Data Collection from Resource-Limited Wireless Sensors for Cloud-Based Applications

Antonios Argyriou
Department of Electrical and Computer Engineering, University of Thessaly, Volos, 38221, Greece.

***Abstract*—We consider a wireless sensor network (WSN) where each sensor node samples a random signal, places the digitized data in a buffer, and transmits the data to an access point (AP) through a wireless packet erasure communication link. The AP communicates the data to a processing center (PC) located in the cloud. Our objective is to maximize the delivery of raw data to the cloud for post-processing given the buffer space and communication bandwidth limitations of each sensor. For this system model, we propose an algorithm that is executed at the sensor and summarizes the miminum subset of the incoming data based on the available buffer space and rate of the end-to-end communication link. The PC located in the cloud uses a sequential minimum mean square error (MMSE) estimation algorithm that fuses the summarized and raw data to estimate the random signal. Our algorithm offers a building block for sensing applications that desire the collection of data in raw form.**

***Index Terms*—Estimation, random signal, sensor data, Big Data, Internet-of-Things, machine-to-machine communication, wireless sensor networks, queueing system.**

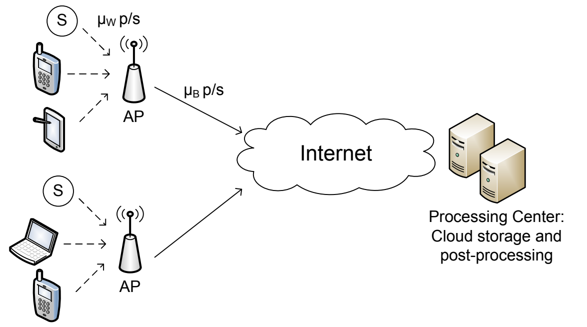


Fig. 1. System model for cloud-based applications that collect data from independent sensors/devices and process at a central processing center. The available bandwidth for each sensor in the wireless and wireline links is μ_W, and μ_B packets/second respectively.

I. Introduction

We live in a world gradually hyper-connected where every device will eventually be connected to the Internet. The volume of data that are generated are tremendous, hence the name Big Data. According to recent reports, nearly 30% of the Internet traffic is expected to be machine-to-machine (M2M) data by 2019 [1]. One of the requirements of several M2M and Internet of Things (IoT) applications, is the storage and post-processing of the collected data. This is true for a class of applications that generate data that have value even if the processing does not occur in real-time. Un-processed data in their raw form are needed for running machine learning or signal processing algorithms that fuse data from several information sources. Consider for example the wireless sensor network (WSN) illustrated in Fig. 1 that is used for an IoT application. Also assume that we desire to execute analytics algorithms that fuse the data from the two WSNs at the processing center (PC) located in the cloud. However, this is possible only if we have raw data measurements from all the information sources.

A first problem in the previous scenario is that the deployment of a large number of sensors and the transmission of their data to the cloud-located PC will result in significant costs for the network operator (e.g., cost of wireless and backhaul communication). Thus, the network operator has also a strong motivation to reduce the communicated data as much as possible. The second problem in this scenario is that when the device that is used for monitoring a random physical signal is a sensor that has resource limitations, then it might not be possible to transfer all the data to a PC leading to loss of valuable information. Resources that might be limited at the sensor and prevent the previous action include energy, processing power, and memory. Even with powerful sensing devices like smartphones and tablets that have ample storage resources, wireless bandwidth usually comes at a premium (especially cellular) and is a limited resource in this scenario.

There is a well-known solution to the bandwidth problem and this is to perform local in-network data compression (summarization) to minimize the costly and resource-demanding communication. Since we are typically interested in collecting data from a random signal, a type of summarization is through its estimation [2]. Estimating a random signal by collecting data from a lossy network has been investigated thoroughly in the literature both from a practical and from a theoretical point of view. A number works investigated the problem of estimation in sensor systems that have communication rate or energy limitations [3]–[7]. Modern massive IEEE 802.11ah-based WSNs were studied in [8]. These estimation algorithms offer excellent performance by using only low-complexity linear processing. The objectives of the aforementioned representative works are either to improve the estimation accuracy, typically measured in terms of the mean square error (MSE), or to reduce energy consumption. Hence, they are not concerned with maximizing the delivery of un-processed data.

In this paper we investigate a scenario where we desire to deliver the largest possible set of data samples to the PC that is located in the cloud by considering bandwidth and memory constraints. We propose an algorithm that uses the

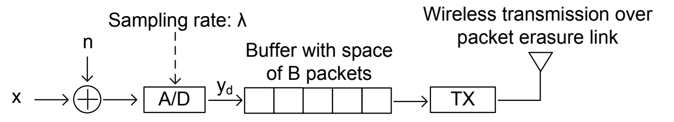

Sampling rate: λ
n
Wireless transmission over packet erasure link
Buffer with space of B packets
x
A/D
y_d
TX

Fig. 2. The sensor node adopts local processing for a subset of the collected samples from the random signal x, when the memory of the sensor and the rate of the communication link do not allow the transmission of each sample.

minimum subset of the collected data for locally compressing/summarizing them, when the sampling rate of the data exceeds the rate of the communication link and the ability of the buffer to store data. After this processing, the data are communicated and estimated at the PC with an on-demand sequential linear minimum mean square error (MMSE) estimation algorithm. Thus, the maximum possible subset of the data is transmitted in raw form. We also study the performance of the algorithm in terms of MSE relative to the available buffer space at the sensor.

II. System Model

The system model we describe next, is illustrated in Fig. 1. Multiple sensors are connected in a star topology with the AP and this represents a fairly typical WSN topology [9]. We focus our subsequent discussion and analysis on a single sensor. Each sensor samples a random signal x with an average rate of λ samples/second (Fig. 2). The samples of the signal are corrupted with additive white Gaussian noise (AWGN). If $y(i)\in\mathbb{R}$ is the i-th observation at the sensor and $n(i)\sim\mathcal{N}(0,\sigma_n^2)$ is the i-th AWGN sample, the observation data model is $y(i) = x(i) + n(i)$. Subsequently, the signal is quantized (compressed) by using R bits. The resulting digital signal of R bits is:

$$y_d(i) = Q(x(i) + n(i)) = x(i) + n(i) + q(i) \tag{1}$$

The variance of the quantization noise (or the distortion), under the use of a uniform probabilistic quantizer captured as the function $Q(\cdot)$ is:

$$\sigma_q^2 = \frac{x_{\max}^2}{(2^R-1)^2} \tag{2}$$

where $2x_{\max}$ is the range of the sensed signal. Note that in this paper we only consider the optimization of the sampling rate and not that of the quantization that can be optimized by selecting properly the value of R. Because the sampling and quantization noise are uncorrelated, we have that for the noise variance at sensor i:

$$\sigma_w^2 = \sigma_n^2 + \sigma_q^2 \tag{3}$$

After sampling and quantization, the sensor node creates a packet for each sample and stores it in a FIFO buffer of B packets. Then, the transmitter (TX) communicates the collected data to the AP through a wireless communication link. The wireless communication link transmits on average μ_w packets/second/sensor and this is a parameter that depends on the number of sensor nodes that share the wireless channel [8]. The wireless channel also generates packet erasures with probability p_e under a two-state Markov chain model. A more realistic model of the wireless channel is out of the scope of this paper but it can be easily supported. All the sensors share the communication channel but each one has an independent packet erasure probability. Thus, samples are lost if the buffer in the sensor is full when a new sample arrives, and when there is a packet erasure. The data that are received at the AP are forwarded through a wireline backhaul link of μ_b packets/sec to the cloud for the estimation of the random signal.

A. Linear Estimation

A classic approach for linear estimation of data that arrive progressively, is to estimate the random signal sequentially at the PC. Let us first describe the sequential MMSE estimation process at the PC. The recursive MMSE estimation formula is written as [2]:

$$\hat{x}(i) = \hat{x}(i-1) + k_i\Big(y(i) - \hat{x}(i-1)\Big) \tag{4}$$

In the above it is

$$k_i = \frac{a_{i-1}}{a_{i-1} + \sigma_w^2}. \tag{5}$$

The parameter α is the Bayesian MSE (BMSE) that is:

$$a_i = \frac{a_{i-1}\sigma_W^2}{a_{i-1} + \sigma_W^2} \tag{6}$$

III. Linear Estimation and Buffer Management Algorithm

For our system model, the problem we desire to address is the following: We want to increase the sampling rate of the random signal beyond the rate of the wireless and wireline communication links. The end-to-end communication link does not allow for the transmission of more than

$$\mu = \min(\mu_w, \mu_b) \text{ packets/sec.} \tag{7}$$

We want to identify what is the optimal course of action at the sensor, so that it transmits as many data as possible, but at the same time satisfies the limitations of communication rate μ and buffer space B. The choices are the following: 1) The reduction of the sampling rate to match the rate of the end-to-end communication link μ. 2) To allow $\lambda > \mu$ but if a new data sample arrives at the sensor, while the buffer is full with data waiting to be transmitted, part of the old data are replaced with the new [10]. This approach can only allow the cloud PC to estimate more accurately the recent value of the random signal x. 3) We propose to allow over-sampling with $\lambda > \mu$ but avoid discarding data by compressing the minimum subset of the data locally. At the same time the system complies with the resource limitations.

A. Decomposed Sequential Linear Estimation

The estimation algorithm is now adapted for our system model. Our algorithm is based on decomposing the estimation

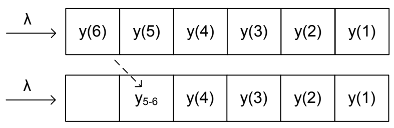


Fig. 3. Contents of the sensor buffer before and after the proposed algorithm is executed.

algorithm across the two network nodes, namely the sensor and the PC. The recursive formula in (4) is re-written as

$$\hat{x}(n) = \hat{x}(-1) + \sum_{i=0}^{n} a_i y(i) = \hat{x}(-1) + \sum_{i=0}^{k} a_i y(i) + \sum_{i=k+1}^{n} a_i y(i) = \hat{x}(k) + \sum_{i=k+1}^{n} a_i y(i). \quad (8)$$

The previous expression is used for initiating the description of our algorithm with the help of the example in Fig 3. Let us assume that the PC has estimated data sample with id 0 as $\hat{x}(0)$. Assume also that the sensor starts generating data samples at rate $\lambda > \mu$. In our example the sensor buffer contains samples $y(1)$ until $y(5)$ and before they are transmitted, sample $y(6)$ arrives. The sensor exits the *transmission mode* and enters *estimation mode* as follows. It replaces the last two entered data samples in the buffer with a scaled value of the two existing samples namely $y(5)$ and $y(6)$ and the result is

$$\tilde{y}_{5,6} = \sum_{i=5}^{6} a_i y(i).$$

This sample is stored in a digital packet and it is marked with an id that indicates that it contains two scaled data samples. *Thus, the stream of transmitted packets multiplexes both raw data and scaled/compressed samples.* The sensor essentially calculates the second term in (8) which means that it does not calculate an actual estimate of y. However, the sensor does not need to store locally the recent estimate $\hat{x}(k)$ but only to account for the impact on the final estimate of the new data samples that cannot be transmitted. In other words the algorithm monitors the tail of the buffer and processes data only when the buffer is full. This is to ensure that no packet is lost and as many as possible data samples are transmitted in a raw form. Note that if a new sample arrives in the system above, say $y(7)$, and no packet has been transmitted, this is combined with $\tilde{y}_{5,6}$ to create the new scaled data $\tilde{y}_{5,7}$.

The sensor processing can be simplified by exploiting the recursive structure of the BMSE in (6). With a little algebra

$$a_i(a_l) = \frac{a_l(\sigma_W^2)^i}{i a_l + (\sigma_W^2)^i}, i \geq l, \quad (9)$$

where a_l is the last calculated value for the BMSE. Thus, the sensor system can nearly instantaneously derive a_i for any sample with id $i > l$. The final formula for the scaling at the sensor is relative to the last value of the BMSE a_l:

$$\tilde{y}_{k+1,n}(a_l) = \sum_{i=k+1}^{n} \frac{a_l(\sigma_w^2)^{i-l}}{(i-l)a_l + (\sigma_w^2)^{i-l}} y(i) \quad (10)$$

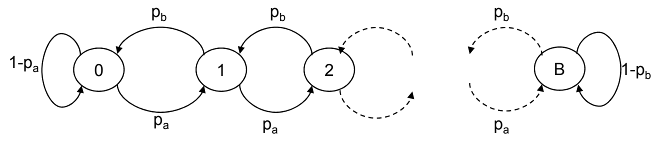


Fig. 4. M/M/1/B queuing model for the buffer of the sensor.

Next, the sensor stores the most recent calculated value a_n as the new last value to allow easier calculations for future batch processing, i.e., $l \leftarrow n$ and $a_l \leftarrow a_n$. At the PC, after data are received the estimation is executed as

$$\hat{x}_{pc}(n) = \hat{x}_{pc}(k) + \tilde{y}_{k+1,n}.$$

The algorithm tries to keep the buffer full and avoid discarding samples. Therefore, ensures that there is always one empty buffer spot in the buffer to accommodate for a new arrival.

IV. Performance Analysis of Buffering

The previous algorithm can be easily implemented in practice. However, for the adopted buffer model we can derive closed-form expressions for the number of packets that are transmitted in un-processed or summarized form as a function of the buffer size. To model the impact of the buffer size on specific performance metrics of the algorithm we design a simple stochastic model. This model does not account for packet erasures since they occur after the transmission. We adopt an $M/M/1/B$ queueing system and Fig. 4 depicts the state diagram of the queue. The queueing system is assumed to have an overwrite-buffer with space for B packets; when a new packet bearing more recent data arrives to the system, and the buffer is full, the last buffered packet, if any, is scaled with the new packet and are stored in the buffer. The parameters of the model include λ and μ that indicate the arrival rate and the service rate. In this figure the transition probabilities are $p_a = \lambda/(\lambda+\mu), p_b = \mu/(\lambda+\mu)$. For this model, the probability that the buffer is in state i (i packets exist in the buffer) is given by [11]:

$$\pi_i = \frac{1-\lambda/\mu}{1-(\lambda/\mu)^{B+1}}(\lambda/\mu)^i \quad i = 0, ..., B \quad (11)$$

A data packet will be transmitted raw if a new sample arrival takes place and at the same time the buffer is in any of the states from $1, ..., B-1$. In the cases above, the algorithm will not be initiated. Thus, the rate of un-processed samples is

$$\lambda_{\text{raw}} = \lambda \sum_{i=1}^{B-1} \pi_i. \quad (12)$$

On the other hand, a sample will be scaled if a packet arrival occurs and the sensor is in state B, which means that the buffer does not have any space left. This is because our algorithm is activated when the buffer is full. In any other case the sensor will transmit the raw data. The rate of scaled samples is $\lambda_{\text{sc}} = \lambda\pi_B$. We can also see that the scaling procedure is executed on average for λ_{sc} packets/sec. From this description we can deduce now further details about the system behavior. When $\lambda < \mu$ then $\lambda_{\text{raw}} = \lambda$ and all packets are transmitted un-processed. However, when $\lambda > \mu$ the average number of transmitted samples will be equal to the rate of both the raw plus the scaled data:

$$\mu = \lambda_{\text{raw}} + \lambda_{\text{sc}} \tag{13}$$

In this case our algorithm adjusts λ_{sc} so that the rate of the raw data matches the rate of the communication link.

V. Simulation and Numerical Results

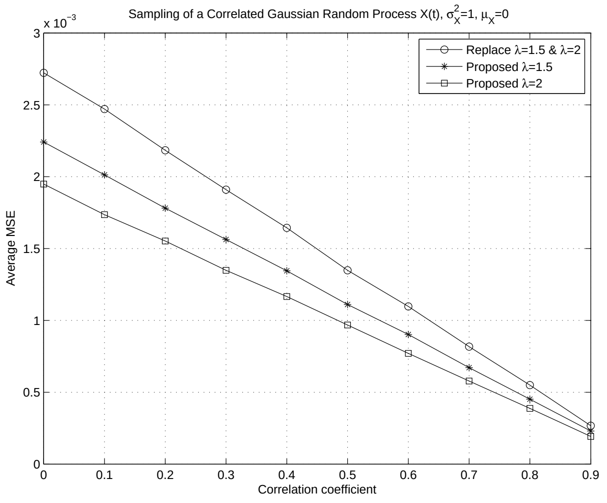
In this section, we provide simulation results to verify the behavior of the proposed algorithm. In our simulation we include four sensors where each sensor samples the corresponding random signal, and places the data in the buffer. Since the sample arrival rate is λ, samples are generated periodically with constant inter-arrival period of $1/\lambda$. The algorithm is executed every time a new sample arrival occurs. We set $\mu = 1$ and we vary λ according to our experiment. The random signals across the four sensors are independent. The MSE results are averaged over all the sensors.

A. MSE vs. Correlation of Sampled Data

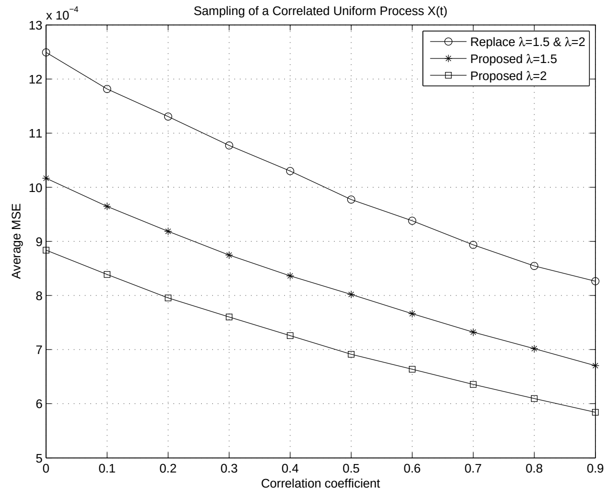
In Fig. 5(a) we present simulation results for the average MSE when the proposed algorithm tracks signal that is a correlated Gaussian process with zero mean and unit variance, while there are no packet erasures. The process is generated according to $x(i) = \rho x(i-1) + (1-\rho)z(i)$, where $z(i)$ provides uncorrelated samples from Gaussian or Uniform distribution depending on the experiment. Hence, it is a first order auto-regressive AR(1) process. The correlation coefficient ρ captures the dependency between current sample and the previous one. The proposed algorithm is compared against a buffer replacing policy where the most recently obtained sample replaces that last sample when the buffer is full. The results are very encouraging since we can observe significant MSE reduction. The Replace policy has the same performance regardless of the difference $\lambda - \mu$ because the sensor sends raw data at a constant rate equal to μ. However, the proposed algorithm utilizes to the fullest extra samples and reduces the MSE further as more data arrive. Performance gains are higher for lower correlation coefficient, i.e., when the generated data are more random, our scheme is even more valuable. Performance is even better for the sampling of a random process that is uniform in $[0, 1]$. The results are shown in Fig. 5(b) and are very good even for higher correlation between the data samples.

B. Sample Rate vs. Buffer Size

Numerical results are presented only for the buffer model analyzed in Section IV and for no packet erasures. Results in terms of λ_{raw} and the size of the buffer can be seen in Fig. 6(a). As the fraction λ/μ remains close to 1, where the sampling rate is equal or lower to the transmission rate, the sensor system experiences performance benefits in terms of the delivery rate of raw data when the buffer space is increased.



(a) Gaussian random process



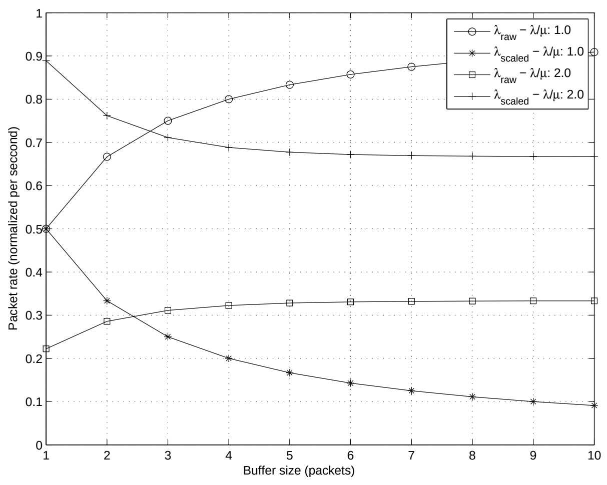
(b) Uniform random process

Fig. 5. MSE results for the proposed algorithm, a buffer replace policy, and two different types of AR(1) random processes.

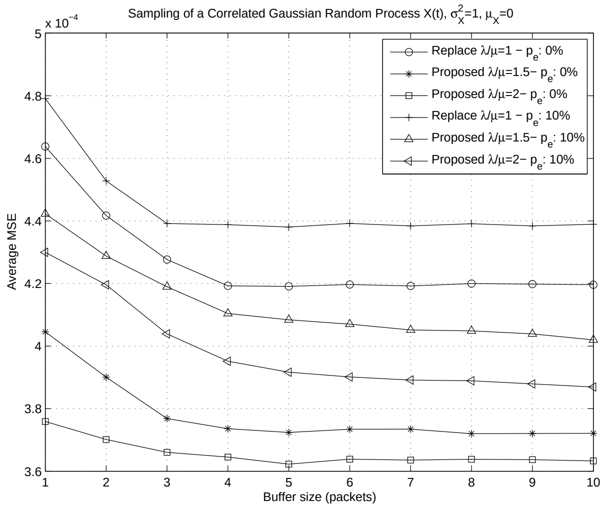
Thus, the important conclusion is that even for lower data rates, the system must provision for buffer space in order to increase the delivery of raw sampled data. The same behavior is not true for $\lambda > \mu$ where we observe that as the ratio λ/μ is increased, the significance of an increased buffer size is diminished. This is because our algorithm executes a higher number of data scaling operations that increase significantly λ_{sc}. On the other hand, λ_{raw} is decreased significantly but it is still proportional to the rate of oversampling $\lambda - \mu$. In this case, even this simple model can provide hints regarding the optimal buffer size which is in this example is approximately 3 packets. A buffer beyond this point is unnecessary.

C. MSE vs. Buffer Size and Packet Erasure Rate

MSE results for a correlation coefficient of ρ=0.5 and different buffer sizes can be seen in Fig. 6(b). For p_e=0% a buffer size increase beyond three packets does not help both



(a) Rate of scaled and raw data vs. buffer size.



(b) MSE vs. buffer size.

Fig. 6. MSE and packetrate results vs. the buffer size.

the typical Replace policy and the proposed scheme. For a ratio λ/μ less than one it is known that a buffer can increase the throughput of the queuing system [11] and that is why we could obtain benefits with the Replace policy. However, the lower sampling rate again would increase the MSE due to the reduction in the volume of data. As expected, for the proposed scheme where oversampling occurs, a higher λ/μ is not related to the buffer size since the algorithm is activated when the buffer is full and no sample is lost.

For p_e=10% the results are shown in the same figure. All the sensor-AP links are assumed to have the same average p_e. The results show that the proposed algorithm is more susceptible to the packet erasures because when a scaled sample is lost then a higher percentage of useful information for estimation is lost leading to worse MSE. As the ratio λ/μ is increased, this leads to an even more significant information loss from a single packet erasure. Even hough a significant MSE reduction with the proposed scheme still exists, it is nonetheless reduced. On the other hand, a packet erasure for the Replace system results in the loss of a single data sample. Despite the above, the MSE is still lower with the proposed scheme since the scaled samples that are successfully transmitted contain compressed information. A practical solution could include an unequal error protection (UEP) mechanism for the most important packets. One final comment is related to the minor MSE reduction that is observed for the proposed scheme and for larger buffer size. This is because with a larger buffer size, a slightly smaller number of samples are scaled in a single packet which means that a packet loss has reduced impact.

VI. Conclusions

In this paper, we presented a decentralized algorithm for sequential linear MMSE estimation that is suitable for sensor systems that have limitations in terms of memory and communication rate resources, but at the same time they want to communicate their raw data to the cloud. The algorithm selects the smallest number of data samples that must be summarized in order to avoid discarding them. At the same time, it ensures the maximum possible raw data rate to the PC by exploiting to the fullest the available buffer space. The proposed algorithm can be extremely useful in sensing applications where offline data processing is critical but the information is extracted from sensors with certain resource limitations.

References

[1] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, February 2015.

[2] S. M. Kay, *Fundamentals of Statistical Signal Processing, Volume I: Estimation Theory*. Prentice Hall, 1993.

[3] A. Fletcher, S. Rangan, and V. Goyal, “Estimation from lossy sensor data: jump linear modeling and kalman filtering,” in *IPSN*, april 2004.

[4] A. Behbahani, A. Eltawil, and H. Jafarkhani, “Linear decentralized estimation of correlated data for power-constrained wireless sensor networks,” *Signal Processing, IEEE Transactions on*, vol. 60, no. 11, pp. 6003 –6016, nov. 2012.

[5] P. Ishwar, R. Puri, K. Ramchandran, and S. Pradhan, “On rate-constrained distributed estimation in unreliable sensor networks,” *Selected Areas in Communications, IEEE Journal on*, vol. 23, no. 4, pp. 765 – 775, april 2005.

[6] A. Ribeiro and G. Giannakis, “Bandwidth-constrained distributed estimation for wireless sensor networks-part i: Gaussian case,” *Trans. Sig. Proc.*, vol. 54, no. 3, pp. 1131–1143, Oct. 2006.

[7] J. Li and G. AlRegib, “Rate-constrained distributed estimation in wireless sensor networks,” in *ICCCN*, 2006.

[8] A. Argyriou, “Power-efficient estimation in ieee 802.11ah wireless sensor networks with a cooperative relay,” in *IEEE International Conference on Communications (ICC)*, June 2015.

[9] H. Karl and A. Willig, *Protocols and Architectures for Wireless Sensor Networks*. Wiley, 2005.

[10] M. Aoun, A. Argyriou, and P. van der Stok, “Performance evaluation of network coding service reneging in ieee 802.15.4-based wireless sensor networks,” in *European conference on Wireless Sensor Netoworks (EWSN)*, 2011.

[11] R. Wolff, *Introduction to Queueing Theory*. Prentice-Hall, Inc., 1989.