QuIC-IoT: Model-Driven Short-Term IoT Deployment for Monitoring Physical Phenomena

Tung-Chun Chang
University of California, Irvine
USA
tungchuc@uci.edu

Tirtha Banerjee
University of California, Irvine
USA
tirthab@uci.edu

Nalini Venkatasubramanian
University of California, Irvine
USA
nalini@uci.edu

Robert York
University of California, Berkeley
USA
ryork@berkeley.edu

ABSTRACT

The Internet-of-things ecosystem has been a driving force in the creation of smart communities where a variety of physical phenomena can be monitored continuously, e.g., air quality, traffic conditions on roads, energy consumption in buildings, etc. In this paper, we address how IoT can be quickly and effectively deployed for short-term and sporadic events (e.g., fire spread in a wildland area and flood propagation), where monitoring the evolving event is critical. In particular, we propose QuIC-IoT, a model-driven planning platform that aims to temporarily deploy a custom IoT infrastructure for monitoring short-term events, where phenomena-spread is driven by models that are physics-based. Our driving usecase event is a quasi-planned *prescribed fire or RxFire* - this is a wildfire resilience technique where intentional small fires are ignited apriori by forestry personnel to destroy fuel and help contain the spread of actual wildfires. Anomalies that may occur during these quasi-planned events must be rapidly captured by the IoT deployment, e.g., escaped RxFires can escalate to catastrophic wildfires under unpredictable conditions of wind, vegetation, etc. QuIC-IoT incorporates domain expert-developed models to guide IoT deployment; the event area is partitioned into subregions and a criticality metric that quantifies the likelihood of anomalies at each location is computed. QuIC-IoT allows us to mix fixed and quasi-mobile IoT devices to flexibly deploy IoT in challenging terrain and as the phenomena (RxBurn) evolves. We evaluate QuIC-IoT in two real-world forest settings (large and small) in Blodgett Forest, CA, USA, with concrete burn plans developed by wildfire experts. Our experimental results reveal that QuIC-IoT enables over 3X improvement in cost-effectiveness and performance (timely detection of anomalies) as compared to baseline IoT deployment algorithms.

CCS CONCEPTS

• **Software and its engineering**; • **Applied computing** → **Physics**; • **Information systems**;

IoTDI '23, May 09–12, 2023, San Antonio, TX, USA
© 2023 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0037-8/23/05.
https://doi.org/10.1145/3576842.3582381

KEYWORDS

IoT deployment problem, short-term, rapid IoT

ACM Reference Format:
Tung-Chun Chang, Tirtha Banerjee, Nalini Venkatasubramanian, and Robert York. 2023. QuIC-IoT: Model-Driven Short-Term IoT Deployment for Monitoring Physical Phenomena . In *International Conference on Internet-of-Things Design and Implementation (IoTDI '23), May 09–12, 2023, San Antonio, TX, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3576842.3582381

1 INTRODUCTION

In recent years, we have seen the emergence of IoT devices, networks, and services in our everyday lives; recent efforts focus on perpetually monitoring of physical phenomena in our communities, e.g., noise monitoring in New York City [6], air quality monitoring in Chicago [10], and water quality monitoring in California [31, 32]. These deployments incorporate sensors to capture physical phenomena and convert them into a digital form; a range of networking devices transmit the captured data to servers that transform it into human-interpretable information. These long-term deployments (sometimes lasting decades) provide decision-makers (humans in the loop) with information to ensure safety of citizens, detect adverse events, and respond to them in a timely manner. Indeed, recent urban planning toolkits [11, 28] incorporate IoT deployment design as an integral part of this long-term urban planning workflow.

In this paper, we address the problem of planning an effective IoT deployment for short-term and sporadic events in communities. Examples include occasional (but planned) activities within a community – county fairs, sporting events, or disaster resilience exercises such as the annual Great CA Shakeout Exercise. Ensuring citizen safety, security and comfort in such settings is critical, especially in unexpected events such as natural or manmade disasters (wildfire, floods, etc.). In planning a short-term deployment, we have partial prior knowledge, e.g., geospatial location of event, planned duration of activities, and an estimate of the number of people involved. However, the evolution of events and associated activities/individuals can deviate significantly from the predetermined plan, leading to uncertainty (e.g., a fire event during a concert). To monitor the flow of events, a rapid pop-up IoT instrumentation is deployed before the start of the event to capture events of interest; the deployment is torn down after the event.

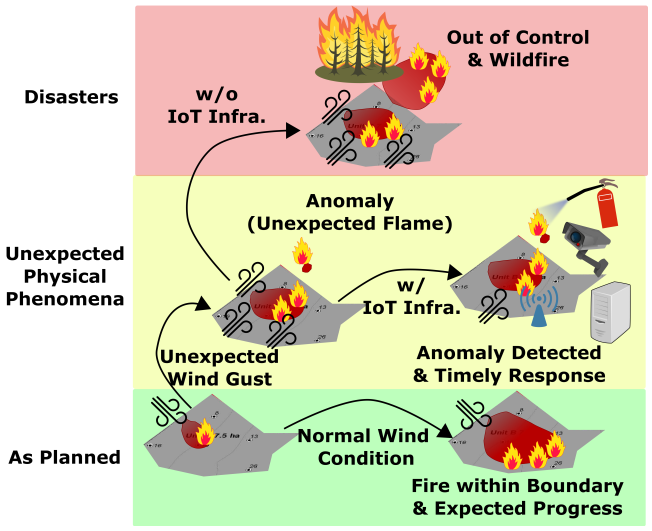

Disasters
w/o IoT Infra.
Out of Control & Wildfire
Unexpected Physical Phenomena
Anomaly (Unexpected Flame)
w/ IoT Infra.
Unexpected Wind Gust
Anomaly Detected & Timely Response
As Planned
Normal Wind Condition
Fire within Boundary & Expected Progress

Figure 1: Uncertainty of Physical Phenomena of Interest.

Particularly, we address the issue of short-term IoT planning through a specific driving usecase of a quasi-planned event - *a prescribed fire or RxFire* that is used to help build community resilience to fire events, such as wildfires. While forestry experts typically provide a burn plan, unpredictable weather and other environmental factors (e.g., high humidity) can impact the feasibility of the burn; a sudden strong wind could blow embers to unnoticed areas transforming the prescribed burn into a catastrophic wildfire. Fig. 1 illustrates how the uncertainty of physical phenomena can result in a large disaster and how IoT deployment can help capture anomalies early on thus avoid potential large-scale damage.

Creating an IoT deployment with custom instrumentation for short-term quasi-planned events, such as RxFires, poses multiple challenges. First, accurately modeling phenomena dynamics, e.g., fire spread, is difficult but useful for effectively deploying sensing technology. A comprehensive understanding of how the event evolves can help determine vulnerability points (e.g., locations prone to winds and high flames) and drive the sensor placement process. Second, cost/budget and time constraints limit the extent to which short-term custom instrumentation can be deployed. In an RxFire, stakeholders must decide how to deploy limited sensing resources - decisions of whether to concentrate sensing resources at locations where there is a higher likelihood of fire escape or provide better coverage for the overall burn site must be made. Third, timely communication of sensed data from the field for in-depth analysis at an edge or cloud server (e.g., fire situational awareness) is critical, but connectivity may be spotty or limited in challenged settings. A joint instrumentation plan that provides both sensing and networking coverage is required; such a plan must often be adapted at the event site, where actual connection qualities can vary.

In this paper, we study the short-term IoT deployment problem that constructs an on-site monitoring system for applications that require safe and effective operations and mission-critical decision-making. The deployment aims to provide a plan for selecting and deploying sensing and networking functionalities with a focus on capturing safety-critical anomalous phenomena before they escalate into disasters. The captured data is sent to an edge server for further analysis and decision-making. We discuss novel methods and algorithms to (i) quantify a location's importance/criticality in terms of sensor deployment decisions, (ii) design sensing configurations that can accurately capture physical phenomena, and (iii) determine the location of networking devices to interconnect the deployed sensing devices. To this end, we design, implement, and evaluate a system called **QuIC-IoT**. Key contributions of this paper include.

• We present a general method that quantifies each location's criticality given phenomena of interest. This is the first step in the foundation of the IoT deployment problem to capture anomalies (Sec. 4).

• We propose QuIC-IoT to generate a plan to construct IoT infrastructure at the event site that provides add-on information to help decision-makers detect and prevent hazardous situations and thus ensure a smooth event (Sec. 4).

• We model the short-term IoT deployment problem into two distinct steps to deploy sensing and networking devices. We quantify the cost-utility functions for different deployments, which are used to formulate the short-term IoT deployment problem. The problem is shown to be NP-hard (Sec. 4).

• We develop a suite of algorithms for the two-step process in QuIC-IoT. Two methods are iteratively executed until it runs out of the deployment budget: (i) sensor deployer for determining a sensing unit's component and location and (ii) network constructor for connecting the deployed sensing unit to the edge server (Sec. 5).

• We conduct extensive evaluations of the proposed QuIC-IoT framework and algorithms in two real-world RxFire settings in Blodgett Forest, CA, USA: small burn and large burn. A real burn plan is leveraged to determine abnormal fire behaviors. Our method outperforms the baseline methods by up to 3.55 times in terms of overall utility (Sec. 6).

2 RXFIRE - A DRIVING USECASE

We explore specific challenges in creating an effective deployment for short-term monitoring of an evolving event through the context of a prescribed fire or RxFire. RxFires are intentional fires that are used to restore ecological processes in forests and to consume accumulations of fuel (vegetation, litter, etc.) that would otherwise burn with high intensity during a wildfire. When practiced at large enough scales and frequencies, prescribed fires have a proven efficacy in reducing the probability of future hazardous wildfires [29]. Unforecasted changes in weather conditions, such as wind, temperature, and relative humidity, can pose challenges to conducting RxFires safely. Occasionally, RxFires can escape, meaning that they burn onto the property of an adjacent landowner and cause damage. Such escapes have consequences involving substantial losses [18, 23]. Two escaped RxFires created the largest wildfire in the history of the state of New Mexico, USA, in April 2022. It destroyed at least 330 houses and affected nearly 500 square miles (1,300 square kilometers) of forests; thousands of residents were evacuated. Around 3,000 firefighters were dispatched to fight the

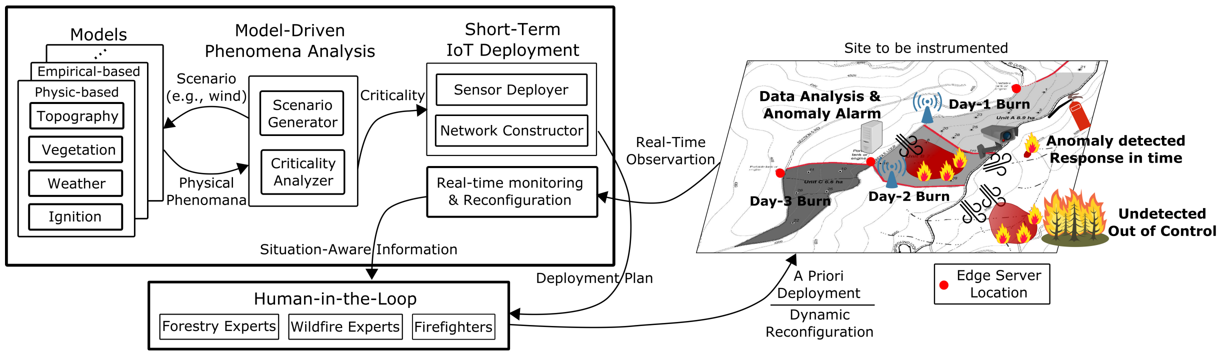


Figure 2: QuIC-IoT's architecture and workflow.

blaze, which cost over 132 million US dollars. The physical phenomena of interest during an RxFire include fire behavior and sudden weather and air quality change. Embers traveling beyond the desired control lines may occur with enough frequency that an escape becomes possible. A sudden change in wind direction is also of concern, even if it does not cause an increase in the risk of escape. Burning during certain wind directions is a key planning factor. A change in wind direction could blow smoke to a sensitive area, such as a road or residential neighborhood.

Typically, planners of RxFires create a burn plan, defining desired weather conditions that are known to result in desired fire behaviors. Once the weather or fire behaviors deviate from the ideal situation, the burn boss must cancel or postpone the operation to mitigate risk. If an RxFire turns into a wildfire, contingency strategies to deal with it are provided - including introducing retardants by fire personnel at the escape locations. Besides, fire monitoring is critical for staff safety and plan adherence. RxFires are usually operated in remote locations that do not have infrastructure (sensing, networking, and computing). The capacity to deploy IoT infrastructure to enable the capture of these physical phenomena during and after RxBurns would be a substantial improvement in identifying and then responding to anomalies. Besides, RxFires usually operate with offline or delayed environmental information; the IoT infrastructure provides near real-time environmental monitoring that assists the burn boss's decision-making process.

Understanding the physical phenomena that govern fire spread requires planners to conduct numerous experiments with different control variables. Controlling and manipulating vital environment variables (e.g., weather conditions) for repeated experiments is challenging and time-consuming. For example, determining how wind conditions affect fire spread is dictated by multiple factors - terrain, vegetation, and other meteorological factors. Leveraging past experience from previous fires in the region to derive accurate models is hard. RxFires in a region are characterized by a long period between burns; new vegetation after a burn may change the characteristics of burn propagation.

Fortunately, expert-developed models/simulators capture the complex physical factors and allow us to experiment with various inputs that can influence phenomena (e.g., fire spread). For instance, fire spread models have been widely used in domains like wildfire prevention [3, 22]. Diverse models exist, such as physics-, empirical-, and semi-empirical-based, with different accuracy and efficiency. In particular, FARSITE [15] is a semi-empirical fire simulator with efficient execution time. Although such models provide a level of understanding of physical phenomena, the underlying assumptions pose issues in practice when fires are actually instantiated. For instance, FARSITE requires weather conditions and ignitions as inputs. Our proposed approach is to address the above problems by integrating model-based techniques with a data-driven approach that is realized through a customized IoT deployment.

3 QUIC-IOT - A MODEL-DRIVEN APPROACH FOR SHORT-TERM IOT PLANNING

We aim to exploit knowledge of physical phenomena (e.g., wildland fire spread) and associated models to drive the deployment of an IoT infrastructure to capture the evolution of the phenomena. In our proposed workflow, data captured by the IoT deployment is communicated via one or more access networks to an (edge) server where further analysis is conducted. The analysis of data from the site can be used for a variety of purposes, including (i) adherence check to a predetermined event plan and (ii) early detection of anomalous behavior, i.e., the event is not proceeding as planned. We argue that a comprehensive short-term IoT deployment must consider the impact of the complex interaction between physical phenomena (e.g., fire) and the changing environment (e.g., wind speed) at the deployment site.

As an initial step, we partition the event site into non-overlapping cells and leverage models (e.g., numerical, physics-based, or empirical) to understand the physical phenomenon's behavior. We then derive a cell's criticality (a measure defined later) by analyzing the probability of an anomaly occurring at the site. The challenge is to ensure coverage of the overall monitoring area so as to capture (not miss) phenomena occurring over the entire site. We also assume that instrumenting a short-term event at a site has cost constraints, i.e., a fixed budget, which imposes a trade-off between deploying sensors at high-criticality cells or maximizing the overall sensing coverage. Moreover, the IoT infrastructure must provide network connectivity for sensors to transmit data to the edge server via the deployed network devices, further complicating cost/coverage/timeliness trade-offs. The crux of our approach is to merge models/simulators of physical phenomena with data from IoT instrumentation. Fig. 2 illustrates a schematic system architecture for QuIC-IoT.

Scenario generation and criticality analysis. First, we aim to identify parameters that characterize the physical *Phenomenon of Interest (PHoI)* - i.e., fire parameters in RxFires. Fire behavior *parameters* can help assess whether an RxFire operation is proceeding or deviating from the prescribed plan. We utilize the FARSITE simulator [15] to help generate multiple features that characterize the fire: (i) flame length, (ii) fire spread rate, and (iii) spotting distance (flying distance of an ember). Environmental factors, including wind speed/direction, temperature, and relative humidity, significantly impact these fire behaviors/parameters. For a comprehensive study of fire behaviors under wind conditions, we leverage WindNinja [16], a wind simulator, to generate wind conditions (speed and direction) in each cell based on the terrain. The composite FARSITE and WindNinja simulators can then help generate fire behaviors for each cell under diverse wind conditions, topography, vegetation, and weather conditions.

The *scenario generator* component in QuIC-IoT generates numerous realistic scenarios/inputs with a range of environmental conditions and ignition locations that can serve as input into WindNinja/FARSITE. Domain experts provide information to create realistic scenarios that might be of specific interest. For instance, forestry experts usually start an RxFire from a higher altitude. A sudden wind gust may blow embers to unnoticed locations, creating an *escape fire* and possibly a wildfire. Such human expert-provided corner situations are important to include in our investigation.

The output of the simulations is a series of values for each scenario for the phenomena of interest being monitored - fire spread rate, flame length, and spotting distance. This information is next analyzed to derive the probability of an anomaly (e.g., escape fire) occurring at a cell, quantified as the cell criticality. The *criticality analyzer* module in QuIC-IoT implements a novel *neural network regression* model that learns a function of the occurrences of different values for each parameter of a cell. We define *criticality* as the *anomaly probability* derived by applying an expert-defined threshold on the regression function. In other words, locations with high criticality are prone to anomalies.

Generating the deployment plan. We next exploit the derived criticality to guide a customized IoT deployment for short-term events with a specified period and PHoI. For instance, a safe and efficient RxFire requires early anomaly detection for staff safety and near real-time fire monitoring. High criticality cells have a high priority of being monitored; however, guaranteeing the smooth operation of the RxFire requires that the system maximizes the sensing coverage area. This naturally induces a trade-off between criticality and coverage that must be addressed by the deployment algorithm. In addition to sensing coverage, communication devices must also be deployed at the site, especially when there is a lack of network infrastructure, as is the case with wildland fires. Sensing requirements can vary based on monitoring needs resulting in varying network bandwidth needs. Wind sensors monitoring a sudden gust have low bandwidth requirements, whereas cameras capturing fire images demand high bandwidth. Cost constraints can induce complicated trade-offs in the choice of access networks. Long-range low-bandwidth communication networks, e.g., LoRa, provide lower cost and lower data accuracy as compared to short-range high-bandwidth communication networks, e.g., WiFi, where more devices are required to cover the whole site.

To generate a deployment plan for sensing and networking infrastructure, QuIC-IoT includes *sensor deployer* and *network constructor* components, executed in an iterative algorithm until the budget constraint is reached. In each iteration, the sensor deployer determines a *sensing unit*'s configuration, including a processing unit (e.g., Raspberry Pi) and a selected set of *sensors* (cameras, temperature, wind, smoke). Each sensor requires a specific *communication technique* for reliable transmission, e.g., cameras require WiFi for high bandwidth transmission. QuIC-IoT 's algorithms choose deployment locations for sensing units based on criticality/coverage measures. The network constructor then establishes connectivity between the sensing unit to the edge server by deploying one or multiple *networking units*. A *transmission (Tx-)aware graph*, embedded with the Tx range information on its edges, is built during this construction. We propose a *Tx-aware shortest path* to minimize the number of units to achieve adequate connectivity.

Run-time monitoring and reconfiguration. Once the IoT infrastructure is constructed, sensing units provide near real-time observation of PHoI at the site. Real-time data, including imagery, temperature, and wind, help the staff at the site decide their next course of action to avoid dangers. As the event (Rxfire) progresses, the deployment can be reconfigured based on situations in the field; burn staff may require additional on-the-fly information from locations with high priority as potential anomalies are reported. Trade-off goals can vary during operation; real-time reconfiguration decisions by the burn boss may favor IoT placement in high-priority locations over maximizing coverage on the site. The addition of specialized devices like drones can help capture emergent data during the event.

4 MODEL-DRIVEN SHORT-TERM IOT DEPLOYMENT PROBLEM

In this section, we mathematically model the geography and physical phenomena and derive the criticality of phenomena at a location with semi-empirical models. Then, we model the IoT infrastructure and formulate the short-term IoT deployment problem, which is proved to be NP-hard. Table 1 summarizes all the notations used in this paper.

4.1 Geography, Physical Phenomena, Criticality

Modeling geography and physical phenomena. A short-term event (e.g., RxFire) is hosted at a *site* composed of a set of *areas* $\mathcal{A}$ (e.g., burn area and surrounding area). An area $a \in \mathcal{A}$ is further divided into a set of cells C, where $c \in C$ is a cell represented by its center for simplicity. A set Φ contains all *parameters* of all *Phenomena of Interest (PHoI)* at the site, where $\phi \in \Phi$ is a parameter. For instance, a parameter stands for flame length, spread rate, or spotting distance for a fire; wind speed and wind direction are the parameters to detect a sudden weather change. A *requirement matrix* M_r indicates whether area a requires monitoring the parameter ϕ, where each element $M_r[a, \phi]$ represents whether ϕ is required in a. If ϕ is required in a, $M_r[a, \phi] = 1$; otherwise, $M_r[a, \phi] = 0$.

Deriving criticality with semi-empirical models. We introduce the method (shown in Fig. 3) to derive the *criticality*, the probability of an abnormal parameter ϕ (e.g., flame length above 1.7 feet) occurring in cell c in area a. However, the actual *anomaly*

Table 1: Notations Used in This Paper

Notation	Description	Notation	Description	Notation	Description	Notation	Description
a	an area	$\mathcal{A}$	a set of areas	c	a cell	C	a set of cells
ϕ	a parameter	Φ	a set of all parameters of PHoI	d	a device	$\mathcal{D}$	a set of devices
t_d	a device's characteristic tuple	r^{tr}	a device's transmission range	r^{sen}	a device's sensing range	τ	a device's type
M_r	requirement matrix	M_ϕ	module matrix	$m_{\phi,i}$	a module monitoring ϕ	u	an unit
$\mathcal{U}^s$	a set of sensing unit	$\mathcal{U}^n$	a set of networking units	$\mathcal{L}$	a set of candidate locations	l	a location
M_p^s	placement matrix of $u \in \mathcal{U}^s$	M_p^n	placement matrix of $u \in \mathcal{U}^n$	$f_{imp}(u, m_{\phi,i})$	implementation function	$dist(l, l')$	distance function
$f_g(u)$	deployment function of u	P	approximate PMF function	α_d	d's sensing parameter	t	anomaly threshold
$U(a, c, \phi, \mathcal{U})$	utility function	$p(u, c, m_{\phi,i})$	sensing coverage	$\sigma(a, c, \phi)$	criticality function	$\kappa(m_{\phi,i}, u)$	connectivity
χ	deployment plan	$P(\chi', \chi, l, \phi, c)$	sensing coverage improvement	B	deployment budget	$\delta(u)$	cost of u
$P(\hat{l}, l_{dst})$	Tx-aware shortest path	$\mathcal{G}(P(\hat{l}, l_{dst}))$	network coverage gain	$\mathcal{G}(u, m_{\phi,i}, l, \phi)$	utility gain	$w(l_1, l_2)$	edge weight on $P(\hat{l}, l_{dst})$
$\Delta C(u, m_{\phi,i})$	cost of deploying u with $m_{\phi,i}$	$\Delta C(P(\hat{l}, l_{dst}))$	cost of deploying u on $P(\hat{l}, l_{dst})$	l_n	nearest net device's location	l_{edge}	edge server's location
e_{net}	marginal network coverage	e_{cov}	marginal utility	w_n	network coverage weight	w_s	sensing weight

probability is hard to derive since we cannot explore all possible scenarios of PHoI. We leverage the *scenario generator* to generate numerous scenarios to drive semi-empirical models. The models generate a set of parameters (fire behaviors) based on the generated scenarios, and each parameter's results from all scenarios are grouped for further analysis. Two separate datasets are generated for training and testing purposes. We leverage the *neural network regression* model to learn the criticality. Then, we design a *criticality analyzer* to find a function $\sigma(a, c, \phi)$ that quantifies the criticality for each ϕ in each c of a to minimize the mean squared error to the training dataset with the following inputs: (i) training dataset of simulation results generated by models for each ϕ and (ii) the abnormal parameters defined by experts. Then, we further evaluate the derived function's performance with the testing dataset.

The criticality analyzer aims to derive each parameter's Probability Mass Function (PMF) for each cell. Its x-axis and y-axis represent a parameter's possible values and the probability of whether each value occurs. Since it is hard to derive the actual PMF, we propose to derive a function P with minimized error to the actual PMF for each parameter ϕ at each cell c in area a. We select the maximum value from ϕ's simulation results and divide it into several bins (ranges) with equal sizes. P's x-axis represents the derived bins, whereas P's y-axis records the occurrence probability of the values in each bin. Let t denote the threshold that a parameter becomes abnormal. The anomaly probability or criticality can be derived by applying the threshold on the PMF, i.e., $\sigma(a, c, \phi) = P(X > t)$. We propose using the regression technique to derive P for an estimate of ϕ. However, canonical regression models require a pre-defined parameter/function as an input, which is difficult to determine due to the limited understanding of PHoI. Therefore, we leverage *neural network regression* that automatically learns arbitrary functions without a priori domain knowledge of the physical phenomena.

4.2 IoT Infrastructure and Sensing Models

IoT infrastructure for data collection. Sensors capture specific parameters of PHoI and convert them into digital form (data); data is then transmitted to an edge server for further analysis. The edge server is usually at a fixed location, e.g., near the burn boss for RxFire. A *sensing unit* is constructed by assembling several *devices*, which usually contain a base unit, such as Raspberry Pi, several sensors, and one or multiple networking interfaces. All possible devices are provided in a set $\mathcal{D}$, where $d \in \mathcal{D}$ is a device. Besides, d is characterized by a tuple $t_d = (r^{tr}, r^{sen}, \tau)$. r^{tr} and r^{sen} are d's *transmission and sensing range*, respectively. Each d belongs to a type τ, such as *sensing*, *networking*, and *base unit.* A *module matrix* M_ϕ indicates the implementations to capture each ϕ. Each row vector indicates the required devices to implement a corresponding *module.* Each element $M_\phi[i, d]$ on the i-th row represents whether the i-th module requires device d. If d is required, $M_\phi[i, d] = 1$; otherwise, $M_\phi[i, d] = 0$. For simplicity, we use $m_{\phi,i}$ to represent a module to implement to monitor ϕ. Besides, a *networking unit* relays data from multiple sensing units to the edge server via a particular communication technique, such as WiFi or LoRa. A set of sensing and networking units are denoted as $\mathcal{U}^s$ and $\mathcal{U}^n$, respectively. For simplicity, $\mathcal{U}^s$ and $\mathcal{U}^n$ are collectively denoted as $\mathcal{U}$.

A sensing unit's coverage for a parameter in a cell. We model the *sensing coverage* of a sensor d installed on sensing unit $u \in \mathcal{U}^s$ deployed at l for ϕ as follows. We extend the truncated attenuated model [33], where d probabilistically covers each cell c in area a with the probability attenuated (decaying) with its Euclidean distance to l, $dist(l, c)$, and truncated by d's sensing range r^{sen}. For simplicity, we assume each sensing unit has only one sensor d for each $m_{\phi,i}$. The probability of u at l capturing ϕ in c by $m_{\phi,i}$ is:

$$p(u, c, m_{\phi,i}) = \begin{cases} e^{-\alpha_d \times dist(l,c)}, \ if \ dist(l,c) \le r^{sen} \ and \ M_r[a, \phi] = 1; \\ 0, \ otherwise, \end{cases} \tag{1}$$

where α_d is a parameter related to d's sensing capability. The sensing coverage of u to capture ϕ is the summation of the probability of capturing ϕ in every c by every installed $m_{\phi,i}$.

Data structure to manage deployment. We design the following data structure for bookkeeping the deployment information of both sensing and networking units.

• A binary *implementation function* $f_{imp}(u, m_{\phi,i})$ indicates whether the i-th module $m_{\phi,i}$ in M_ϕ is installed on u, i.e., all corresponding devices are installed. If u implements $m_{\phi,i}$, $f_{imp}(u, m_{\phi,i}) = 1$; otherwise, $f_{imp}(u, m_{\phi,i}) = 0$. Each u implements up to one module for each ϕ.

• A site has a set of *candidate locations* $\mathcal{L}$ for deploying sensing and networking units. Each unit $u \in \mathcal{U}$ has a *deployment function* $f_g(u) = l$, where $l \in \mathcal{L}$ is u's deployment location.

• A *placement matrix of sensing units* M_p^s indicates whether each location $l \in \mathcal{L}$ can deploy sensing unit u's module $m_{\phi,i}$. If $m_{\phi,i}$ can be deployed at l and, thus, u, $M_p^s[l, m_{\phi,i}] = 1$; otherwise,

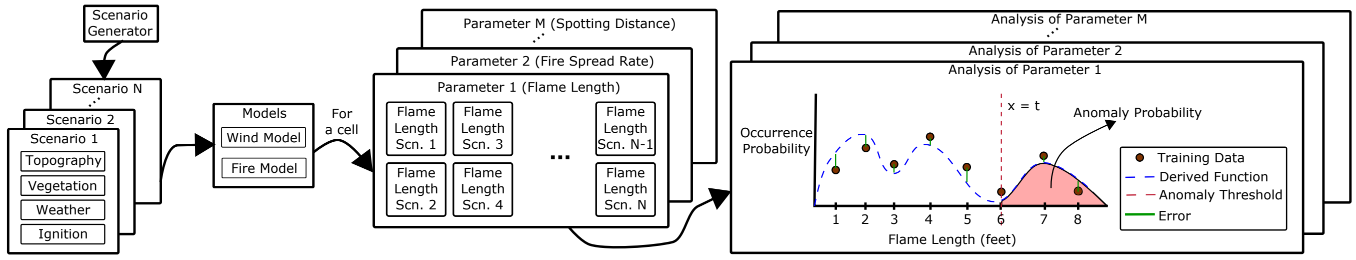


Figure 3: Criticality Analyzer's workflow to derive the criticality of each parameter for each cell.

$M_p^s[l, m_{\phi,i}] = 0$. Similarly, a *placement matrix of networking units* M_p^n indicates the possible networking units to deploy for each location $l \in \mathcal{L}$.

Connectivity of a module installed on a sensing unit. We define *connectivity* $\kappa(m_{\phi,i}, u)$ to capture if sensing unit u's module $m_{\phi,i}$ with network interface d has a network connection. We assume that u is deployed at l, i.e., $f_g(u) = l$. Let u' denote u's nearest networking unit with an identical communication technique as d, and $f_g(u') = l'$. If l' is within d's transmission range r^{tr}, u is connected; otherwise, u is unconnected by d. The connectivity of $m_{\phi,i}$ on u is:

$$\kappa(m_{\phi,i}, u) = \begin{cases} 1, & if\ dist(l, l') \le r^{tr}; \\ 0, & otherwise. \end{cases} \tag{2}$$

Note that a sensing unit can have multiple network interfaces with different connectivity.

Utility of sensing units for a parameter in a cell. We argue that deploying a sensing unit u's *utility* contains three main factors: (i) u's *sensing coverage* for capturing parameter ϕ in area a, (ii) the *criticality*, $\sigma(a, c, \phi)$, of ϕ in each c in a, iii) the *connectivity*, $\kappa(m_{\phi,i}, u)$, of u's module $m_{\phi,i}$ capturing ϕ. We design the utility as the sensing probability of deploying u to capture ϕ in c, i.e., Eq. (1), weighted by ϕ's criticality in c. Then, u's connectivity is a binary decision of whether the utility is zero, i.e., the captured information cannot be transmitted without a network connection. Thus, for deploying a set of sensing and networking units (collectively denoted as $\mathcal{U}$), the *utility* to capture ϕ in cell c in area a is:

$$U(a, c, \phi, \mathcal{U}) = \max_{\substack{\forall f_{imp}(u, m_{\phi,i})=1, \\ \forall u \in \mathcal{U}^s}} (\sigma(a, c, \phi) \times \kappa(m_{\phi,i}, u) \times p(u, c, m_{\phi,i})). \tag{3}$$

Costs and budget. Each unit has a deployment cost incurred by purchasing or renting devices. Sensing unit u's deployment cost $\delta(u)$ is the summation of each installed device's cost. Each networking unit $u \in \mathcal{U}^n$ has a deployment cost $\delta(u)$. Besides, each short-term event has a deployment budget B for obtaining the sensing and networking units.

4.3 Problem Formulation – Short-Term IoT Deployment

Given the site's geography information and the criticality of each parameter in each cell, this problem aims to maximize the overall utility under a fixed budget constraint by (i) creating a set of optimal sensing units $\mathcal{U}^{s^*}$ and networking units $\mathcal{U}^{n^*}$ and (ii) determining the deployment function for all units. We formulate the deployment problem as follows.

$$\max \sum_{\forall a \in \mathcal{A}} \sum_{\forall c \in C} \sum_{\forall \phi \in \Phi} U(a, c, \phi, \mathcal{U}^*), \tag{4a}$$

$$\text{subject to:} \sum_{\forall u \in \mathcal{U}^*} \delta(u) \le B. \tag{4b}$$

Eq. (4a) is the objective function that determines the components and deployment function of units in $\mathcal{U}^* = \mathcal{U}^{s^*} \cup \mathcal{U}^{n^*}$ to maximize the overall utility. Eq. (4b) is the budget constraint to deploy sensing and networking units collectively.

Theorem 1. *Short-Term IoT deployment is NP-hard and can not be approximated within $1 - 1/e$.*

Proof. We consider a special case where every sensing and networking unit has an unlimited transmission range and costs 1 and 0, respectively. The site has only an area where all cells have uniform criticality. Only one PHoI exists with a parameter, and only one sensing unit can monitor it. Let the budget equal K. The special case aims to cover all cells with K sensing units. Hence, the special case, equivalent to the K-cover problem, cannot be approximated within $1 - 1/e$ [14]. Thus, the short-term IoT deployment problem is NP-hard. □

Algorithm 1: Sensor Deployer

```
Input: $\chi$, $\mathcal{L}$, $\Phi$, $M_\phi$, $M_p^s$
1  $e \leftarrow 0$ {keep track of the maximum utility gain}
   {examine all possible configurations of sensing units at each candidate location}
2  for $\phi \in \Phi, m_{\phi,i} \in M_\phi, l \in \mathcal{L}$ do
3      if $M_p^s[l, m_{\phi,i}] == 1$ then
4          for $u \in \hat{\mathcal{U}}^s$ do
5              if $\hat{f}_{imp}(u, m_{\phi,i}) = 1, \hat{f}_g(u) = l$ then
6                  {leverage existing sensing unit with the required sensing module}
7                  $\hat{e} \leftarrow \mathcal{G}(\sim, \sim, l, \phi)$
8              else if $\hat{f}_{imp}(u, m_{\phi,i}) = 0, \hat{f}_g(u) = l$ then
9                  {upgrade the existing sensing unit with an additional sensing module}
10                 $\hat{e} \leftarrow \mathcal{G}(\sim, m_{\phi,i}, l, \phi)$
11             if $\hat{e} > e$ then $e \leftarrow \hat{e}, \hat{u} \leftarrow u, \hat{m_{\phi,i}} \leftarrow m_{\phi,i}, \hat{l} \leftarrow l$ ;
12         if $\hat{f}_g(u) \neq l, \forall u \in \hat{\mathcal{U}}^s$ then
13             $\hat{e} \leftarrow \mathcal{G}(u, m_{\phi,i}, l, \phi)$ {install a new sensing unit}
14             if $\hat{e} > e$ then $e \leftarrow \hat{e}, \hat{u} \leftarrow u, \hat{m_{\phi,i}} \leftarrow m_{\phi,i}, \hat{l} \leftarrow l$ ;
15 {update the deployment plan for sensing units}
16 $\hat{\mathcal{U}}^s \leftarrow \hat{\mathcal{U}}^s \cup \hat{u}, \hat{f}_{imp}(\hat{u}, \hat{m_{\phi,i}}) \leftarrow 1, \hat{f}_g(\hat{u}) \leftarrow \hat{l}$
17 Execute Algorithm 2 to construct a network connection for the newly
   added sensing unit $\hat{u}$
```

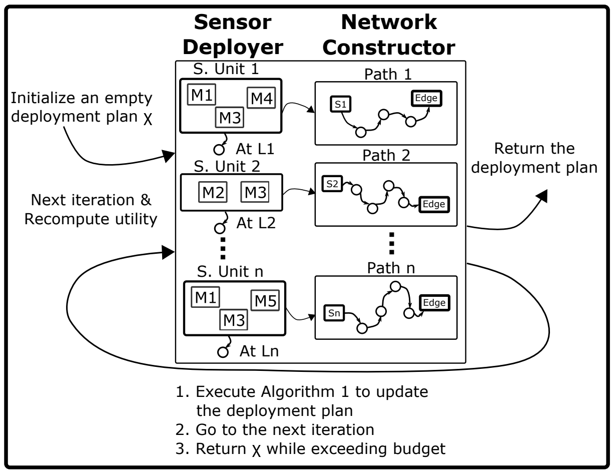

Sensor Deployer
Network Constructor
S. Unit 1
M1
M4
M3
At L1
Path 1
S1
Edge
S. Unit 2
M2
M3
At L2
Path 2
S2
Edge
S. Unit n
M1
M5
M3
At Ln
Path n
Sn
Edge
Initialize an empty deployment plan χ
Next iteration & Recompute utility
Return the deployment plan
1. Execute Algorithm 1 to update the deployment plan
2. Go to the next iteration
3. Return χ while exceeding budget

Figure 4: General Workflow of QuIC-IoT's Algorithms.

```
Algorithm 2: Network Constructor
  Input: $\hat{l}$, $l_{edge}$, $l_n$, $\mathcal{L}$, $w(l,l')$, $\forall l, l' \in \mathcal{L}$
1 $P(\hat{l}, l_{edge})$ ← the Tx-aware shortest path from $\hat{l}$ to $l_{edge}$
2 $P(\hat{l}, l_n)$ ← the Tx-aware shortest path from $\hat{l}$ to $l_n$
3 $\hat{P}$ ← $P(\hat{l}, l_{edge})$, $P(\hat{l}, l_n)$ with fewer hops
```

5 QUIC-IOT ALGORITHMS

The short-term IoT deployment is NP-hard and complex due to the intertwined relationships between deploying sensing and networking units. A sensing unit embedded with multiple modules for different purposes may require multiple network interfaces. For example, both low and high-bandwidth transmission techniques can transmit regular sensor data, whereas high-resolution images require high-bandwidth transmission like WiFi. Besides, the budget limits the number of deployed sensing and networking units. Exploiting long-range low-bandwidth transmission techniques like LoRa has a higher coverage per device but supports fewer modules. Reusing common devices for multiple parameters, such as using a camera to capture flame length and fire spread rate, increases the possible number of deployed devices and, thus, the coverage. Thus, proposing a one-off method to solve the problem is challenging. We divide the problem into two parts and address them by proposing two components in QuIC-IoT with the following intuition. A *sensor deployer* first determines a sensing unit's location and composition. Then, a *network constructor* guarantees that the edge server receives the captured information by connecting the sensing unit to it.

Fig. 4 shows QuIC-IoT's workflow to generate a deployment plan, which is initialized as an empty set. First, the sensor deployer examines all possible deployment and composition of all possible sensing units and selects a sensing unit with various intuitions. Then, the network constructor identifies a path to deploy networking units that connect the sensing unit to the edge server if necessary. QuIC-IoT executes the process iteratively to derive a deployment plan until the accumulated deployment cost exceeds the budget. The sensor deployer's efficiency is closely related to the number of possible compositions of a sensing unit. If a sensing unit can be built with N modules, the number of compositions is 2^N. However, we find its impact relatively small in practice. A short-term event usually has only a few PHoI and hence, the number of possible modules installed on a sensing unit. A limited number of ports to sensors and network interfaces (usually <10) of a base unit like Raspberry Pi bounds the number of possible compositions. We propose a basic QuIC-IoT that maximizes the overall utility and connects each sensing unit by the shortest path capturing the transmission characteristic. Then, we propose to enhance QuIC-IoT by selecting the sensing and networking units with the maximum marginal utility and network coverage in each iteration.

5.1 Basic QuIC-IoT

We design a basic QuIC-IoT, BAS-QuIC, which aims to maximize the overall *utility* during the deployment process. In each iteration, the sensor deployer selects the sensing unit with the *maximum utility gain*, which is derived by assuming the unit has a network connection to the edge server. Then, the network constructor establishes the connection for the selected sensing unit by deploying network units on the shortest path if necessary. Each pair of locations on the path has maximized transmission coverage. BAS-QuIC includes the sensing unit and networking units on the path in the deployment plan and executes the next iteration. The process ends when the deployment cost exceeds the budget.

Sensing unit with the maximum utility gain. We define the *utility gain* as the *sensing coverage improvement* weighted by each cell's criticality after (i) deploying a new sensing unit, (ii) upgrading (installing a new module) a sensing unit in the deployment plan, and (iii) leveraging the sensing unit in the plan for another parameter. In each iteration, the sensor deployer (Algo. 1) has the following inputs: (i) the current deployment plan χ, (ii) a set of candidate locations $\mathcal{L}$, (iii) a set of all PHoI's parameters Φ, (iv) a module matrix M_ϕ that describes the implementation methods to capture Φ, and (v) a placement matrix M_p^s that shows possible deployment locations for each module. A set of sensing units $\hat{\mathcal{U}}^s$ is recorded in χ. The deployment information of each $u \in \hat{\mathcal{U}}^s$ consists of two functions, $\hat{f}_{imp}(u, m_{\phi,i})$ and $\hat{f}_g(u)$, representing its current implemented modules and deployment function, respectively.

The sensor deployer computes the *utility gain* for all possible compositions of a sensing unit at each candidate location l for capturing each parameter ϕ (line 3). A possible utility gain can be derived from: (i) installing a new sensing unit (line 12), (ii) upgrading a sensing unit in χ (line 9), and (iii) reusing a sensing unit in χ (line 7). The one with the maximum utility gain is included in $\hat{\mathcal{U}}^s$ of χ (line 14). In particular, let $\mathcal{U}^{s'} = \hat{\mathcal{U}}^s \cup u$ denote the sensing unit set of a new plan χ' after including u deployed at l in χ. $f'_{imp}(u, m_{\phi,i}) = 1$ and $f'_g(u) = l$ are u's implementation and deployment functions in χ'. The *sensing coverage improvement* for cell c by adopting χ' is:

$$P(\chi', \chi, l, \phi, c) = \sum_{\substack{\forall f'_{imp}(u, m_{\phi,i})=1, \\ \forall u \in \mathcal{U}^{s'}}} p(u, c, m_{\phi,i}) - \sum_{\substack{\forall \hat{f}_{imp}(u, m_{\phi,i})=1, \\ \forall u \in \hat{\mathcal{U}}^s}} p(u, c, m_{\phi,i}). \tag{5}$$

Then, the *utility gain* to adopt χ' is:

$$\mathcal{G}(u, m_{\phi,i}, l, \phi) = \sum_{\forall c \in C, \forall a \in \mathcal{A}} \sigma(a, c, \phi) \times P(\chi', \chi, l, \phi, c). \quad (6)$$

Note that, in Algo. 1, we let $u =\sim$ represent upgrading a sensing unit in χ by adding module $m_{\phi,i}$; if $u =\sim$ and $m_{\phi,i} =\sim$, a sensing unit in χ can be leveraged for other parameters.

Network construction with the Tx-aware shortest path. Once a new sensing unit or module is installed at candidate location $\hat{l}$, the network constructor (Algo. 2) checks the existence of a network connection to the edge server provided by the current plan. Two routes that have possibly fewer hops are from $\hat{l}$: (i) directly to the edge server at location l_{edge} (line 2) and (ii) to the nearest networking unit at location l_n with a connection to the edge server (line 3). We design a *Transmission range (Tx)-aware graph* that captures the transmission characteristic if networking units are deployed. Each vertex represents a candidate location $l \in \mathcal{L}$, and the edge between $l \in \mathcal{L}$ and $l' \in \mathcal{L}$ has a weight $w(l, l')$ capturing the characteristic. The network constructor selects the *Tx-aware shortest path* of the aforementioned two routes with fewer hops to deploy networking units to reduce cost (line 4). In particular, let $l \in \mathcal{L}$ and $l' \in \mathcal{L}$ denote two locations in a Tx-aware graph; moreover, the newly included sensing unit or module has network interface d with a transmission range r^{tr}. The weight between l and l' is:

$$w(l, l') = \begin{cases} \frac{r^{tr} - dist(l,l')}{r^{tr}}, \ if \ dist(l, l') \leq r^{tr}, \\ \infty, \ otherwise, \end{cases} \quad (7)$$

where $dist(l, l')$ is the distance between l and l'.

5.2 Enhanced QuIC-IoT – Maximizing Marginal Utility and Network Coverage

We design an enhanced QuIC-IoT, EN-QuIC, that further optimizes the deployment process of BAS-QuIC. We propose to deploy the devices with a higher marginal performance (improvement per budget spent) in sensing and networking to optimize the overall utility, especially under a stringent deployment budget.

Marginal utility in sensor deployer. We introduce the sensor deployer's *marginal utility*, which is the utility gain per budget spent on deploying/upgrading a sensing unit. EN-QuIC and BAS-QuIC share the same algorithm structure (Algo. 1). Instead of utility gain, EN-QuIC computes the marginal utility for all possible compositions of a sensing unit at each candidate location l for capturing each parameter ϕ (line 3). Three possible deployments are: (i) installing a new sensing unit (line 12), (ii) upgrading a sensing unit (line 9), and (iii) reusing a sensing unit (line 7). In particular, let $\Delta C(u, m_{\phi,i})$ denote the cost of deploying a new sensing unit u with module $m_{\phi,i}$. Similar to the utility gain, in Algo. 1, if $u =\sim$, it represents the cost of adding an extra $m_{\phi,i}$ to u. Hence, the marginal utility of the sensor deployer is:

$$e_{cov} = \mathcal{G}(u, m_{\phi,i}, l, \phi) / \Delta C(u, m_{\phi,i}). \quad (8)$$

e_{cov} is computed at lines 9 and 12 instead of $\mathcal{G}(u, m_{\phi,i}, l, \phi)$. If a sensing unit can be reused (line 7), $e_{cov} = \infty$.

Marginal network coverage in network constructor. We define the network constructor's *marginal network coverage* as the *network coverage gain* per budget spent on deploying networking units. Similar to BAS-QuIC, the network constructor (Algo. 2) identifies the Tx-aware shortest path from a sensing unit's location $\hat{l}$ to (i) the edge server's location l_{edge} (line 2) and (ii) the nearest networking unit's location l_n (line 3), which are denoted as $P(\hat{l}, l_{edge})$ and $P(\hat{l}, l_n)$. The network constructor selects the shortest path with higher marginal network coverage for each sensing unit. In particular, let l_{dst} denote the destination of a Tx-aware shortest path. The *network coverage gain* is $\mathcal{G}(P(\hat{l}, l_{dst}))$, the number of additional candidate locations covered by deploying network units on $P(\hat{l}, l_{dst})$. Besides, let $\Delta C(P(\hat{l}, l_{dst}))$ denote the cost to deploy network units on $P(\hat{l}, l_{dst})$. If a network unit has been deployed at a location, there is no extra cost since we can reuse it. The marginal network coverage of deploying networking units on $P(\hat{l}, l_{dst})$ is:

$$e_{net} = \mathcal{G}(P(\hat{l}, l_{dst})) / \Delta C(P(\hat{l}, l_{dst})), \quad (9)$$

where l_{dst} is either l_{edge} or l_n.

EN-QuIC provides user-specified weights w_s and w_n of marginal performance in deploying sensing and networking units. EN-QuIC computes the following for each sensing unit and its Tx-aware shortest path: $w_s \times e_{cov} + w_n \times e_{net}$. The one with the highest value is included in the deployment plan in each iteration.

5.3 Reconfiguration During A Short-Term Event

Once the IoT infrastructure is deployed, it captures PHoI and transmits the data to the edge server. QuIC-IoT reconfigures the IoT deployment based on the captured evolution of PHoI and experts' domain knowledge. First, QuIC-IoT re-analyzes each cell's criticality by leveraging the captured data (e.g., wind speed) to drive models (e.g., fire simulators) to generate a new set of parameters (e.g., fire behavior). Additional specialized devices (e.g., drones) are then leveraged to capture emergent data by giving a set of possible candidate locations. QuIC-IoT re-executes the algorithms with the new inputs, such as candidate locations, criticality estimates, devices, etc., to generate a new deployment plan/reconfiguration.

5.4 Practicality of QuIC-IoT

QuIC-IoT can be applied to monitor other impending disaster scenarios, such as floods, hurricanes, and wildfires, where there may be days/hours of warning time. Tools to model physical phenomena (flood modeling, hurricane prediction) have been developed by domain experts and can be utilized to instrument communities for damage and loss monitoring. QuIC-IoT can help identify locations experiencing a higher probability of severe damage and impact. Besides, devices may fail or disconnect from communication networks in unpredictable ways. Sensing quality may be damaged due to high temperatures despite fireproofing of components. Candidate locations generated by QuIC-IoT may be inconvenient to instrument - local connectivity and sensing issues may exist, and this will require data recalibration or even repositioning of devices during deployment/operation. Our ongoing research aims to address these issues via reliability methods, including health monitoring of the IoT deployment using heartbeat protocols, exploiting mobile data collectors (e.g., drones) when connectivity issues arise during operation, and redundant sensors and sensing modalities.

Moreover, QuIC-IoT is designed for IoT deployment in a relatively small and bounded region. Our prescribed fire usecase usually

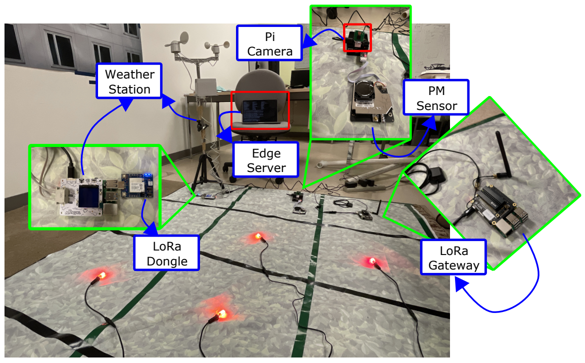


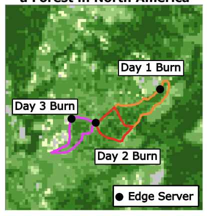
Figure 5: Prototype system for RxFire monitoring.

Snippet of the Real Burn Plan

	Environmental Prescription Range			
	Burn Area		Surrounding Area	
Parameter	**Low**	**High**	**Low**	**High**
Relative Humidity	65	23	65	25
Wind Speed	5	10	5	10
Temperature (°F)	38	80	38	80
	Fire Behavior Prescription Range			
Parameter	**Low**		**High**	
	Burn Area	**Surrounding Area**	**Burn Area**	**Surrounding Area**
Flame Length (feet)	2.4–3.2	1.7	4.3–4.8	3.6
Rate of Spread (chains/hr)	2.1–3.1	1	6.5–8.0	3.9
Spotting Distance (miles)	0.1	0.1	0.3	0.3
	Historical Weather Range			
	Wind Speed	Wind direction	Temperature	Relative Humidity
Maximum	21	359	104	99
Minimum	0	0	21	10

(b)

Actual Burn Site in a Forest in North America



(a)

Figure 6: Snippet of a real burn plan: (a) canopy cover of the burn site and (b) environmental/fire behavior prescription.

Table 2: Module Composition and Devices' Cost

	Raspberry Pi	Weather Station	PM Sensor	Pi Camera	Fire-Proof Case	LoRa Dongle
Cost	139.95	113.05	33.99	14.99	100	84.99
Fire#1	X			X		
Fire#2	X			X	X	
Air Quality#1	X		X			
Air Quality#2	X		X			X
Weather#1	X	X				
Weather#2	X	X				X

involves a few personnel (<10), a few square miles at most, and takes around a day to finish the operation. Hence, the level of instrumentation expected is smaller (i.e., fewer devices) than a city-scale deployment, like the one considered in SONYC [6]. The fact that these scoped situations are short-term and also a factor in keeping the complexity within bounds. In general, a larger, long-term city-wide deployment involves many stakeholders/communities with various requirements of sensing, communication, and computing, which complicates the problem. In fact, our previous work [11] addresses the nature of IoT planning in urban/suburban communities (like Irvine, CA, USA) where the reuse of deployed devices for multiple purposes over time is of value.

6 EVALUATING THE QUIC-IOT APPROACH

In this section, we evaluate the QuIC-IoT approach for short-term IoT deployment. We developed a prototype system for our driving usecase, monitoring RxFires, in a lab setting. This allowed us to integrate and refine system components and their interaction, e.g., the criticality analyzer and physics-based modeling tools to estimate the anomaly probability of Rxfires. We conduct a broad range of experiments with various settings to evaluate QuIC-IoT's algorithms. Two burns with different scales are explored: (i) a small regular-size burn from a real burn plan and (ii) a large burn to explore the possibility of increasing the burn site's size and, thus, RxFire's efficiency. We evaluate the efficacy of our placement methods under multiple weather conditions (normal, abnormal, and extreme) and burn scales. QuIC-IoT's capability to capture anomalies is evaluated and quantified.

6.1 QuIC-IoT Lab Prototype

For a RxFire operation, we build a prototype system that monitors the following physical phenomena: (i) fire spread, (ii) sudden weather change, and (iii) sudden air quality change. Ensuring a safe RxFire operation requires monitoring the following parameters of fire spread: (i) flame length, (ii) fire spread rate, and (iii) spotting distance (distance an ember travels). Parameters of a sudden weather change include temperature, wind speed/direction, and relative humidity. Particulate Matters (PM) 2.5 and 10 are regular parameters for detecting a sudden air quality change. The fire breaks partition a burn site into two areas: the operation region (burn area) and the others (surrounding area). Both areas require fire monitoring, whereas only the surrounding area requires weather and air quality monitoring. Table 2 lists the modules' composition (required devices) that can monitor each parameter and each device's cost. A transparent fire-proof case enables a sensing unit, e.g., a Raspberry Pi with a camera, to be deployed in the burn area. The prototype's primary communication technologies include WiFi and LoRa. A camera requires a higher bandwidth provided by WiFi, whereas other sensors accept both technologies. The Raspberry Pi's onboard WiFi antenna and a LoRa dongle via USB are the network interface for transmission. The data exchange between the Raspberry Pis and the edge server relies on a WiFi router and a LoRa gateway, which cost $110 and $299.9, respectively.

We build a prototype system to demonstrate and verify the required functionality at a real burn site, such as sensing, communication, and computing, before taking them to the field. Fig. 5 shows the prototype at a mock burn site created by a grid mat. Each red light mimics a flame in a grid. A weather station consists of a Raspberry Pi connected via GPIO to an off-the-shelf weather shield embedded with temperature and relative humidity sensors; the weather shield connects to an anemometer and a wind vane via RJ11 cables. The weather station transmits its data through a LoRa dongle. A multi-functional sensing unit has a PM sensor and a Pi camera for air and fire monitoring with built-in WiFi functionality. A WiFi router and a LoRa gateway forward the data to the edge server, which is a laptop in the middle. The Raspberry Pis execute SCALE client [7] that (i) publishes the data to an MQTT broker on the edge server and (ii) transmits the camera's images via restful

API to the edge server. The edge server receives the data via subscribing and restful API and stores the data in a local PostgreSQL database.

6.2 Experimental Setup

Input from domain experts and a real dataset. We exploit an existing dataset from the LANDFIRE project [1] to drive the simulation environment - this includes detailed features such as vegetation, slope, canopy, etc., derived from satellite images. The dataset's resolution bounds the minimum cell size we use (30mX30m); the chosen resolution in the dataset reflects a single vegetation type in a specific cell; this allows for faster fire simulation. The domain experts provide us with a detailed burn plan derived from past experiences and simulations that take into account the dimensions of the burn site (with clearly identified fire breaks), topography, fuel, and expected weather conditions. Domain experts on the paper also provided us with potential candidate locations to deploy devices in this burn plan. However, to model uncertainty, we explore numerous scenarios with varied inputs, such as budgets to instrument burn sites, weather conditions, and varied candidate deployment locations, which are introduced as follows.

Settings derived from a real burn plan. Fig. 6 shows a snippet of the real burn plan in Blodgett Forest, CA, USA, where Fig. 6(a) is the canopy cover of the burn site that hosts three separate RxFires on three consecutive days. We conduct the experiments in two real-world burn sites (rectangles centered at approximately the Day-2 burn center) derived from the burn plan: (i) small burn (≈500 m^2) and (ii) large burn (≈3000 m^2). For each burn site, we derive the topography (elevation, slope, aspect) and vegetation (fuel model, canopy cover) data from LANDFIRE [1]. Fig. 6(b) records the prescribed range of environmental and fire-related parameters in each area (burn and surrounding) in an actual burn plan. It also provides the forest's historical environmental parameters[1]. We further define three weather types: (i) normal weather with prescribed parameters, (ii) abnormal weather with parameters from high to the maximum, e.g., 10–21 for wind speed, and (iii) extreme wind with a wind speed of force 6–8 in the Beaufort wind scale [35].

Utilizing wind and fire spread models. We exploit a wind simulator, WindNinja [16], and a fire simulator, FARSITE [15], to generate RxFires under different conditions. Static inputs include the information in the data from LANDFIRE, where a burn site is divided into 30m×30m cells by default. WindNinja generates sophisticated wind parameters (speed, direction) for each cell based on the terrain and initial wind speed/direction. The initial wind speed and direction are uniformly sampled from the range defined in each weather type and from 0–360 degrees, respectively. FARSITE generates fire behaviors/parameters for each cell with environmental inputs, a fire break, and an ignition. Environmental inputs, including the generated wind parameters and other parameters (temperature and relative humidity), are uniformly sampled within the range defined in each weather type. The small burn uses the Day-2 burn fire break, i.e., the red polygon in Fig. 6(a). The large burn's fire break is a quadrilateral, with four corners uniformly generated within a distance of up to 1 km to each of the nearest two borders of the burn site. Each simulation has a uniformly generated ignition line within the fire break with a length of 25 and 150 m for small and large burns.

Parameters of sensing and networking units. We derive the parameters from experts and our prototyping experience. In the experiments, we exploit the prototype's real settings, e.g., cost, modules, and monitoring requirements of each area. Each sensor's sensing range (in meters) is: (i) weather station–300, (ii) PM sensor–200, and (iii) camera–100; the sensing parameter, α_d, in Eq. (1) is the reciprocal of sensing range for each sensor. The transmission ranges of WiFi and LoRa are 100 m and 1 km, respectively. Finally, the candidate locations are generated in two ways: (i) each cell's center and (ii) uniformly sampled on the fire break. Only sensing units equipped with a fire-proof box can be deployed in the burn area. All devices can be deployed on the fire break and in the surrounding area. The edge server location for the small burn is indicated in Fig. 6(a). The edge server for the large burn is at a fixed location near the fire break.

6.3 Evaluating QuIC-IoT Criticality Analysis

We evaluate the criticality analyzer in QuIC-IoT with the following settings: (i) small burn under all three possible weather types and (ii) large burn with normal and abnormal weather conditions. The burn plan is used to determine abnormal parameters, i.e., the value exceeds the prescribed range of low parameters in Fig. 6(b). We generate the training and testing datasets of identical sizes to derive a similar probability distribution for a fair comparison. We use the Mean Squared Error (MSE) as the evaluation metric and report the average MSE of the probability in the derived PMF.

We compare the QuIC-IoT's criticality analyzer (developed with neural network regression) with three other regression models: (i) polynomial, (ii) SVM, and (iii) random forest. The criticality analyzer's architecture includes two hidden layers of 64 nodes with a ReLU activation function. Its output layer contains only one node with a Sigmoid activation function. We train the criticality analyzer with 100 epochs. We report the best results for the polynomial and random forest regression, derived with a degree of 7 and 200 trees, respectively. A hundred equal-sized bins are used to count the occurrences in the PMF function for all models.

We use solid and dashed lines to represent the training and testing results in Fig. 7. Generally, the QuIC-IoT's criticality analyzer has the lowest average MSE in testing. Note, however, in Fig. 7(c), while polynomial regression has a slightly lower MSE during testing, its average MSE is far worse in quantifying the criticality for flame length and fire spread rate. In Fig. 7(c) and (f), random forest regression has much lower MSEs in training than our method; however, its performance deteriorates in testing, representing possible overfitting. Moreover, since the criticality analyzer's learning process relies on the occurrence of anomalies, it performs better in small burns with extreme wind scenarios. That is, its average MSE is below 2% in Fig. 7(a)–(b) and 5% in Fig. 7(d)–(e). Moreover, Table 3 shows the training times of the proposed criticality analyzer and traditional baseline methods with 1000 data points. The mean training time of all cells is reported. Since neural networks have higher complexity, QuIC-IoT has the longest mean training time and the highest standard deviation; however, its training time is relatively short and bounded (<7.4s). Note that we train the model

[1]Weather Underground: https://www.wunderground.com/history

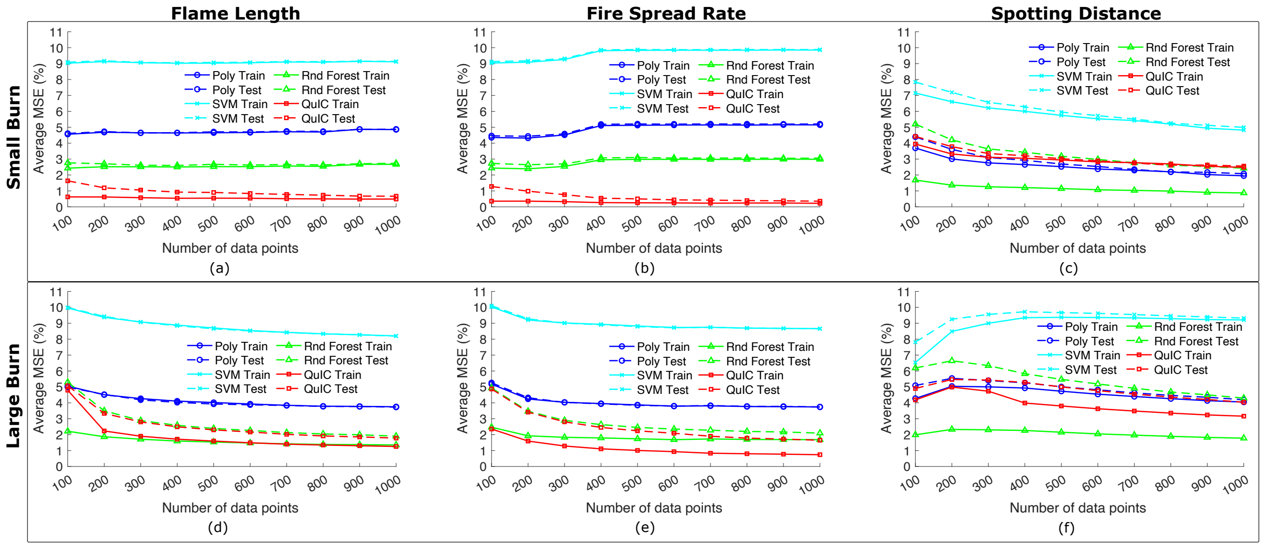


Figure 7: Average MSE of the fire spread's parameters in different burns. (a) Flame length in small burns. (b) Fire spread rate in small burns. (c) Spotting distance in small burns. (d) Flame length in large burns. (e) Fire spread rate in large burns. (f) Spotting distance in large burns.

on a CPU; hence the training time could be further optimized by leveraging a GPU for parallelization.

6.4 Experimental Results

RxFire in Blodgett Forest, CA, USA. We first evaluate QuIC-IoT's algorithms in the small burn setting. To the best of our knowledge, the short-term IoT deployment problem has no prior solutions. Thus, we develop two simple baseline sensor deployers: (i) traditional (TRAD) that purely maximizes the overall sensing coverage with a similar greedy heuristic in [33] and (ii) maximum criticality (Crit) that deploys a sensing unit to the location with the maximum criticality in each iteration. For comprehensive studies, we separately evaluate QuIC-IoT's sensor deployer and network constructor by various combinations. We denote the basic and enhanced QuIC-IoT's sensor deployer as QuIC and QuIC$^+$. QuIC-IoT's network deployers, Tx-aware shortest path and maximum marginal network coverage, are denoted as S and M. A hyphen concatenates a sensor deployer and a network constructor to form a deployment method, e.g., QuIC-S and QuIC$^+$-M represent BAS-QuIC and EN-QuIC, respectively. We use the following performance metrics to evaluate all 8 combinations under a range of budgets for instrumentation - (i) the overall utility, Eq. (4a), of the IoT instrumentation, (ii) the cost performance (C/P) index, the ratio of the overall utility, Eq. (4a), to the overall budget spent in the deployment, Eq. (4b), (iii) mean sensing coverage, averaging the ratio of the covered area to the overall area of all parameters, and (iv) the number of implemented data flows. Note that different sensors have different sensing/coverage capabilities, which are captured by the α_d parameter in Eq. (1). We capture the difference by setting α_d related to each sensor's sensing range, i.e., the reciprocal of its sensing range. The covered area is composed of the cells that are covered by (with a positive value) Eq. (1) for each parameter. These performance metrics allow us to evaluate the IoT deployment under each setting.

By conducting comprehensive experiments, we derive the best weight of the marginal performance in sensing w_s and networking w_n, which are 0.8 and 0.2, for EN-QuIC. We exploit the results from the previous experiments to set the criticality for fire behaviors in each cell; other parameters' criticality in each cell is uniformly set to 1. We let solid and dashed lines in Fig. 8 represent the network constructor with S and M. Deployment methods with S network constructor (excluding QuIC$^+$-S) are *non-marginal network constructor* methods, i.e., solid lines excluding QuIC$^+$-S. In Fig. 8(a), EN-QuIC derives the highest overall utility with improvements up to 49.33% compared to the baseline methods. BAS-QuIC's overall utility improves up to 27.21% compared to other non-marginal network constructor methods. Maximizing the marginal network coverage improves the overall utility by up to 30.29% compared to non-marginal network constructor methods due to efficient budget management. Fig. 8(b) confirms that EN-QuIC has the highest C/P index. Besides, the C/P index decreases for all methods when the budget increases since the sensing coverage increases, as shown in Fig. 8(c). The marginal gain of deploying units decreases when the sensing coverage increases.

Fig. 8(c) shows that EN-QuIC derives comparable sensing coverage as TRAD-M, which maximizes the sensing coverage; however, EN-QuIC derives the highest overall utility, as shown in Fig. 8(a), due to the rigorous consideration of sensing coverage, criticality, and connectivity. Though BAS-QuIC has the lowest sensing coverage, it still has the highest utility among non-marginal network constructor methods due to the effective deployment that maximizes the overall utility. Moreover, the sensing coverage improves while applying the M network constructor, meaning that maximizing the marginal network coverage spares the budget such that more sensing units can be deployed. Fig. 8(d) shows that EN-QuIC

	Large Burn								**Small Burn**							
	Mean Training Time (ms)				**Standard Deviation (ms)**				**Mean Training Time (ms)**				**Standard Deviation (ms)**			
Fire Behevior	Poly	SVM	Rnd Forest	QuIC	Poly	SVM	Rnd Forest	QuIC	Poly	SVM	Rnd Forest	QuIC	Poly	SVM	Rnd Forest	QuIC
Flame Length	1.758	0.641	256.746	7220.582	0.269	0.122	11.129	273.549	1.553	0.63	256.506	7336.509	0.124	0.15	10.09	452.831
Fire Spread Rate	1.778	0.666	266.562	7339.404	0.269	0.114	11.765	319.565	1.642	0.631	263.987	7369.863	0.214	0.099	9.954	321.488
Spotting Distance	1.772	0.681	256.282	7297.73	0.266	0.13	10.744	273.575	1.747	0.691	258.642	7259.192	0.272	0.133	9.449	435.992

Table 3: Execution Time of each Learning Technique

implements the most data flows due to effective budget management. By adopting the M network constructor, the number of implemented data flows increases by around 1.17 times compared to non-marginal network constructor methods, which all implement a similar number of data flows.

Large scale RxFire. We evaluate QuIC-IoT's ability to support RxFire operations in large burn area settings. Increasing the burn size improves the efficiency of RxFire, e.g., burning the whole area in Fig. 6(b) in one day rather than three days. We also explore the methods' efficacy under limited deployment budgets. We assume all sensing and networking units are fire-proof and can be deployed in both burn and surrounding areas. The burn site is resized into 100m×100m cells, and cell centers are candidate locations. Fire is the only PHoI. The rest of the settings are identical to the experiments in small burns. We compare QuIC-IoT with the baseline methods and report the results in Fig. 9.

Fig. 9(a) shows that EN-QuIC derives the highest overall utility and improves up to around 3.55 times compared to the baseline methods. TRAD-S has the second highest overall utility since the traditional method derives a higher sensing coverage in a large area where the units cannot fully cover, as shown in Fig. 9(c). Note that even though TRAD-S seems to perform comparably to EN-QuIC, EN-QuIC still outperforms TRAD-S by 13.57% of utility (12K budget). The two algorithms have the same time complexity since TRAD-S only adopts a different greedy metric in Algo. 1. Thus, EN-QuIC is still the most appropriate method in this setting. Typically, applying the M network constructor has a significant improvement, especially for QuIC$^+$, by up to 3.29 times. However, the traditional method's utility deteriorates (e.g., TRAD-M) since the method cannot trade off deploying sensing and networking units for maximizing sensing and networking coverage. Fig. 9(b) shows that EN-QuIC maintains the highest C/P index, as TRAD-S's index decreases when the budget increases. More networking units are deployed due to lacking optimizing the network coverage.

Fig. 9(c) shows that both QuIC$^+$ methods significantly increase sensing coverage with an increasing budget. Maximizing the overall utility and maximizing the marginal performance in both sensing and networking improve the sensing coverage as the traditional method. When the budget is sufficient for effective deployment, EN-QuIC even derives a higher sensing coverage than purely maximizing the sensing coverage, i.e., TRAD-S. Fig. 9(d) shows that the number of implemented data flows has a similar trend as the sensing coverage. That is, the number of connected data flows bounds the number of locations covered by sensing units in large area settings. Overall, the results show that EN-QuIC has an effective deployment in a large burn site. Even with insufficient budgets, EN-QuIC still maintains the best performance compared to the baseline methods.

Evaluating anomaly capture rate. We evaluate the effectiveness of a generated deployment plan to capture anomalies using experimental settings for small and large burns as discussed earlier. The wind and fire spread model are exploited to generate simulations with three fire behaviors, i.e parameters: (i) flame length, (ii) fire spread rate, and (iii) spotting distance. For each cell, the simulator generates a unique value for each of the fire behavior parameters above. We identify the cell as having an anomaly when the parameter value exceeds an abnormality threshold (as defined by the domain expert). A sensor can only capture a cell's anomaly if the cell is within its sensing range. Thus, we approximate the ability of a deployment to capture anomalies using a metric called the anomaly capture rate, i.e. the ratio of captured anomalies to the total number of anomalies. Small burns use abnormal weather and extreme wind to generate anomalies, whereas large burns leverage only abnormal weather. Each setting has 100 simulations. We report the average percentage of the captured anomalies of all three behaviors. Fig. 10(a) shows that EN-QuIC captures the most anomalies in small burns. Almost all anomalies (up to 97.5%) are captured with sufficient deployment budgets. Since BAS-QuIC has the lowest sensing coverage, its ability to capture anomalies is the worst. Fig. 10(b) shows that EN-QuIC captures more anomalies than TRAD-S with sufficient budgets for large burns. Its anomaly capture rate increases significantly compared to TRAD-S, whereas these two methods' sensing range differs relatively insignificant, as shown in Fig. 9(c). That is because EN-QuIC captures more anomalies per sensing coverage.

7 RELATED WORK

The IoT deployment problem has been studied across different domains; much of this work focuses on issues of sensor placement for capturing physical phenomena in the focus domain and gateway deployment for ensuring network access, given the geospatial range of interest. The canonical sensor placement problem (also known as the point coverage problem) [33] aims to maximize the overall sensing coverage by instrumenting sensors using models that capture a sensor's coverage and capacity. Multiple variants/constraints are studied while maximizing coverage, including under a limited number of sensors, minimizing network cost, under different coverage criteria, etc. The problem can be equated to the well-known NP-hard *set-cover problem*, which is usually solved by greedy heuristic methods. Sensing coverage may depend on the sensor's detection angle using an unlimited model (360 degrees) [9, 34, 39] or a limited angle model [2, 8, 17, 24]. Additionally, the distance between a sensor and a possible location where the phenomena occur impacts the coverage in a stochastic manner; the coverage probability decays with distance [25] and is further truncated by a threshold [40, 41]. Domain-specific sensor placement techniques have been developed

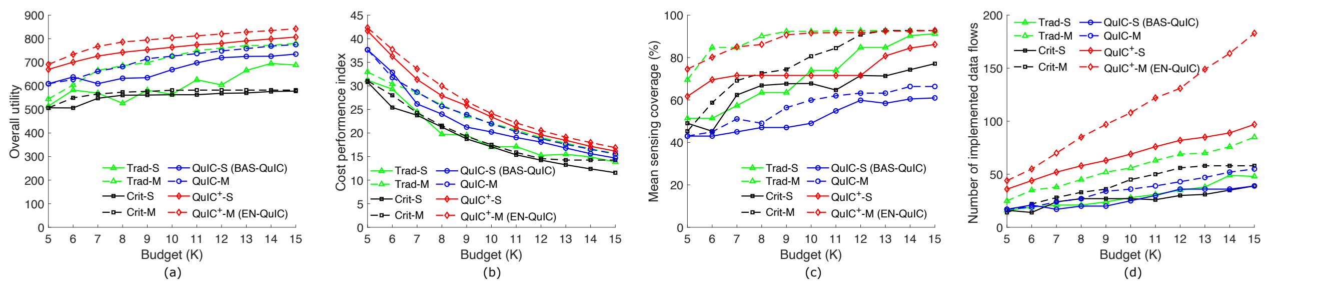


Figure 8: Experiment results in an RxFire in a forest in North America: (a) overall utility, (b) cost performance index, (c) mean sensing coverage, and (d) the number of implemented data flow.

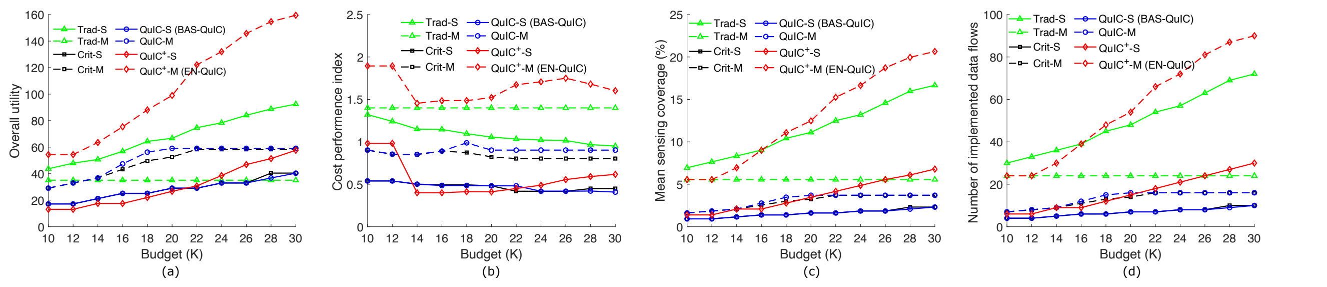


Figure 9: Experiment results in a large RxFire burn: (a) overall utility, (b) cost performance index, (c) mean sensing coverage, and (d) the number of implemented data flows.

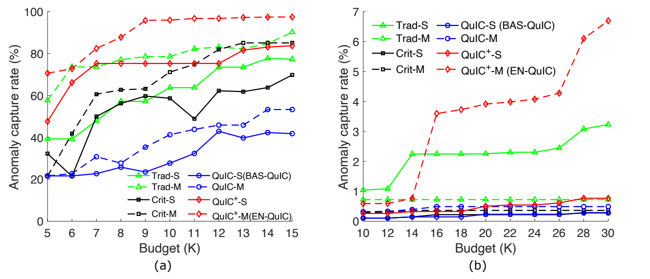


Figure 10: The ratio of captured anomalies in: (a) small burns and (b) large burns.

for wide-area infrastructure such as smart agriculture [36], water infrastructure [31, 32, 37], and smart power grids [21, 30]. An event's impacts on the area in terms of demography and economy are considered during the deployment [31, 32]; however, the impacts are relatively static compared to physical phenomena, which are highly dynamic and complex to model. In contrast to the above efforts, we exploit a more in-depth approach to integrate expert-developed models to identify and prioritize critical locations where sensors must be placed to help detect potential anomalies, desirably even before they occur.

The gateway deployment problem aims to deploy gateways to provide network access to sensors in specific regions. This problem has been studied in the context of various access networks, including mesh networks [4], LoRa [26], and 5G ultra-dense networks [27]. Metrics or constraints used in related studies include throughput and coverage and their joint maximization [38], energy efficiency [5, 12], and QoS constraints [4, 13, 19, 20]. In several of these efforts, network component placement focuses on network construction, assuming known locations of sensors that have often already been deployed (or vice-versa). A joint planning that uses a cross-layer approach is proposed in SmartParcels [11] to deploy sensors and network devices in an integrated manner. A key distinguishing feature of the work proposed in this paper is the role played by expert-developed models in addressing joint placement.

8 DISCUSSION AND FUTURE DIRECTIONS

We presented QuIC-IoT, a framework to construct and deploy IoT infrastructure for short-term events by leveraging expert-developed models to understand the parameters (e.g., fire behaviors) of physical phenomena of interest (e.g., fire spread). QuIC-IoT analyzes each location's criticality (anomaly probability) by integrating simulation results of numerous scenarios capturing real-world physical conditions (e.g., weather conditions, terrain, vegetation, etc.). The platform considers the generated criticality, sensing coverage, network connectivity, and marginal performance in deploying sensing and networking units while constructing IoT infrastructure. We plan to extend our model-based approach to study other complex scenarios and physical phenomena where rapid and flexible instrumentation can help gain situational awareness. We plan to more deeply understand the role of observation granularity to gain deeper insights into real events and design adaptation mechanisms for dynamic reconfiguration of sensors, data collection, and analytics in challenged network scenarios. Such adaptation can enable the deployment of resilient infrastructure and systems that can cope with the dynamics of physical phenomena in real time. Moreover, during the operational phase (e.g., RxBurn) - we can exploit UAVs to gather information to augment the in-situ instrumentation deployed apriori (days before burn in our case) by QuIC-IoT. This paper focuses on addressing an offline problem given a burn plan and geospatial information about the burn site. The emphasis is to utilize the physics-based modeling tools used by domain experts to help with planning the instrumentation of the target site. An offline cost-benefit analysis can help wildfire management authorities to determine sensing/communication components (that may need to be purchased) and where to deploy them. We expect to use UAVs in conjunction with QuIC-IoT during the dynamic setting - key problems to be addressed include the determination of live monitoring tasks, energy-efficient and task-dependent motion planning, and ensuring wireless connectivity in the wildland setting. For example, only images that contain an abnormal fire are useful to analyze for a quick response, such as releasing retardant to the area identified with anomalies.

ACKNOWLEDGMENTS

We acknowledge the funding support from the University of California Laboratory Fees Research Program funded by the UC Office of the President (UCOP), grant ID LFR-20-653572 and NSF IIS Award #40284388.

REFERENCES

[1] [n. d.]. LANDFIRE: LANDFIRE Existing Vegetation Type layer. (2013, June - last update). U.S. Department of Interior, Geological Survey, and U.S. Department of Agriculture [Online]. Available: http://landfire.cr.usgs.gov/viewer/ [2013, May 8].

[2] Jing Ai et al. 2006. Coverage by directional sensors in randomly deployed wireless sensor networks. *Journal of Combinatorial Optimization* (2006).

[3] Ilkay Altintas et al. 2015. Towards an integrated cyberinfrastructure for scalable data-driven monitoring, dynamic prediction and resilience of wildfires. *Procedia Computer Science* 51 (2015), 1633–1642.

[4] B. Aoun et al. 2006. Gateway Placement Optimization in Wireless Mesh Networks With QoS Constraints. *IEEE JSAC* 24, 11 (2006), 2127–2136.

[5] Usman Ashraf. 2017. Energy-Aware Gateway Placement in Green Wireless Mesh Networks. *IEEE Communications Letters* 21, 1 (2017), 156–159.

[6] Juan P Bello et al. 2019. SONYC: A system for monitoring, analyzing, and mitigating urban noise pollution. *Commun. ACM* 62, 2 (2019).

[7] Kyle Benson et al. 2015. Scale: Safe community awareness and alerting leveraging the internet of things. *IEEE Communications Magazine* (2015).

[8] Yanli Cai et al. 2007. Target-oriented scheduling in directional sensor networks. In *IEEE INFOCOM*. 1550–1558.

[9] Mihaela Cardei et al. 2005. Maximum network lifetime in wireless sensor networks with adjustable sensing ranges. In *IEEE WiMob*, Vol. 3. 438–445.

[10] Charles E Catlett et al. 2017. Array of things: a scientific research instrument in the public way: platform design and early lessons learned. In *Proceedings of the 2nd international workshop on science of smart city operations and platforms engineering*. 26–33.

[11] Tun-Chun Chang et al. 2021. SmartParcels: Cross-Layer IoT Planning for Smart Communities. In *ACM/IEEE IoTDI*.

[12] Cemil Can Coskun et al. 2015. A greedy algorithm for energy-efficient base station deployment in heterogeneous networks. In *IEEE ICC*. 7–12.

[13] Y. Drabu et al. 2008. Gateway Placement with QoS Constraints in Wireless Mesh Networks. In *Seventh International Conference on Networking*.

[14] Uriel Feige. 1998. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)* 45, 4 (1998), 634–652.

[15] Mark A Finney. 1998. *FARSITE, Fire Area Simulator–model development and evaluation*. Number 4. US Department of Agriculture, Forest Service, Rocky Mountain Research Station.

[16] Jason M Forthofer. 2007. Modeling wind in complex terrain for use in fire spread prediction. *Thesis. Fort Collins, CO: Colorado State University.* (2007).

[17] Giordano Fusco et al. 2009. Selection and orientation of directional sensors for coverage maximization. In *6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*. IEEE, 1–9.

[18] Bill Gabbert. 2021. Two escaped prescribed fires in California. *Wildfire Today* (2021).

[19] Ilias Gravalos et al. 2018. Efficient Network Planning for Internet of Things With QoS Constraints. *IEEE Internet of Things Journal* 5, 5 (2018).

[20] Ilias Gravalos and other. 2016. Efficient Gateways Placement for Internet of Things with QoS Constraints. In *IEEE GLOBECOM*.

[21] Joe-Air Jiang et al. 2020. A Novel Sensor Placement Strategy for an IoT-Based Power Grid Monitoring System. *IEEE Internet of Things Journal* (2020).

[22] Kostas Kalabokidis et al. 2016. AEGIS: a wildfire prevention and management information system. *Natural Hazards and Earth System Sciences* (2016).

[23] Morgan Lee and Cedar Attanacio. 2022. Largest wildfire in New Mexico history linked to planned burns. *Public Broadcasting Service (PBS)* (2022).

[24] Liang Liu et al. 2008. On directional k-coverage analysis of randomly deployed camera sensor networks. In *IEEE ICC*. 2707–2711.

[25] Seapahn Megerian et al. 2002. Exposure in wireless sensor networks: Theory and practical solutions. *Wireless Networks* 8, 5 (2002), 443–454.

[26] B. Ousat et al. 2019. LoRa Network Planning: Gateway Placement and Device Configuration. In *IEEE International Congress on Internet of Things*.

[27] Mital Raithatha et al. 2021. A Fast Heuristic for Gateway Location in Wireless Backhaul of 5G Ultra-Dense Networks. *IEEE Access* (2021).

[28] M. Rathore et al. 2016. Urban planning and building smart cities based on the internet of things using big data analytics. *Computer networks* (2016).

[29] Kevin C Ryan, Eric E Knapp, and J Morgan Varner. 2013. Prescribed fire in North American forests and woodlands: history, current practice, and challenges. *Frontiers in Ecology and the Environment* 11, s1 (2013), e15–e24.

[30] A. N. Samudrala et al. 2020. Sensor Placement for Outage Identifiability in Power Distribution Networks. *IEEE Transactions on Smart Grid* (2020).

[31] Praveen Venkat. et al. 2019. Augmenting in-situ with mobile sensing for adaptive monitoring of water distribution networks. In *ACM/IEEE ICCPS*.

[32] Praveen Venkateswaran et al. 2018. Impact driven sensor placement for leak detection in community water networks. In *ACM/IEEE ICCPS*.

[33] Bang Wang. 2011. Coverage Problems in Sensor Networks: A Survey. *ACM Computing Surveys (CSUR)* 43, 4 (2011), 1–53.

[34] Jiong Wang et al. 2007. Energy efficient coverage with variable sensing radii in wireless sensor networks. In *IEEE WiMob*. 61–61.

[35] Force Land Water. 2005. The beaufort wind scale. (2005).

[36] F. Yang et al. 2020. A partition-based node deployment strategy in solar insecticidal lamps Internet of Things. *IEEE Internet of Things Journal* (2020).

[37] Deze Zeng, Lin Gu, Lu Lian, Song Guo, Hong Yao, and Jiankun Hu. 2016. On cost-efficient sensor placement for contaminant detection in water distribution systems. *IEEE Transactions on Industrial Informatics* 12, 6 (2016), 2177–2185.

[38] Yue Zhang et al. 2020. Joint Optimization of Placement and Coverage of Access Points for IEEE 802.11 Networks. In *IEEE ICC*.

[39] Zongheng Zhou et al. 2009. Variable radii connected sensor cover in sensor networks. *ACM Transactions on Sensor Networks (TOSN)* 5, 1 (2009).

[40] Yi Zou et al. 2004. Sensor deployment and target localization in distributed sensor networks. *ACM TECS* (2004).

[41] Yi Zou et al. 2005. A distributed coverage-and connectivity-centric technique for selecting active nodes in wireless sensor networks. *IEEE Trans. Comput.* 54, 8 (2005), 978–991.