results of Real Body Experiments in an Android Phone-centric BSN

Fig. 6 (a) plots the the timely throughput delivery ratio. We observe that both BodyT2 data streams on average maintain $\sim 100\%$ timely throughput delivery ratio. However, the best effort data stream on average has $< 100\%$ timely throughput delivery ratio which also fluctuates significantly. For example, the best effort stream achieves only 77% ratio at $210s$ and 79.5% ratio at $260s$, but BodyT2 data streams' ratios never go below 95.5%. This demonstrates BodyT2's effectiveness and best effort's ineffectiveness in supporting multiple data streams' throughput and time delay performance requirements. Here, for the same reason as we have presented when explaining Fig. 5 (a), we also observe that the percentage of delivered throughput fluctuates above and below the 100% line.

Fig. 6 (b) depicts the data packet deadline miss ratio. We observe a near zero deadline miss ratio for both BodyT2 data streams but up to 22% deadline miss ratio for the best effort data stream. This is because on the one hand, best effort uses the resources remaining after QoS resource scheduling, and on the other hand, the best effort approach does not consider deadline when scheduling resources.

Fig. 6 (c) shows the energy consumption per timely delivered application data byte. We observe that while both the BodyT2 data streams and the best effort data stream



(a) Delivered Bandwidth% (b) Deadline Miss Ratio (c) Energy Consumption Per Byte

Fig. 6: BodyT2 Performance Evaluation Through Real Body Experiments in an Android Phone-centric BSN

have similar energy efficiency on average, the energy efficiency fluctuation of the best effort data stream is much higher than that of the BodyT2 data streams. The maximum energy consumption per timely delivered data byte on the two BodyT2 data streams are $1.88\times10^{-5}J$ and $1.92\times10^{-5}J$, respectively. But the maximum value of the best effort data stream is $2.24\times10^{-5}J$, which is $17\% \sim 20\%$ higher than that of BodyT2. This is because the group-polling scheme and also adaptive resource scheduling in BodyT2 can absorb and tolerant fluctuations of link qualities but best effort can not.

V. Conclusions

Joint throughput and time delay performance assurance is critical for many BSN applications. This paper proposes a novel approach to provide this joint assurance in a radio-agnostic manner. Our approach is based on a group-polling scheme that is essential for radio-agnostic BSN design. We rigorously prove that with the group-polling scheme resource scheduling for the throughput performance assurance is P, while the joint throughput and time delay assurance is NP-hard. For practical system deployment, we propose the BodyT2 framework that assures throughput and time delay performance in a heterogeneous BSN. Through both TelosB mote lab tests and real body experiments in an Android phone-centric BSN, we demonstrate that BodyT2 achieves superior performance over existing solutions.

References

[1] 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.

[2] 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.

[3] T. Gao, C. Pesto, L. Selavo, Y. Chen, J. Ko, J. Lim, A. Terzis, A. Watt, J. Jeng, B. r. Chen, K. Lorincz, and M. Welsh, "Wireless Medical Sensor Networks in Emergency Response: Implementation and Pilot Results," in *IEEE HST*, 2008.

[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] G. Zhou, T. He, S. Krishnamurthy, and J. A. Stankovic, "Models and Solutions for Radio Irregularity in Wireless Sensor Networks," in *ACM TOSN*, 2006.

[7] A. Natarajan, B. D. Silva, K. K. Yap, and M. Motani, "Link Layer Behavior of Body Area Networks at 2.4 GHz," in *ACM MobiCom*, 2009.

[8] 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.

[9] G. Zhou, J. A. Stankovic, and S. H. Son, "Crowded Spectrum in Wireless Sensor Networks," in *IEEE EmNets*, 2006.

[10] R. C. Shah, L. Nachman, and C.-Y. Wan, "On the Performance of Bluetooth and IEEE 802.15.4 Radios in a Body Area Network," in *BodyNets*, 2008.

[11] A. Rowe, R. Mangharam, and R. Rajkumar, "RT-Link: A Time-Synchronized Link Protocol for Energy-Constrained Multi-hop Wireless Networks," in *IEEE SECON*, 2006.

[12] A. Eswaran, A. Rowe, and R. Rajkumar, "Nano-RK: An Energy-aware Resource-centric RTOS for Sensor Networks," in *IEEE RTSS*, 2005.

[13] M. Barry, A. T. Campbell, and A. Veres, "Distributed Control Algorithms for Service Differentiation in Wireless Packet Networks," in *IEEE INFOCOM*, 2001.

[14] Y. Yang and R. Kravets, "Achieving Delay Guarantees in Ad Hoc Networks Through Dynamic Contention Window Adaptation," in *IEEE INFOCOM*, 2006.

[15] C. Hu, H. Kim, J. C. Hou, D. Chi, and S. Shankar N, "Provisioning Quality Controlled Medium Access in UltraWideBand WPANs," in *IEEE INFOCOM*, 2006.

[16] 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 *ACM/IEEE IPSN*, 2008.

[17] V. Namboodiri and A. Keshavarzian, "Alert: An Adaptive Low-latency Event-driven MAC Protocol for Wireless Sensor Networks," in *ACM/IEEE IPSN*, 2008.

[18] K. Lorincz, B. r. Chen, J. Waterman, G. Werner-Allen, and M. Welsh, "Pixie: An Operating System for Resource-aware Programming of Embedded Sensors," in *IEEE EmNets*, 2008.

[19] D. Narayanan and M. Satyanarayanan, "Predictive Resource Management for Wearable Computing," in *MobiSys*, 2003.

[20] M. Verloop and S. Borst, "Heavy-Traffic Delay Minimization in Bandwidth-Sharing Networks," in *IEEE INFOCOM*, 2007.

[21] L. Gu and J. A. Stankovic, "$t-kernel$: Providing Reliable OS Support to Wireless Sensor Networks," in *ACM SenSys*, 2006.

[22] M. J. Neely, "Delay Analysis for Maximal Scheduling in Wireless Networks with Bursty Traffic," in *IEEE INFOCOM*, 2008.

[23] V. Raghunathan, V. Borkar, M. Cao, and P. R. Kumar, "Index Policies for Real-time Multicast Scheduling for Wireless Broadcast Systems," in *IEEE INFOCOM*, 2008.

[24] T. F. Abdelzaher, S. Prabh, and R. Kiran, "On Real-time Capacity Limits of Multihop Wireless Sensor Networks," in *IEEE RTSS*, 2004.

[25] C. Lu, B. M. Blum, T. F. Abdelzaher, J. A. Stankovic, and T. He, "Rap: A Real-time Communication Architecture for Large-scale Wireless Sensor Networks," in *RTAS*, 2002.

[26] T. He, J. A. Stankovic, C. Lu, and T. F. Abdelzaher, "SPEED: A Stateless Protocol for Real-Time Communication in Sensor Networks," in *IEEE ICDCS*, 2003.

[27] I-H. Hou and P. R. Kumar, "Admission Control and Scheduling for QoS Guarantees for Variable-bit-rate Applications on Wireless Channels," in *ACM MobiHoc*, 2009.

[28] A. Pyles and G. Zhou, "Implementing USB Host Mode on the Android Platform," in *College of William and Mary Technical Report WM-CS-2010-06*, 2010.