Geometric Monitoring in Action: a Systems Perspective for the Internet of Things

Charalampos Stylianopoulos, Magnus Almgren, Olaf Landsiedel, Marina Papatriantafilou
Chalmers University of Technology, Sweden - {chasty, magnus.almgren, olafl, ptrianta}@chalmers.se

***Abstract*— Applications for IoT often continuously monitor sensor values and react if the network-wide aggregate exceeds a threshold. Previous work on Geometric monitoring (GM) has promised a several-fold reduction in communication but been limited to analytic or high-level simulation results. In this paper, we build and evaluate a full system design for GM on resource-constrained devices. In particular, we provide an algorithmic implementation for commodity IoT hardware and a detailed study regarding duty cycle reduction and energy savings. Our results, both from full-system simulations and a publicly available testbed, show that GM indeed provides several-fold energy savings in communication. We see up to 3x and 11x reduction in duty-cycle when monitoring the variance and average temperature of a real-world data set, but the results fall short compared to the reduction in communication (4.3x and 44x, respectively). Hence, we investigate the energy overhead imposed by the network stack and the communication pattern of the algorithm and summarize our findings. These insights may enable the design of protocols that will unlock more of the potential of GM and similar algorithms for IoT deployments.**

***Keywords*- geometric monitoring, IoT, lifetime improvement**

I. Introduction

Monitoring system-state or the environment-conditions are fundamental uses for Wireless Sensor Networks (WSNs). Given N sensor nodes with individual readings that vary over time, we want to continuously track whether a function f, defined over the network-wide weighted average of the readings, is larger than a threshold. Keeping track of such a function serves as a basis for many applications, e.g. detecting outliers [2], hot-spots [13] or denial-of-service attacks [4] [5], [9]. The challenge is to let all nodes accurately determine whether the function is above or below the threshold locally, without having to share every reading. For simple, linear functions (average), local constraints can be derived to minimize communication, while for non-linear functions (variance), deriving such constraints is challenging.

Distributed monitoring has received high interest, where Sharfman et al. [14] proposed a general method, called *geometric monitoring* (GM), that can monitor *any* function, linear or not, computed over network-wide aggregates and can keep track of its value with respect to a threshold. The method suppresses unnecessary communication by deriving local constraints that individual nodes can check without communication. The effectiveness of the method has been thoroughly studied showing impressive communication reductions. Variations of GM have been enhanced with sketches [7] and prediction models [8] and have been applied on outlier detection [2] and data stream queries [6]. The above mentioned extensions are orthogonal to the original GM algorithm. In this paper, we focus on the basic principles of GM and tackle the challenges described next.

Research Challenges: Even though GM and similar threshold monitoring algorithms are designed with sensor networks in mind, there is lack of insights from full-system perspective. Existing work on GM has focused on the algorithmic part, backed up with numerical, high level simulations where communication is assumed instant and reliable. However, the reality is different: packet losses are frequent, nodes have severe constraints on processing power and lifetime, and message propagation is costly in terms of energy and latency. Recent work on data aggregation [12] has shown that properties of the network stack greatly influence the lifetime savings that can be achieved in practice for similar applications.

Thus, the feasibility of GM for WSNs and the impact of the system's properties raise questions, such as: (1) What are the actual battery lifetime savings achievable on real nodes? (2) How can such methods be implemented on commodity IoT network stacks? To address these, we take a step beyond the existing analysis and consider the whole system stack, through (i) extensive, cycle-accurate, full-system simulations and (ii) validation from a real deployment. We are thus able to evaluate up to what degree and condition emerging results on distributed continuous monitoring can benefit real WSN deployments.

Contributions:

1) We bridge the gap between high level numerical simulation results on threshold monitoring and real IoT environments.
2) We study the algorithmic implementation and the actual performance on a real deployment, using real data sets and offer new insights. Specifically:

- We show that the practical energy lifetime improvements may vary significantly and are often far from the savings estimated analytically. Specifically, we find that the overhead of idle listening is a dominant factor that can sometimes limit the effectiveness of the monitoring algorithm.
- Communication patterns under GM vary greatly over time, with periods of no activity and bursts of concurrent updates. This presents a challenge to the underlying networking protocol.

II. Applied GM perspectives and algorithmic implementation on Wireless IoT Sensors

For GM-based continuous threshold monitoring in IoT environments, we argue that focal points are: (i) Design challenges from the application's point of view and (ii) System properties and parameters affecting the design.

A. Addressing communication challenges

Multi-hop, all-to-all communication: In traditional WSN communication scenarios, either all nodes send to a single, fixed node (data collection) or all traffic is disseminated from a single, fixed node to all the others (data dissemination). With GM's communication requirements, every node can potentially be a source of information that needs to be disseminated to all other nodes (all-to-all communication) and even concurrently with other nodes (as shown in § III). In addition, sensor networks are commonly multi-hop, so an individual update generated by a single node needs to be propagated in a reliable manner to all nodes.

We consider mesh, unstructured networks that follow a simple approach for multi-hop propagation: every node that receives a packet with new information will broadcast it further on. Obviously, this leads to an increased number of broadcasts for every update. This is a commonly considered baseline, motivated by its inherent property that the update will eventually propagate throughout the network with a high degree of reliability, without the need to maintain a routing topology. As the goal of GM is to reduce the number of updates that need to be propagated, we expect that network-wide flooding of updates will not happen often. We evaluate this further in § III.

Recovery from losses. The related literature on GM and similar methods does not typically consider packet losses. In WSNs, losses are common and an important consideration for application design. If an update from node A fails to reach B, node B will have stale information and the estimate vector will be *out of sync* (with respect to A), until the next update from A. Such a node has an inaccurate view of the network-wide aggregate being monitored and might miss a threshold violation or report a non-existing one. We allow updates to get lost and rely on the application layer to eventually converge to the correct estimate vector.

B. Tunable parameters of the network stack

We implemented geometric monitoring in Contiki [3], a well-known operating system for IoT applications. We targeted the TelosB platform and used a mainstream stack that relies on ContikiMac for radio duty cycling. That means that nodes will mostly have their radio off and only periodically turn it on to check if there is any traffic to receive. In this setting, the period at which the nodes check the channel for transmissions (*channel check rate*, CCR) is the main parameter of interest.

CCR affects: (i) how often nodes wake up to check the medium for possible transmissions and (ii) for how long a broadcasting node should keep re-transmitting a packet to make sure that all nodes receive it.

As the main goal of GM is to reduce the number of inter-node updates, one would generally expect GM to benefit from lower CCR values, compared to a naive approach that shares every sensor reading. On the other hand, as we show later in § III, GM's communication behaviour is highly data-dependent and unpredictable. There are periods with high activity and low activity, depending on the monitored variable and how close its value is to the threshold. In addition, when an update is received by a node, the resulting recalculated estimate might also cause a threshold violation, forcing the node to broadcast its readings immediately, creating periods of burst traffic.

Based on the above, it is clear that: (i) CCR is a system parameter that will affect the expected energy savings of the method, and (ii) it is hard to find a suitable CCR that can match the communication of GM at all times. We evaluate this further in § III.

III. Evaluation from a holistic system perspective

In this section, we assess the performance of GM in practice over a real network stack.

Experiment Setup: We run our experiments in two settings: (i) A *full-system evaluation* on the Flocklab [10] testbed, a deployment of 26 TelosB nodes in a university building with a four hop topology. (ii) A *full-system simulation* on Cooja [11], a cycle-accurate simulator where the *whole network stack* is simulated in software at every node. The topology here is similar to the testbed (26 nodes, 4 hops).

Data set: We use the Intel Lab data set [1], commonly used in the WSN literature. We select 20 hours of temperature readings from 26 nodes, using the original sampling frequency (31 seconds).

Monitoring functions: We experiment with both linear and non-linear monitoring functions, namely the global *average* and *variance* of the sensor readings. For the average we choose a threshold of $T = 20°C$ and for the variance a threshold of $T = 2°C^2$, which is crossed twice during the daily cycle. Unless stated otherwise in the description of the experiment, the variance is used.

As a comparison, we adopt the *baseline* method (also used in [14]) where nodes broadcast at every *epoch*, i.e. every sensor reading, as soon as they get it.

A. *Full-system simulations*

In the full system simulation, 20 hours of data are used for each configuration. We collect results both for the geometric method and for the baseline. In every configuration, the GM method achieves **4.31 times** communication reduction when monitoring the variance (Table I, col 9) i.e. only 23.2% of the sensor readings are actually propagated.

Duty cycle: In Table I we report the measured duty cycle as the CCR value ranges from 8 to 64 Hz, showing that GM results in significant reduction in duty cycle. As an example (Table I, col 5, CCR=12 Hz), using GM reduces the duty cycle from 7.73% to just 2.57%, a three-fold improvement. However, this improvement diminishes as the CCR increases. The lifetime improvement between the best configurations is 2.8x (compare duty cycles between 12 Hz for GM and 24 Hz for the baseline), which is far from the 4.31x communication reduction achieved by the method.

A brief look at the respective results from monitoring the *average* (Table I, col 10), shows that the effects mentioned above for the variance are even more pronounced now. In this case, GM manages to reduce communication by 44 times,

Intel Lab Dataset (20 hours), Monitored Functions: **Variance** (Threshold T=2) and **Average** (Threshold T=20)													
Channel Check rate (CCR) in Hz	Baseline (variance/average) [1]			GM, Function: variance					GM, Function: average				
	Duty cycle	Loss rate (%)	Latency (ms)	**Duty cycle**	Loss rate (%)	Latency (ms)	Lifet. Impr.	Comm. Red.	**Duty cycle**	Loss rate (%)	Latency (ms)	Lifet. Impr.	Comm. Red.
8	8,44	9,98	18348	2,68	1,19	3862	3,15x		0,76	0,46	3716	11,04x	
12	7,73	0,59	760	2,57	0,22	590	3,01x		1,03	0,41	2465	7,50x	
16	7,28	0,18	314	2,70	0,11	336	2,69x		1,37	0,02	2078	5,31x	
24	6,88	0,07	189	3,11	0,07	212	2,21x	4,31x	1,96	0,20	1078	3,51x	44x
32	7,26	0,04	162	3,70	0,04	190	1,96x		2,68	0,97	906	2,71x	
48	7,81	0,12	139	4,88	0,12	151	1,60x		3,85	0,45	581	2,03x	
64	9,11	0,14	134	6,18	0,12	157	1,47x		5,31	1,01	703	1,71x	

TABLE I

FULL SYSTEM SIMULATIONS: DUTY CYCLE, LOSS RATE AND LATENCY FOR THE GM METHOD VS THE BASELINE, WITH VARYING CCR.

Flocklab Testbed, Monitored Function: **Variance**								
		Duty Cycle (%)		Communication Reduction	Lifetime Improvement	Loss rate (%)		Latency(ms)	
CCR	Dataset section	GM	Baseline			GM	Baseline	GM	Baseline
12	low comm. (0:00 - 03:00) high comm. (8:00 - 12:00)	1,33 4,28	6,98	45,74x 2x	5,26x 1,63x	0,45 0,88	5,43	908 634	1704
24	low comm. (0:00 - 03:00) high comm. (8:00 - 12:00)	2,05 5,50	6,42	45,74x 2x	3,13x 1,17x	0,91 0,49	0,84	223 262	268

TABLE II

TESTBED EXPERIMENTS: RESULTS FROM THE FLOCKLAB TESTBED, ON TWO 3 HOUR PERIODS FROM THE DATASET.

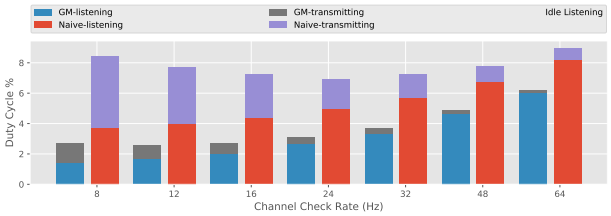


Fig. 1. **Full system simulations**: The duty cycle, broken down to sending and listening, as well as the cost of idle listening.

keeping nodes mostly quiet throughout the execution. In terms of duty cycle, GM reduces it by an impressive amount (up to 11 times for a CCR value of 8 Hz), but still, 4 times less than the achieved reduction in communication.

Duty cycle decomposition: A detailed look on the duty cycle *explains the aforementioned differences*. Figure 1 shows the duty cycle for GM and the baseline method (when tracking the variance), as well as its individual components. First, the percentage of the duty cycle that is spent on sending is greatly reduced using the GM method, directly matching the communication reduction ratio achieved by the algorithm. Subsequently, the GM version spends less time receiving data at each node. Also, notice that the time spent on transmitting decreases as the channel check rate grows. This is simply because broadcasts are shorter when the CCR is high. In this figure, we have also included the cost of idle listening, i.e. the cost of turning on the ratio periodically to check for traffic, even though there is nothing to receive. This cost is the same for both methods and is computed from the CCR value. It is evident that this cost dominates the duty cycle when the channel check rate increases. Even for small CCRs, *the idle cost represents a significant overhead*, that reduces the potential lifetime savings of the algorithm.

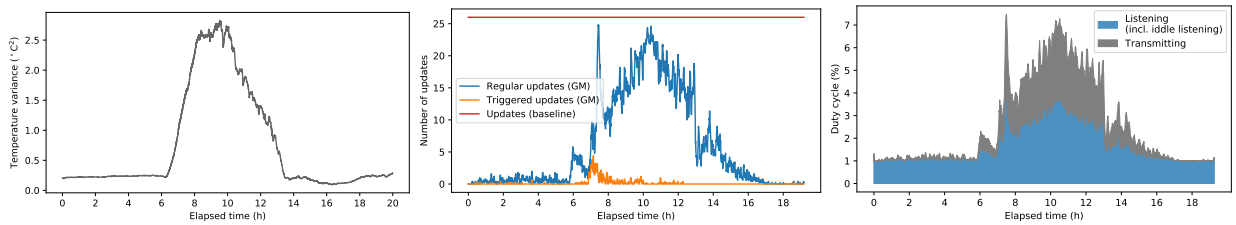
[1]The baseline method behaves the same way, regardless of the function.

B. *Validation through Testbed Experiments*

Goal: We use the testbed experiments of this section as a way to validate the insights and trends gained from the full-system simulations that we presented above.

Experiment settings: In this section, we present the results from the execution on the Flocklab testbed. Due to usage restrictions on the testbed, we do not replay full day measurements from the dataset. Instead, we focus on sections of particular interest. We select two sections from the data set, 3 hours each (midnight and morning) where, as we detail further in the next experiment series, the communication pattern of the GM method is expected to be very different. For these experiments, we have picked a channel rate of 12 Hz, where the GM method had the lowest duty cycle on the simulations, as well as a rate of 24 Hz for comparison.

Results: Table II shows the overall results for the two sections of the dataset. The topology of the testbed is slightly different than the one used in the simulation (more sparse), so the absolute values are different than Table I, but we expect the general trends to hold. On the morning section (08:00 to 12:00), GM communicates 2 times less than the baseline. The lifetime improvement follows closely, and the duty cycle is reduced by 1.63 times. For the midnight section (00:00 to 03:00), GM achieves remarkable communication savings, reducing the number of readings that need to be propagated by 46 times, but the associated lifetime improvement is more modest (5.26 times). This indicates that, during this section (and unlike the previous one), even a check rate of 12 Hz is excessively high, and *most of the energy is spent on idle listening*. Similar results can be seen when CCR is set to



(a) Variance of the temperature between nodes. (b) Number of updates per epoch. (c) The average duty cycle across nodes, during the execution.

Fig. 2. **Runtime insights** from the execution of GM over a period of 20 hours.

24 Hz. Here, the lifetime improvements decrease for both data set sections, especially for the period with low communication (3.13x lifetime improvement).

C. *Runtime insights: a closer look*

We now take a closer look into a single experiment and provide insights into the communication behaviour of the algorithm. We set the CCR to 12 Hz (that resulted in the best duty cycle for the GM case) and elaborate on detailed observations from GM, in order to distill deeper insights about the interplay between GM and the communication stack.

Monitored value: Figure 2a shows the actual value that is being monitored: the variance of the temperature readings. Due to variation in the temperature between different rooms during working hours, the data set exhibits a period of approximately 7 hours where readings between nodes have increased variance, up to 2.8 $°C^2$.

Temporal variation in comm. reduction: Figure 2b shows the number of updates per epoch (31 seconds), for the baseline and GM, computed over a 1.5 minute sliding window and averaged across all nodes. The baseline induces the same number of updates per epoch, equal to the number of nodes. On the contrary, the GM method significantly reduces the number of sensor readings that need to be updated per epoch. Especially during the periods when the variance is small and away from the threshold, almost all communication is suppressed. As the variance comes closer to the specified threshold, nodes start detecting frequent violations and update more of their readings. We can also see a period close to the threshold where these updates trigger violations on other nodes (triggered updates), and a peak in the number of updates when the actual value of the variance is close to the threshold of 2 $°C^2$.

Temporal variation in duty cycle: In Figure 2c we take another look at the duty cycle and monitor how it changes during the execution. The duty cycle is computed at every epoch and averaged across all nodes. The duty cycle follows the same trend as the number of updates in Figure 2b: at periods where communication is high, the radio needs to stay on longer in order to send or receive the extra traffic. From Figure 2c, it is also evident that *the idle listening cost is a dominant factor* that affects the duty cycle and limits the potential of the GM method: even during periods with no activity, nodes waste a constant amount of time to check the radio for transitions.

IV. Conclusions and Future Work

We take a full-system approach studying Geometric Monitoring in IoT deployments. Our results confirm significant reduction in communication, but also show that the lifetime improvements on the nodes can differ from the expected savings from communication reduction; this is due to baseline energy overhead of the network stack. Our described insights may enable the design of custom protocols that will unlock more of the potential of GM and related algorithms for IoT deployments.

References

[1] P. Bodik, W. Hong, C. Guestrin, S. Madden, M. Paskin, and R. Thibaux, "Intel Lab Data," 2004. [Online]. Available: http://db.csail.mit.edu/labdata/labdata.html

[2] S. Burdakis and A. Deligiannakis, "Detecting Outliers in Sensor Networks Using the Geometric Approach," in *IEEE International Conference on Data Engineering*, 2012.

[3] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki - a lightweight and flexible operating system for tiny networked sensors," in *29th IEEE Int'l Conf. on Local Computer Networks*, 2004.

[4] L. Feinstein, D. Schnackenberg, R. Balupari, and D. Kindred, "Statistical approaches to DDoS attack detection and response," in *Proc. DARPA Information Survivability Conference and Exposition*, 2003.

[5] Z. Fu, M. Almgren, O. Landsiedel, and M. Papatriantafilou, "Online temporal-spatial analysis for detection of critical events in cyber-physical systems," in *IEEE Int'l Conf. on Big Data*. IEEE, 2014, pp. 129–134.

[6] M. Garofalakis, "Approximate geometric query tracking over distributed streams," *IEEE Data Eng. Bull.*, 2015.

[7] M. Garofalakis, D. Keren, and V. Samoladas, "Sketch-based Geometric Monitoring of Distributed Stream Queries," *Proc. VLDB Endow.*, 2013.

[8] N. Giatrakos, A. Deligiannakis, M. Garofalakis, I. Sharfman, and A. Schuster, "Prediction-based geometric monitoring over distributed data streams," in *ACM SIGMOD Int'l Conf. on Manag. of Data*, 2012.

[9] V. Gulisano, M. Almgren, and M. Papatriantafilou, "METIS: a two-tier intrusion detection system for AMI," in *Int'l Conf. on Security and Privacy in Comm. Sys.* Springer, 2014, pp. 51–68.

[10] R. Lim, F. Ferrari, M. Zimmerling, C. Walser, P. Sommer, and J. Beutel, "Flocklab: A testbed for distributed, synchronized tracing and profiling of wireless embedded systems," in *2013 ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2013.

[11] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with cooja," in *31st IEEE Conf. on Local Computer Networks*, 2006.

[12] U. Raza, A. Camerra, A. L. Murphy, T. Palpanas, and G. P. Picco, "Practical Data Prediction for Real-World Wireless Sensor Networks," *IEEE Trans. on Knowledge and Data Eng.*, 2015.

[13] I. Sharfman, A. Schuster, and D. Keren, "Aggregate threshold queries in sensor networks," in *IEEE Int'l Parallel & Distr. Process. Symp.*, 2007.

[14] I. Sharfman, A. Schuster, and D. Keren, "A geometric approach to monitoring threshold functions over distributed data streams," in *ACM SIGMOD Int'l Conf. on Management of Data*, 2006.