Event-Based Communication for IoT Networking

Panayiotis Kolios, Christos Panayiotou, Georgios Ellinas, and Marios Polycarpou

Abstract—**In the Internet of Things (IoT), smart devices embedded on a myriad of different objects will enable monitoring, control, and optimization with potentially transformative impact to the society. Evidently, operational efficiency has become an essential factor in the proliferation of these devices since higher efficiency will prolong device lifetime and achieve greater autonomy.**

In this work, the operational efficiency challenge is being looked at from the device point of view as well as from the networking side of things in which local and remote hosts need to exchange information for monitoring, control, and optimization functionalities. In the former case, a data-driven event triggering technique is developed to minimize the interaction between devices, while in the latter case an event-based communication strategy is investigated to minimize the overall energy consumption of the network. Both experimental results and results for a real-world application are presented to demonstrate the manyfold operational efficiency gains that can be realized using the proposed solution.

Index Terms—**Internet of Things, Event-triggering, Monitoring, optimization and control**

I. Introduction

IoT is a technology that promises to revolutionize social and economic activity by enabling horizontally integrated productivity solutions as opposed to the current vertical paradigm [1]. To do so, IoT devices combine computing capabilities with sensing, communication, and actuation capabilities to: a) collect and exchange information in order to gain new insights on processes/systems (i.e., monitoring), b) take preemptive or responsive actions based on feedback (i.e., control), and c) review and revise the employed activities when deemed necessary (i.e., optimize) [2] [3].

The increased complexity associated with these three tasks together with the necessity for greater autonomy, of mostly battery-powered embedded devices, calls for higher efficiency across all executed operations. For instance, upcoming communications standards (such as the 802.11ah [4] and LTE-MTC [5]) that are specifically designed for IoT applications define active/sleep states to enable sporadic exchange of information and extended power-down periods to conserve battery capacity. This intermittent operation has also attracted considerable attention by the research community that is currently investigating possible event-triggering techniques to reduce sensing and actuation tasks as well [6] [7] [8]. As opposed to continuous operation, and even periodic triggering, event-triggering deals with actions that take place only when specific events have occurred. These events could either be due to simple discrete state changes, predefined interrupt sequences, or spontaneous violations to some set threshold. Most often the choice is governed by the degree of predictability of the system to be managed:

1) Discrete state changes are employed for systems with unpredictable behavior,
2) Interrupt sequences are designed for systems that can be precisely expressed mathematically,
3) Threshold techniques are developed for systems that exhibit behavioral patterns which adequately reflect recurrent behavior. Events are triggered whenever the actual observations made exceed the set thresholds defined by the system model.

The third category of systems appeals to many real-life settings that experience diurnal patterns, including daily commuting and home dwelling. Importantly, IoT devices have dramatically simplified the collection and analysis of data for the extraction of the underlying behavior patterns, and have enabled the design of new data-driven event-triggering approaches for monitoring, control, and optimization [9].

In accordance, the primary focus of this work is on the study of systems that exhibit recurrent patterns. Individual IoT devices are deployed to conduct the aforementioned management actions and to inform remote hosts of any unanticipated changes. To conserve resources, these devices are assumed to operate in duty cycles, i.e., they sporadically get activated to execute their actions and power down computing and communication circuitry for the rest of the time.

Under this regime, a novel data-driven event-triggering technique is developed based on behavior models (to capture the underlying patterns that suffice). The proposed technique considers the tradeoff between estimation accuracy and volume of event triggers generated. Obviously, fewer event triggers result in less information being exchanged by local and remote hosts, and reduce the energy consumption of each individual host (both for processing the interrupts and for communicating the events between the local and remote hosts). Moreover, the overall energy consumption of the emerging event-based communication paradigm (from inter-networking devices employing event triggering) is also studied. Notably, the idlying energy consumption while waiting for a remote host to wake up may prolong the active duration and reduce energy savings. In addition, the end-to-end delay in exchanging event interrupts may be unacceptable for different applications. Both these challenges are addressed in the sequel.

In the following section, related work is reviewed and background material is included. Section III develops the proposed data-driven event-triggering technique and demonstrates its applicability in a real-world public transport scenario. Section IV studies the derived event-based communication paradigm and conducts extensive experimental results to study the

P. Kolios, C. Panayiotou, G. Ellinas, and M. Polycarpou are members of the KIOS Research Center for Intelligent Systems and Networks and the Department of Electrical and Computer Engineering, University of Cyprus, e-mail:{pkolios, christosp, gellinas, mpolycar}@ucy.ac.cy

performance of this networking approach. Finally, Section V concludes this work with key insights obtained and elaborates on future research efforts.

II. Background and Related Work

Event-triggering is becoming an increasingly popular computation and communication approach especially for embedded and mobile devices where resources must be thriftily used. The potential impact of this approach is already experienced on consumer electronics such as smartphones and tablets that advertise standby times that are several orders of magnitude larger than active times. A typical smartphone, for instance, provides 10 hours of active Internet usage while standby time reaches 250 hours. Event-triggering tries to strike a balance between the two by performing necessary actions whenever particular events have occurred and power down at every other time.

Among the various event-triggering techniques that are explored in the literature, threshold techniques are the most popular for several reasons. Firstly, these techniques assume that a system model exists. Evidently, the majority of application scenarios have to do with systems that experience recurrent patterns which can be extracted using data analytics, and can be used to build a behavior model. Noticeably, extracting behavioral patterns is one of the main reasons why data analytics have been very popular in recent times, with a large volume of research work steered towards that direction (including the works in [10] and [11]). Secondly, thresholds can be purposefully designed to intelligently trade off accuracy with the number of events triggered and, thirdly, thresholds can be dynamically adapted to reflect changes in the system behavior.

Threshold techniques can be broadly classified into sample- and stream-based approaches. As their names suggest, sample-based thresholds deal with individual data elements while stream-based thresholds consider data streams and the properties that they entail. An early survey is provided in [12]. Limit checking is the simplest and most popular sample-based approach with events triggered whenever sample data exceeds the set limits. The alternative change checking approach triggers events based on the differences detected on the current sample from previously sampled data. Trend checking is one such approach whereby variations in the samples are used to create thresholds instead of the absolute values [13]. Doing so allows for earlier detection of events at the expense of increased sensitivity. Statistical values are also commonly used instead of absolute values in checking for changes [14]. The first few moments are frequently used in conjunction with statistical tests such as hypothesis testing and run-sum testing (as elaborated in [15]). The advantage of statistical values is that small changes can be detected more effectively without penalizing sensitivity.

As previously emphasized, stream-based approaches consider properties of data streams to set triggering thresholds. Signal processing is utilized for this category, whereby various different features are extracted and analyzed [16]. Transforms (including Fourier and Z transforms, wavelets, etc) are often used for feature extraction while statistical relationships are subsequently used for analysis. Another approach is process analysis of either single or multiple signals that deals with changes in the signals to detect unanticipated events [17]. This is mainly achieved by analytical redundancy, whereby various excitation inputs to the system are used to identify behavioral patterns. Correlation methods are frequently used to identify relevant patterns, in addition to linear and non-linear parameter estimation methodologies such as least squares and artificial neural networks [18]. Residuals of various forms are also used to describe the difference between the actual system behavior and that of the model [13]. These residuals are computed using parity equations (through transfer functions or state-space formulations) or state-observers (that compute the changes between the triggering events and the associated system behavior). Thresholds are then set based on the latest residual values. The main drawback of this approach is that residuals change continuously due to the inherent model uncertainty resulting in increased complexity.

Statistical analysis greatly simplifies feature extraction and analysis. Similar to statistical solutions for sample-based approaches, stream-based statistical analysis focuses on purely data-driven tools to detect unanticipated events [19]. Principal component analysis is one such tool, whereby a small number of uncorrelated variables (i.e., principal components) are used to describe a larger multi-variable model. The principal components maintain most of the information describing the system but greatly speed up the analysis.

In the sequel, statistical analysis is used to develop a novel data-driven event-triggering technique for IoT applications. This technique is used to intelligently generate triggering events that balance estimation accuracy to activity level. The resulting event-based communication paradigm (that suffices for networking devices operating based on event-triggering) is also investigated. Key insights are offered on the optimizations that need to take place in order to reduce the overall energy consumption when practical considerations are introduced.

III. Data-driven Event Triggering

The proposed technique first quantizes data streams into a number of consecutive segments, builds a model of the system based on statistical measures on these segments, and derives thresholds across the segments for event triggering. Depending on the application scenario, data streams could be split into segments of a specified dimensional space, with the temporal and spatial dimensions being the prominent two examples. It is assumed that recurrent patterns start at the first segment and end at the last segment of the data stream.

Let each segment (l, f) belong to the set $\mathcal{L} = \{1, \ldots, L\}$. Then, $t_{lf}(n), \{l, f\} \in \mathcal{L}$ is set to be the n^{th} data sample across segment (l, f). With a total of N available samples, the first moment $\theta_{lf}(n)$ can be calculated as follows:

$$\theta_{lf}(N) = \frac{1}{N} \sum_{n=1}^{N} t_{lf}(n), \, \forall \, \{l, f\} \in \mathcal{L} \tag{1}$$

and subsequent sample moments can be computed as $\theta^k_{lf}(N) = \frac{1}{N} \sum_{n=1}^{N} t_{lf}(n)^k$, for the k^{th} moment. Clearly, higher order moments provide better description of each segment's distribution. More importantly though is the fact that sample moments can be updated recursively with every new data sample collected and thus there is no need to collect and store raw data samples but instead the respective moments only. For the k^{th} sample moment, the updating step is derived as follows:

$$\theta_{lf}(N+1)^k = \frac{1}{N+1}\sum_{n=1}^{N+1}\left[t_{lf}(n+1)\right]^k \tag{2}$$

$$= \frac{1}{N+1}\left(\sum_{n=1}^{N}\left[t_{lf}(n)\right]^k + \left[t_{lf}(N+1)\right]^k\right) \tag{3}$$

$$= \frac{N\theta_{lf}(N)^k + \left[t_{lf}(N+1)\right]^k}{N+1} \tag{4}$$

As shown in eq. (4), the updating step requires knowledge of the latest moment value, $\theta_{lf}(N)$, and the actual number of samples N used to compute that value. Hence, each sample moment can be updated dynamically with every new sample using only these two elements. In addition, central moments can be computed from sample moments through the binomial transform, with the k^{th} central moment given as $\vartheta_{lf}(N)^k = \sum_{\kappa=0}^{k}\binom{k}{\kappa}(-1)^{k-\kappa}\theta_{lf}(N)^k(\theta_{lf}(N)^1)^{k-\kappa}$.

The proposed technique assumes that a system model consists of the first K moments for each segment of the data stream which are used to approximate the distribution of the data stream. This model is then used to calculate probability bounds that are to be used to set the event-triggering threshold. An event is triggered within a particular segment if the latest data sample exceeds the threshold. With $\theta_{lf}(N)$ being the first sample moment (i.e., the mean) as defined in eq. (1) for segment (l, f), then the upper and lower probability bound are given by $\theta_{lf}(N)+\alpha$ and $\theta_{lf}(N)-\alpha$ respectively, where α is the threshold value. An out-of-bound event is triggered whenever the data sample in segment (l, f) violates the probability bounds.

In the derivations that follow these threshold values are calculated for a target S of triggered events. The combination of event interrupts triggered across particular segments is given by the binomial coefficients in matrix $\mathbf{C} = \binom{L}{S}$ with rows identifying all possible combinations of choosing S number of segments out of the total number of segments L. Hence, entry $\mathbf{C}_{rs}$ of the matrix, determines the s^{th} segment ($s = \{1, \ldots, S\}$) where an event is triggered in the r^{th} possible combination of events ($r = 1, \ldots, R$ with $R = \frac{L!}{(L-1)!S!}$). An illustrative example of matrix $\mathbf{C}$ is shown below to aid understanding. For this example it is assumed that $L = 40$ and $S = 5$.

$$\mathbf{C} = \begin{array}{c|ccccc|} & \mathit{1} & \mathit{2} & \mathit{3} & \mathit{4} & \mathit{5} \\ \mathit{1} & 3 & 6 & 10 & 17 & 23 \\ \mathit{2} & 7 & 17 & 24 & 38 & 40 \\ & & & \ddots & & \\ r & 20 & 27 & 30 & 37 & 40 \\ & & & \ddots & & \end{array}$$

In this example, the 1st combination of events is assumed in segments 3, 6, 10, 17, and 23. At every other segment the accumulated values of the sampled data does not fall outside the thresholds and thus no event is triggered. Also, with every event trigger the uncertainty associated with the actual system state, vanishes. Then, the probability of consecutive events triggered between an arbitrary pair of segments can be expressed as follows:

$$p(l,f) = \bar{p}(l,l{+}1)\times\bar{p}(l,l{+}2)\times\ldots\times\bar{p}(l,f{-}1)\times(1{-}\bar{p}(l,f)) \tag{5}$$

where $\bar{p}(l, f)$ expresses the probability of no event triggers between segment (l, f). Probability $\bar{p}(l, f)$ is defined as follows:

$$\bar{p}(l,f) = p(\theta_{lf}(N)-\alpha \le S_{lf} \le \theta_{lf}(N)+\alpha) \tag{6}$$

Moreover, in addition to the probabilities between consecutive events, border conditions should also be incorporated. The leftmost border deals with the probabilities from the first segment up to segment l of the 1st event interrupt. Similar to (5), this probability can be expressed as follows:

$$p(1,l) = \bar{p}(1,2)\times\bar{p}(1,3)\times\ldots\times\bar{p}(1,l{-}1)\times(1{-}\bar{p}(1,l)) \tag{7}$$

This is to say that all remote hosts are informed of the start of a new phase by either a real event interrupt (from the local host) or a virtual indication from a predefined schedule. The rightmost border includes the probabilities for all segments that follow after the last event trigger. This probability can be expressed as follows:

$$p(f,L) = \bar{p}(f,f+1)\times\bar{p}(f,f+2)\times\ldots\times\bar{p}(f,L) \tag{8}$$

Then, the probability of all possible combinations of events is expressed as follows:

$$\grave{p} = \sum_{r=1}^{R}\prod_{s=1}^{S-1}\left[p(\mathbf{C}_{rs},\mathbf{C}_{rs+1})\right]\times p(1,\mathbf{C}_{r1})\times p(\mathbf{C}_{rS},L) \tag{9}$$

with $p(\mathbf{C}_{rs}, \mathbf{C}_{rs+1})$ defined by eq. (5), $p(1, \mathbf{C}_{r1})$ by eq. (7) and $p(\mathbf{C}_{rS}, L)$ by eq. (8). Since all possible combinations of events are included in (9), the total probability should be equal to $\grave{p} = 1$. It should be noted here that the only unknown parameter in this system of equations is threshold α in the bound described by eq. (6).

The bisection method [20] can be used to solve the system of equations in (5) in order to compute the unknown parameter α, as described by Algorithm 1 below. The bisection method assumes that the solution lies within the interval (a_ι, a_u) which is iteratively reduced to reach a favorable precision. To do so, the center point (i.e., $a = 0.5*(a_u+a_\iota)$) of this interval is first calculated. This value is subsequently used to evaluate eq. (9) and test feasibility. The center point a is then assigned to either a_ι or a_u depending on which section retains feasibility (i.e., $\grave{p} \le 1$). The procedure continues until a favorable precision ϵ is reached i.e., $a_u - a_\iota \le \epsilon$ and finally the threshold value α is returned.

In summary, the proposed technique uses the first K moments for each segment of a particular data stream to build a model of the system. Using these moments, a threshold α is computed to set a probabilistic bound that anticipates a total of

Algorithm 1 Bisection Method

```
Ensure: $a_\iota = 0,\ a_u = A$
 1: while $(a_u - a_\iota) > \epsilon$ do
 2:     $a = \frac{(a_u + a_\iota)}{2}$
 3:     Solve $\grave{p}$ in (9)
 4:     if $\grave{p} > 1$ then
 5:         $a_u = a$
 6:     else
 7:         $a_\iota = a,\ \alpha = a$
 8:     end if
 9: end while
10: return $\alpha$
```

S event interrupts triggered along the identified segments. As exemplified above, higher precision can be achieved by computing higher order moments for each segment and reducing the ϵ parameter in the bisection method.

With regards to the complexity of the proposed technique, there are $\frac{L(L-1)}{2}$ data sample calculations between all segments and K moment calculation updates (expressed in eq. (4) above) for each segment. With regards to bound updates, there are R combinations of events that need to be considered (as expressed by eq. (9) above) that exponentially increase with S. Finally, the bisection method has linear convergence with a total of $log_2(\frac{\epsilon_0}{\epsilon})$ iterations carried out, where ϵ_0 is the initial interval at the start of the iterative procedure.

A. *Public Transport Case Study*

To demonstrate the applicability of the proposed technique, a public transport scenario is considered in which vehicles operating along various service routes are remotely monitored. Current state-of-the-art solutions employ mostly periodic triggering whereby onboard units report location and mobility information to remote hosts at constant time intervals. Using the proposed technique, we investigate the tracking accuracy that can be achieved based on the proposed event-triggering approach and for varying number of target event triggers, S.

Specifically, bus route 150 of the Transportation Organization of Nicosia District (OSEL) in Cyprus, was considered. Road segments between consecutive bus stops on this route were used to quantize the data streams and onboard units were installed on all buses operating on the specific route to measure the travel times along consecutive road segments. The mobility model (consisting of travel times between consecutive road segments) was built using the first K=4 moments of the measured travel times. Of the 500,000 samples collected, 80% were used to build the mobility model and the rest were used to test the derived thresholds.

The proposed technique was implemented in Matlab, and for the bisection method parameter $A = 60$ minutes was used that covers all round trip times and a stopping precision $\epsilon = 1$ sec was set. Figure 1 depicts the tradeoff that exists between the event triggering volume and the length of the travel time thresholds. More specifically, Fig. 1a shows the threshold values obtained for a range of target event triggers while Fig. 1b provides box plots for the event triggers created

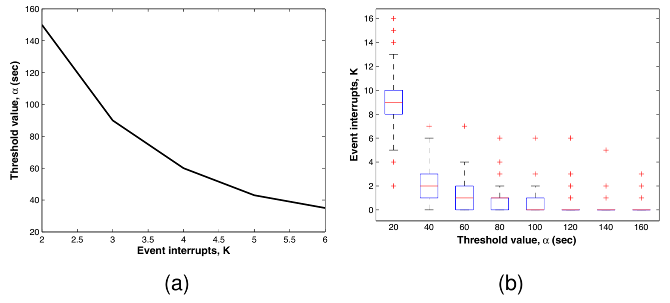

Fig. 1. a) Travel time thresholds for varying number of target event triggers, and b) experimental performance with the test sample.

based on the set threshold values when simulating the buses' mobility using the test sample.

As clearly shown in the figure, the threshold drops exponentially with increasing number of triggered events. For instance, a total of $S = 4$ target interrupts results in a threshold of around 1 minute. This is to say that of the total travel time of each bus along the particular service route that lasts approximately 50 minutes, only 4 event interrupts need to be handled in order to maintain an estimated tracking accuracy of 1 minute. This is also demonstrated with the simulation results depicted in Fig. 1b. Compared to periodic triggering with interrupts triggered every minute during the entire trip duration, event-triggering achieves a $12\times$ gain.

As emphasized above, the reduction in triggering events allows devices to power down for longer periods in order to conserve battery and minimize the wireless channel utilization, since only intermittent connectivity is needed. In practice however, remote hosts may consume significant energy while listening over the wireless channel for a specific local host to wake up and exchange information. In the following section, the networking aspects of event-based communication are detailed and the transmit/idling tradeoff that arises is thoroughly investigated.

IV. Event-based Communication

As previously emphasized, the key benefit of event-triggering is that devices sporadically wake up to process and communicate event interrupts and at every other time unutilized computing and communication circuitry powers down to conserve resources. This operation is in contrast to current Internet practices where devices, have perpetual availability and communication can be instantaneously be established between any pair of nodes in the network.

Of course, several techniques have been introduced to date to achieve energy savings based on periodic triggering. For instance, the IEEE 802.3az standard [21] for energy-efficient Ethernet powers down the transmitters when no exchange of data is necessary and periodically probes the communication link to check if there is data waiting for exchange. A similar strategy is also employed in the upcoming IEEE 802.11ah standard for the IoT where extended power down periods are defined. However, as shown in the previous section, event-triggering can achieve significantly higher gains compared to periodic triggering and, as such, event-based communication is an excellent candidate for future IoT applications.

The difference between periodic and event triggering is that in the former case there is a fixed predetermined period of time that devices need to wait before the host becomes active again to be able to exchange data. In the latter case, the duration of time that a local host needs to wait (until a remote host becomes available) is randomly distributed and only probabilistically bounded (as previously discussed). Hence, the energy consumed by an active device waiting for a particular remote host to wake up can grow substantially and thus this overhead needs to be considered in the overall energy budget.

Let τ be the transmit time duration in communicating an event interrupt. The mean waiting time incurred while waiting for an event interrupt to be triggered at remote host j is $\frac{A}{2\times s_j}$, where A is the total duration of each monitoring phase (similar to the previous section), and s_j is the set number of event triggers by remote host j. Evidently, the latter calculation assumes that event triggers follow the uniform distribution across all segments. Then, the total energy consumption of local host i in the active state that needs to communicate an event to remote host j is given by eq. (10):

$$e(i,j) = s_i(\frac{A}{2\times s_j} + \tau)P_{TX} \tag{10}$$

with P_{TX} being the transmit power used by the communications circuitry. As shown in [22], the power consumed by communications circuitry on mobile devices is approximately the same for both listening and transmitting data. Thus, only P_{TX} is necessary in the energy budget expressed by eq. (10). Equation (10) demonstrates that waiting times decrease with an increasing number of event interrupts that are triggered by remote hosts (effectively forcing them to wake up and exchange data). At the same time, an increasing number of event triggers raises the energy consumption of the local host. Hence, the challenge is to devise intelligent ways of selecting the volume of events that should be triggered (and the respective thresholds) by each device in order to minimize the overall energy consumption.

The networking scenario considered hereafter assumes that each host $i \in \mathcal{V} = \{1, \ldots, V\}$ wants to communicate its event interrupts to a subset $\mathcal{Z}(i) \subset \mathcal{V}$ of remote hosts. Upon an event trigger, the local host wakes up and probes connection requests to a particular remote host until that host wakes up and establishes communication. The local host powers down immediately after data exchange. Problem (P1) derives a fractional integer programming formulation to find the number of event interrupts s_i that each device should trigger in order to minimize the overall energy consumption.

$$\text{(P1)} \min \sum_i \sum_{j\in\mathcal{Z}(i)} e(i,j) \tag{11}$$

$$\text{s.t. } s_i \geq S(i),\ \ s_i \in \mathbb{Z},\ \ \forall i \in \mathcal{V} \tag{12}$$

where the $e(i,j)$ term in the objective function is expressed in eq. (10) and $S(i)$ is the minimum number of event triggers necessary at node i to meet the target threshold values of the system model, as expressed in the previous section. It should be emphasized here that problem (P1) assumes that node i has direct access to each of the remote hosts in $\mathcal{Z}(i)$. When a multi-hop path from source $i \in \mathcal{V}$ to destination $j \in \mathcal{Z}(i)$ is necessary to reach j, then the end-to-end delay in waiting for intermediate nodes to wake up (with set $\mathcal{H}(i,j)$ containing all nodes in the end-to-end path) equals $\sum_{h\in\mathcal{H}(i,j)} \frac{A}{2\times s_h}$. To ensure timely delivery of event interrupts, this delay is usually bounded by some deadline D_{ij}. Problem (P2) is an extension to problem (P1) whereby this deadline constraint is also considered.

$$\text{(P2)} \min \sum_i \sum_{j\in\mathcal{Z}(i)} e(i,j) \tag{13}$$

$$\text{s.t. } \sum_{h\in\mathcal{H}(i,j)} \frac{A}{2\times s_h} \leq D_{ij} \tag{14}$$

$$s_i \geq S(i),\ \ s_i \in \mathbb{Z},\ \ \forall i \in \mathcal{V} \tag{15}$$

A. Performance Analysis

To study the performance of event-based communication a network of $V = 30$ nodes is considered and the set $\mathcal{Z}(i)$ is randomly populated. A fixed transmit power is assumed for all devices at $P_{TX} = 0.1$W and $A = 60$ minutes as used in Section III-A. $S(i),\ \forall i \in \mathcal{V}$ is assumed to be uniformly distributed in the interval $U(1,10)$. Both (P1) and (P2) have been implemented in Matlab and Yalmip's BnB solver has been used to compute the optimal solution. For each scenario depicted below, 1000 Monte Carlo simulations were conducted.

Figure 2 depicts average values obtained by solving (P1) to optimality by varying τ and for varying cardinality of set $\mathcal{Z}(i)$. The plots in Fig. 2a depict the normalized energy consumption of the whole network against varying values of τ, the plots in fig. 2b depict the volume of event interrupts that are in excess of the minimum target $S(i)$ of each node, and the plots in Fig. 2c depict the total number of nodes for which the volume of event triggers have been raised above $S(i)$ in order to minimize the total energy consumption for the entire network.

As expected, the energy consumption increases substantially with τ and the cardinality of set $\mathcal{Z}(i)$. More interestingly, the excess volume of event triggers (as shown in Fig. 2b) varies substantially with τ. Smaller transmission times force nodes to increase their triggering volume in order to minimize waiting time while higher transmission times force nodes to remain active for longer periods and thus the waiting time diminishes (and so does the need to raise the volume of event triggers above $S(i)$). The effect on the cardinality of set $\mathcal{Z}(i)$ is less severe for this networking scenario. As shown in the figure, the plot with cardinality 1 has marginally higher accumulated volume of event triggers compared to the rest of the cases while the difference diminishes for higher cardinality values. At the same time, a comparison between Figs. 2b and 2c shows that lower cardinality in set $\mathcal{Z}(i)$ causes a few nodes to raise their volume of event triggers substantially higher than the rest.

Similar trends are observed by solving problem (P2) as well and thus the results are omitted. However, the key difference

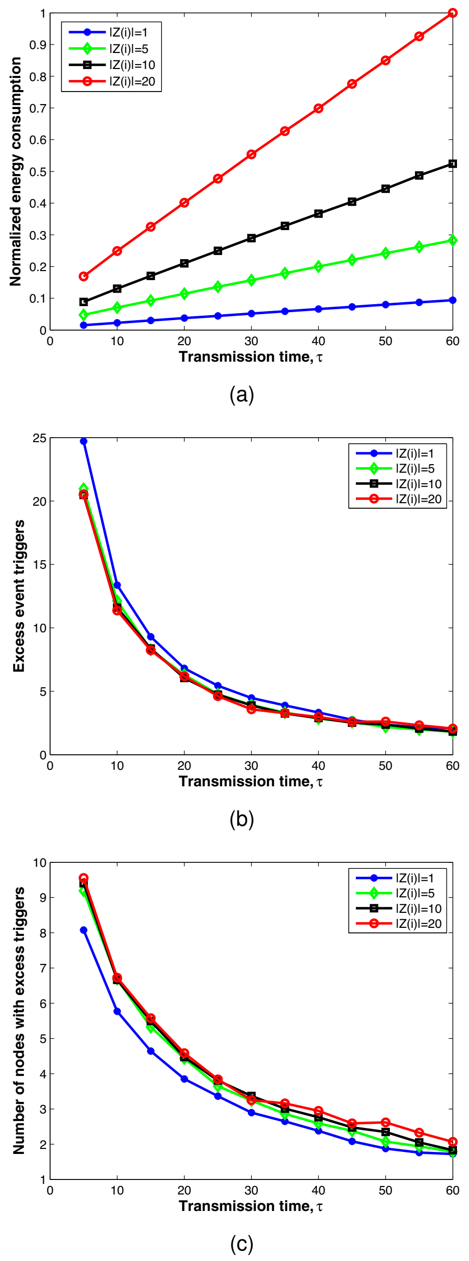

Fig. 2. Performance analysis of problem (P1) showing (a) the normalized energy consumption in the network, (b) the accumulated volume of excess event triggers, and (c) the total number of nodes with excess event triggers.

is that a larger number of event interrupts are triggered as D_{ij} (i.e., the end-to-end deadline) is reduced. This is due to the fact that more frequent triggers reduce waiting times for the exchange of events and thus achieve lower end-to-end communication delays.

V. Conclusions

Event triggering is well suited for a broad range of IoT solutions that employ embedded devices with limited resources. In this work, special emphasis is given in application scenarios where IoT devices are used to monitor, control, and optimize the operation of systems that experience recurrent patterns. A novel data-driven event-triggering technique has been developed to extract recurrent patterns and to enable event triggering that trades off estimation accuracy with the number of triggering events. The great potential of the proposed technique has been demonstrated for a real-world application (tracking the mobility of public transport buses).

Moreover, the paper considers the networking challenges that emerge when devices (that employ event-triggering) are interconnected. Both the energy-efficiency aspects and the quality of experience perceived by devices have been analyzed and mathematical programming formulations were derived to enable optimized parameter setup. Future work will delve further into the connectivity aspects of this novel networking approach and experiment with the applicability of event-based networking for different IoT solutions.

Acknowledgments

This work is supported by the European Research Council under the Advanced Grant FAULT-ADAPTIVE ERC-2011-AdG-291508.

References

[1] J. Rifkin, *The Zero Marginal Cost Society: The Internet of Things, the Collaborative Commons, and the Eclipse of Capitalism*, Palgrave Macmillan, Apr. 2014.
[2] GSMA, “Connected Living Linking the Physical and Digital Worlds”, *The Mobile Economy 2014*, 62-64, April 2014.
[3] J.A. Stankovic, “Research Directions for the Internet of Things”, *IEEE Internet of Things Journal*, 1(1):3-9, Feb. 2014.
[4] E. Khorov, “Low Power Wi-Fi: How IEEE 802.11ah is Transforming M2M (Tutorial)”, *IEEE ISWCS*, Aug. 2014.
[5] Qualcomm, “LTE MTC: Optimizing LTE Advanced for Machine-Type Communications”, www.qualcomm.com/lte-mtc, Nov 2014.
[6] C. Cassandras, “Event-driven Control, Communication, and Optimization”, *Chinese Control Conference*, July 2013.
[7] M. Lemmon, “Event-Triggered Feedback in Control, Estimation, and Optimization”, *Networked Control Systems - Lecture Notes in Control and Information Sciences*, 406:293-358, 2010.
[8] W. Heemels, K.H. Johansson, and P. Tabuada, “An Introduction to Event-triggered and Self-triggered Control”, *IEEE Conference on Decision and Control*, Dec. 2012.
[9] O. Vermesan, and P. Freiss, *Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems*, River Publishers Series in Communications, 2014.
[10] H. Hu, Y. Wen, T.S. Chua, and X. LI, “Toward Scalable Systems for Big Data Analytics: A Technology Tutorial”, *IEEE Access*, 2:652-687, June 2014.
[11] X. Wu, X. Zhu, G.Q Wu and Wei Ding, “Data Mining with Big Data”, *IEEE Trans. on Knowledge and Data engineering*, 26(1):97-106, Jan. 2014.
[12] M. Basseville, “Detecting changes in signals and systems- a survey”, *Automatica*, 24(3):309-326, 1988.
[13] R. Isermann, *Fault-Diagnosis Systems: An Introduction from Fault Detection to Fault Tolerance*, Springer, 2006.
[14] H. Stark, and J. Woods, *Probability, random processes and estimation theory for engineers*, Prentice Hall, 1994.
[15] H. Akaike, “A new look at the statistical model identification”, *IEEE Trans. on Automatic Control*, 19(6):716-723, 1974.
[16] S. Stearns, *Digital signal analysis*, Hayden Book Company, 1975.
[17] D. Hall, and J. Llinas, *Handbook of Multisensor Data Fusion*, CRC Press LLC, 2001.
[18] X.W. Chen, and X. Lin, “Big Data Deep Learning: Challenges and Perspectives”, *IEEE Access*, 2:514-525, May 2014.
[19] K. Slavakis, G.B. Giannakis, and G. Mateos, “Modeling and Optimization for Big Data Analytics”, *IEEE Signal Processing Magazine*, 31(5):18-31, Sept. 2014.
[20] R.L. Burden, J. D. Faires, “The Bisection Algorithm”, Numerical Analysis (3rd ed.), PWS Publishers, 1985.
[21] K. Christensen, et al., “IEEE 802.3az: the road to energy efficient ethernet”, *IEEE Communications Magazine*, 48(11):50 - 56, Nov. 2010.
[22] K. Pentikousis, “In Search of Energy-Efficient Mobile Networking”, *IEEE Communications Magazine*, 48(1):95 - 103, Jan. 2010.