# Data-Driven Event Triggering for IoT Applications

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

***Abstract*—Event-triggering (ET) is an up-and-coming technological paradigm for monitoring, optimization, and control in the Internet of Things (IoT) that achieves improved levels of operational efficiency. This paper first defines the envisioned event-triggering architecture for the IoT domain. It then classifies and reviews the various different event-triggering approaches obtained from the available literature for the three phases of ET, namely behavior modeling, event detection, and event handling. Thereafter, a novel data-driven technique is developed to address all three phases of ET in an efficient and reliable manner. Finally, the applicability of the proposed data-driven technique is showcased in a real-world public transport scenario, demonstrating a substantial improvement in energy and spectrum efficiency compared to existing periodic techniques.**

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

## I. Introduction

The Internet of Things (IoT) has recently received significant attention by researchers and industrial companies alike. This is clearly due to its potential to offer new monitoring capabilities on processes/systems, to optimize operations, especially for cost savings, and to improve operational resilience, [1], [2].

To facilitate the interaction between the physical and the cyber space, IoT devices need to support a variety of sensors and actuators in addition to customary processing, storage, and communication circuitry [3], [4]. Already, the plethora of consumer electronics such as smartphones, tablets, and wearables (currently available in the market) boast several different sensors (including accelerometer, gyro, proximity, compass, barometer, microphone, and camera) and actuators (including display, vibrator, and speaker). Interestingly, recent efforts to modularize smartphone hardware (primarily to meet the diverse consumer needs) has demonstrated the potential of smartphone platforms to support IoT applications [5]. However, contrary to the design priorities for consumer electronics, IoT devices aim to support applications and services that have ubiquitous presence in the world [6], [7]. And to do that, a very large number of IoT devices needs to be deployed and operated with minimal human intervention [8].

The latter design priority raises significant challenges with respect to the devices' operational efficiency (as exemplified in [9]). Firstly, most IoT devices are expected to operate on batteries or with limited power supply for extended time periods. Thus, energy efficiency becomes a crucial design consideration. Secondly, the necessity for a wide range of capabilities may translate to higher computational complexity which in turn can influence battery lifetime. Hence, intelligent algorithmic solutions are needed to improve computational efficiency and maintain performance. Thirdly, the inherent inter-networking between these devices will dramatically increase contention over the available communication channels, as shown in [10]. Therefore, curbing data exchange will improve channel utilization and further help minimize the energy expenditure.

Event-triggering (ET) is a promising paradigm that has the potential to adequately address the aforementioned challenges [11]. With ET, computation/communication actions take place only when a particular *event* or a certain *series of events* have taken place. Events represent unexpected changes, abnormalities, or faults of the process/system that alter the system expected state. In contrast to continued and periodic triggering, ET ensures that normal operation is interrupted only after the relevant events have taken place [12]. In doing so, the available resources (including battery capacity, processing cycles, and communication channels) are thriftily used while communicated information reduces from raw data to simple event descriptions [13], [14].

In this work, a novel data-driven technique is developed to address all phases of ET, based solely on intelligence gained from quantifiable measurements. This data-driven paradigm is a natural fit for the IoT, in which data is produced with ease, it can be effectively processed and readily communicated as shown in [15]. Most importantly, data-driven approaches can offer the level of automation necessary to deploy IoT solutions in large numbers and to sustain their operational efficiency. Overall, the paper provides the following key contributions:

1) Provides an extensive literature review of existing event-triggering technologies.
2) Designs, develops and implements a purely data-driven (based on statistical measures) event-triggering technique for behavior modeling, event detection and event handling.
3) Demonstrates the applicability of the proposed technique utilizing simulations and via experimentation in operational environments.
4) Demonstrates the significant gains of the proposed technique compared to traditional periodic solutions that are currently being employed in state-of-the-art tools.

Related work is provided in Section II and the envisioned ET architecture is detailed in Section III in which, the three phases of event triggering, namely *behavior modeling*, *event detection*, and *event handling* are described. The proposed data-driven ET technique is then developed in Section IV

P. Kolios, C. Panayiotou, G. Ellinas, and M. Polycarpou are with 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.

within the scope of a public transport scenario. Thereafter, experimental results are provided in Section V to analyze the performance of the proposed data-driven ET technique under real-world settings. The whole process demonstrates the multiple efficiency gains of the proposed technique and the ease by which the proposed technique can automatically adapt to dynamic changes in an online fashion and provide scalability to IoT solutions. Finally, Section VI concludes this work.

## II. Related Work

To date, the Internet has been deployed and operated based on the assumption of perpetual availability of networked machines. Traditionally, these machines consisted mainly of computing units with ample power availability and as a result, continuous monitoring and control could easily be realized. However, the aforementioned assumption is becoming increasingly invalidated by IoT applications [16].

It is envisioned that, within the realm of the IoT, embedded devices will make use of the Internet to conduct machine-to-machine operations in either semi- or fully-autonomous fashion [17], [18]. Thus, operational efficiency is crucial for a sustainable deployment in large scales. Evidently, the tight form-factor necessitated by most of these devices puts stringent constraints on the hardware design. This necessity, in turn, has severely limited the onboard energy resources, since battery technology has failed to catch up to the demand as compared to transistor technology for computing, storage, and communication (that follows Moore's Law). Consumer electronics (including smartphones, phablets, and tablets) already experience this technology mismatch. The percentage contribution of battery packs to the total weight and volume of smart devices is disproportionately set very high in order to support adequately-long recharging cycles. Fortunately, the great improvement in computation and communication capabilities of these devices enables a plethora of new ways to increase operational efficiency. This is the primary reason that such devices have consistently being employed for various IoT applications in sectors such as the automotive, health-care, and security as noted in [19], [20] and [21].

Applications (that employ smart devices to implement IoT solutions) have demonstrated that time-based periodic triggering is primarily used for both monitoring and control [22]. The appealing proposition of this strategy is that there is a definite, predetermined interaction of the local and remote host. Hence, absence of communication results in an indisputable indication of a fault in the system. The advantage of periodic triggering is that devices are able to duty-cycle between the active and sleep states in order for them to improve their resource utilization. Nevertheless, periodic triggering can result in an unnecessarily high number of computation and communication actions (that convey no new information or do not indicate a change in state) simply due to the inherent periodicity of the paradigm [23]. At the same time, network scalability issues can arise due to this periodicity.

As exemplified above, ET compensates for these shortfalls by carrying out computation and communication actions only when certain events have taken place and which can potentially change the state of the system at hand [24], [25], [26]. In this way, the resource utilization efficiency is improved, (i.e., when no events are triggered, processing can scale down and communication circuitry can be put to sleep).

In its different forms, the ET technology is already heavily used in practice. It has been used to drive interrupts on computer machinery and provide traceability through event data logs. It has also been extensively referenced within the control community for more than three decades as the alternative more pragmatic solution to control applications. As of late, the communications community is also looking into the potentials of ET for improved efficiency through device duty cycling.

Still, most existing ET implementations within the IoT domain consider spontaneous events and thus fail to take advantage of the recurrent patterns and the system dynamics. Processing and analyzing streams of events can reveal recurrent patterns which can in turn be used to extract accurate behavior models and eventually more meaningful information. In doing so, all anticipated events are incorporated into a behavior model and relevant actions take place only when some unanticipated events have occurred. This greatly improves resource efficiency which is becoming increasingly important for sustainable IoT operation.

Interestingly, the proposed approach capitalizes on a number of technologies that have received considerable attention as of lately. For one, enormous effort is placed on stream processing and data analytics within the *big data* domain (an overview of the topic can be found in [27]). Data-driven solutions proposed within this realm can be used to build system behavior models that are necessary for event triggering. At the same time, a plethora of model-based event-triggering strategies have been developed for system *monitoring and control* (a review of such triggering strategies can be found in [28] [29]). In addition to reviewing the most prominent solutions that could enable ET, the work described in this paper introduces a novel data-driven ET paradigm and demonstrates its multiple gains in terms of efficiency and scalability compared to existing periodic techniques.

## III. Event-triggering Architecture

The basic system architecture under consideration assumes a local and a remote host. The local host is an embedded IoT device that is used for monitoring, control, and optimization of physical systems and the remote host can be any other entity (such as another device, server, or cloud infrastructure) that would also like to manage the system. Within the envisioned ET architecture, both the local and remote hosts obtain a model that approximates the system behavior. The local host uses this model to detect and trigger unanticipated event interrupts. The local host can take corrective actions in response to this event and communicate with the remote host in order to inform it of the updated system behavior which will allow the remote host to respond accordingly as well. A schematic of the proposed system architecture and the possible flow of information is presented in Figure 1 to aid understanding.

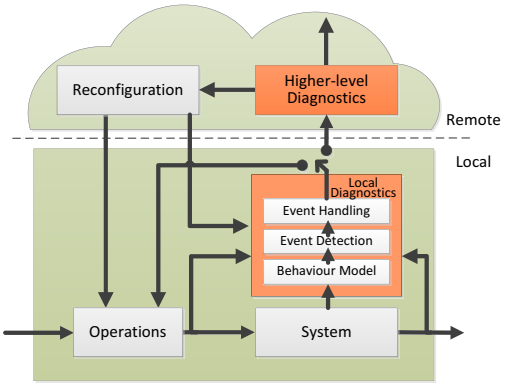



Fig. 1. Collaborative event triggering architecture.

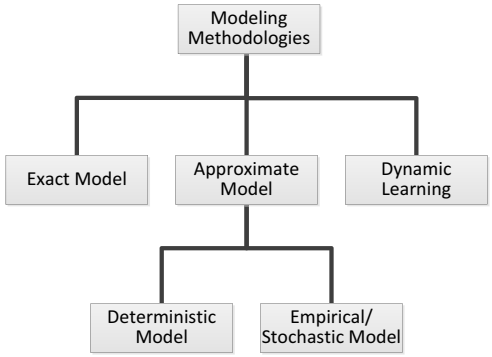



Fig. 2. Behavior modeling taxonomy

With respect to the application scenarios, the primary focus is on systems that exhibit recurrent patterns for which a behavior model suffices. Under this regime, events are defined as follows:

**Definition 1.** *Events are all those unanticipated changes that are detected to be outside the normal system behavior model. A behavior model captures all those patterns that suffice during normal operating conditions.*

Overall, the three main diagnostic phases that emerge from the aforementioned ET architecture are the following:

**Phase 1** : Behavior Modeling
**Phase 2** : Event Detection
**Phase 3** : Event Handling

Obviously, for events to be detected and acted upon, there should exist a model of the system to be monitored and controlled. This model is used as a reference for the normal behavior of the system. In practice the actual system behavior will be compared against this model to detect abnormal events. Therefore, a prerequisite to build such a model is for the system to exhibit patterns that can adequately characterize its evolution. Clearly, changes that may occur over long time horizons can be incorporated into the model by iterative updates and adaptations. On the contrary, when the system characteristics are completely unpredictable, then discrete indicators (e.g., on/off states) can be used for event triggering [29]. Further details on behavior modeling are provided in Section III-A.

Assuming the presence of a behavior model, unanticipated events can be effectively detected. An extensive literature, especially on control theory, has detailed numerous techniques for detecting events varying from simple threshold-checking solutions, to statistical changes of the system behavior, state estimation, and principal component analysis as elaborated in [28]. Section III-B provides a review of these techniques and then discusses data-driven solutions that are highly relevant to IoT applications.

Depending on the hierarchy to be used, these events are handled either locally, by a remote host, or collaboratively between the hosts as discussed in [30]. Originally, the unavailability of communication capabilities allowed only handling of events at a local level, while the intelligence built into monitoring and control algorithms was limited to the available processing and storage units. With the introduction of communication circuitry, networked solutions began to emerge. Centralized supervisory levels were setup and studied due to their relative simplicity [29]. Of course, distributed solutions were, and still are, among the most popular approaches due to their ability to significantly reduce system complexity [31]-[36]. More recently, the collaborative approach has gained considerable attention due to its ability to strike a balance between computation and communication and thus improve flexibility while reducing complexity [37], [38]. The latter approach also adheres to the state-of-the-art computing solutions in which local hosts perform as many computing actions as needed to maintain adequate quality of service (QoS) while intensive tasks are dealt with by remote hosts (usually in a distributed and parallel fashion) supported by scalable cloud services [39], [40]. Event handling is examined in detail in Section III-C.

Each of the three phases of ET are described in detail below.

### A. Phase 1: Behavior Modeling

Differential changes in the system state can form the simplest event triggers. For instance, discrete events (e.g., on/off switching) are commonly used in practice to indicate changes [29], [41]. However, there could be cases where such changes might be part of routine tasks that repeat over time or occur in tandem with other recurrent tasks. In effect, routine changes become highly predictable and offer no valuable information. Nevertheless, the patterns that arise can be used to build models of the system's normal behavior and hence capitalized upon to detect actual abnormalities.

Depending on the degree of available information with regards to the target system, several different behavior modeling methodologies exist as shown by the classification in Figure 2. When the actual physics of the system are available, its behavior can be characterized exactly by the underlying mathematical equations. Alternatively, the system might have an overall behavior that could be closely approximated by either some deterministic model or even an empirical model that most frequently incorporates some stochastic component. In the case when no a priori information is available or when

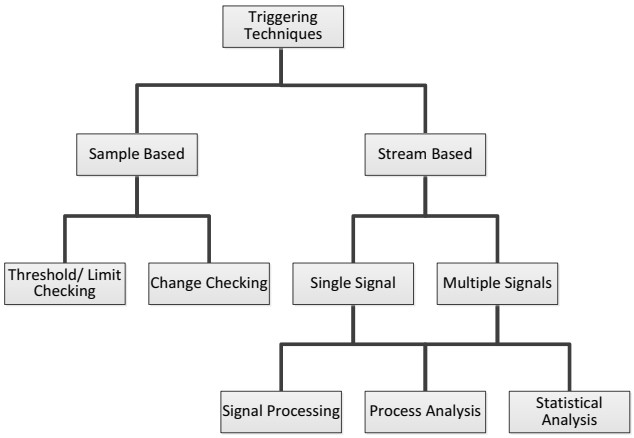



Fig. 3. Event detection techniques.

the system can change dynamically, then statistical modeling and machine learning approaches are employed to learn and update its behavior model on the fly, [42].

Among the alternatives, dynamic learning is the most favorable approach for IoT solutions for a multiplicity of reasons. Firstly, behavior modeling is greatly simplified since no a priori information is necessary to build the system model. Moreover, the data-driven nature of this methodology is well suited for IoT applications in which sampled data is readily available. Equally important is also the fact that the model can be constantly updated by the changing statistics in the data and thus easily adapted to evolutionary changes. Finally, the degree of detail built into the model is highly flexible, with increasing levels of detail easily incorporated into the system at any time. In Section IV that follows, the latter modeling methodology is adopted to build a simple mobility model for a transportation scenario.

### *B. Phase 2: Event Detection*

Event detection encompasses all the steps taken to ensure that unanticipated changes in the system's normal operation are detected. As emphasized above, a prerequisite for the correct detection of these events is the availability of an adequately accurate behavior model of the recurrent patterns that suffice. With the behavior model at hand, one or more different techniques can be employed to detect behavioral changes.

Viewed broadly, event detection can be classified into two main categories as shown in Figure 3. Sample-based techniques consider only individual samples to detect events while stream-based techniques explore the information provided by the flow of data samples to trigger an event.

The simplest and most commonly used sample-based technique is threshold-checking (with either static or adaptive thresholds) whereby an event is detected when the measured data sample crosses either a lower or an upper limit [43]. Change-checking is an alternative approach, where changes in the sample data are detected and evaluated. Examples of change-checking are techniques that learn trends in the changing data or changes in statistical measures as discussed in [44]. Obviously, sample data moments (especially the mean and variance) are frequently used, together with statistical tests (e.g., hypothesis testing, run-sum testing, etc.) for change detection [45]. With statistical measures, small changes can be detected more effectively without increasing the detection sensitivity.

The alternative to sample-based detection is stream-based detection as shown in Fig. 3. Stream-based techniques utilize information of data flows to build up knowledge of behavior changes. Signal processing techniques are particularly popular solutions in this category, in which specific features of the signal are extracted and evaluated [46]. Signal transforms of various kinds (including, Fourier, Z-transform, and wavelets) are extensively used for this type of analysis. Therein, auto- and cross-correlation operations are also used when the system model contains stochastic components that need to be considered.

Process analysis (either for single or multiple signals) looks into the problem from a higher level by trying to detect changes in the behavior model that happened due to obfuscated events [47]. This is mainly achieved by analytical redundancy whereby measurements are taken based on various excitation inputs. Several different process-identification techniques have already been devised to gain information on the model dynamics. As before, correlation methods can also be employed here. In addition, linear and non-linear parameter estimation techniques are also utilized [48]; with least squares being the most prominent solutions in the former case, and artificial neural networks in the latter.

Residuals is yet another stream-based approach that describe the difference between the actual system behavior and that of the model [28]. Parity equations can be employed to derive residuals through transfer functions or state-space formulations. State observers and state estimation can also be used as residuals by computing the discrepancies between the changes in the behavior model and the associated changes to the triggering events. Subsequently, threshold-checking is used to trigger events that are associated with residuals that exceed limiting values. Unfortunately, residuals can change continuously due to the inherent uncertainty in the model, the state, and the system noise. Hence, considerable effort should be placed in dealing with these disturbances when residuals are computed in this way.

To simplify the entire effort, (multivariate) statistical analysis has been extensively used. This technique focuses on data-driven tools to detect unanticipated events [42]. Deriving probability bounds, for certain random variables, given some of their moments, is a problem that appears frequently in the literature [49]. Values that fall outside these bounds can be used, in turn, to trigger event interrupts. Another important tool in this category is principal component analysis (PCA) [42]. PCA attracted considerable attention by the research community due to its ability to reduce the problem dimensionality and thus greatly reduce the number of processing requirements. Instead of using a large number of correlated variables, PCA produces a small number of uncorrelated variables (i.e., principal components) that preserve most of

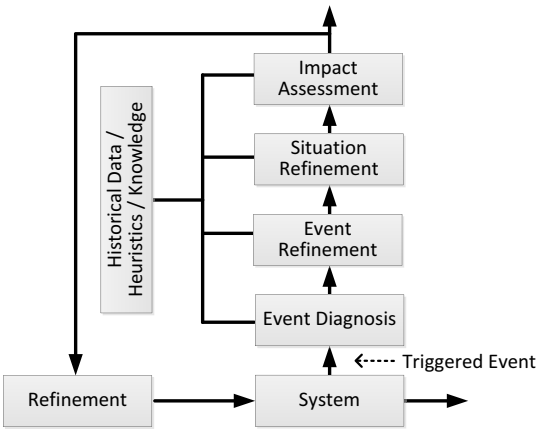



Fig. 4. JDL multisensor data fusion model applied to CEP.

the information describing the system. These new variables are then used to compute the residuals to be used in event-triggering.

The simplicity of statistical analysis tools is demonstrated in the section that follows, whereby a data-driven event detection approach is developed.

### C. Phase 3: Event Handling

Having detected an event, intelligence should be built to extract higher level knowledge, so as to be in a position to achieve actionable decisions. Over time, this procedure has shifted from simple event processing (with synchronous and sequential handling of events) to complex event processing (that supports asynchronous and batch execution) [50]. Complex event processing (CEP) elaborates on all necessary steps that need to be followed in order to build adequate knowledge out of a set of events either probabilistically (based, for example, on the theory of belief) or in a rule-based manner (based, for example, on detection theory). The Joint Directors of Laboratories (JDL) multisensor data fusion model is one of the most popular reference models in detection theory with broad applicability to various domains [51], [52]. As a note, legacy models including command and control [53] and condition-based maintenance [54] have also been proposed but address only specific aspects of event handling. JDL decomposes the whole functionality into five steps as shown in Fig. 4.

The first step deals with the decoding of primitive events into distinguishable interrupts. A diagnostic procedure is followed to determine the root cause of the event through the collection and processing of as much information as possible with regards to the event, including the event timestamp, and its location. Depending on the level of available information, a diagnostic method is then used to provide event metadata. Example diagnostic methods include pattern recognition, Bayes and polynomial classifiers.

Various data-driven methods have also become popular diagnostic methods in recent times. Clustering is one of the most popular methods of this type that relies solely on the event characteristics for classification. Events that are near each other in an arbitrary feature space are clustered together and labeled to belong to the same class of events. Notably, clustering is not as adaptive as neural network approaches but requires much less prior input and can be shown to improve its performance with higher volumes of data. Clearly, the latter fact is also true for other data-driven methods that are extensively used in practice, including regression, linear support vector machines, and expectation maximization [55], [56].

When the root cause of an event is unclear, inference methods such as event-tree-analysis, heuristic reasoning and fuzzy logic are also utilized that simplify the conditional rules necessary for event diagnosis [57].

The second step in the JDL model considers the refinement of events to determine attributes that are relevant to decision making. An important aspect of event refinement is the information gained by individual events in relation to their predecessors. These relations can be with respect to time, causality, ontology, or taxonomy. The information gained by these dimensions can then be used to build knowledge models on monitoring and control actions that should take place under different circumstances. Eventually, a combination of time with causality, knowledge models, and reasoning techniques can result in hybrid solutions that benefit from the knowledge build-up over time to better support decision making.

In the third step, a train of events (each containing metadata extracted from event processing, as discussed above) is analyzed to gain situational awareness. This correlation is done to estimate the current situation and to suggest, identify, and predict the system evolution. Step four in JDL carries out an assessment of the impact and the consequences for the system to be in the current state. This assessment also takes into account not only the current state but also the predicted reactions (based on historical data, knowledge, heuristics, etc.) and known plans (using for example user-based rules as in [58]). Given this assessment, the final step is to make decisions if further action is necessary. As discussed in Section III, the decision may be to deflect further action to a remote host (including re-synchronization), take local action to refine the system operation (including update of the behavior model), or do both. As shown in Fig. 1, information from this last diagnostic step is fed back to local operations for immediate action or sent to a remote host for higher-level management.

In the sequel, a simplified data-driven event handling approach is developed to extract higher level knowledge from triggered events.

## IV. Event Triggering Technique

Without loss of generality, a transportation scenario is considered hereafter to aid understanding. However, the same steps can be employed for any other use-case scenario that exhibits recurrent behavior patterns. The transportation scenario under consideration assumes public transport vehicles (i.e., buses) that service specific routes in an urban setting. The proposed data-driven ET technique deals with tracking trip changes conducted by these vehicles.

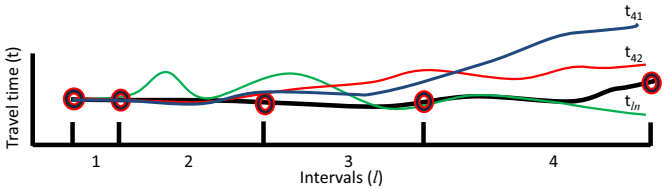



Fig. 5. Trips conducted with traveling times along different segments.

This scenario is one of the most prominent IoT application scenarios in the automotive sector. As such, the problem received substantial attention from both academia and the industry. However, most current solutions consider either continuous or periodic signaling, both causing excessive overhead (as shown in [59] and [60]) as opposed to ET that can greatly improve efficiency. Moreover, with the proposed data-driven technique all three phases of ET can be easily generated and automatically updated on demand, resulting in improved scalability as well.

The solutions for all three phases (as detailed above) are considered for the following setup. An onboard device (i.e., the local host discussed in Section III) is used to track the actual vehicle movement (using GPS traces), while the remote host uses a predetermined mobility model to estimate this movement. Clearly, no interaction is necessary between the two entities whenever the model closely matches the actual vehicle movement. On the contrary, the local host will trigger an event interrupt whenever it detects a substantial deviation (e.g., delay in transit) between the actual movement and the model estimate. This interrupt is then used to make model adjustments, re-synchronize the remote host, and deal with any incidents that could be inferred.

For each bus route service a set of $\mathcal{B} = \{1, \ldots, B\}$ stops are considered (intermediate check points can also be considered to improve resolution when deemed necessary). It is assumed that bus stop locations are known and the requirement is to monitor the bus schedule and track trip changes.

### A. Phase 1: Mobility Model

Consider the problem of building a mobility model based on travel times between the different legs of the particular route. Let $t_{ln}$ be the travel time observed on the $n^{\text{th}}$ journey along an arbitrary leg of the path $l = (i \mapsto j)$, $\{i, j\} \in \mathcal{B}$ with $l \in \mathcal{L}$. Obviously, the travel time is the measured elapsed time between the GPS samples received at bus stops $i$ and $j$. Hence $t_l$ is the travel time observed each time $l$ is traversed (an example with n=3 distinct trips is shown in Fig. 5). Then, the first sample moment $\theta_{lN}$ after $N$ sample measurements is given as follows:

$$\theta_{lN} = \frac{1}{N} \sum_{n=1}^{N} t_{ln}, \; \forall \, l \in \mathcal{L} \tag{1}$$

while higher order moments (i.e., $\theta^k_{lN} = \frac{1}{N} \sum_{n=1}^{N} \left[t_{ln}\right]^k$ for the $k^{\text{th}}$ moment) improve the knowledge with regards to the actual probability distribution of the random variable.

Importantly, sample moments can be updated recursively with each new sample. The updating for an arbitrary moment can be expressed as follows:

$$\begin{aligned} \theta^k_{l(N+1)} &= \frac{1}{N+1} \sum_{n=1}^{N+1} \left[t_{l(n+1)}\right]^k \\ &= \frac{1}{N+1} \left( \sum_{n=1}^{N} \left[t_{ln}\right]^k + \left[t_{l(N+1)}\right]^k \right) \\ &= \frac{N\theta^k_{lN} + \left[t_{l(N+1)}\right]^k}{N+1} \end{aligned} \tag{2}$$

where, as shown in eq. (2), the updating simply requires the total number of samples already fed into the calculation, $N$, and the latest moment value, $\theta_{lN}$. Each moment can be updated dynamically with only these two elements. In this way, the behavior (i.e., the mobility model) can be characterized by the first $K = 1, \ldots, k$ moments for each leg of the path instead of the raw sample data, greatly reducing the processing and storage/memomry load of the computing units. Of course, better accuracy can be achieved by introducing higher-order moments. In addition, the central moments can be computed from the raw moments using the binomial transform $\vartheta^k_{lN} = \sum_{i=0}^{k} \binom{k}{i} (-1)^{k-i} \theta^k_{lN} (\theta^1_{lN})^{k-i}$.

Alternatively, with a total of $N$ measurements made, a simple mobility model could be the ensemble of the latest $M$ raw measurement samples made along each leg of the path, i.e., $\mathcal{S} = \{\mathcal{S}_{12}, \mathcal{S}_{13}, \ldots, \mathcal{S}_{ij}, \mathcal{S}_{(B-1)B}\}, \{i, j\} \in \mathcal{B}$ resulting in a time window of measurements $t^n_{ij} \in \mathcal{S}_{ij}$, $n = \{N - M + 1, \ldots, N-1, N\}$. Evidently, maintaining the first $K$ moments greatly reduces the memory requirements (since $M$ is usually several order of magnitude larger than $K$) while ensuring that adequately accurate statistics of the sample distributions are used.

Given this information with regards to the vehicle journey, probabilistic estimates can be computed on the travel times, as well as the journey frequency and schedules, while unanticipated changes can be detected and acted upon.

### B. Phase 2: Time Deviation (Event) Detection

Noticeably, recurrent journeys along the particular route reinforce the information regarding the expected travel times for each leg of the path. This information can be used to set bounds on the travel times anticipated on the route and in turn trigger interrupts when deviations in the schedule occur. Evidently, having loose bounds can reduce the frequency of event interrupts, at the cost, however, of low-tracking accuracy. On the other hand, tight bounds may result to an unnecessarily high number of interrupts, since corrective control actions may be required for small schedule deviations. Therefore, a good bound should be computed to strike a balance between tracking accuracy and resource efficiency.

Let $\tau_{ij}, \{i, j\} \in \mathcal{B}$ be the mean travel time in segment $(i \mapsto j)$. Then, the bound in the travel time is set to be the time interval around the mean, given as $\tau_{ij} \pm \alpha$, where $\alpha$ is the threshold value. Given this travel time bound, a time-deviation event occurs whenever the measured travel time in segment $(i \mapsto j)$ is not within the travel time bound.

Let $E$ be the target number of deviation events triggered in a single trip on a particular route. Noticeably, if $E$ is high the tracking accuracy increases but too much communication may take place. On the other hand, if $E$ is set too low tracking inaccuracy may become unacceptable and the system may even fail without even the central controller ever knowing it.

In this work, $E$ is left as a design parameter that can be adjusted by the route operator. Then, the number of distinct combinations of events that can possibly occur along the different legs of the route can be found by the binomial coefficients that are expressed with matrix $\mathbb{C}$ whose rows consist of all possible combinations of the $B$ route stops taken $E$ at a time. Hence, matrix entry $\mathbb{C}_{ce}$ determines the $e^{\text{th}}, e = 1, \ldots, E$ deviation event that occurs in the $c^{\text{th}}$ instantiation of events, $c = 1, \ldots, C$, where $C = \frac{B!}{(B-1)!E!}$ is the number of all possible combinations that could occur. An example of this matrix is shown below with both the index of the rows and columns as well as the entries of the matrix. The matrix illustrates the case of $B = 40$ and $E = 5$.

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

As an illustrative example, consider the $c = 1^{\text{st}}$ combination of events in the matrix above. $\mathbb{C}_{11}$ indicates that a time-deviation event is triggered at the $3^{\text{rd}}$ stop while the next event is triggered at the $\mathbb{C}_{12} = 6^{\text{th}}$ stop, and so on. Hence, no interrupt occurs between the $4^{\text{th}}$ or the $5^{\text{th}}$ road segment for the particular pair. This is to say that after an event interrupt was triggered at the $3^{\text{rd}}$ stop, the time deviation that accumulated on the subsequent two segments was within the set bounds while the measured deviation on the $6^{\text{th}}$ stop exceeded those bounds and another interrupt was triggered. Noticeably, with every event interrupt, the estimated vehicle movement is reset to its true value.

Then, the probability of an event occurring at $j \in \mathcal{B}$ given that the previous event occurred at segment $i \in \mathcal{B}$ can be expressed as follows:

$$P(i,j) = \bar{P}(i,i{+}1){\times}\bar{P}(i,i{+}2){\times}\ldots{\times}\bar{P}(i,j{-}1){\times}(1{-}\bar{P}(i,j)) \tag{3}$$

where $\bar{P}(i,j)$ expresses the probability of no event triggered when traversing segment $i \mapsto j$. Formally, this probability is defined as follows:

$$\bar{P}(i,j) = P(\tau_{ij} - \alpha \leq \mathcal{S}_{ij} \leq \tau_{ij} + \alpha) \tag{4}$$

where $\mathcal{S}_{ij} = \{t_{ln}, l = (i \mapsto j), n = 1, \ldots, N\}$ is the set of samples observed between stops $i \mapsto j$, $\{i,j\} \in \mathcal{B}$. Left and right border cases in the combination of events should also be considered. The leftmost border deals with the probabilities from the start of the route up to the $1^{\text{st}}$ event. Similar to (3), these probabilities can be expressed as follows:

$$P(1,i) = \bar{P}(1,2) \times \bar{P}(1,3) \times \ldots \times \bar{P}(1,i-1) \times (1 - \bar{P}(1,i)) \tag{5}$$

In reality, an event is triggered at the start of each trip to synchronize the local and remote hosts. This interrupt can be due to a real event trigger or automated indicators triggered virtually by the bus schedule.

The rightmost border deals with the probabilities of the route segments that are traversed from the last event interrupt to the end of the route. These probabilities can be expressed as follows:

$$P(j, |\mathcal{B}|) = \bar{P}(j, j+1) \times \bar{P}(j, j+2) \times \ldots \times \bar{P}(j, |\mathcal{B}|) \tag{6}$$

Overall, the probability of all possible combinations of events results in the following expression:

$$P_T = \sum_{c=1}^{C} \prod_{e=1}^{E-1} [P(\mathbb{C}_{ce}, \mathbb{C}_{ce+1})] \times P(1, \mathbb{C}_{c1}) \times P(\mathbb{C}_{cE}, |\mathcal{B}|) \tag{7}$$

where $P(\mathbb{C}_{ce}, \mathbb{C}_{ce+1})$ is defined in equation (3) and the last two terms are defined by equations (5) and (6), respectively. Clearly, the total probability expressed in equation (7) should be equal to 1 since it includes all possible combinations of events that could occur for a target of $E$ interrupts.

In this expression, the only unknown parameter is threshold $\alpha$ of the bound in equation (4). The bisection method can be used to solve (7) and the method is described by the iterative algorithm shown below. The method begins by assuming that the solution lies within a definite interval $(a_l, a_u)$. Thereafter, the interval is halved (i.e., $a = 0.5 * (a_u + a_l)$) and a new value of the function is calculated. Then, the feasibility of the result is tested and the interval is updated according to the range that satisfies feasibility. The procedure continues until a favorable precision is reached with respect to the threshold, i.e., $a_u - a_l > \epsilon$.

**Algorithm 1** Bisection Method

**Ensure:** $a_l = 0$, $a_u = A$
1: **while** $(a_u - a_l) > \epsilon$ **do**
2: $a_= \frac{(a_u + a_l)}{2}$
3: Compute $P_T$ in (7)
4: **if** $P_T > 1$ **then**
5: $a_u = a$
6: **else**
7: $a_l = a$, $\alpha = a$
8: **end if**
9: **end while**

In summary, the proposed event detection approach relies only on a set of travel time measurements to build a mobility model of the vehicle movement and for tracking schedule changes. For tracking schedule changes, probability bounds are calculated based on a target number $E$ of out-of-bound event interrupts to be triggered in each trip. The bisection method, that is used to calculate these bounds, terminates when a favorable precision $\epsilon$ is reached. Noticeably, the mobility model can be dynamically updated with each journey to

incorporate the latest vehicle traveling behavior. The same is true for the event detection bounds that can be easily recomputed on the fly. Specifically, the complexity of this solution, involves the calculation of $\frac{|\mathcal{B}|(|\mathcal{B}|-1)}{2}$ travel times in the mobility model and an update of the $K$ sample moments. Furthermore, at every update of the bounds using the proposed solution, there are $C$ combinations that are considered in (7), a number which exponentially increases with $E$. Finally, the bisection method converges linearly with a total of $log_2(\frac{\epsilon_0}{\epsilon})$ iterations carried out, where $\epsilon_0$ is the initial interval at the start of the iterative procedure.

### C. Phase 3: Mobility Incident Inference

As discussed in Section III-C, event processing can offer higher-level knowledge of the system operating conditions and allow proactive decisions to be made. Evidently, sporadic event interrupts are triggered based on the design requirements as discussed above. On the other hand, sequences of events that do not reflect usual behavior patterns could allow for various different incidences (such as the build-up of congestion, speeding, and system faults) to be detected quickly, allowing for prompt responses.

One such event processing approach is described herein. Let $P^{\Xi}(1,i)$ be the probability for a total of $\Xi$ events occurring from the start of every new trip until bus stop $i$. It should be noted here that the volume of event interrupts may differ from the target number $E$ set in computing the event detection bounds (as elaborated above). Hence, historical data can be used to extract the probability distribution of $P^{\Xi}(1,i)$. Given $P^{\Xi}(1,i)$ and the probability $1-\bar{P}(i-1,i)$ of an event triggered within segment $(i-1 \mapsto i)$ (as defined in the previous section), then the probability of a sequence of events occurring up to segment $l$, can be computed as follows:

$$P_S(l) = \prod_{e=1}^{\Xi} \left[P^e(1,i-1)(1-\bar{P}(i-1,i))\right] \qquad (8)$$

As before, thresholds on this probability can be used to detect unlikely event sequences that would suggest some form of an incident. Then, various rules can be used to identify particular incidents and make decisions, as in [61], [62].

## V. Experimental Results

A bus service route operated by the Transportation Organization of Nicosia District in Cyprus is considered for the experimental set-up. The particular route considered, together with the designated bus stops, is shown in Figure 6. Service on this route is maintained by three buses (at any one time) that carry out a total of 25 trips each day (on weekdays).

These buses have been retrofitted with tracking devices and location, speed, and timing data have been collected for analysis. As a result, a total of approximately 500,000 samples have been collected over a six-month period which were subsequently used to extract the mobility model and compute the event triggering bounds as elaborated above. Of those, $M = 1000$ samples were used to build the mobility model of raw samples (as detailed in Section IV-A), while the

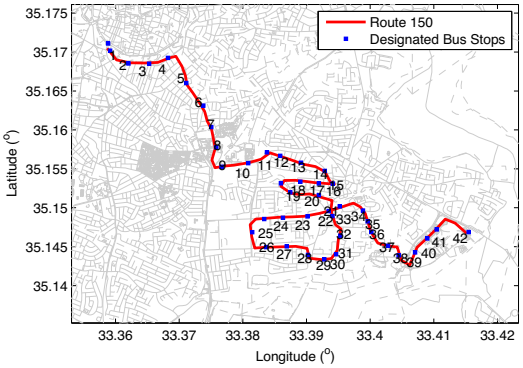



Fig. 6. OSEL service route 150 with designated bus stops.

first $K = 4$ moments were used to describe the travel time distribution of the model as detailed in Section IV-A.

### A. Phase 1: Mobility Model Results

Figure 7 depicts the road segments of the measurements made and the corresponding travel time histograms (including both raw samples and sample moments) for two distinct cases. In each case, an arbitrary pair of bus stops is selected and the travel times calculated from the collected GPS traces is shown. For instance, the plots in Figure 7a depict the travel times for road segments starting from the first bus stop on the route while Figure 7b shows the travel times from a bus stop further down the route. From these plots, it is clear that the spread of travel times is well defined and the sample moments (as shown in the rightmost plot in each subfigure) can in fact closely resemble the distributions observed by the raw samples (as shown by the middle plot in each subfigure). Moreover, these distributions describe precisely the mobility of the buses between a pair of stops along the particular route and thus can be used to approximate the vehicle trip.

### B. Phase 2: Event Detection Results

As detailed in Section IV-B, to calculate probability bounds on these measurements the bisection method (as shown in Alg. 1) has been coded in Matlab and was used to calculate probability bounds using $80\%$ of the collected samples. The remaining $20\%$ was used as a test set to validate the methodology. A threshold $\epsilon = 1$ sec is used in the algorithm that is both realistic and highly accurate for this application. Figure 8a presents the bounds achieved when both raw samples are used and when samples are produced from the distributions characterized by the first $K = 4$ moments. In the results, the target number of events, $E$, varies from 1 to 6 in total. As shown in the figure, the bound closes significantly for both mobility modeling approaches with only a few interrupts. Indicatively, $E = 4$ interrupts achieve a bound of less than a minute. This is to say that, on average, the proposed ETA will trigger a mere $4$ deviation detection events along the complete trip of the bus. At any other time, the bus movement can be adequately approximated by the mobility model within the

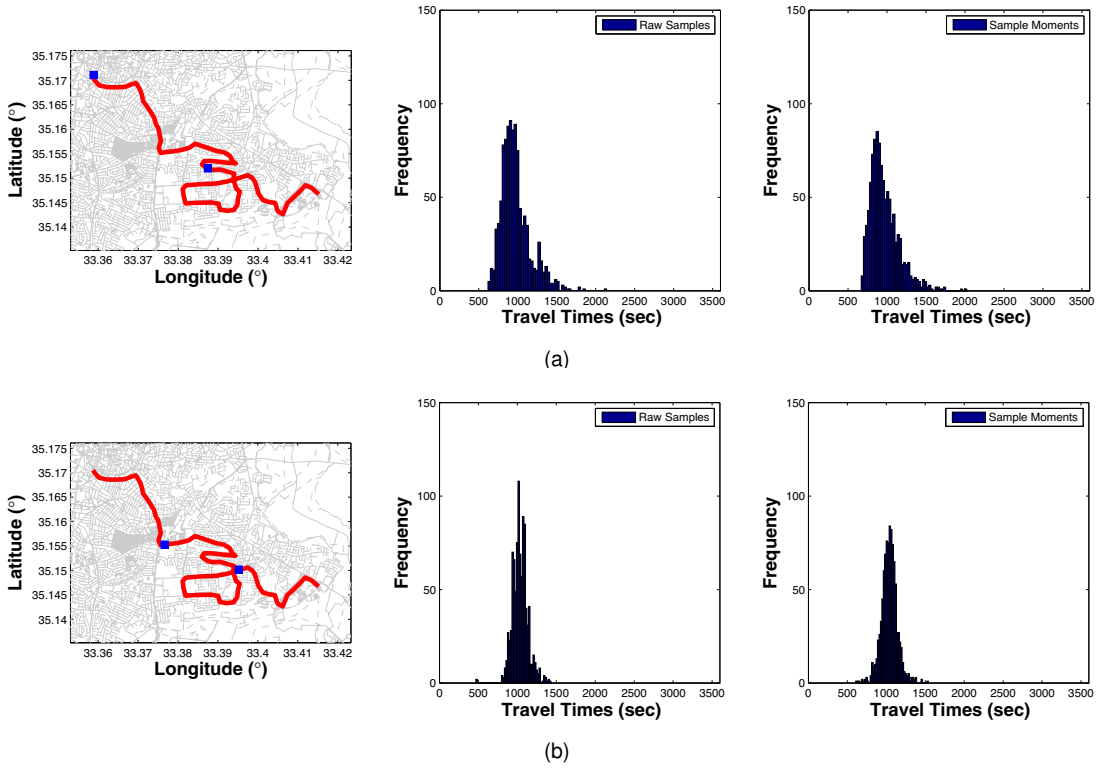



Fig. 7. Distribution of travel times between two arbitrary bus stops.

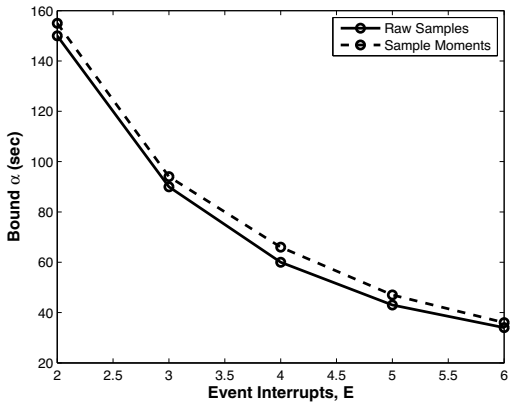
achieved 1 minute bound. During all this time (the total trip time for service route 150 is about 50 minutes), there is no need for communication between the local and the remote host and there is no need for any additional computation actions to be performed by the remote host. As such, the proposed ET technique provides a gain of 12 times compared to basic periodic triggering (that updates the vehicle status every minute[1], since 50 interrupts will be needed by the periodic scheme as compared to only 4 by the proposed ET technique as shown in the experiments). It also offers a far greater improvement if the basic periodic scheme follows more frequent updating. Noticeably, the scheme is realized simply by keeping travel time statistics and dynamically updating event detection bounds as discussed in Section IV.

Figure 8b depicts the number of interrupts triggered (using the achieved 1-minute deviation bounds) based on the sample test set (i.e., the 20% of the collected samples). In the box plots, the central mark is the median, the edges of the box are the 25th and 75th percentile, the whiskers (i.e., the lines above and below the box) extend to those points not considered outliers (i.e., whiskers extend to cover approximately 99% of the values, assuming points are normally distributed), and all outliers are plotted individually. Similar to the experimental results shown in Fig. 8a, the total number of events drops dramatically for wider bounds for the test set as well. More importantly though, it is evident from the comparison of the two plots in Fig. 8 that the volume of triggered events is within the computed bounds.

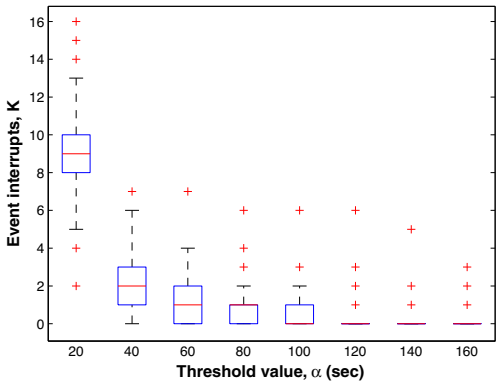
*1) Implementation Results:* As an additional validation step, the proposed ET technique has been implemented as an application for Android devices and has been used for live testing on route 150 of OSEL's fleet. A function was written to: a) acquire GPS signals for location information at intervals of 15sec, b) calculate the travel time between successive pairs of bus stops, and c) check for bound violations. The mobility model derived from the collected traces (as discussed in the previous section) was used to set a threshold value of $\alpha = 60$sec for a target of $K = 4$ interrupts. A second function was used to collect and log event interrupts. The application was installed on 3 mobile devices which were subsequently placed onboard the buses operating that particular route.

Figure 9 presents the results for a total of 100 trips conducted by buses on route 150. As shown in the figure, the proposed solution successfully maintains the target volume of interrupts for more than 90% of the trips conducted while for the rest of the cases analysis of the collected traces has shown that those interrupts occurred due to delays experienced at the start of the trips. Interestingly, the cumulative distribution curve of Fig. 9 has a sharp start indicating that many of the trips trigger only very few interrupts. For instance, more than 40% of the cases trigger less than 1 event interrupt while 70% of the trips trigger less than 2 out-of-bound events. In that respect, this implementation demonstrates in practice the great potential of the proposed ET technique in building more efficient tracking applications by minimizing processing and communication overhead (since only out-of-bound event interrupts need to be handled) while maintaining good tracking

[1] The proposed ET technique uses 1 minute bounds on event triggers and thus it is compared against the performance of periodic triggering that exchanges messages every 1 minute.




(a)




(b)

Fig. 8. a) Travel time bounds for varying number of target event interrupts, and b) experimental performance with a test set.

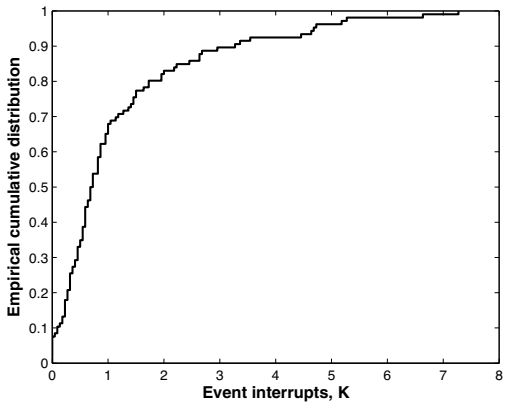



Fig. 9. Empirical results of event interrupts created on route 150.

accuracy using the predetermined mobility model.

### C. Phase 3: Event Handling Results

The potential to infer on the various mobility incidents is investigated herein. This is achieved by following the sequence of event interrupts that are produced by the proposed event detection algorithm using the collected traces and assuming

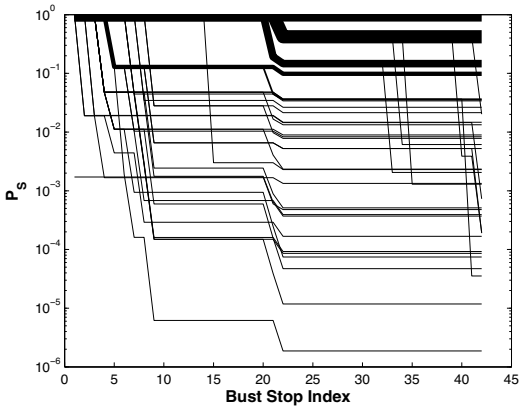



Fig. 10. Evaluating $P_S$ for a number of trips conducted on route 150.

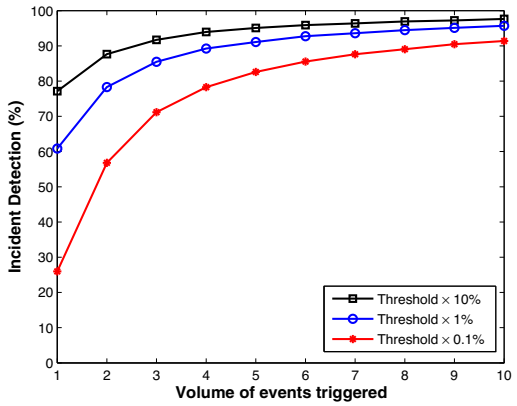
$K = 4$, as before. The variability in the volume of event triggers has already been discussed above (and shown in Fig. 8b), while the empirical distribution that suffices (as expressed in eq. (2)) is used to compute $P^{\Xi}(1, i), \forall i \in \mathcal{B}$ (as elaborated in Section IV-C).

Figure 10 shows the accumulated drop in probability based on equation (8) for a number of trips conducted along route 150. In the figure, the frequency with which a sequence of event interrupts takes place is highlighted with bolder lines. Noticeably, each drop in every line is associated with the triggering of an event interrupt and for the majority of the cases only very few interrupts are triggered. This is in line with the results shown in Fig. 8b, where for a target of $K = 4$ the mean volume of interrupts is around 1. It is also evident that in the majority of the cases the accumulated probability drop does not exceed two orders of magnitude while greater drops occur only infrequently.

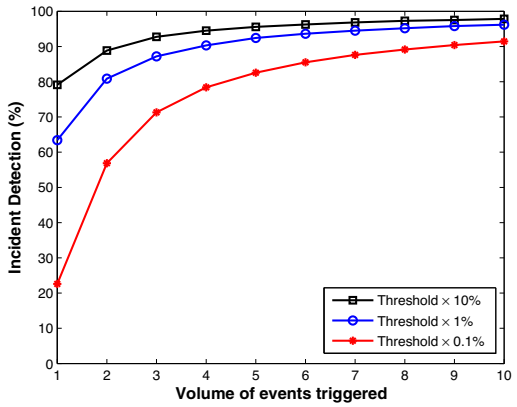
In the simulations that follow, the mean probability from the observed results is used as a threshold for detecting unlike event sequences. Moreover, the following random event triggers are considered:

- **Spontaneous:** Individual random event interrupts are injected and the volume of event interrupts follows the uniform distribution across all route segments.
- **Continuous:** A series of event interrupts, each occurring in tandem along successive route segments, is considered. The initial route segment where the first event is triggered is randomly and uniformly selected from all route segments and so is the volume of the event interrupts that follow.
- **Continuous Repeated:** Similar to the previous case, with multiple event interrupts occurring within each route segment as well. The volume of event interrupts occurring within each segment is also randomly and uniformly distributed in the interval $U\ (1, 4)$.

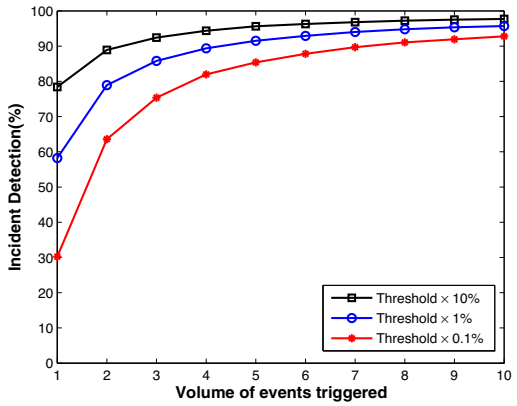
Figure 11 provides simulation results by varying the volume of event interrupts triggered for all three cases considered above (i.e., Fig. 11a for the case of spontaneous triggers, Fig. 11b when continuous triggering takes place, and Fig. 11c for the case of continuous and repeated event triggers). As




(a)




(b)




(c)

Fig. 11. Performance evaluation for a) Spontaneous, b) Continuous, and c) Continuous Repeated event interrupts.

discussed above, an incident is inferred if the probability drop is lower than the mean probability drop obtained by the actual performance illustrated in Fig. 10. In the results presented in Fig. 11, the threshold is varied by $1-3$ orders of magnitude from the mean probability drop in order to investigate the incident detection potentials.

As shown in the results, the percentage of detected incidents is substantially lower when the volume of triggered interrupts is lower than $K = 4$ for all three cases. This is expected since the event triggering technique derived in Section IV-B is designed to anticipated up to $K$ interrupts; a number that was set to 4 in the analysis conducted in the previous section. Even so, the percentage of incidents detected in all cases grows non-linearly both with increasing number of triggered events and with tighter thresholds.

Comparing the results obtained for the three cases, it is evident that the order by which events are triggered only marginally affects the percentage of incidents detected. Clearly, this is due to the fact that only loose bounds are needed for $K = 4$ which do not substantially penalize $P_S$ in eq. (8). When tighter bounds are set, the impact on the probability of an out-of-bound event would be more severe and thus incident detection would be more probable. In any case, the order by which events are triggered can be used with higher level reasoning to infer on the different incidents, as discussed in Section IV-C.

## VI. Conclusions

In this work, the highly-promising event-triggering architecture has been reviewed with particular emphasis placed on data-driven approaches for the IoT domain. Specifically, the three distinct phases of ET, namely behavior modeling, event triggering, and event handling have been studied and analyzed. Additionally, and to aid understanding of each phase, a practical scenario considering road mobility has been investigated and data-driven solutions have been derived for all three phases. Finally, the potential of data-driven ET has been demonstrated through a real-world public transport tracking scenario. As shown by this case study, data-driven ET can provide many-fold improvement in resource utilization efficiency (including energy and spectrum efficiency) and offers the level of autonomy necessary for scalability.

## Acknowledgments

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

## References

[1] J.A. Stankovic, "Research Directions for the Internet of Things", *IEEE Internet of Things Journal*, 1(1):3-9, Feb. 2014.
[2] C. Cassandras, "Event-driven Control, Communication, and Optimization", *Chinese Control Conference*, July 2013.
[3] J. Anderson and L. Rainie, "Digital Life in 2025 - The Internet of Things will Thrive by 2025", *Pew Research Center - Numbers, Facts and Trends Shaping the World*, May 2014, Available at: http://www.pewinternet.org/2014/05/14/internet-of-things/.
[4] Lopez Research, "An Introduction to the Internet of Things (IoT)", *The IoT Series, Part 1*, Nov. 2013.
[5] Google's Advanced Technology and Projects group, "Project Ara", *http://www.projectara.com/*.
[6] GSMA, "Connected Living Linking the Physical and Digital Worlds", *The Mobile Economy*, 62-64, April 2014.
[7] A.M. Ortiz, D. Hussein, S. Park, S.N. Han, and N. Crespi, "The Cluster Between Internet of Things and Social Networks: Review and Research Challenges", *IEEE Internet of Things Journal*, 1(3):206-215, June 2014.
[8] C. Perera, C.H. Liu, S. Jayawardena, M. Chen, "A Survey on Internet of Things from Industrial Market Perspective", *IEEE Access*, 2:1660 - 1679, Jan. 2015.
[9] L. Li, S. Li, and S. Zhao, "QoS-Aware Scheduling of Services-Oriented Internet of Things", *IEEE Transactions on Industrial Informatics*, 10(2):1497-1505, May 2014.

[10] W. Heemels, A.R. Teel, N. Wouw, and D. Nešić, "Networked Control Systems with Communication Constraints: Tradeoffs between Transmission Intervals, Delays and Performance", *IEEE Transactions on Automatic Control*, 55(8):1781-1796, Aug. 2010.

[11] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context Aware Computing for The Internet of Things: A Survey", *IEEE Communications Surveys & Tutorials*, 16(1):414-454, 2014.

[12] W.P.M.H. Heemels, and N. van de Wouw, "Stability and Stabilization of Networked Control Systems", *Networked Control Systems*, Lecture notes in control and information sciences, Springer 406, pp. 203-253.

[13] M. Lemmon, "Event-Triggered Feedback in Control, Estimation, and Optimization", *Networked Control Systems - Lecture Notes in Control and Information Sciences*, 406:293-358, 2010.

[14] X. Meng, L. Xie, Y.C. Soh, C. Nowzari, and G.J. Pappas, "Periodic Event-Triggered Average Consensus over Directed Graphs", *IEEE Conference on Decision and Control*, Dec. 2015.

[15] Q. Wu, G. Ding, Y. Xu, S. Feng, Z. Du, J. Wang, and K. Long, "Cognitive Internet of Things: A New Paradigm Beyond Connection", *IEEE Internet of Things Journal*, 1(2):129-143, Apr. 2014.

[16] R. Saracco, "The Internet of Things (and With Things)", *IEEE Communications Society Blogs*, http://www.comsoc.org/blog/internet-things-and-things.

[17] Commission of the European Communities, "Internet of Things An Action Plan for Europe", *Communication from the Commission to the European Parliament, the Council, the European Economic and Social Committee and the Committee of the Regions*, COM(2009) 278, June 2009.

[18] The Working Party on the Protection of Individuals with Regard to the Processing of Personal Data, "Opinion 8/2014 on the Recent Developments on the Internet of Things", *Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995*, Articles 29 and 30 thereof, Sept. 2014.

[19] GSMA, "Understanding the Internet of Things (IoT)", *Connected Living*, July 2014.

[20] S. Chen, H. Xu, D. Liu, B. Hu, and H. Wang, "A Vision of IoT: Applications, Challenges, and Opportunities With China Perspective", *IEEE Internet of Things Journal*, 1(4):349-359, Aug. 2014.

[21] D. Evans, "The Internet of Things - How the Next Evolution of the Internet is Changing Everything", *Cisco Internet Business Solutions Group*, Apr. 2011.

[22] O. Vermesan and P. Freiss, "Internet of Things - From Research and Innovation to Market Deployment", *River Publishers Series in Communications*, 2014.

[23] M. Mahmoud and M. Sabihb, "Networked Event-triggered Control: An Introduction and Research Trends", *International Journal of General Systems*, 43(8):810-827, Apr. 2014.

[24] 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.

[25] O. Vermesan and P. Freiss, "Internet of Things - Converging Technologies for Smart Environments and Integrated Ecosystems", *River Publishers Series in Communications*, 2014.

[26] P. Huang, C. Wang, and L. Xiao, "RC-MAC: A Receiver-Centric MAC Protocol for Event-Driven Wireless Sensor Networks", *IEEE Transactions on Computers*, DOI: 10.1109/TC.2014.2308174, Feb. 2014.

[27] 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.

[28] R. Isermann, *Fault-Diagnosis Systems: An Introduction from Fault Detection to Fault Tolerance*, Springer, 2006.

[29] C. Cassandras, and S. Lafortune, *Introduction to Discrete Event Systems*, Springer, 2nd Ed., 2008.

[30] M. Blanke, M. Kinnaert, J. Lunze and M. Staroswiecki, *Diagnosis and Fault Tolerant Control*, Springer, Ed. 2, 2006.

[31] D. Castanton, and D. Teneketzis, "Distributed Estimation Algorithms for Nonlinear Systems", *IEEE Transactions on Automatic Control*, 30(5):418-425, May 1985.

[32] A. Nayyar, A. Mahajan, and D. Teneketzis, "Decentralized Stochastic Control with Partial History Sharing: A Common Information Approach", *IEEE Transactions on Automatic Control*, 58(7):1644-1658, July 2013.

[33] M. Zhong, and C. Cassandras, "Asynchronous Distributed Optimization with Event-Driven Communication", *IEEE Transactions on Automatic Control*, 55(12):2735-2750, Dec. 2010.

[34] Y. Khazaeni, and C. Cassandras, "A New Event-Driven Cooperative Receding Horizon Controller for Multi-agent Systems in Uncertain Environments", *IEEE Transactions on Automatic Control*, 55(12):2735-2750, Dec 2010.

[35] P. Panagi and M. Polycarpou, "A Coordinated Communication Scheme for Distributed Fault Tolerant Control", *IEEE Transactions on Industrial Informatics*, 9(1):386-393, Feb. 2013.

[36] M. Mazo and P. Tabuada, "Decentralized Event-Triggered Control Over Wireless Sensor/Actuator Networks", *IEEE Transactions on Automatic Control*, 56(10):2456-2461, Oct. 2011.

[37] T. Schlage, "Remote Diagnosis of Technical Systems", *Institute of Automation and Computer Control*, May 2009.

[38] T. Schlage and J. Lunze, "Modeling of Networked Systems for Remote Diagnosis", *International Conference on Control and Fault Tolerant Systems*, Oct. 2010.

[39] V. Cevher, S. Becker, and M. Schmidt, "Convex Optimization for Big Data: Scalable, Randomized, and Parallel Algorithms for Big Data Analytics", *IEEE Signal Processing Magazine*, 31(5):32-43, Sept. 2014.

[40] S. Ren and M. van der Schaar, "Efficient Resource Provisioning and Rate Selection for Stream Mining in a Community Cloud", *IEEE Transactions on Multimedia*, 15(4):723-734, June 2013.

[41] Y.C. Tseng, T.Y. Lin, Y.K. Liu, and B.R. Lin, "Event-Driven Messaging Services Over Integrated Cellular and Wireless Sensor Networks: Prototyping Experiences of a Visitor System", *IEEE Journal on Selected Areas in Communications*, 23(6):1133-1144, June 2005.

[42] K. Slavakis, G. Giannakis, and G. Mateos, "Modeling and Optimization for Big Data Analytics: (Statistical) Learning Tools for Our Era of Data Deluge ", *IEEE Signal Processing Magazine*, 31(5):18-31, Sept. 2014.

[43] M. Basseville, "Detecting Changes in Signals and Systems - A Survey", *Automatica*, 24(3):309-326, 1988.

[44] H. Stark, and J. Woods, *Probability, Random Processes and Estimation Theory for Engineers*, Prentice Hall, 1994.

[45] H. Akaike, "A New Look at the Statistical Model Identification", *IEEE Trans. on Automatic Control*, 19(6):716-723, 1974.

[46] S. Stearns, *Digital Signal Analysis*, Hayden Book Company, 1975.

[47] D. Hall, and J. Llinas, *Handbook of Multisensor Data Fusion*, CRC Press LLC, 2001.

[48] X.W. Chen, and X. Lin, "Big Data Deep Learning: Challenges and Perspectives", *IEEE Access*, 2:514-525, May 2014.

[49] D. Bertimas and I. Popescu, "Optimal Inequalities in Probability Theory: A Convex Optimization Approach", *SIAM Journal on Optimization*, 15(3):780-804, 2005.

[50] P. Vincent, "Complex Event Processing", *Workshop on Distributed Object Computing for Real-time and Embedded Systems*, June 2008.

[51] D. Hall and J. Llinas, *Handbook of Multisensor Data Fusion: Theory and Practice*, Taylor & Francis, 2nd Ed., Sept. 2008.

[52] D. Luckham, *The Power of Events*, Addison Wesley, 2002.

[53] RAF, "The RAF Fighter Control System", *Background to the Battle of Britain*, 1940, http://www.raf.mod.uk.

[54] A. Steinberg, and C. Bowman, "Data Fusion for Developing Predictive Diagnostics for Electromechanical Systems", *Handbook of Multisensor Data Fusion*, CRC Press, 2001.

[55] X. Wu, X. Zhu, G.Q Wu and Wei Ding, "Data Mining with Big Data", *IEEE Transactions on Knowledge and Data Engineering*, 26(1):97-106, Jan. 2014.

[56] H. Peng, F. Long, and C. Ding, "Feature Selection Based on Mutual Information: Criteria of Max-Dependency, Max-Relevance, and Min-Redundancy", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1226-1238, Aug. 2005.

[57] Y. Wang, and K. Cao, "Context-aware Complex Event Processing for Event Cloud in Internet of Things", *International Conference on Wireless Communications & Signal Processing*, Oct. 2012.

[58] C.Y. Chen, J.H. Fu, T. Sung, P.F. Wang, E. Jou, and M.W. Feng, "Complex Event Processing for the Internet of Things and its Applications", *IEEE International Conference on Automation Science and Engineering*, Aug. 2014.

[59] J. Biagioni, A.B.M. Musa, and J. Eriksson, "Thrifty Tracking: Online GPS Tracking with Low Data Uplink Usage", *ACM International Conference on Advances in Geographic Information Systems (SIGSPATIAL)*, 2013.

[60] IBM Ireland, "Dublin City Council", *Travel & Transport*, July 2013.

[61] S. Hasan, E. Curry, M. Banduk, and S.O. Riain, "Toward Situation Awareness for the Semantic Sensor Web: Complex Event Processing with Dynamic Linked Data Enrichment", *International Workshop on Semantic Sensor Networks*, Oct. 2011.

[62] Y. Wang, and K. Cao, "A Proactive Complex Event Processing Method for Large-Scale Transportation Internet of Things", *International Journal of Distributed Sensor Networks*, Mar. 2014.