# BodyQoS: Adaptive and Radio-Agnostic QoS for Body Sensor Networks

Gang Zhou, Jian Lu[†], Chieh-Yih Wan[‡], Mark D. Yarvis[‡], and John A. Stankovic[†]
Computer Science Department, College of William and Mary
*gzhou@cs.wm.edu*
[†]Computer Science Department, University of Virginia
[†]{*jl3aq,stankovic*}*@cs.virginia.edu*
[‡]Intel Corporation
[‡]{*chieh-yih.wan,mark.d.yarvis*}*@intel.com*

***Abstract*—As wireless devices and sensors are increasingly deployed on people, researchers have begun to focus on wireless body-area networks. Applications of wireless body sensor networks include healthcare, entertainment, and personal assistance, in which sensors collect physiological and activity data from people and their environments. In these body sensor networks, quality of service is needed to provide reliable data communication over prioritized data streams. This paper proposes BodyQoS, the first running QoS system demonstrated on an emulated body sensor network. BodyQoS adopts an asymmetric architecture, in which most processing is done on a resource rich aggregator, minimizing the load on resource limited sensor nodes. A virtual MAC is developed in BodyQoS to make it radio-agnostic, allowing a BodyQoS to schedule wireless resources without knowing the implementation details of the underlying MAC protocols. Another unique property of BodyQoS is its ability to provide adaptive resource scheduling. When the effective bandwidth of the channel degrades due to RF interference or body fading effect, BodyQoS adaptively schedules remaining bandwidth to meet QoS requirements. We have implemented BodyQoS in NesC on top of TinyOS, and evaluated its performance on MicaZ devices. Our system performance study shows that BodyQoS delivers significantly improved performance over conventional solutions in combating channel impairment.**

## I. Introduction

In the last few years commercial sensor network applications have emerged that require pervasive sensing of people and the environment, e.g., assisted living [1] [2] [3] and smart homes [4] [5] [6]. For these applications, it is essential to be able to reliably collect physiological readings from humans via Body Sensor Networks (BSN). Such networks could benefit from Quality of Service (QoS) mechanisms that support prioritized data streams, especially when the channel is impaired by interference or fading. For example, heart activity readings (e.g., EKG waveforms) are often considered more important than body temperature readings, and hence can be assigned a higher priority in the system. In another example, a glucose data stream might be assigned a low priority when the readings are in normal range, but a higher priority might be re-assigned to the data stream by user applications when readings indicate hyper- or hypo-glycemia. QoS support is needed to ensure reliable data collection for high priority data streams and to dynamically reallocate bandwidth as conditions change, especially when the effective channel bandwidth is reduced by interference or fading.

QoS research is not new; prior research has focused on managing and reserving resources [7] [8] [9] [10] in the Internet, wireless networks, and ad hoc networks. However, for body sensor networks, three new QoS challenges arise: (1) A body sensor network consists of two levels of devices: multiple simple sensor nodes and a more powerful aggregator. A sensor node can be very resource constrained. For example, a sweat sensor can be designed as a Band-Aid, which uses a film battery and is attached to the wrist to obtain sweat levels. In contrast, an aggregator is comparatively more powerful. An aggregator might be a cell phone or an electronic watch that includes a 32-bit processor and rechargeable batteries. This asymmetric, star-shaped architecture requires an asymmetric QoS solution, not typical of prior QoS research. (2) Existing medical sensor devices often use different radio technologies, such as CC1000 [11], IEEE 802.15.4 [12], or Bluetooth [13]. There is a need for a radio-agnostic QoS solution, which can be easily ported from one radio platform to another. (3) Low power radios are susceptible to channel fading and interference. This issue is especially evident in BSN due to human body factor. A number of studies [14] [15] [16] [17] showed that the human body presents various adverse fading effects to wireless communication channels that are dependent on body size and posture. As a result, the communication channel characteristics in a BSN are highly variable and difficult to predict. Furthermore, when co-existing networks or RF emitting devices such as microwaves are present, packets may be lost due to interference, reducing the effective channel bandwidth [18]. In both cases, the QoS software needs to re-allocate resources from lower priority streams to higher priority streams. Adaptive QoS scheduling is thus needed to provide statistical bandwidth guarantees, which are essential for reliable data collection in body sensor networks.

Driven by these new challenges, we propose BodyQoS, an adaptive and radio-agnostic QoS solution for body sensor networks. The main contributions of this work are:

* Other names and brands may be claimed as the property of others.

- *Prioritized Data Stream Service:* Through well-defined interfaces, applications submit their QoS requirements for each stream, assigning a specific priority according to the data type and level of criticality. If the available wireless resource is not sufficient to serve all QoS reservations, the higher priority streams are served first.
- *Asymmetric QoS framework:* Most admission control functions and resource scheduling computations in BodyQoS are managed by the aggregator, while little processing is required at sensor nodes.
- *Radio Agnostic QoS:* A virtual MAC is implemented in BodyQoS to represent and schedule channel resources. It separates the QoS scheduler from the underlying MAC implementation. The virtual MAC design allows the QoS system to be ported from one radio platform to another (less than 100 lines of modified code, in one instance).
- *Adaptive Bandwidth Scheduling:* Besides conventional RSVP-Light [19] QoS scheduling, we also implement and evaluate adaptive QoS scheduling in BodyQoS that provides statistical bandwidth guarantees.
- *Testbed Implementation:* BodyQoS has been implemented in TinyOS [20] and evaluated on the MicaZ [21] platform. Our evaluation shows that BodyQoS exhibits improved performance compared to conventional solutions.

The rest of the paper is organized as follows. In section II we present an overview of the BodyQoS architecture. We then explain design details of the BodyQoS components: the Virtual MAC in section III, the QoS scheduler in section IV and admission control in section V. The performance of a TinyOS implementation of BodyQoS is evaluated in section VI. Section VII describes related work. Finally, in section VIII, we present conclusions and future work.

## II. BodyQoS Overview

A BSN is a collection of small, low power sensing devices (such as EKG, pulse oximeter, temperature, sweat) wirelessly connected to a resource-rich aggregation device (such as a watch or cell phone), all within one communication hop of each other. BodyQoS is a set of software modules that sits between the transport and the MAC layers. BodyQoS receives QoS and data transmission requests from the transport layer and uses the underlying MAC protocol to transmit data. BodyQoS consists of three components: Admission Control, QoS Scheduler and Virtual MAC (VMAC). BodyQoS adopts an asymmetric architecture. The Admission Control and Scheduler components are implemented as a master and slave module on the aggregator and sensor nodes respectively. As shown in Figure 1, the slave admission control and slave QoS scheduler execute on the sensor motes, while the main part of admission control and the QoS scheduler execute on the aggregator.

The admission control module has two main functions. The first function is to accept or reject new QoS reservations. The decision is based on the requested bandwidth, time delay requirement, and priority, as well as the available wireless resources, previously accepted reservations, and the effective channel bandwidth. The second function is to continuously monitor and estimate the effective bandwidth, which varies according to interferences in the environment and body fading effect. QoS reservations are dynamically adjusted as necessary.

The main function of the QoS scheduler is to control use of the channel such that reservation requirements are attained. It schedules wireless resource for all admitted streams and provides admission control signaling between sensor nodes and the aggregator. The scheduler also allocates unused wireless resources for best-effort communications. The QoS scheduler is also responsible for measuring the effective channel bandwidth at runtime and for reporting it to the admission control module. Finally, the QoS scheduler can schedule a sensor node to sleep if it predicts a lull in the communication schedule.

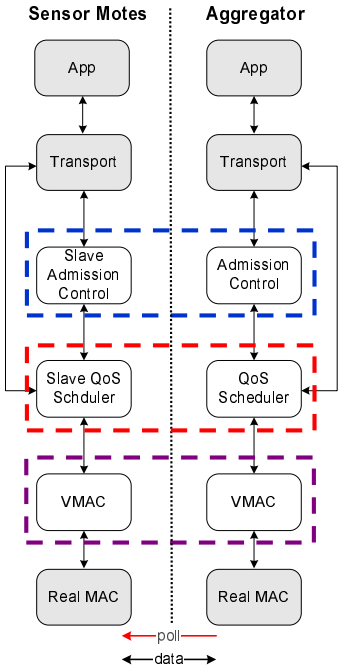



Fig. 1. BodyQoS Architecture

The VMAC isolates BodyQoS from the details of a specific MAC protocol. The VMAC design is challenging because it must control and reserve the wireless resource to support QoS scheduling, while abstracting away the details of the underlying MAC, including both CSMA-based and TDMA-based MACs. To achieve this goal, the VMAC exposes an abstract representation of wireless resource. It also provides an interface for the QoS scheduler to schedule wireless resource.

As illustrated in Figure 1, both admission control and the QoS scheduler consist of two parts that are distributed across sensor nodes and the aggregator. The aggregator makes admission control decisions, computes the communication schedule, and polls individual sensor nodes for data. Use of polling simplifies resource scheduling and supports an efficient asymmetric QoS design. Polling also simplifies coordination among multiple aggregators, allowing co-existing body sensor networks (future work).

## III. VMAC Design

Today, several different radio platforms are available for sensor devices, e.g., the CC1000 [11], the CC2420 [22] or Bluetooth [13]. In some cases, multiple radio platforms may exist within a single system. To prevent QoS scheduling from being tied to a specific radio implementation, a virtual MAC design is required.

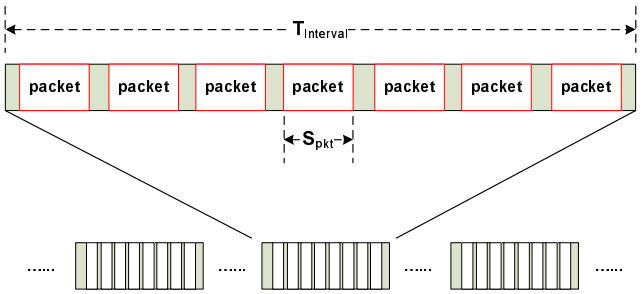



Fig. 2. VMAC Abstraction

The VMAC design is particularly challenging. We first identify three MAC features that are essential for QoS support and commonly available in most MAC protocols. First, a MAC should be able to send a packet when requested. It should return control of the radio back to the scheduler when the packet is either transmitted successfully or fails after exceeding the maximum number of backoffs or retransmissions. As a result, there is a maximum time period for a MAC to handle a packet transmission request, after which the MAC should return control back to the QoS scheduler. Second, when a packet is received, the MAC should notify the QoS scheduler with the received packet. Third, it should track the time and energy overhead for transitioning between sleeping/waking states, to allow efficient sleep scheduling for sensor nodes.

Figure 2 presents our VMAC abstraction for representing and scheduling wireless resources. Time in VMAC is divided into intervals. Within each interval, VMAC is able to send out a certain number of packets with the specified data payload length, each within a specified time period. The following parameters are essential for the VMAC abstraction, and their values should be configured according to measurements of underlying MAC implementations.

- $T_{interval}$: the length of each interval in seconds.
- $N_{pkt}$: the maximum number of packets that the QoS scheduler can transmit or receive within each time internal, assuming a clean channel.
- $S_{pkt}$: the effective data payload size in bytes.
- $T_{minPkt}$: the minimum time duration for the MAC to transmit a packet assuming a clean channel, i.e., the minimum MAC response time for handling a packet transmission request, after which the MAC returns radio control to the QoS scheduler.
- $T_{maxPkt}$: the maximum time duration for the underlying MAC to transmit a packet or report giving up after exceeding the maximum backoffs or number of retransmissions. This is the maximum MAC response time for handling a packet transmission request, after which the MAC returns radio control to the QoS scheduler.
- $T_{minSleep}$: the minimum sleep duration for radio duty cycling, taking into account the delay to transition the radio between sleep and normal mode, as well as a user specified value to account for other energy costs.

The above parameters define a virtual MAC that the QoS scheduler can directly use for scheduling wireless resource. Setting these parameter values, however, is not trivial and is MAC dependent[1]. For example, $S_{pkt}$ is the actual data payload length used in the underlying MAC. Likewise, $T_{maxPkt}$ needs to account for both the worst-case backoff time and the maximum number of MAC-layer retransmissions. One could obtain these values either analytically or experimentally. Once the above parameters are configured, VMAC is able to represent and reserve the wireless resource regardless of the underlying MAC implementation.

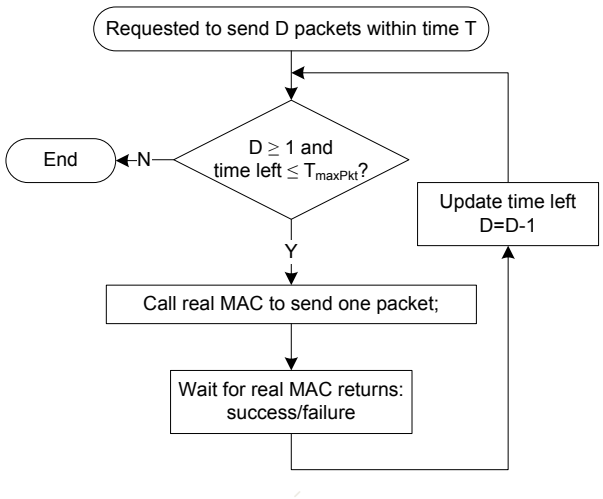



Fig. 3. Packet Transmission in VMAC

Figure 3 presents the packet transmission process in VMAC. In each interval, the VMAC attempts to transmit $D$ packets iteratively within the time period $T$, until either all packets have been sent or the remaining time is less than $T_{maxPkt}$.

VMAC design is generic, refraining from functions that are only available in a specific MAC. For example, overhearing is possible in CC1000 and IEEE 802.15.4, but not in Bluetooth. This design choice allows the QoS implementation to be ported from one radio platform to another with minimum code modification. Researchers from U.C. Berkeley have developed a virtual MAC [23], in which priorities and reliability can be configured for a set of underlying MAC protocols. However, this solution is designed for general use, without specific features for QoS reservation. Specifically, there is no support for bandwidth specification or reservation in their solution. In our BodyQoS design, as we will discuss later in Section IV and V, users' high level bandwidth specification can be translated into low level VMAC parameter configuration, which as shown in Figure 3 can be enforced regardless of the underlying real MAC implementation.

## IV. QoS Scheduler Design

The QoS scheduler is the core of BodyQoS. We first describe the basic scheduling framework that BodyQoS uses to reserve the channel for different types of data traffic. Then we

[1]Notice that the MAC dependency is encapsulated entirely in the VMAC.

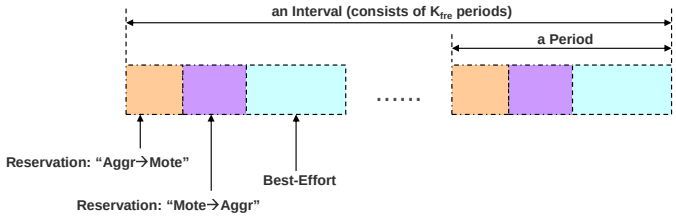



Fig. 4. QoS Traffic Scheduling

describe a mechanism to measure the effective bandwidth in a radio-agnostic manner. Finally, we present two QoS scheduling algorithms: RSVP-Light [19] QoS scheduling and adaptive QoS scheduling. We also consider the impact of different delay requirements on scheduling.

### A. Basic Scheduling Framework

There are three types of traffic in our system: (1) reserved aggregator→mote communication, (2) reserved mote→aggregator communication, and (3) best-effort communication. Each QoS reservation has a delay requirement that is either high or low sensitivity. A reservation that has low delay sensitivity is only scheduled once for communication within an interval. For a reservation with high delay sensitivity, a data communication slot is scheduled $K_{maxFre}$ times within each interval, where $K_{maxFre}$ is a system-wide parameter[2].

The average scheduling delay for a low delay sensitivity reservation is $\frac{T_{interval}}{2}$, and the worst case scheduling delay is $T_{interval}$. For a high delay sensitivity reservation, the average scheduling delay is $\frac{T_{interval}}{2\times K_{maxFre}}$, and the worst case scheduling delay is $\frac{T_{interval}}{K_{maxFre}}$. Depending on delay sensitivity requirements, an interval can be divided into either 1 or $K_{maxFre}$ periods. Within each period, each high delay sensitivity reservation is scheduled once for data communication. For ease of explanation, assume that an interval consists of $K_{fre}$ periods, which can be either 1 or $K_{maxFre}$.

For a mote→aggregator QoS reservation, the aggregator generates polling packets to poll each sensor mote for data within each time interval. If the number of packets returned for each poll is too small, the control overhead will be high. Hence, we aggregate multiple short polling packets into a larger one. However, the requested number of packets can not be arbitrarily large within a single polling packet because losing such a polling packet leads to significant wireless bandwidth loss. In BodyQoS, a dynamically configurable parameter $PL$ is used to set the maximum number of packets requested within a single polling packet.

To take advantage of polling aggregation, we group all QoS reservations that request mote→aggregator traffic, as shown in Figure 4. Within each period, the wireless bandwidth is first assigned to aggregator→motes traffic. If the requested data packets are successfully delivered before the assigned bandwidth is consumed, the unused bandwidth is used for scheduling motes→aggregator traffic. The remaining bandwidth is used for best-effort communication.

For each QoS reservation $R_i$, the aggregator knows how many QoS reservations remain to be scheduled before serving the best-effort traffic. Thus, the aggregator can assign reservation $R_i$ a sleep period $T_{sleep}$, during which $R_i$ will not be scheduled for data communication and the corresponding radio can be scheduled for sleeping. The $T_{sleep}$ value is piggybacked in each polling packet to notify corresponding sensor motes. Each sensor mote checks all of its QoS reservations for the overlapping sleep period. If the overlapping period is greater than the configured sleep overhead $T_{minSleep}$, the radio is put into sleep mode for the overlapping period.

### B. Measuring Effective Bandwidth

In order for BodyQoS to perform efficient QoS scheduling, it is essential to measure the effective bandwidth that the QoS system can use during runtime. Conventional wisdom for measuring effective bandwidth usually depends on underlying MAC functions. For example, in CODA [24], each node samples the channel load periodically to compute how busy the channel is. This scheme works for radios that have the carrier sense ability, and may not work for frequency hopping spread spectrum radios such as Bluetooth [13]. The effective bandwidth can also be measured through utilizing ACK packets such as in B-MAC [25] or the RTS-CTS-DATA-ACK handshake in IEEE 802.11b [26].

In BodyQoS, we propose a radio-agnostic method to measure effective bandwidth by taking advantage of the global knowledge of the QoS scheduler. In BodyQoS, if QoS reservation $R_i$ requests bandwidth $b_i$, the QoS scheduling algorithm, discussed later in this section, interprets the request as: within each time interval, request VMAC to send/receive $D_i$ packets within time $T_i \times D_i$, where $T_i$ is the time for MAC to send a packet. The $D_i$ and $T_i$ information is included in a polling packet and sent from the aggregator to a mote. When the mote receives this packet, it continues sending the requested $D_i$ packets until the specified time period $T_i \times D_i$ expires. When the requested reservation has no packet to send, a "NoData" packet is returned, which ensures that the mote tries its best to send back $D_i$ packets within the time period $T_i \times D_i$.

On the aggregator side, after a polling packet is sent out, it waits for the mote response for a time period of $T_i \times D_i + T_{maxPkt}$, where $T_{maxPkt}$ is used to account for the maximum backoff or retransmissions of the polling packet. The effective bandwidth computation needs to consider three possible cases:

1) If the aggregator receives $D_i$ packets within time $T_i \times D_i + T_{maxPkt}$, it stops waiting for more packets, and records the actual wait time $T_{waitTime}$. The number of packets received is $D_i$, and the effective bandwidth is $BW_{effective} = \frac{D_i * Bytes\,per Packet + Polling\,Packet\,Size}{T_{waitTime}}$, within the recorded actual waiting time.
2) If the aggregator does not receive $D_i$ replies, it continues waiting for packets until the specified time period $T_i \times D_i + T_{maxPkt}$ expires. Then it records the actual number of received packets

[2]In the current implementation, $K_{maxFre}$ is configured during system initialization. In the future, we plan to configure it dynamically during runtime.

to compute the effective bandwidth within the time period $T_i \times D_i + T_{maxPkt}$ as $BW_{effective} = \frac{Num\,of\,Recived\,Packet * Bytes\,Per\,Packet + Polling\,Packet\,Size}{T_i \times D_i + T_{maxPkt}}$

3) If the aggregator does not receive any packets before the time period $T_i \times D_i + T_{maxPkt}$ expires, it knows that either the polling packet was lost, or all replies were lost. The aggregator assumes that the single polling packet was lost (since this is far more likely), and the effective bandwidth is 0 within the time period $T_{maxPkt}$.

As discussed above, whenever a polling packet is transmitted, the aggregator gets a chance to measure the effective bandwidth almost for free. Even though these bandwidth measurements are only samples of the channel when polling occurs, they are accurate enough for the QoS support since the time interval is relatively short (2 seconds in our system).

The aforementioned is a direct way to compute the effective bandwidth in BodyQoS. To reduce the processing load, we propose a second, simpler, and indirect scheme: $BW_{effective} = BW_{ideal} * \frac{Num\,of\,Delivered\,Packet}{Num\,of\,Requested\,Packet}$. For each polling packet, the QoS scheduler knows the number of requested packets. The number of delivered packets can also be easily obtained at the aggregator. The ideal bandwidth can be computed as follows:

$$BW_{ideal} = \frac{N_{pkt} \times S_{pkt} \times 8}{T_{interval}} \quad (1)$$

In our performance evaluation, we use this indirect scheme to compute effective bandwidth, and also to maintain a moving average of effective bandwidth with the decay factor of $\delta$:

$$\begin{aligned} BW_{movAvg}(i+1) &= \delta \times BW_{effective} \\ &\quad +(1-\delta) \times BW_{movAvg}(i) \end{aligned} \quad (2)$$

### C. Advanced Scheduling Algorithms

Next, we describe a way to compute $T_i$ and $D_i$ values at runtime. In BodyQoS, we provide two algorithms for computing these values: RSVP-Light QoS scheduling and adaptive QoS scheduling.

*1) RSVP-Light QoS Scheduling:* In the standard RSVP [19] protocol, an audio/video stream reservation consists of a fixed bandwidth and time for data communication, and lost packets are not retransmitted. This is reasonable for audio/video streams because it does not make sense to play late or unordered portions of audio/video data. BodyQoS supports this conventional wisdom, by reserving a fixed wireless bandwidth according to the QoS reservation, disregarding the current channel condition and measured effective bandwidth.

If a QoS reservation $R_i$ requests $b_i$ Kbps bandwidth, the number of bits sent within each time interval $T_{interval}$ is $b_i * T_{interval}$. Since the effective application data payload size for each VMAC packet is $S_{pkt}$ bytes, the number of data packets that should be scheduled for $R_i$ is:

$$D_i = \lceil \frac{b_i * T_{interval}}{S_{pkt} \times 8} \rceil \quad (3)$$

In section III, we defined $T_{minPkt}$ as the minimum time period for the real MAC to send out a packet when the channel is clean. For VMAC to send out $D_i$ packets, BodyQoS should reserve a time period of $T_{minPkt} \times D_i$

*2) Adaptive QoS Scheduling:* To cope with channel impairment, it is essential for the QoS system to manage wireless resources adaptively, based on channel conditions. For example, an EKG sensor continuously samples heart activities and places data into a limited size buffer in a sensor mote. During times of channel impairment, the effective bandwidth reduces and packets are lost. The lost packets should receive more opportunities to be retransmitted. The question is how much time should the lost packets get for retransmission. For example, if the packet needs 4ms to be sent when the effective bandwidth is 100%, then 40ms is needed to service the reservation when the effective bandwidth reduces to 10%.

Now we must determine how to allocate the 40ms. One approach is to give the MAC more time for retransmission. However, MAC protocols typically limit the number of backoffs and retransmissions. Radio control is returned to QoS scheduler after time $T_{maxPkt}$. If the scheduler waited any longer, the radio would remain idle, wasting precious resources. Instead, a portion of the bandwidth should be given to the transport layer so that the transport layer can retransmit the lost packet. Traditional transport layers do not ask for more resources when packets are lost. For example, communication failure in TCP is considered a result of congestion. TCP actually backs off and uses less bandwidth when packets are lost. However, packet losses in a BSN are mainly due to channel impairment, rather than congestion, since the radio range is small and BSNs typically have a single use. As a result, transport layer retransmissions make sense in BSN. Without the adaptive bandwidth provision, buffer space in sensor motes may be depleted and data samples could be lost.

In order to schedule adaptive bandwidth, we need to compute $D_i$ and $T_i$ values dynamically, according to the effective bandwidth $BW_{movAvg}$ from Formula 2. For ease of explanation, we denote the dynamically computed new values of $D_i$ and $T_i$ as $D_i^\star$ and $T_i^\star$. $D_i$ and $T_i$ are used to denote the values obtained in the ideal case when there is no channel impairment. Therefore, $T_i$ in the ideal case is the minimum time for the MAC to send out a packet, that is, $T_{minPkt}$. When there is channel impairment and the effective bandwidth is $BW_{movAvg}$, the MAC needs to increase the time for sending one packet to $T_i^\star$:

$$T_i^\star = min\{T_{minPkt} \times \frac{BW_{ideal}}{BW_{movAvg}}, T_{maxPkt}\} \quad (4)$$

Where $BW_{ideal}$ is computed according to Formula 1.

When the channel degrades to the point at which the maximum MAC retransmissions fails to deliver most packets, some additional time must be allocated so that the transport layer can retransmit lost packets. The total time assigned to both the transport and MAC layers should be the adaptive bandwidth the reservation requests under the current channel condition. In the ideal case, the time assigned to the reservation is $D_i \times T_i$. When there is channel impairment, the time assigned to it should be $D_i^\star \times T_i^\star$. So we have the following

equation:

$$\frac{D_i^\star \times T_i^\star}{D_i \times T_i} = \frac{BW_{ideal}}{BW_{movAvg}}$$

From this equation, we can obtain the following formula:

$$D_i^\star = D_i \times \frac{BW_{ideal}/BW_{movAvg}}{T_i^\star/T_{minPkt}} \quad (5)$$

*3) High Delay Sensitivity:* Thus far we have only considered QoS reservations that require low delay sensitivity. For high delay sensitivity QoS reservations, the algorithms are slightly different.

For RSVP-Light QoS scheduling, we modify Formula 3 and calculate $D_i$ as follows:

$$D_i = max\{\lceil \frac{b_i \times T_{interval}}{S_{pkt} \times 8} \rceil, K_{maxFre}\} \quad (6)$$

For adaptive QoS scheduling, we can still use Formula 3 to compute $D_i$, and use Formula 5 to compute $D_i^\star$. However, the minimum allowable value for $D_i^\star$ is now $K_{maxFre}$.

## V. Admission Control Design

This section discusses three important aspects of admission control: (1) Computing the required bandwidth, including both data and signaling bandwidth for new QoS reservations; (2) Making decisions to admit/reject new QoS reservations. The same algorithm is used to adjust existing reservations when the effective bandwidth is reduced as a result of channel impairment; and (3) Coordinating slave admission control modules on the motes from a master module on the aggregator.

### A. Computing Bandwidth Requirements

For each QoS reservation, the requested data bandwidth is directly obtained from its QoS specification. We also compute the control overhead that is needed for scheduling reservations, which varies depending on the type of QoS reservation.

For aggregator→mote QoS reservations, no polling packets are needed. The required data bandwidth is given by:

$$D_{AM} = \sum_i D_i, \text{ where } D_i = \lceil \frac{b_i \times T_{interval}}{S_{pkt} \times 8} \rceil \quad (7)$$

For mote→aggregator QoS reservations with low delay sensitivity, the required data bandwidth is given by:

$$D_{MAL} = \sum_i D_i \quad (8)$$

For mote→aggregator QoS reservations with high delay sensitivity, the required data bandwidth is given by:

$$D_{MAH} = \sum_i max\{D_i, K_{maxFre}\} \quad (9)$$

For mote→aggregator QoS reservations, regardless of the delay requirements, polling packets are needed. As discussed in Section IV, neither long polls nor short polls are optimal in terms of bandwidth efficiency. As a result, a standard polling length $PL$ is used, which denotes the standard packet size that is desired to be polled for each polling. Since all mote→aggregator traffic is grouped together for scheduling, the (polling) overhead bandwidth[3] can be computed as:

$$P = K_{fre} \times \lceil \frac{(D_{MAH} + D_{MAL})/K_{fre}}{PL} \rceil \quad (10)$$

Combining all aforementioned data and polling overhead bandwidths together, the total required bandwidth for all QoS reservations is:

$$D_{required} = D_{AM} + D_{MAL} + D_{MAH} + P \quad (11)$$

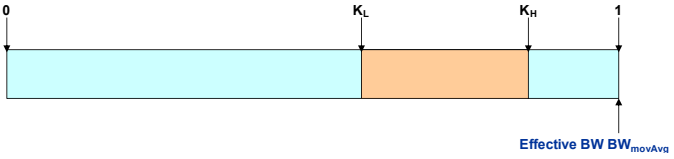



Fig. 5. Water Mark and Admission Decision

### B. Admission Decisions

As shown in Figure 5, a high water mark $K_H$ and a low water mark $K_L$ are set within the resource range [0, 1], where 1 corresponds to the effective bandwidth $BW_{movAvg}$. If the total required bandwidth, including the new reservation, is no greater than the low water mark $K_L \times BW_{movAvg}$, then the new reservation is accepted.

The total required bandwidth may also fall between the low water mark and the high water mark $K_H \times BW_{movAvg}$. In this case, if the new QoS reservation's priority is no less than the highest priority among all admitted reservations, this new reservation is admitted. Otherwise, it is rejected.

Finally, if the required bandwidth is greater than the high water mark, BodyQoS considers removing existing, lower priority reservations to make room for the new reservation. If enough bandwidth can be reclaimed by removing lower priority requests, the new reservation is admitted. Otherwise, it is rejected. By removing existing reservations, the total bandwidth requirement may be pushed below the high water mark, which can prevent thrashing when the effective bandwidth oscillates. When less important reservations are removed, they are temporarily treated as best-effort communication, and the requesting applications are notified of the QoS termination. Applications can choose to resubmit their request with different QoS requirements, perhaps setting a higher priority or a lower bandwidth requirement.

The ejection policy is now described in more detail. First, admission control decides how much bandwidth to reclaim. Then, it sorts all admitted reservations in increasing priority order. When two QoS reservations tie in priority, the one with the higher bandwidth requirement is considered first. The sorted list is then evaluated from head to tail. As the reservation list is processed, three situations may occur:

[3]We assume in this paper the resources consumed by polling and data packets is roughly the same, at a first order approximation. We are aware that large data packets do exist, and we will consider an additional variable for controling packet size in future work.

1) If the aggregated bandwidth of all QoS reservations to be removed, including the current reservation being checked, is greater than the bandwidth that is needed to be reclaimed, then the current reservation is ignored; admission control checks the next reservation in the list.
2) If the aggregated bandwidth of all QoS reservations to be removed, including the current reservation being checked, is smaller than the bandwidth that is needed to be reclaimed, then the current reservation is removed; admission control checks the next reservation.
3) If the aggregated bandwidth of all QoS reservations to be removed, including the current reservation being checked, is equal to the bandwidth that is needed to be reclaimed, then the current reservation in the list is removed and the algorithm stops.

Once admission control determines which QoS reservations to remove, it notifies the QoS scheduler to stop reserving resources for each.

The above algorithm executes when a new reservation request is received and when the effective bandwidth drops below the amount needed for admitted QoS reservations.

### C. Admission Control Signaling

Because the BodyQoS design is based on an asymmetric architecture, admission control is split between the master and slave modules on the aggregator and sensor nodes. Signaling between the master and slave is needed in three cases: when a new reservation request is submitted by an application, when the application must be notified of a stopped reservation, and when an application requests the removal of an existing reservation. Signaling is required between master and slave modules to notify each of application requests and changes in the set of admitted reservations. Details of admission control signaling is omitted due to space constraints.

## VI. Performance Evaluation

We have implemented BodyQoS in TinyOS [20] with the CC2420 IEEE 802.15.4 [12] radio. The VMAC implementation, which is the only MAC-dependent component in the QoS system, has less than 100 lines of code while the whole BodyQoS system has about 3700 lines of code. As a result, porting the QoS system from one radio platform to another should be relatively straightforward, requiring modification of less than 3% of the code.

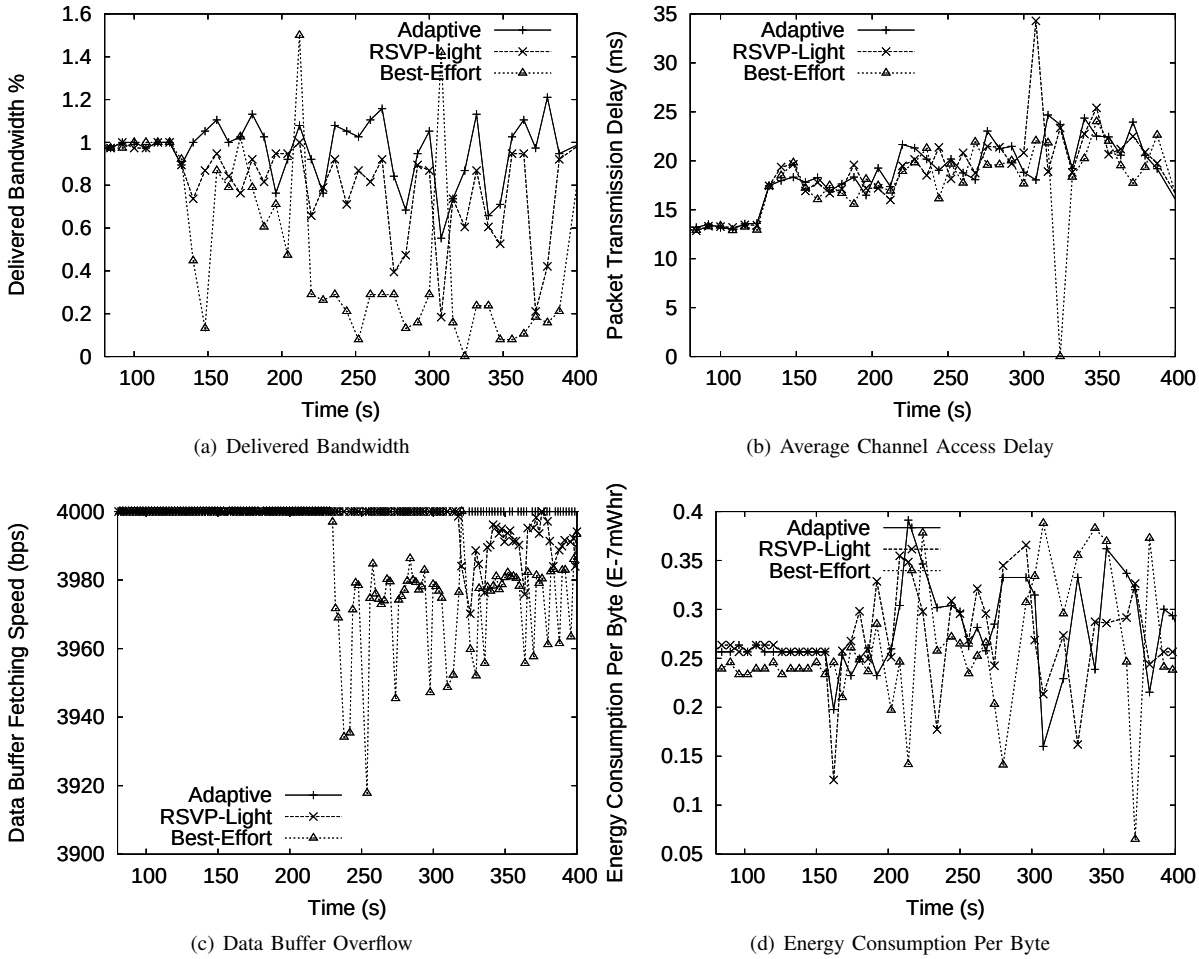
Our experimental setup mimics a typical on-body sensor network, in which sensors measure a person's physiological parameters. A MicaZ mote [21] is configured to report heart activity readings from an EKG sensor to the aggregator. A second MicaZ mote is configured to report a patient's location information, and a third MicaZ mote is configured to report body temperature readings. The aggregator is emulated by a MicaZ mote that is connected to a serial port of a laptop for data collection. Within the body sensor network, we compare the performance of three solutions: adaptive QoS that we propose in BodyQoS, conventional RSVP-Light QoS that we also implement in BodyOoS, and best-effort communication (no resource reservations). In our experiments, the EKG data stream uses adaptive QoS, the location data stream uses RSVP-Light QoS, and the temperature reading data stream requests best-effort communication. For a fair comparison, all three motes generate data at 4 Kbps.

Four metrics are used for performance evaluation. First, we measure the ratio of delivered bandwidth over requested bandwidth, demonstrating the reliability of data collection. Second, we measure the average time delay of data communication, which starts when a packet is fetched from a sensor's data buffer and ends when the packet is delivered to the aggragator. Third, we compare the rate at which sampled data is removed from the transmission buffer, reflecting an ability to preserve the limited storage space of sensor nodes. Finally, we measure the energy consumption per delivered data byte.

To evaluate BodyQoS's ability to cope with channel impairment, the level of interference is varied over the course of each experiment. In the first time period, 0s$\sim$135s, there is no explicit noise. In the second time period, 135s$\sim$225s, a MicaZ node is used to generate noise signals. It broadcast a packet every 30ms. In the third time period, 225s$\sim$315s, the noise level is increased to one packet every 25ms. In the fourth time period, 315$\sim$400s, the noise level is again increased to one packet every 20ms. The experiments were repeated over 10 times, and similar results were obtained in each trial. Figure 6 presents a representative result from one trial.

As presented in Figure 6(a), in the first time period (before interference is introduced), all three solutions maintain 100% bandwidth delivery ratio. However, in the face of interference, both RSVP-Light QoS and best-effort communication suffer severe packet loss and reduced bandwidth delivery ratios. For example, the bandwidth delivery ratio of RSVP-Light QoS decreases to about 90% in the second time period, to about 80% in the third time period, and finally to about 60% in the fourth time period. Similarly, the bandwidth delivery ratio of best-effort communication decreases to 45$\sim$90% in the second time period, and 0$\sim$30% in the third and fourth time periods. The performance of RSVP-Light QoS is decreased because it always reserves a fixed level of wireless resources, regardless of existing interference. As a result, the higher the interference level, the more the performance degrades. For best-effort communication, the performance decreases for two reasons. First, when interference increases, more wireless resources are allocated to adaptive QoS and less to best-effort communication. Second, it relies on the MAC layer to recover lost packets, but it does not benefit from the adaptive bandwidth scheduling, in which a collaborative effort from both the MAC and transport layers is made.

Note that, even when different interference levels are present in time periods 2$\sim$4, the adaptive QoS solution still maintains close to 100% bandwidth delivery ratio. Even though the bandwidth delivery ratio varies with time, which is due to the adaptation to interference, statistically speaking, the delivered bandwidth meets the requested bandwidth. This performance is mainly due to adaptive QoS's novel bandwidth scheduling. When the BSN suffers from channel impairment due




(a) Delivered Bandwidth

(b) Average Channel Access Delay

(c) Data Buffer Overflow

(d) Energy Consumption Per Byte

Fig. 6. Performance Evaluation with MiicaZ Devices

to RF interference or human body fading effect [14], the QoS scheduler allocates more resources to accommodate more retransmissions, so that statistically the delivered bandwidth is able to meet the requested bandwidth. Notice that sometimes the bandwidth delivery ratio for adaptive QoS exceeds 1. In this case, additional bandwidth has been allocated to adapt to interference, and the sensor node is able to send out accumulated data (backlogged in the transmission buffer because of lack of resource in the previous time intervals) in a burst.

As shown in Figure 6(b), BodyQoS pays only a small delay penalty in providing highly reliable data communication. We observe that the average time delay for all three cases is only slightly higher than the case in which there is no QoS reservation. As discussed in section IV, this time delay can be significantly reduced when $K_{fre}$ is greater than 1. We plan to evaluate this approach in future work.

The data buffer fetching speed for each QoS strategy is illustrated in Figure 6(c). It is clear that adaptive QoS is able to fetch data from the buffer at the same pace that the data is generated, regardless of interference levels. This feature is essential for storage limited sensor nodes to prevent data loss due to buffer overflow.

For RSVP-Light QoS, fixed wireless resources are reserved during all four time periods. When interference level increases, the MAC layer uses more time to send each packet, and hence the number of packets that can be fetched for transmission is reduced within the fixed time period. As shown in Figure 6(c), the data buffer fetching speed of RSVP-Light QoS reduces to 3.9$\sim$4 Kbps in the fourth time period. Best-effort communication receives less resources when more resources are given to the adaptive QoS in the presence of interference. As shown in the figure, the data fetching speed drops below 4 Kbps in the third and fourth time periods.

Figure 6(d) illustrates energy consumption per delivered data byte. In the first time period, adaptive QoS and RSVP-Light QoS consume slightly more energy than best-effort. This is because adaptive QoS and RSVP-Light QoS incur a polling overhead. Note that all three cases have similar energy efficiency when interference is present in time periods 2$\sim$4.

In summary, BodyQoS provides a high bandwidth delivery ratio (Figure 6(a)) and a high data fetching speed (Figure 6(b)) while maintaining similar delay (Figure 6(c)) and energy efficiency (Figure 6(d)) properties as compared with existing solutions. Interested readers can refer to [27] for a more

detailed performance evaluation.

## VII. Related Work

There has been extensive research in the networking community on QoS solutions for the Internet and general wireless ad hoc networks. Particular focus has been given to use of QoS to manage and reserve communication resources [7] [8] [9] [10]. However, those QoS solutions are designed for much more powerful devices, such as Internet routers and wireless access points, which are often line-powered. Most of these solutions do not apply to BSN applications, which use resource constrained sensor devices powered by small form factor batteries (e.g., AA, coin, or film).

Several sensor network protocols have been proposed that provide QoS features. For example, a VMAC like solution [23] was developed at U.C. Berkeley, in which priorities and reliability can be configured for a set of underlying MAC protocols. However, their solution does not support bandwidth specification and reservation. This statement is also true for the Bluetooth [13] protocol and several other sensor network protocols [28] [29] [30] that provide limited QoS features. For body sensor networks, real system experiences can be found in [2] [3], and a number of initial results are available in the BSN proceedings [31]. The SATIRE paper [1] describes a wearable personal monitoring service that is transparently embedded in user garments. However, none of these systems support QoS.

## VIII. Conclusions and Future Work

This paper presents the design and implementation of a QoS system for body sensor networks, called BodyQoS. Different from conventional QoS designs, BodyQoS addresses three unique challenges brought by BSN applications. First, BodyQoS adopts an asymmetric architecture, in which most processing is done at the resourceful aggregator while little is done at the resource limited sensor nodes. Second, a virtual MAC is developed in BodyQoS to make it radio-agnostic, so that it can control and schedule wireless resources without knowledge of the implementation details of the underlying MAC protocol. This approach supports a wide variety of different MACs, including CSMA, TDMA, and hybrid approaches. Third, BodyQoS adopts an adaptive resource scheduling strategy during times of channel impairment, either due to RF interference or fading effects. This makes it possible to provide statistical bandwidth guarantees as well as reliable data communication in BSN. BodyQos has been implemented in NesC on top of TinyOS, and evaluated in a MicaZ testbed. Our performance evaluation demonstrates that BodyQoS achieves greatly improved performance as compared with conventional solutions, with minimal overhead.

In the future, we plan to extend our work to address multiple co-existing body sensor networks, in which resource scheduling and reservation must be coordinated across multiple mobile networks. We also plan to develop a new transport protocol that can take advantage of the adaptive bandwidth provided by BodyQoS.

## References


[1] R. K. Ganti, P. Jayachandran, T. F. Abdelzaher, and John A. Stankovic, "SATIRE: A Software Architecture for Smart AtTIRE," in *ACM MobiSys*, 2006.

[2] "CodeBlue: Sensor Networks for Medical Care," http://www.eecs.harvard.edu/ mdw/proj/codeblue/.

[3] "MIThril," http://www.media.mit.edu/wearables/mithril/.

[4] "UVA Smart House," http://www.marc.med.virginia.edu.

[5] C. D. Kidd, R. J. Orr, G. D. Abowd, C. G. Atkeson, I. A. Essa, D. Macintyre, E. Mynatt, T. E. Starner, and W. Newstetter, "The aware home: A Living Laboratory for Ubiquitous Computing Research," in *CoBuild*, 1999.

[6] "UFL Smart House," http://www.rerc.ufl.edu.

[7] H. Zhu and G. Cao, "On Supporting Power-efficient Streaming Applications in Wireless Environments," in *IEEE Transactions on Mobile Computing*, 2005.

[8] I. Aad and C. Castelluccia, "Differentiation Mechanisms for IEEE 802.11," in *IEEE INFOCOM*, 2001.

[9] P. Garg, R. Doshi, R. Greene, M. Baker, M. Malek, and X. Cheng, "Using IEEE 802.11e MAC for QoS over Wireless," in *IEEE IPCCC*, 2003.

[10] G. S. Ahn and A. Campbell and A. Veres and L. H. Sun, "Supporting service differentiation for real-time and best effort traffic in stateless wireless ad hoc networks (swan)," in *IEEE Transactions on Mobile Computing*, 2002.

[11] "Chipcon CC1000 Low Power Radio Transceiver," http://www.chipcon.com.

[12] "IEEE 802.15.4, Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)," 2003, http://www.ieee802.org/15/pub/TG4.html.

[13] "IEEE 802.15.1: Wireless medium access control (MAC)and physical layer (PHY) specifications for wireless personal area networks (WPANs)," 2002, http://www.ieee802.org/15/pub/TG1.html.

[14] 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," Third International Conference on Body Area Networks (BodyNets'08).

[15] A. Natarajan, M. Motani, B. de Silva, K. Yap, and K. C. Chua, "Investigating Network Architectures for Body Sensor Networks," in *HealthNet 2007*, June 2007.

[16] L. Roelens, Van den Bulcke, S. Joseph, W. Vermeeren, and L. G. Martens, "Path loss model for wireless narrowband communication above flat phantom," *IEEE Electronics Leters*, 2006.

[17] A. J. Johansson, "Wave-Propagation from medical implants-influence of body shape on radiation pattern," in *2nd Joint EMBS/BMES Conference*, Oct 2002.

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

[19] "Resource ReSerVation Protocol," http://www.ietf.org/rfc/rfc2205.txt.

[20] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System Architecture Directions for Networked Sensors," in *ASPLOS-IX*, 2000.

[21] "XBOW Mote Specifications," http://www.xbow.com.

[22] "CC2420 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver," http://www.chipcon.com.

[23] J. Polastre, J. Hui, P. Levis, J. Zhao, D. Culler, S. Shenker, and I. Stoica, "A unifying link abstraction for wireless sensor networks," in *ACM SenSys*, 2005.

[24] C. Wan, S. Eisenman, and A. Campbell, "CODA: Congestion Detection and Avoidance in Sensor Networks," in *ACM SenSys*, 2003.

[25] J. Polastre, J. Hill, and D. Culler, "Versatile Low Power Median Access for Wireless Sensor Networks," in *ACM SenSys*, 2004.

[26] "IEEE 802.11, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification," 1999, ANSI/IEEE Std. 802.11.

[27] Gang Zhou, *Taming the Sensor Networking Challenges*, Ph.D. thesis, University of Virginia, Charlottesville, VA, 2007.

[28] 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.

[29] Y. Liu, E. I., and H. Qi, "An energy-efficient QoS-aware media access control protocol for wireless sensor networks," in *IEEE MASS*, 2005.

[30] Q. Zhao and L. Tong, "QoS Specific Medium Access Control forWireless Sensor Networks with Fading," in *SPSC*, 2003.

[31] "BSN Proceedings," http://www.bsn-web.org.