GeM-REM: Generative Model-driven Resource efficient ECG Monitoring in Body Sensor Networks

Sidharth Nabar[*], Ayan Banerjee[†], Sandeep K.S. Gupta[†] and Radha Poovendran[*]
[*]Network Security Lab (NSL), Electrical Engineering Department, University of Washington, Seattle
[†]IMPACT Lab, Arizona State University, Tempe.
Email: {snabar, rp3}@uw.edu, {abanerj3, sandeep.gupta}@asu.edu

***Abstract*—With recent advances in smartphones and wearable sensors, Body Sensor Networks (BSNs) have been proposed for use in continuous, remote electrocardiogram (ECG) monitoring. In such systems, sampling the ECG at clinically recommended rates (250 Hz) and wireless transmission of the collected data incurs high energy consumption at the energy-constrained body sensor. The large volume of collected data also makes data storage at the sensor infeasible. Thus, there is a need for reducing the energy consumption and data size at the sensor, while maintaining the ECG quality required for diagnosis. In this paper, we propose GeM-REM, a resource-efficient ECG monitoring method for BSNs. GeM-REM uses a generative ECG model at the base station and its lightweight version at the sensor. The sensor transmits data only when the sensed ECG deviates from model-based values, thus saving transmission energy. Further, the model parameters are continually updated based on the sensed ECG. The proposed approach enables storage of ECG data in terms of model parameters rather than data samples, which reduces the required storage space. Implementation on a sensor platform and evaluation using real ECG data from MIT-BIH dataset shows transmission energy and data storage reduction ratios of 42.1:1 and 37.3:1 respectively, which are better than state of the art ECG data compression schemes.**

***Keywords*-body sensor networks; BSN; model-based communication; ECG monitoring; resource-efficient; generative model**

I. Introduction

Electrocardiogram (ECG) is a time-varying signal representing the electrical activity of the heart, and is an effective, non-invasive diagnostic tool for cardiac monitoring. Recently, several systems have been developed for continuous, remote ECG monitoring using Body Sensor Networks (BSNs) [1]. Such systems typically consist of a wireless, battery-operated, body-worn sensor that collects ECG data and transmits it to a gateway device such as a smartphone. The gateway reports this data over the internet to a remote base station, which is typically a hospital server or caregiver's computer. Such remote monitoring allows collection of data during a person's daily routine and enables early detection of conditions such as tachycardia or angina. Further, the availability of continuous long-term data can help identify gradual, long-term trends in the cardiac health of at risk patients.

A key challenge in BSN-based ECG monitoring is the large volume of data collected by the sensor in a short time interval. For example, with a sampling rate of 250 Hz and resolution of 12 bits/sample, more than 2 KB of data is collected within 6 seconds. Local storage of this data on the sensor or the gateway device is impractical due to storage limitations. Further, wireless transmission of this data consumes significant power at the energy-constrained sensor. At the same time, the quality and continuity of the reported ECG signal must be maintained at the base station to allow effective investigation and diagnosis by a physician.

In this paper, we focus on this problem of resource-efficient ECG monitoring for BSNs, and develop GeM-REM: a Generative Model-driven Resource-efficient ECG Monitoring method. Our key observation is that ECG is a fairly periodic signal, with a known morphology and well-understood temporal variations. A set of key features are collected from a patient's ECG and incorporated into a generative model that is stored on the sensor and the base station. The sensor transmits data only when the sensed ECG deviates from the assumed model. When no data is received from the sensor, the base station uses the generative model to generate a synthetic signal closely resembling the patient's ECG. This signal can be used as the patient's ECG for clinical investigations. Through experiments on real-life ECG data, we show that GeM-REM significantly reduces sensor energy consumption, while preserving the diagnostic quality of the reported ECG.

We make the following contributions in this paper:

- We develop a novel generative model based scheme for ECG monitoring, where the sensor and base station store a common ECG model, and the sensor transmits data only when the sensed ECG deviates from this model.
- We design a base station module that can learn a model based on training data and generate synthetic ECG signals using the trained model. We also develop a lightweight sensor module that performs comparison of the sensed ECG data to the model.
- We implement the proposed system on a sensor platform and show the resultant savings in energy consumption and data storage memory requirement.

The rest of the paper is organized as follows. Section II presents background and related work. Section III discusses the overall architecture and operation of GeM-REM. Sections IV and V present the design and implementation of the base station and sensor modules respectively. We present our results in Section VI, and Section VII concludes the paper.

The work of A. Banerjee and S.K.S. Gupta was funded in part by National Science Foundation Grant CT-0831544.

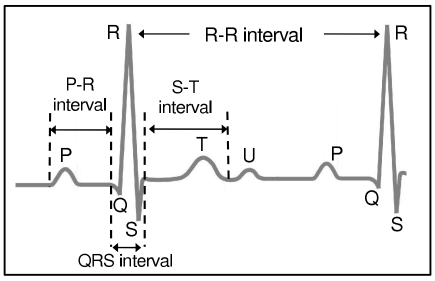


Fig. 1. Morphology features of an ECG beat. A beat is comprised of P, Q, R, S and T waves, with a U wave present in some cases.

II. Background and Related Work

The ECG signal has been extensively studied and used for cardiac diagnosis. As shown in Figure 1, a single beat of ECG consists of P, Q, R, S and T waves, with a U wave present in some cases. The Q, R and S waves are often jointly considered as a QRS complex. The shape, amplitude and relative locations of the constituent waves are key features of an ECG, referred to as *morphology features*. The distance between two consecutive R peaks is called the R-R interval, and its reciprocal gives the instantaneous heart rate. The R-R interval typically varies over time, and this variation is described using temporal features such as mean and standard deviation of heart rate, and spectral features such as Low Frequency/High Frequency (LF/HF) ratio [2]. In this paper, we refer to these features as *inter-beat features*.

ECG is a low amplitude electrical signal and is often corrupted by noise from various sources such as electrical mains, muscle noise and patient movement. As a result, the measured signal must be filtered to extract the underlying ECG. The QRS complex can be extracted using computationally lightweight algorithms [3], while extracting P and T waves requires advanced filtering techniques that are computationally expensive to implement on sensors. Further, several clinical conditions can be diagnosed from the QRS complex alone. As a result, in the current version of GeM-REM, we focus only on the QRS complex. Future extensions will include improved filtering methods that can extract P and T waves.

Generative models: A key aspect of GeM-REM is the use of a generative ECG model. Such models can generate synthetic ECG signals, given a set of input parameters. In this paper, we use the widely accepted dynamical generative model ECGSYN, proposed in [2]. In ECGSYN, the inter-beat features of ECG (mean and standard deviation of heart rate and LF/HF ratio) are modeled using 3 parameters: *hrmean*, *hrstd* and *lfhfratio* respectively. For the morphology features, each wave (P, Q, R, S and T) is represented by 3 parameters: (a, b, θ), which determine its height, width and distance to R peak, respectively[1]. The authors of ECGSYN provide a MATLAB implementation [4] to generate synthetic ECG, given a set of input parameters. To avoid duplication, we use this implementation in GeM-REM.

[1]Note that the height of a wave is not exactly equal to the value of a. Similarly, the width of the wave is not directly equal to value of b.

A. Related Work

Several approaches based on data compression using wavelets, Huffman coding and priority-based encoding have been proposed in literature to reduce energy consumption and data size in ECG monitoring [1]. A feature extraction-based method was proposed in [5]. These schemes, unlike GeM-REM, need to continuously transmit data, thus limiting their energy savings. In [6], we discuss how these schemes can be combined with GeM-REM to further improve energy savings. Recently, a compressive sensing approach has been proposed [7], which uses the sparsity of the ECG signal in specific wavelet transformations to reduce sampling rate. However, reconstruction of the received signal is complex and strongly depends on error-free transmission of all coefficients.

III. Model-Based Operation of GeM-REM

In this section, we describe the overall architecture and operation of GeM-REM. The assumed BSN system model is shown in Figure 2a, where the generative ECG model used at the sensor and the base station is denoted as $\mathcal{G}$.

Architecture: As shown in Figure 2b, GeM-REM consists of the following two modules:

1) Base station module (M_{BS}): This module uses $\mathcal{G}$ to generate synthetic ECG data given a set of input parameter values. Additionally, it includes a model learning function, which is used to train $\mathcal{G}$ based on a specific patient's ECG. This function derives suitable input parameter values for $\mathcal{G}$ from the given ECG data.
2) Sensor module (M_{LITE}): This module is intended for use in the ECG sensor and uses a lightweight implementation of the model $\mathcal{G}$ to generate an expected ECG signal. It also performs a comparison between this signal and the sensed ECG, to decide when to transmit data to the base station.

Initialization: Prior to deploying the system for a patient, the learning functionality of M_{BS} is used to train $\mathcal{G}$ using the patient's ECG data. This training process outputs a set of parameter values, which are stored on the base station as well as the sensor. These values are intended to be used as inputs to $\mathcal{G}$, for generating synthetic ECG data closely resembling the patient's actual ECG.

Basic operation: During regular operation of the system, the sensor compares the sensed ECG signal to that generated by $\mathcal{G}$, and if these signals match within a pre-defined threshold, the sensor does not report any data to the base station.[2] Conversely, if the sensed data deviates from the model, the sensor transmits updates to the base station, as follows:

- Feature updates: Representative features are calculated from the sensed ECG data, and when these values change significantly, the sensor updates the corresponding parameters of the model $\mathcal{G}$ in M_{LITE}. Further, the sensor reports these values to the base station. Such reports,

[2]To differentiate between absence of data reports and sensor or network failure, we use a periodic sensor heartbeat scheme, where the sensor periodically sends 'HELLO' messages to the base station.

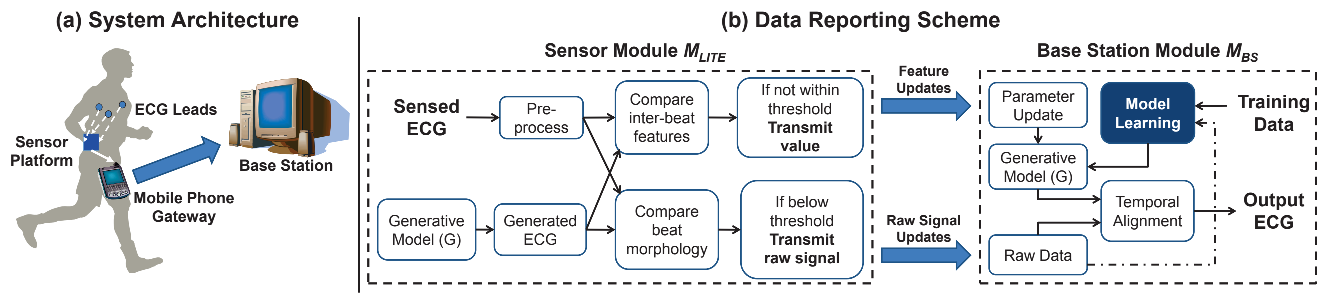


Fig. 2. System architecture and data reporting scheme of GeM-REM. The computationally intensive model learning component is implemented in M_{BS}, while M_{LITE} only has lightweight components. The model $\mathcal{G}$ in M_{LITE} is a simpler version of the one in M_{BS}, and can only generate individual beats.

called feature updates, are used by the base station to update the model in M_{BS}. This approach is suitable for features that can be calculated on the computationally-limited sensor.

- Raw signal updates: For some other model parameters, the corresponding ECG features are hard to compute on the sensor. In such cases, when the sensed ECG deviates from the model, the sensor sends raw sensed data to the base station, which is called a raw signal update. Based on received data, the base station derives new parameter values for $\mathcal{G}$ using its model learning functionality. These values are then communicated to the sensor, to update the model in M_{LITE}.

These updates allow the parameter values of the model $\mathcal{G}$ in M_{BS} and in M_{LITE} to be continually updated as the patient's ECG varies over time. At the base station, when raw data is received from the sensor, it is directly recorded as the patient's ECG. For remaining time intervals, the corresponding parameter values of $\mathcal{G}$ are used to generate a synthetic ECG signal. This signal is then temporally aligned with the raw ECG snippets to form the final, reported ECG signal.

A. *Advantages of Proposed Model-Based Architecture*

The model-based architecture of GeM-REM provides two main advantages: flexible energy consumption-data accuracy tradeoff, and reduced data size for ECG storage.

By defining suitable thresholds for the comparison between the sensed and model-generated ECG, a large fraction of data transmission at the sensor can be suppressed, thus significantly reducing sensor energy consumption. These threshold values can be specified by the physician and adjusted over time based on the data accuracy requirements of the application.

The proposed approach also provides data storage savings by representing ECG using model parameters instead of data samples. For example, during a time interval denoted $[t_A, t_B]$, if the model parameter values are $[p_1,p_2, \cdots p_N]$, the data can be stored as: "$[t_A, t_B]$: $[p_1, p_2, \cdots p_N]$". These values can later be used as inputs to $\mathcal{G}$ to regenerate the corresponding ECG data. This representation significantly reduces data size, and can enable local storage of ECG data on the patient's smartphone, which is not feasible with direct ECG storage.

We note that the approach outlined in this section does not depend on specific characteristics of the generative model $\mathcal{G}$, and can be applied to any given ECG data model. In this paper, we use the ECGSYN [2] model as an example.

IV. Base Station Module M_{BS}

In this section, we describe the design and implementation of the base station module M_{BS}. The ECGSYN model is used as an example of the generative model $\mathcal{G}$ for ECG.

The two main functions of the M_{BS} module are learning the input parameter values for $\mathcal{G}$ from given training data and generating a synthetic ECG when no data is received from the sensor. For the synthetic signal to closely match the actual, sensed ECG, we need to derive suitable input parameters for the generative model $\mathcal{G}$. This functionality is provided by the model learning function of M_{BS}, which uses real ECG data as input and calculates suitable input parameter values for the specific model used as $\mathcal{G}$. As discussed in Section II, the ECGSYN model used in this paper uses two groups of inputs: inter-beat parameters and morphology parameters. The learning method for each of these groups is as follows:

Inter-beat parameters: This group includes the parameters *hrmean*, *hrstd* and *lfhfratio*, corresponding to the mean heart rate, standard deviation of heart rate and LF/HF ratio features of ECG respectively. To calculate the LF/HF ratio, a set of 256 R-R interval values is obtained from the given ECG data and the Power Spectral Density (PSD) of this set is computed. The Low Frequency (LF) and High Frequency (HF) components are then obtained by integrating the PSD over the ranges (0.04Hz - 0.15Hz) and (0.15Hz - 0.4Hz) respectively. The ratio between these components gives the value of the *lfhfratio* parameter. The *hrmean* and *hrstd* values are obtained by performing averaging and standard deviation calculations on a set of 60 R-R interval values.

Morphology parameters: This group includes the nine parameters (a_Q, a_R, a_S, b_Q, b_R, b_S, θ_Q, θ_R, θ_S) that represent the morphology of the QRS complex. Out of these, θ_Q and θ_S are calculated using the distance of the R peak from the Q and S peaks respectively while θ_R is zero, by definition. For learning the remaining parameters (a_Q, a_R, a_S, b_Q, b_R, b_S) we use a curve fitting approach. A set of initial values for these parameters is obtained by solving a system of linear

TABLE I
QRS DETECTION ALGORITHM USED IN THE SENSOR MODULE

```
For each incoming data sample x[i] //-0.4 $\leq$ x[i] $\leq$ 1.2
  if (x[i] $>$ MAX) then Set MAX = x[i] //Initially, MAX = -100
  if (x[i] $<$ MIN) then Set MIN = x[i] //Initially, MIN = 100

  if (Looking for upward peak) then
    if (x[i] $<$ MAX) then Mark x[i-1] as PossiblePeak
    if (x[i] $<$ (MAX - thresholdUp)) then
      Mark latest PossiblePeak as RealPeak
      if (magnitude(RealPeak) $>$ thresholdR) then
         Mark peak as R peak
      Start Looking for downward peak
  else    // looking for downward peak

    if (x[i] $>$ MIN) then Mark x[i-1] as PossiblePeak
    if (x[i] $>$ (MIN + thresholdDown)) then
      Mark latest PossiblePeak as RealPeak
      if ((magnitude(RealPeak) $>$ thresholdQ)
             AND (PreviousPeak is S)) then
         Mark peak as Q peak
      if ((magnitude(RealPeak) $>$ thresholdS)
             AND (PreviousPeak is R)) then
         Mark peak as S peak
      Start Looking for upward peak
```

equations using six points on the ECG signal. Starting with these initial values, we use a least square error curve fitting function to adjust the values till noise floor is reached. Due to space limitations, the detailed initial value equations and optimization method are presented in [6].

Thus, a total of 12 input parameters (*hrstd*, *hrmean*, *lfhfratio*, a_Q, a_R, a_S, b_Q, b_R, b_S, θ_Q, θ_R, θ_S) are learned from the patient's true ECG and used to generate a matching synthetic ECG. We note that the morphology of ECG depends on the lead configuration and may vary across patients. Hence, the data used for learning the model should be obtained from the intended user of the system, and using the same lead configuration as the final system.

V. SENSOR MODULE M_{LITE}

M_{LITE} is intended for use in ECG sensors and contains the model $\mathcal{G}$ along with functions to compare the sensed ECG signal to $\mathcal{G}$. From our experience with BSNBench [8], we conclude that the full version of $\mathcal{G}$ may not be feasible for implementation in sensors due to resource limitations. As a result, we developed a suitable, lightweight TinyOS implementation of ECGSYN for use in GeM-REM.

The energy savings provided by GeM-REM can be maximized through efficient implementation of the set of components in M_{LITE} shown in Figure 2. In this paper, as a proof of concept, we implement these components in software, on the commercially available TelosB platform. In future versions, some of these components will be converted to hardware implementations, which are expected to lower the energy consumption. The first task performed by M_{LITE} is preprocessing the sensed ECG to convert it into a format suitable for comparison with the stored model. This involves the following operations:

1) Scaling: The amplitude of the sensed ECG signal is highly dependent on the sensor hardware and the ECG lead configuration. To ensure an accurate comparison between the sensed data and the model data, they must both be converted to a common, device-independent scale. This is achieved by linearly scaling each signal to a maximum of 1.2 mV and minimum of -0.4 mV.

2) Filtering: As discussed in Section II, the sensed ECG is typically noisy, and must be filtered to extract the underlying signal. Extracting the QRS complex requires a passband of 5-12 Hz, which is achieved by cascading lowpass and highpass filters with cutoff frequencies 5 Hz and 12 Hz respectively. For low computational overhead, we use a Finite Impulse Response (FIR) filter of 6 taps and order 32. A similar filter design used in [3] was shown to achieve good performance.

3) Peak detection: Measuring ECG features such as R-R intervals or QRS complex width requires detecting Q, R and S peaks. In order to perform this peak detection at low computational overhead, we developed a lightweight QRS peak detection algorithm shown in Table I. This algorithm detects all the positive and negative peaks in a signal, and then imposes relative thresholds (*thresholdQ*, *thresholdR*, *thresholdS*) on the amplitude to qualify peaks as Q, R and S respectively. The derivation of threshold values is presented in [6]. Further, false positives are reduced by imposing conditions based on the previous peak. For example, for a negative peak to be declared as 'S', the previous peak must be an R peak.

Once preprocessing is complete, the sensed ECG is compared to the signal generated by the model. Such a comparison is performed in two ways:

Feature-based comparison: In this approach, a set of features are extracted from both signals and their values are compared. We use this method for inter-beat features (mean and standard deviation of heart rate, and the LF/HF ratio) since they can be calculated from the sensed ECG at low computational cost. The mean and standard deviation of the heart rate are obtained by calculating the mean and standard deviation of a set of 30 consecutive R-R intervals. The LF/HF ratio is calculated as in Section IV for M_{BS}. However, to optimize computation speed and power consumption, we developed an efficient TinyOS implementation for Fast Fourier Transform (FFT). Once these calculations are complete, the feature values are compared to model parameter values *hrmean*, *hrstd* and *lfhfratio* respectively.

Direct signal comparison: Calculating the morphology features involves complex curve fitting and is not feasible for computationally-limited sensors. As a result, we use the direct signal comparison approach for the ECG morphology. A sample, representative beat, referred to as *MeanBeat*, is obtained by averaging 10 consecutive beats of the sensed ECG. On the other hand, we use our lightweight ECGSYN implementation to generate a sample ECG beat, referred to as *ModelBeat*. The *ModelBeat* and *MeanBeat* are aligned by superimposing the respective R peaks, and the fit is compared using the mean square error metric. The mean square metric is chosen since it captures shape as well as amplitude of the Q, R and S waves. Since generating the *ModelBeat* was found to be computationally expensive, it is performed only when morphology parameter values are updated. The generated

ModelBeat is then stored in memory for future use.

The sensor periodically computes the mean squared error between the *MeanBeat* with the *ModelBeat* and transmits raw sensed ECG samples to the base station if this error exceeds a specified threshold. For inter-beat features, if the difference in the sensed signal features and model parameters exceeds a pre-defined threshold, the model in M_{LITE} is updated and the new parameter values are transmitted to the base station.

VI. Experimental Results

In this section, we evaluate the following aspects of GeM-REM: (i) Accuracy of model learning functionality of M_{BS}, (ii) Energy consumption of M_{LITE}, (iii) Reduction in energy consumption and data storage, and (iv) Accuracy of GeM-REM. The data used for evaluation is a set of real life 3-lead ECG traces sampled at 250 Hz, obtained from the MIT-BIH database [4]. This ECG is further scaled and filtered, as described in Section V. This filtered version of ECG is referred to as ECG$_{raw}$ for the remainder of this section. Similarly, the ECG reported by GeM-REM at the base station is referred to as ECG$_{GR}$. The M_{BS} module is implemented in MATLAB, while the M_{LITE} module is implemented in TinyOS 2.x, and run on the TelosB platform.

A. Learning Function in M_{BS}

The model learning function of M_{BS} was tested over 20 ECG traces of two different types: (i) 10 Normal ECG, (ii) 10 Congestive Heart Failure (CHF) ECG. For each ECG$_{raw}$, a sample beat was obtained by averaging 10 consecutive beats. An ECGSYN model was trained on this beat, and then used to generate a synthetic ECG, which was compared to ECG$_{raw}$. As shown in Figure 3, the trained model achieves very good fit for different morphologies of ECG$_{raw}$. An average mean square error of 2.13 $\%$ was observed over 20 ECG traces.

B. Energy Consumption of M_{LITE}

We evaluate the energy consumption of M_{LITE} since it can be an important factor in the battery life of the ECG sensor. Some tasks in M_{LITE}, such as filtering, are executed continuously, while others, such as *hrmean* calculation, occur once for a set of several beats. To jointly consider these tasks, a per beat energy consumption is calculated for each task, and their sum is considered as the total energy consumption per beat. Table II shows that the highest contribution is from

TABLE II
Energy consumption on TelosB motes for different computational tasks in M_{LITE}

Task	Occurrence	Energy (mJ)	Energy/Beat (mJ/beat)
Scaling, Filtering and Peak detection	Every beat	0.63	0.62
Generate *ModelBeat* using ECGSYN *	-	22.176	-
Mean squared error between *MeanBeat* and *ModelBeat*	10 beats	1.008	0.1
Calculate heart rate mean, std. deviation	60 beats	3.434	0.056
Calculate LF/HF ratio	256 beats	3.072	0.012
Total Computational energy/beat			0.79

* Being a one-time task, it is not considered in per beat energy.

simple tasks such as scaling and filtering, since they are executed continuously. Using hardware implementations of these tasks in future versions is expected to drive down the energy consumption of M_{LITE}.

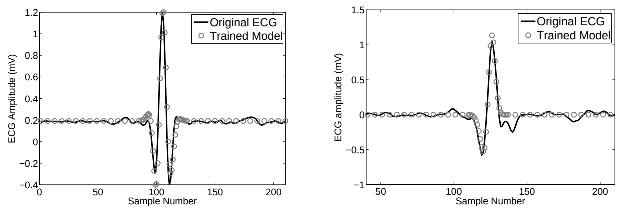
C. Energy and Memory Savings of GeM-REM

We now discuss the reduction in transmission energy consumption and data storage provided by GeM-REM, and compare our results to state of the art compression schemes proposed for ECG monitoring.

Transmission energy reduction: Transmission energy is an important performance metric for ECG monitoring schemes, since data transmission is the main factor in energy consumption on ECG sensors [5]. Since transmission energy consumption is directly proportional to the size of data being transmitted, compression-based methods use the compression ratio (CR) as an indication of transmission energy reduction [9]. To enable comparison of GeM-REM to state of the art compression schemes, we define a similar ratio for GeM-REM:

$$CR_{GR} = \frac{\text{Total ECG data}}{\text{Data transmitted by GeM-REM}}$$

To calculate CR_{GR}, we ran GeM-REM for a set of ECG$_{raw}$ signals from five different subjects. Each ECG signal is 5.4×10^6 samples long (approx. 6 hours). Using 16 bits/sample, the total data is 10.8 MB. The thresholds for *hrmean*, *hrstd* and *lfhfratio* parameter comparison were 3 beats per min (bpm), 2 bpm and 4, respectively. For morphology comparison, the



(a) Normal ECG (b) ECG showing CHF

Fig. 3. Comparison of model generated ECG and filtered original ECG for different ECG morphologies. The model achieves $> 97\%$ fit.

TABLE III
Compression Ratio of GeM-REM compared to existing compression schemes

Scheme/Algorithm	Compression Ratio
AZTEC [1]	10 : 1
CORTES [1]	4.3 : 1
Wavelet and Huffman [1]	9.4 : 1
DCT & arithmetic [1]	14.73 : 1
DCT & LZW [1]	9 : 1
SPITH [9]	21.4 : 1
DWLT & MSVQ [9]	29.3 : 1
GeM-REM	42.09 : 1

TABLE IV
PERFORMANCE OF GEM-REM FOR DIFFERENT THRESHOLD VALUES

Threshold Values		$\mathbf{CR_{GR}}$	Feature Error (%)	
(hrmean, hrstd, lfhfratio)	Morphology MSE		Max	Mean
(3,2,4)	0.07	42.08:1	6.92	4.03
(2,1,3)	0.07	40.19:1	6.32	3.24
(10,5,6)	0.2	7892:1*	14.3	8.64
(10,5,6)	0.01	1.16:1†	0.5	0.1

* This case does not send any raw signal updates
† This case transmits almost entire ECG$_{raw}$

TABLE V
PERCENT ERROR BETWEEN ECG$_{raw}$ AND ECG$_{GR}$ FOR 6 DIAGNOSTICALLY RELEVANT FEATURES

Feature	Error(%)
R-R interval	6.47
QRS complex width	6.92
Polarity of QRS	0.6
Number of peaks in QRS	1.1
QRS: Max amplitude	4.8
QRS: Min amplitude	4.13

threshold for mean squared error between *ModelBeat* and *MeanBeat* was set as 0.07. As shown later, these values preserve the diagnostic content of ECG.

With these threshold values, the average data transmission per ECG_{raw} was found to be 4420 bytes in feature updates (2210 updates) and 252.2 KB in raw signal updates (65 updates). Thus, total data transmitted is 256.62 KB, giving: $CR_{GR} = 10.8$ MB$/256.62$ KB $= 42.086 : 1$. Table III shows the comparison between GeM-REM and state of the art compression schemes [1], [9]. The variation in CR_{GR} based on the chosen threshold values is shown in Table IV. We observe that a larger threshold for morphology mean square error (MSE) gives very high gain in CR_{GR} since it almost eliminates data-intensive raw signal updates.

Storage space reduction: To measure the effectiveness of the data representation described in Section III-A, we used it to store ECG$_{GR}$ in memory. For each feature value update, a new time interval is started, and the set of model parameters is appended to the file. The data samples received through raw signal updates are stored directly. On an average, for an ECG$_{raw}$ file size of 32.9 MB, we obtained a size of 903 KB for ECG$_{GR}$ (Compression Ratio of 37.3), which is significantly higher than existing compression schemes. We note that this ratio relates to the storage requirements and is different from CR_{GR} which is related to data transmission.

D. Accuracy of GeM-REM

We consider the following two aspects of accuracy: diagnostic quality of the ECG$_{GR}$ signal and the ability of GeM-REM to detect occurrence of unexpected events in the sensed ECG.

Signal quality: The Percent Root-mean-square Distortion (PRD) metric typically used by compression schemes to measure distortion is not applicable to GeM-REM since the output signal ECG_{GR} is not intended to match the input ECG_{raw} sample-to-sample. Instead, the goal is to preserve the diagnostic information. As a result, we measure signal quality by evaluating a set of 6 diagnostically relevant features related to the QRS complex and R-R intervals for ECG$_{raw}$ and the corresponding ECG$_{GR}$ signal [10]. A percent error is then obtained for each feature. As shown in Table V, the average error over 5 different ECG traces is below 7% for the same set of threshold values used for calculating CR_{GR}. This indicates that the diagnostic quality of ECG is retained by GeM-REM. Like CR_{GR}, the signal quality also varies with chosen threshold values, as shown in Table IV. It was observed that lower threshold values for inter-beat features improve accuracy, with minimal effect on CR_{GR}.

Detection of unexpected events: To evaluate the response of GeM-REM to occurrence of unexpected events such as acute cardiac failure, we appended a trace of a CHF ECG to a normal ECG waveform. This was used as ECG$_{raw}$, to simulate occurrence of a heart failure. As expected, the change in ECG was detected in the morphology comparison, and was reported to the base station within 5 beats (approximately 5 s).

VII. CONCLUSION AND FUTURE WORK

In this paper, we proposed GeM-REM, a generative model-based method for ECG monitoring using BSNs. Based on validation with real ECG data, GeM-REM is observed to significantly reduce energy consumption and data storage, while maintaining the diagnostic quality of the reported ECG. Future versions of GeM-REM will include modeling of P,T waves and hardware implementation of some parts of M_{LITE}. Due to space limitations, we discuss these in [6], along with other possible extensions to GeM-REM.

REFERENCES

[1] B. Yu, L. Yang, and C.-C. Chong, “ECG Monitoring over Bluetooth: Data Compression and Transmission,” in *Wireless Communications and Networking Conference (WCNC)*. IEEE, 2010, pp. 1–5.

[2] P. McSharry, G. Clifford, L. Tarassenko, and L. Smith, “A dynamical model for generating synthetic electrocardiogram signals,” *Biomedical Engineering, IEEE Transactions on*, vol. 50, no. 3, pp. 289–294, 2003.

[3] L. Ren-Guey, I. Chou, L. Chien-Chih, L. Ming-Hsiu, and M. Chiu, “A Novel QRS Detection algorithm applied to the analysis of heart rate variability of patients with sleep apnea,” *Biomedical Eng. Application, Basis & communication*, vol. 17, no. 5, 2005.

[4] http://www.physionet.org/.

[5] R. Jafari, H. Noshadi, S. Ghiasi, and M. Sarrafzadeh, “Adaptive electrocardiogram feature extraction on distributed embedded systems,” *IEEE Transactions on Parallel and Distributed Systems*, pp. 797–807, 2006.

[6] S. Nabar, A. Banerjee, S. Gupta, and R. Poovendran, “Generative Model-driven Resource efficient ECG Monitoring,” *Technical Report # 008, Network Security Lab (NSL), University of Washington, Seattle.*, 2011.

[7] C. Eduardo, P. Octavian Adrian, and S. Pedro, “Implementation of Compressed Sensing in Telecardiology Sensor Networks,” *International Journal of Telemedicine and Applications*, 2010.

[8] S. Nabar, A. Banerjee, S. Gupta, and R. Poovendran, “Evaluation of body sensor network platforms: a design space and benchmarking analysis,” in *Wireless Health 2010*. ACM, 2010, pp. 118–127.

[9] B. Kim, I. Jung, I. Lee, and Y. Kim, “DWLT compression method based on MSVQ for a real-time ECG monitoring system in WSNs,” in *Proceedings of the International Conference on Mobile Technology, Applications, and Systems*. ACM, 2008, pp. 1–5.

[10] Y. Zigel, A. Cohen, and A. Katz, “The weighted diagnostic distortion (WDD) measure for ECG signal compression,” *Biomedical Engineering, IEEE Transactions on*, vol. 47, no. 11, pp. 1422–1430, 2002.