2016 IEEE First International Conference on Internet-of-Things Design and Implementation

Resilient Overlays for IoT-based Community Infrastructure Communications

Kyle E. Benson*, Qing Han*, Kyungbaek Kim†, Phu Nguyen‡, Nalini Venkatasubramanian*
∗ Donald Bren School of Information and Computer Science, UC Irvine
† Department of Electronics and Computer Engineering, Chonnam National University
‡ The Henry Samueli School of Engineering, UC Irvine.
∗ {kebenson, qhan3, nalini}@ics.uci.edu

***Abstract*—This paper considers the reliable delivery of sensor data from Internet of Things (IoT) devices to distributed cloud service instances in the face of localized failures for event detection, community infrastructure management, and emergency response. We develop and explore the notion of GEOgraphically Correlated Resilient Overlay Networks (GeoCRON) to capture the localized nature of community IoT deployments and their impact/vulnerability in the context of small failures and massive disruptions such as that caused by a large seismic event. As a focus use case, we study the viability and utility of GeoCRON in two different community sensor networks. The first, a large participatory sensing-style network of small inexpensive seismic sensors, reports ground motion in homes and businesses to a cloud service for detecting and characterizing earthquakes. The second, a smaller infrastructure-based network of wireless devices embedded in the water distribution network, reports water pressure changes at pipe junctions that may be indicative of leaks that require human intervention. Using realistic topologies derived from experiences with real world community scale deployments, we study the impact of geo-correlated failures on the combined network infrastructure and evaluate a range of GeoCRON heuristics to improve commnunications resilience in this context. We validate the promise of the proposed GeoCRON-based approach using extensive simulations and an initial prototype system implementation.**

I. Community IoT Deployments

Advances in sensing, cloud computing, and wireless communications has created an emerging Internet-of-things (IoT) ecosystem - it has enabled the proliferation of small inexpensive devices and the ability to connect and manipulate sensor/actuator systems to create instrumented smartspaces, smart civil infrastructures and associated applications for the public. Examples of such smartspaces include personal spaces such as homes, offices, senior facilities, critical infrastructures such as airports, shipping and port facilities, and organizations such as schools and hospitals. Applications range from surveillance and security to personal safety and situational awareness in emergency response scenarios. While IoT devices are often deployed for a dedicated purpose, such as monitoring critical infrastructures, their data may be repurposed and exploited for new applications if access to this information are made available through open APIs.

As these devices penetrate more homes, offices, and community spaces, new efforts have aimed to improve the safety and quality of life for occupants of these instrumented spaces. For example, the Quake-Catcher Network [8] and Community Seismic Network [3] use small inexpensive accelerometers driven by volunteers' computers to monitor homes/offices for possible indications of seismic activity, as evidenced by anomalies in the detected ground shaking. Recently, the Safe Community Awareness and Alerting Network (SCALE) Project [9] uses multi-sensor multi-network devices deployed in homes along with cloud-based data exchange and analytics services, and an Internet phone service to alert residents and emergency personnel about potential emergencies in instrumented homes. SCALE is an example of a public-private partnership initiated in response to the NIST/WhiteHouse SmartAmerica Challenge, aiming to transform the lives of our general publics, irrespective of their economic strata or physical ability through the creation of community wide smartspaces. It demonstrates the ability to develop CPS platforms that are scalable, flexible, inexpensive and easy to deploy/manage at a community wide scale and support the creation of actionable situational awareness for the safety of our publics. This will help in alleviating an often-overlooked chasm in the digital divide where such technologies are often only available to those with the ability to afford it and the expertise to setup/manage it. Other recent projects look at instrumenting critical infrastructure or city spaces to improve efficiency, lower maintenance costs, and make data available for scientific studies. Such examples include Sentilo [5], an IoT platform developed in Barcelona to expose a RESTful API for both sensor and actuator devices, and AquaSCALE [19], an extension to SCALE for monitoring water distribution networks and quickly identifying potential water leaks.

As more individuals come to rely on these systems, especially for personal safety, we must clearly ensure a high degree of confidence in their reliable operation. IoT devices often tend to be low-powered, inexpensive, embedded systems with little intelligence and are therefore susceptible to a variety of problems including including faulty components, inaccurate sensing, and software bugs. As IoT systems scale and increasingly rely on cloud services to operate, resilience of these services, the software and algorithms comprising them, and the networks that link them with end devices are also crucial considerations. Resilient operation of applications and services in the presence of failures and disruptions is a key issue, especially as seen in recent disasters (e.g. Haiti/Japan earthquakes, Hurricane Sandy). While IoT deployments can be used to create dependable awareness and consequently improved decision making in disaster settings, this data must be quickly delivered in the face of massive geo-correlated network outages caused by large events.

978-1-4673-9948-7/16 $31.00 © 2016 IEEE
DOI 10.1109/IoTDI.2015.40

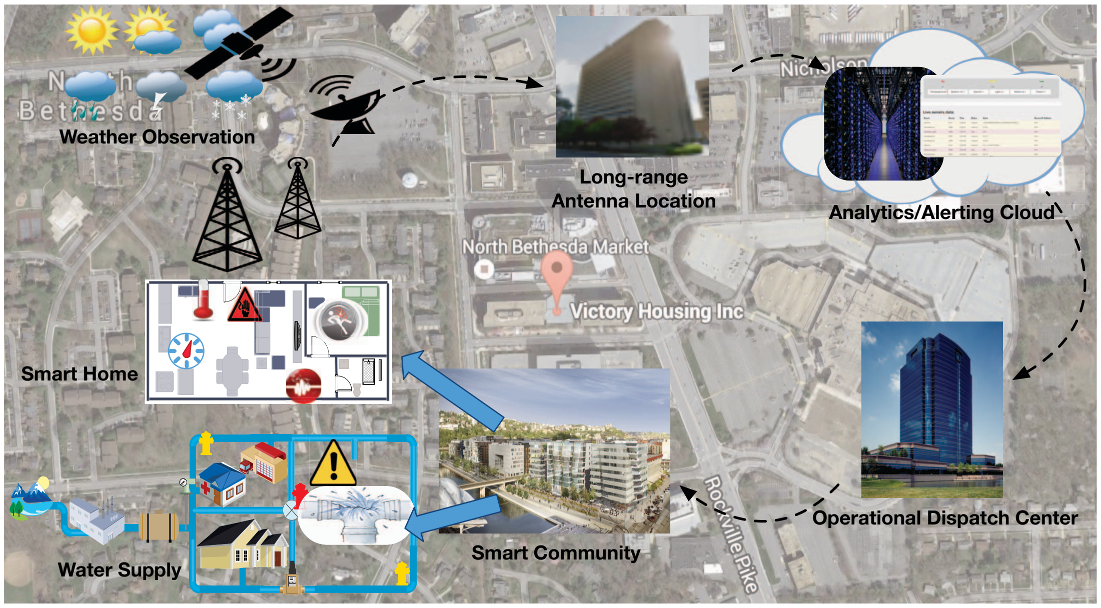

Fig. 1. A general IoT community example with emphasis on water infrastructure. As an example, the cloud constantly collects data from instrumented water meters and seismic sensors via basestations and the regional internet. Based on the analysis of results, a first responder might be dispatched for rescuing.

In this paper, we focus on a specific resilience concern - that of resilient communications, in particular during large-scale geographically-correlated failures, e.g. due to a high-magnitude earthquake. Resilience techniques to deal with limitations/disruptions in community IoT networking infrastructures can be addressed at two different levels - a lower connectivity level (how to deliver reliably) and a higher messaging level (what to deliver for improved utility). We focus on the former, i.e. reliable delivery, by exploiting redundant paths to ensure commnunication of critical safety data to the backend cloud where it is stored and analyzed. For this, we extend the the notion of resilient overlay networks (RON), originally designed in the context of wired networks to IoT deployments. RONs, originally proposed in [6], have been studied extensively for routing around Internet failures and congestion. To capture the geographically correlated nature of the deployed infrastructure and associated damage regions in disasters, we explore the use of *Geographically-Correlated Resilient Overlay Network (GeoCRON)* [7] and design a middleware that extends the RON concept with awareness of the geographic placement of nodes in the underlay. It uses this information, as well as knowledge about the underlying routing infrastructure, to choose multiple geo-diverse routes in order to improve the chances of a message reaching the destination during large-scale geo-correlated failures.

To contextualize our work further, we focus on deployments and use cases surrounding massive network failures affecting two different community IoT systems within the same geography. The first, a large **participatory sensing-style seismic network** of small inexpensive seismic sensors modeled after CSN [3] (a Community Seismic Network in Pasadena, CA), reports ground motion in homes and businesses to a cloud service for detecting and characterizing earthquakes. The second, a smaller infrastructure-based network of wireless devices embedded in a **smart water distribution network**, reports water pressure changes at pipe junctions that may be indicative of leaks that require human intervention. We consider a realistic water distribution network provided by EPANet, a simulation framework from the Environmental Protection Agency (EPA). In this network, we place the water sensors at pipe junctions, wireless basestations to cover this whole network, and then construct a realistic network topology, using IGen [20], based on population density derived from the demands on this water network. Corresponding to the two deployment scenarios, we consider a resilient overlay network (RON) comprised of two types of nodes, i.e. seismic sensing nodes and the wireless basestations that receive the water sensor readings. We develop cost-effective mechanisms to support fast and reliable data transfer over the multiple physical network connectivities, taking into account the shared nature of access and dynamic workloads in a community setting. We study the similarities, differences, and interplay between these two community IoT systems in such a situation in order to identify common patterns that improve communications resilience in both systems as well as strategies for enabling them to collaborate effectively. Some of these differences include timing constraints on the data delivery, last-hop connectivity, and physical placement of network nodes.

We use simulations to study geo-correlated failures on this combined network infrastructure and the use of our proposed GeoCRON middleware to route messages along geo-diverse paths in order to improve the chances of data delivery. GeoCRON measures path geo-diversity through various formulations, and so we apply these various measures to create a family of heuristics for choosing geo-diverse paths and compare their performance with each other.

Key contributions of this paper include:

- Development of a methodology for inferring IoT communication networks structure for multiple community infrastructures in a geography (seismic/water in this case) towards the creation of a common GeoCRON topology (Sec 2)

- Design of the state-of-the-art algorithms to exploit path geo-diversity for the construction, maintenance and effective use of GeoCRON overlays in a realistic multi-network geo-correlated failure scenario (Sec 3)
- Validation of the diverse GeoCRON techniques with appropriate geo-diversity metrics using extensive simulations and analysis of improved resilience to communications network failures, gained by sharing network resources between IoT deployments using an overlay middleware (Sec 4)
- A Prototype implementation of the GeoCRON reliable communication methodology in the SCALE platform to understand operational and deployment challenges in a real world community setting (Sec 5).

II. IoT Communications Resilience

In this section, we begin discussion of resilient IoT communications in the Internet. We present two IoT networks as concrete examples and propose a strategy for designing a realistic resilient communications network to support them: a volunteer seismic sensor network and a water distribution sensor network.

A. *Improving Network Resilience*

The original Internet architecture was designed as a scalable high-performance resilient network for delivering data. Throughout its evolution, researchers and engineers applied a variety of techniques to improve its resilience. Some of these techniques generalize and reappear at various points in different networking stacks. For example, the concept of redundancy is applied in terms of transmitting over multiple wireless channels to avoid interference, maintaining redundant networking equipment (e.g. routers, wireless basestation coverage) in case of failure, aggregating multiple physical links in a wired topology in case one fails, and even retransmitting unacknowledged packets when using TCP. Another application of redundancy that we will exploit in this paper is maintaining multiple network paths between two nodes. This may take the form of recomputing alternative routes in routing infrastructure given knowledge of the complete topology (e.g. OSPF), maintaining different flows and either intelligently choosing between them with a higher level logic or copying data over both (e.g. MPLS, TCP multi-homing), or maintaining loops for fast re-routing of traffic in response to failures (e.g. SONET). These techniques, while capable of quickly and effectively handling transient network failures such as packet loss due to congestion or faulty networking equipment, break down during massive failure scenarios.

In the face of failures and congestion, the underlying routing infrastructure may take several minutes to identify alternative routes due to reconvergence of the routing protocols. Studies have identified issues with Internet routing, in particular with the Border Gateway Protocol (BGP), during massive failure scenarios. For example, [27] found paths that needlessly passed through other continents following a major earthquake in Taiwan. It also claimed that BGP policies negatively impact resilience on the Internet by not allowing certain paths. The authors of [10] found that most visible failures in the Internet did not exceed 5-15 minutes while the authors of [16] found that BGP route update convergence could take up to 15 minutes after a fault. Hence, we see that we cannot always rely on network routing infrastructure to address all failure scenarios. Part of the reason for this is the simplification of intelligence in the core of the network to improve performance and scalability.

A common design paradigm for the Internet architecture as well as protocols/applications that rely on it is that of pushing intelligence as close to the edge as possible. Not only does this allow the network core to focus specifically on high-speed packet routing by limiting the amount of logic those devices contain, it also allows for a hierarchical design methodology in terms of deploying heterogeneous devices to handle different tasks within a network. For example, customer devices run TCP in order to ensure reliable delivery of data (if possible) through the best-effort Internet service. Gateway routers at the edges of networks handle these customer devices' traffic and may send packets through different flows in the network to offer a particular quality-of-service or shape certain traffic types (e.g. p2p applications). In enterprise networks, middleboxes offer services such as firewalls or caches that would overly complicate finely-engineered routers. This hierarchical design also allows these networks to scale in an efficient and cost-effective manner. Routers at the edges of the network aggregate data flows from many customer devices and forward them along a fewer number of links into the core of the network. The core routers, which are much fewer in number, will then forward this traffic along high-speed long-haul links to other core routers and back up this hierarchy to the destination customer devices. In this manner, the more complex logic to handle applications and new network services resides in customer devices. With the advent of IoT systems, designers often disrupt this paradigm by further extending the hierarchy as IoT devices are less capable of supporting more complex logic. They often deploy gateway nodes or basestations to collect traffic from sensor devices (or send commands to devices) and determine how to handle it (e.g. send to cloud or process immediately). These nodes offer opportune points at which to add software intelligence for complementing the Internet's resilience mechanisms in order to address the aforementioned problems, a concept we will return to in Section III-B.

B. *Resilient IoT Network Design*

Here we describe our two motivating IoT deployments and a process for how we would construct a realistic resilient multi-technology communications network to serve them.

1) Seismic Sensing Network: We model our seismic network use case after the Community Seismic Network (CSN) [3] system deployed in California. CSN uses low-cost accelerometers deployed in homes, businesses, and schools. Volunteers provide space and power for these sensors, which may be connected to a desktop computer or a dedicated low-power computer, such as a plug computer or Raspberry Pi. We envision such a deployment expanding in the future as more individuals deploy IoT sensor devices in their homes that often incorporate accelerometers (e.g. home security, smartphones), which could contribute shaking measurements to the CSN system. These sensor devices monitor background shaking for anomalies and report data about them when detected in messages called *picks* to a cloud service (e.g. Google AppEngine for CSN, IBM BlueMix for SCALE), which sits outside the earthquake-prone region and analyzes recent *picks* possible earthquakes. By effectively identifying and categorizing earthquakes in a timely manner, the CSN system, in conjunction with traditional seismographs, could lead to a real-time fine-grained targeted early warning system and provide people precious seconds to take shelter before a seismic wave propagates to their location. To accomplish this despite severe congestion and packet loss due to sudden traffic spikes from people contacting each other as well as failures caused by the tremor, the *picks* must arrive at the server for analysis within a few seconds of the event. Such failures will often be localized due to the geographic scope of an earthquake as well as cascading failures introduced by e.g. failures in the power grid. This motivates the failure model proposed in Section IV-B that we use in this study.

2) Water Sensing Network: We model the water sensor network based on the realistic water network provided by EPANet [22], [26]. It consists of 118 pipelines, 96 junctions (i.e. pipe joints), a pump, a valve, a storage tank, and 2 reservoirs located throughout a 9.26 km × 7.77 km geographical region. Each junction has its own level of demand (i.e. consumption), and each pipeline has different properties including length, diameter, roughness coefficient, and status (i.e. open or closed). To mimic a real-world setting, we place approximately 50 sensors in this network at sporadic junctions since these interconnection points are more prone to failures [2]. The sensors periodically send pressure/flow rate values to the cloud service for leak detection. Because modern water networks typically do not have such sensor instrumentation currently built into them, we assume that these sensors are mostly retrofitted into the infrastructure. Therefore, we assume that sensor data collection will be done wirelessly. Because these sensor devices will likely be low-powered and frequently battery-operated, we model the wireless network after a low-power long-range technology such as Sigfox's ultra-narrowband, which we used extensively in the SCALE project. Multiple wireless base stations (BSs) would cover the water network (see Section II-B3) and forward sensor data to the cloud.

3) Topology Construction: To construct two realistic network topologies (one less redundant and one more redundant) for this study, we started with the sensorized water distribution network described in Section II-B2 as our target geography from which to build communications and seismic networks alongside. By applying real-world network design techniques inspired by the resilience considerations outlined in Section II-A and proposed in [20] , we performed the following steps to generate the topologies:

1) Use the water demand values at each junction to derive an estimate of population density around that point. Normalize these demands across the total demand of the whole network to get the density.
2) Use these densities to randomly place 225 nodes representing customer locations (businesses, residential

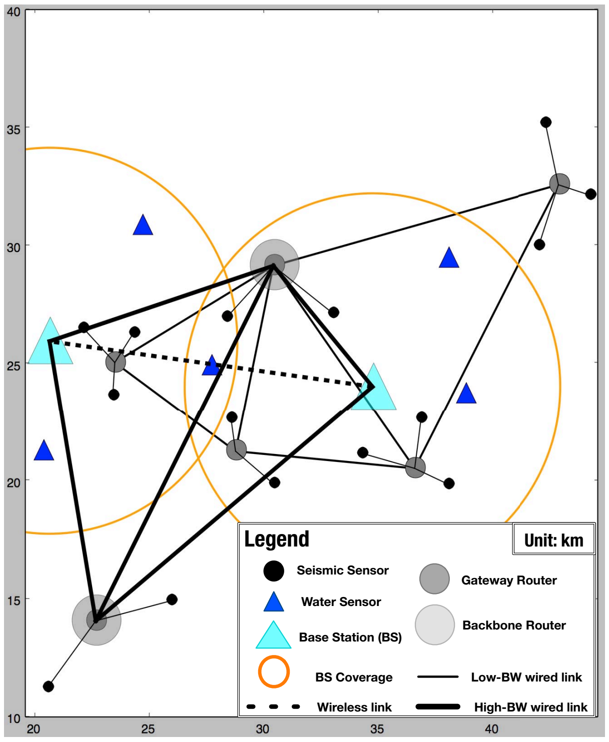

Fig. 2. Partial network topology as an example to show different node and link types. Other topology plots will follow this legend and exclude the legend for clarity.

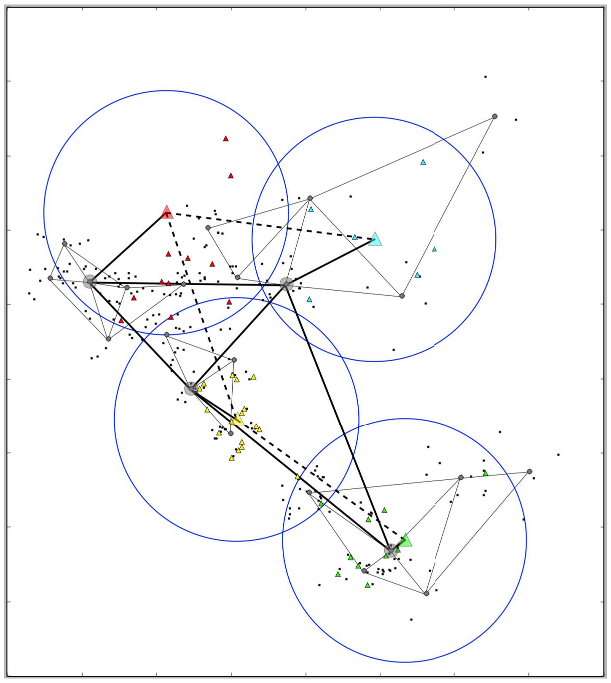

Fig. 3. The less redundant of our two network topologies.

neighborhoods, etc.), each of which have community seismic sensors running on their network.

3) Place 20 additional router nodes based on K-Means with respect to the seismic nodes' locations.
4) Use the techniques adopted by the network topology generator IGen [20], we build a full network topology interconnecting the routers and splitting them into gateway routers and backbone ones. 4 backbone routers are chosen from all the router nodes using K-Medoids. The backbone routers are linked in a full mesh topology, whereas the gateway routers are linked into a sparse mesh using a Delaunay Triangulation. This is an efficient way of obtaining a cost-effective topology with redundancy. It produces alternate paths between nodes, while minimizing the number of such paths [20]. We also removed a few redundant long-haul connections between backbone routers as this would reduce network construction costs.
5) Augment the resulting network to have some long-haul connections outside of the city through a few different paths to represent connections to the cloud service where the data should all be uploaded.
6) Connect each of the customer nodes to the closest gateway routers.
7) Place long-range wireless basestations (BSs). We deploy 4 in the less redundant topology and 7 in the other using K-Means [13] to place each BS at the center of sensor clusters and then move them slightly to ensure full coverage (2km range) over the water sensors. The basestations, which we modeled after the Sigfox ultra-narrowband network we worked with during the SCALE project, are each linked to the closest backbone router via a wired link as well as to the closest basestation via a wireless link. The wireless links between basestations represent microwave links used to link radio towers, especially useful during emergencies when wired connectivity may be affected. We modeled this after Montgomery County, which deploys microwave technology on many its government buildings and radio towers to ensure continuity of operations even during large outages.
8) Placed the cloud data center (server node) far outside target geography because we assume nodes will always try to send sensor data outside of the affected region. This mimics the CSN system, which runs its cloud service outside of California to lessen the possibility of data being trapped within a failed region. The server is connected to 3 of the backbone routers so that there exist redundant options for the overlay to utilize if the default path fails. While we only place a single server node in this topology, it actually represents a connection to the cloud. That is, it represents many servers in different locations, but for the purposes of studying connectivity to any of them we only need a single node.

III. Resilient Overlays for IoT

This section introduces the concept of Resilient Overlay Networks (RONs) and our proposed Geographically-Correlated Resilient Overlay Network (GeoCRON) system. It details the architecture and algorithms used in GeoCRON for constructing and using the resilient overlay as well as choosing geo-diverse paths.

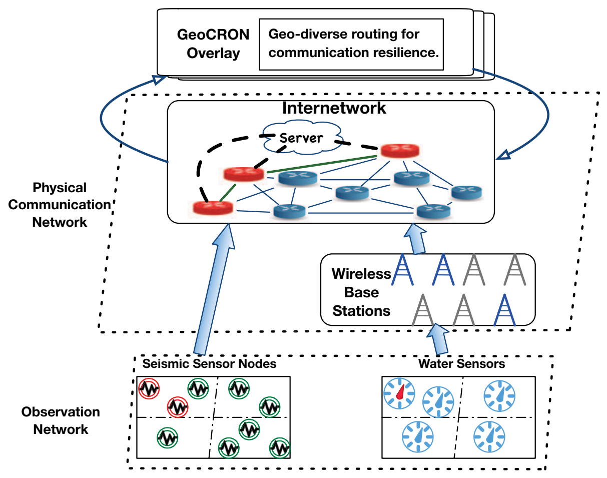

Fig. 4. An overview of the layered multi-system architecture being studied.

A. Resilient Overlay Networks

In this paper, we make use of *Resilient Overlay Networks (RONs)* to improve the resilience of our large scale IoT system. Previous research [6], [10] has shown that RONs can substantially improve delivery of messages, as well as latency, during failures, unavailability, and congestion. Indeed, [10] found that "overlay networks can typically route around 50% of failures." RONs accomplish this by routing messages along functioning alternative paths disjoint with the presently-unavailable or congested Internet-routed ones. Peers in the overlay facilitate this alternative routing by acting as intermediary hops, which tricks the underlying routing infrastructure into sending the packet first to this intermediary and then from the intermediary to the destination (in the two-hop case) as shown in Figure 5. When an end-host perceives a failure, or simply wishes to improve chances of delivery by utilizing an alternate path, it chooses such an intermediary overlay peer according to some metrics (e.g. latency, current load) and requests that it route traffic to the destination.

B. GeoCRON: Exploiting Network Edge Intelligence for Failure Recovery

As discussed in Section II-A, networked systems commonly follow the paradigm of pushing intelligence towards the edges of the network to streamline the core and allow heterogeneous logic on these edge devices. Because many IoT devices, including our model seismic sensing nodes, feature general CPUs, we can easily add software intelligence to enable middlewares for sharing resources between IoT devices. This allows us to share networking capabilities and implement the aforementioned RON approach to significantly enhance communications resilience in IoT systems. Thus, we exploit the existence of a multitude of end-host devices without requiring Internet Service Providers to offer additional services in their networks. Our proposed IoT deployments allow us to add this logic by running the GeoCRON overlay on the seismic nodes, which run on commodity computers, and water sensor basestations, which are assumed to be running high-performance hardware.

When a GeoCRON node tries to send sensor data to the cloud server(s), it tries to maximize the delivery rate of the data by sending multiple copies along disjoint paths to each known server. In our previous work [7], we explored sending one message and awaiting a timeout before trying a different path, up to a predetermined number of retries. However, we found that this frequently required a full 5-10 seconds to deliver the majority of the messages, and so, given the low-latency requirements of our seismic sensing scenario, we instead opted to send all message attempts at the same time to decrease latency. We note that other IoT systems, such as the water sensing network, may opt to relax this constraint and use a hybrid of these two techniques, but we save this study for future work. We define the *multi-path fanout k* as the number of message copies sent to each server. This value is configurable within the client code and is experimentally determined to maximize data delivery without introducing too much congestion into the network. The first of these packets is sent directly to the server without any overlay hops to minimize latency. The remaining $k-1$ are sent using geo-diverse overlay paths chosen from the available peers as described in Section III-D. In [11], the authors discovered that the vast majority of node pairs only require a single overlay hop in order to exhibit the same diversity as multiple hops. Therefore, we use as each possible path a 2-hop overlay path where the first hop is some overlay peer and the second is the destination server. The list of overlay peers from which to choose is based on the overlay construction described below.

C. Overlay Construction

In order to pick an overlay peer to help create a geo-diverse path to the destination, each GeoCRON client must know about other clients in the network. All GeoCRON nodes must at least know the locations and IP addresses of these peer clients; specific heuristics will request additional information (e.g. physical route to the other node) in order to construct the overlay. As IoT networks can scale to a large number of nodes, it is clearly impractical for each client, which typically runs on a low-powered embedded computer, to maintain this information for every other client in the overlay. Therefore, we must restrict the number of other peers known by each client to a subset of the entire network.

Because GeoCRON nodes report data to a cloud service for analysis and storage, we consider GeoCRON a *hybrid peer-to-peer network*. The cloud service instances maintain knowledge of the full network and associated metadata, including underlying route information, to simplify bootstrapping new nodes in the network. When a GeoCRON node comes online, it contacts one of the servers and retrieves a list of geo-diverse peers according to the metrics described in Section III-D. These peers use tools such as traceroute to gather information about the paths between them and both the server(s) and other nodes, including physical routers and link latencies. This data is uploaded to the server(s), where the path geo-diversity algorithms are run to select the best overlay path choices for each node.

The *maintenance* of the overlay would be a crucial consideration if our system involved more *node churn* (e.g. mobile nodes). However, the in situ placement of the sensors means the overlay topology and node locations are expected to remain essentially static except due to failures, in which case nodes are not expected to come back online quickly. Our current architecture thus does not concern itself with overlay maintenance, but we do intend to incorporate mobile nodes and study it in the future. It is well-known that maintaining an peer overlay where each node in the overlay knows about $O(\log(n))$ (n being the number of nodes) other nodes will allow the network to scale to practical sizes. Similar to other overlay-based systems [15], [17], GeoCRON could bootstrap nodes with the aforementioned centralized approach and then allow nodes to gossip with known peers to refine these initial choices according to some metric. For example, [17] used an simulated annealing-like approach in which connections are chosen to decrease the distance between peers while assuring a low probability of disconnect during random failures by keeping an average node degree. To address more sophisticated and realistic route choice strategies, we are exploring assigning peers based on geo-spatial metrics so that clients are aware of both nearby peers as well as those in diverse areas. We are currently considering three different approaches:

- Assigning peers with a probability inversely proportional to their distance from the client node
- Choosing peers to be as uniformly distributed spatially as possible by breaking the area under consideration into a grid and picking some number of peers from each cell
- Choosing peers based on clusters (a structured approach) by breaking the area into a hierarchical grid and choosing a predetermined number of peers from each cluster

These approaches would guarantee a client knowing about peers that are both nearby and distant. This would allow for a client to contact another peer far away in order to request information about additional peers in this distant region, similar to the method used by Pastry [23] for contacting far away peers based on some key, which in this case would be location.

Furthermore, by introducing such gossip mechanisms to GeoCRON in addition to overlay routing, nodes can further coordinate with each other during a disaster to learn about new peers and paths, especially if contact with the server(s) has been disrupted. In this manner, they may share information about perceived failures and known good paths to further improve adaptation to dynamic events. This can be further improved if multiple devices are located within a local area network, especially if equipped with wireless technologies. Note also that such a mechanism can be exploited to implement rich techniques for compression, local event detection and content aggregation, which is a topic for further work.

D. Geo-diverse Route Selection

Resilience to Internet failures has been extensively studied, though few works address massive geo-correlated failures.

Large-scale failures tend to be geographically-correlated in nature, whether due to a particularly impactful natural phenomenon (e.g. earthquake, tornado), a cascading failure from another network (e.g. power grid), or perhaps a targeted attack by human adversaries (e.g. electro-magnetic pulse weapon). Some research attempts to formally model these failures and extrapolate design methodologies for improving network infrastructure reliability from them. Straight line segments drawn through a network topology, failing any intersected links, were used in [18] to study geo-correlated failures. In [12], the most damaging link cuts possible for a given network provider is used to plan a more resilient network. Further general network resilience challenges are discussed in [24] and [25]. Geo-diverse multipath routing within an autonomous system (AS) is studied in [21], and we borrow and expand on their geo-diversity metrics in this work.

1) Model and Notation: This section introduces notation that we use to model and reason about GeoCRON formally.

Network Model Let $G = (V, E)$ be the graph defining the network under consideration, where V is the set of nodes representing routers and end hosts and E is the set of undirected edges representing physical links between nodes. Each edge $e \in E$ is assigned a weight $w \in \mathbb{N}$ to represent the latency (as measured in milliseconds) of the links. For the purposes of this study, we assume the latency of a link is constant (other than queueing delays) and bidirectional.

Node Locations Each node $v \in V$ is assigned a physical location as measured in geographic coordinates. Let $loc : v_i \rightarrow (x_i, y_i)$, for $x_i, y_i \in \mathbb{R}$, be the function mapping each node v_i to its physical coordinates and $dist : (v_0, v_1) \rightarrow \mathbb{R}$ be the function mapping each node pair to the physical distance between their locations.

Overlay and Server Nodes Let $S \subset V$ be the servers (sinks) to which each sensor node reports data to.

Network Paths When a sensor node sends a message to some server $s \in S$ the packet will travel a particular path through the network as determined by the underlying infrastructure. Let $p = (v_0, e_0, v_1, ..., e_{n-1}, v_n)$, for $v_i \in V, e_i \in E$, be a sequence of nodes and interconnecting edges that represent such a path. When a sensor node chooses to send such a message using overlay nodes as intermediaries, we may consider only the sequence of overlay peers rather than the entire physical path. As such, let $h = (o_0, ..., o_n)$, for $o_i \in O$, be the sequence of overlay peer hops taken by such a message, where $o_n \in S$ is the final destination server. Currently, we only consider 2-hop overlay paths in which the second hop is the destination server s_0. That is, $h = (o_0, ..., s_0)$. Let also $P = \{p_0, p_1, ...\}$ be the set of all possible paths in G and $path : h \rightarrow p \in P$ map overlay paths to physical topology paths. This is done by joining together the physical paths from the source peer o_0 to the first hop o_1 with with the path from the first hop o_1 to the second hop o_2 and so on.

Modeling Path Diversity In order to assess the diversity of a potential path choice, we must define some model for quantifying it. We define diversity as a measure of how different two paths are in terms of shared components, proximity of component locations, or even both. The goal is to identify paths that are less likely to suffer geo-correlated disruptions

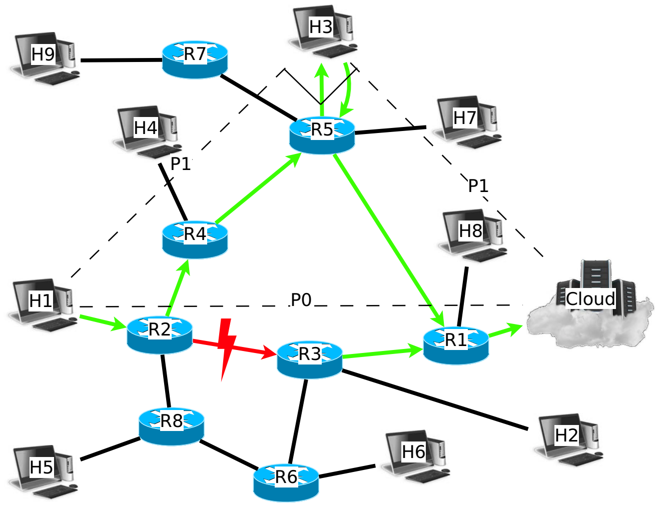

Fig. 5. A GeoCRON node chooses an overlay based on geographic information.

at the same time. Let $D(p_a, p_b) : \{p_a, p_b\} \rightarrow \mathbb{R}$ be the abstract diversity function for comparing two physical paths. Let $D(h_a, h_b) : \{h_a, h_b\} \rightarrow \mathbb{R}$ be the abstract diversity function for measuring the diversity between two overlay paths. These function templates are used to measure the diversity according to one of the concrete heuristics defined in Section III-D2.

2) Geo-diverse Path Heuristics: In this section, we propose a family of techniques and heuristics used to rank the geo-diversity of various overlay path options. When a GeoCRON node o_1 has data to send to the server, it first sends the data directly without use of the overlay as described in Section III-B. For the sake of discussion, let us just consider the case of a single server s acting as the destination for this data. Let $p_0 = (o_1, e_1, v_2, \ldots, s)$ be this non-overlay direct path to the destination s. If configured with $k > 1$, o_1 will also send out additional packets along geo-diverse overlay routes $P^* = (p_1, \ldots, p_{k-1})$ to s. Clearly it is impractical to assume that o_1 will have knowledge of the failures along overlay path $p_1 = (o_1, e_1, v_2, \ldots, o_i, \ldots s)$ before either attempting the path or receiving some (possibly out-of-band) communications regarding the failures. Therefore, GeoCRON nodes must make the best local decision possible regarding which geo-diverse paths to use.

In our previous work [7], we tested various heuristics that chose paths $(p_1, \ldots, p_{k-1})$ using only the locations of the overlay nodes. We found little benefit from these heuristics over the baseline (random choice of peer), and so in this paper we consider knowledge of the underlying physical network path p_1 and even the locations of the routers therein. Therefore, when determining p_i's *diversity*, o_1 may consider information such as the components of the path $v, e \in p_1$, the location of each router $v \in p_i$, and the location of the overlay peer choice $o_1 \in O$. By exploiting this knowledge, these heuristics aim to pick paths that are more diverse (less correlated) and therefore improve the resilience of the system to failures.

Below we discuss the implementation of various heuristics (other than *Random* and *Ideal* since they are straightforward) used to choose geo-diverse overlay paths. They have varying degrees of awareness regarding existence and locations of network infrastructure, including the end devices. In order of decreasing topology and location knowledge, these are:

- *Ideal* (oracle heuristic that finds any working path),
- *Gsford* (router proximity-aware),
- *AreaDistance* (router minimum distance and path area-aware),
- *Intersection* (shared router and link-aware),
- *Random* (uniformly random path choice).

Gsford: This heuristic is derived from some our previous work [14], [15]. In [14], we considered lessening the impact of geographically-correlated failures by picking geo-diverse paths considering whether their routers were within some threshold distance (T_{dist}) of one another. As detailed in Algorithm III-D2, the very first overlay peer chosen is the one with the lowest latency. All subsequent path choices are compared with all other currently chosen paths and penalized by one point for each router on those paths within T_{dist} of a router on this path. The diversity value is the inverse of this penalty, and so the chosen path will be the one with the lowest penalty score. This technique was applied to a geo-social notification system (GSFord) in [15], where the heuristic gets its name from.

Algorithm 1 Path diversity scoring function used in the GSFord system [14], [15]. Paths are penalized for having routers within a threshold distance of a router on another path.

```
Require: $T_{dist} \leftarrow$ distance threshold for path components
 1: function GsfordGetDiversityScore($h$)
 2:     $H \leftarrow$ GetCurrentMultiPath
 3:     if $H$.size() $= 1$ then
 4:         return $1/$GetLatency($h$)
 5:     proximity $\leftarrow 0$
 6:     $p_0 \leftarrow$ path($h$)
 7:     for $h' \in H$ do
 8:         $p_1 \leftarrow$ path($h'$)
 9:         for router $v_1 \in p_0$ do
10:             for router $v_2 \in p_1$ do
11:                 if dist($v_1$, $v_2$) $< T_{dist}$ then
12:                     proximity $\leftarrow$ proximity $+1$
13:     diversity $\leftarrow 1/$(proximity$+1$)        $\triangleright$ avoid divide by 0
14:     return diversity
```

AreaDistance: This heuristic is based on a geodiversity metric proposed in [21]. The goal of this heuristic is to choose physical paths that are as far away from each other as possible. They define the geodiversity of two paths according to:

$$D_g(P_b, P_a) = \alpha d_{min}^2 + \beta A \tag{1}$$

Where $\alpha, \beta \in [0, 1]$ are configurable weight parameters, d_{min}^2 is the square of the minimum distance between any two routers in P_b, P_a, and A is the area of the polygon bounded by the locations of all components in P_b and P_a. *AreaDistance* is called to evaluate the geodiversity of each currently chosen path and the current possible path choice. For each path, its aggregate geodiversity is chosen as the minimum of those when compared with all of the other paths. The justification for this method is that although a path may be very diverse from another, it could also be extremely correlated with another yet, and so it is geodiverse only in so far as it is geodiverse from *all* currently chosen paths. Ties are broken by taking the choice with the lower path length (number of routers and links).

Path Intersection: This heuristic is also based on a diversity metric proposed in [21]. However, it is not truly a geodiversity metric as it does not consider the physical locations of the topology components. Rather, it simply considers whether or not the two paths share the same components. It measures the size of the paths' intersection, hence its name, and measures the diversity of two paths according to the following metric:

$$D(P_b, P_a) = 1 - \frac{| P_b \cap P_a |}{|P_a|} \tag{2}$$

Where $|P_a| \leq |P_b|$. Note that our implementation takes $min(|P_a|, |P_b|)$ as the factor in the denominator to ensure $|P_a| \leq |P_b|$. Note also that we must ensure both paths have at least one component each in order to avoid dividing by 0. Just like *AreaDistance*, *Intersection* computes the diversity between the path under consideration and each currently chosen path, assigning the aggregate diversity for this path as the minimum of all these. It also breaks ties by choosing the path of shorter length.

IV. Experimental Evaluation

Due to practical considerations with respect to deploying an IoT network within critical infrastructure and testing its performance in an earthquake, we implemented and studied our system in a simulation-based environment. We opted to use the ns-3 [4] network simulator because we wanted to be able to modify the source code to fit our scenario. Below we describe our simulation design and implementation, the code for which is freely available for others to download, use, and modify [1]. We then present the experiments we ran with this simulation and discuss the results.

A. Simulation Design

To support collecting physical path information, we extended ns-3's NixVectorRouting model, which is used for more computationally efficient routing in large networks. This new function returns the physical path that will be used to reach a destination (identified by its IP address).

To read our network topology and sensor locations (see Section II-B3), we created a new TopologyReader model that extends the InetTopologyReader. This new model reads in all of the node and link information the same as for Inet, but it stores the different types of nodes in different NodeContainers so that we can properly configure each group separately. It also sets the locations of nodes for use in the failure model.

We implemented a GeoCRON Application in ns-3 that attempts to upload sensor data to the server(s) via multiple geo-diverse routes at a specific time. Each Application adds a slight random delay (uniformly random within a range of 1.5 seconds) to this send time when sending packets via the overlay so as to avoid high packet loss rates that we

[1]The interested reader can find all of our ns-3 code on the *geocron* branch at https://github.com/KyleBenson/ns3

encountered initially due to too many nodes sending messages at the exact same instant. We chose a small delay and to send all multi-path messages at once because the seismic scenario requires low latency. Furthermore, each node would be sending messages at a slightly different time in a real scenario due to e.g. detecting the seismic wave at different times. Overlay forwarding was handled by adding a new header that specifies the IP address of the peer hops, which is used by overlay peers to determine where to forward a packet to. When a server receives a message, it logs this fact in a trace file and responds with an ACK through the same overlay path.

The parameters to consider for an experiment are all specified by command line configuration. These include: disaster location, base failure probability p(fail) , number of geo-diverse paths to attempt, which heuristics and their (optional) model parameters, and the number of times to run each unique configuration so as to average results over many different applications of the failure model, random heuristic choices, etc. The simulator iterates over all combinations of specified parameters applying them in a particular order so as to ensure a consistent comparison, with respect to one parameter e.g. application of failure model, between configurations. This lessens the amount of variance we see between treatment groups and gives a more accurate estimation of the underlying distribution(s) that determine the results.

B. Failure Model

The failure model assumes an earthquake occurs immediately before each client reports sensor data and that all non-server nodes and all links within the region under study are failed with a particular probability (p(fail)). We assume the server (cloud data center) resides far enough away from the region under study so as to remain unaffected by the disaster. The motivation for this assumption comes from the CSN deployment, which specifies its server instance to run outside of California in order to lessen the chance of data being trapped within the area affected by the earthquake. Non-server nodes and links fail with a probability inversely proportional to the network component's distance from the earthquake's epicenter according to the following equation:

$$\frac{p(fail)}{2^{D*S/B}} \tag{3}$$

Where p(fail) is the base failure rate input into the simulation, D is the distance from the component to the earthquake epicenter, S is a factor that determines how quickly $p(fail)$ decreases with distance D, and B is the length of the square boundary representing the entire region under study.

All failures happen at the same time (before the Applications attempt data upload), although we are exploring the use of a more sophisticated model that would allow for dynamic evolving failures to represent e.g. propagating seismic waves, secondary failures, aftershocks, and other disasters such as tornadoes, floods, etc.

C. Experimental Results

This section describes the results of our simulation experiments. We ran each unique configuration of simulation parameters 24 times. For each unique parameter configuration, we averaged the results of all the runs to lessen the effects of edge cases and better compare the experimental groups with each other.

To quantitatively compare each experimental group, we use the delivery rate of the individual nodes' original sensor data message. That is, a message counts as delivered if at least one copy of it reaches at least one server. This message count is normalized by the number of *active* (non-failed) nodes so that this delivery rate falls in the range $[0, 1]$. The plots below show this delivery rate as a function of time. Note that the starting point of the curves in these plots represents the performance without the overlay. This is because the messages sent directly to the server go out first and the overlay messages are sent after a short time delay. Note that in the legends each experimental group is labeled with the name of the heuristic and the following parameters in brackets: f - failure probability; k - multipath fanout; D - distance threshold for *Gsford* heuristic.

1) Comparison of Heuristics: Before comparing the various heuristics with each other, we had to identify a somewhat narrow set of parameters to run them with in order to avoid the combinatorial explosion of exploring every possible configuration, which would make each simulation run prohibitively long. We settled on a p(fail) of 0.1, though we also experimented with 0.2, 0.3, and 0.5 (see below). For the multipath fanout, we used $k = 5$ (though we also used $k = 3$, $k = 9$, and $k = 17$ as described below) based on the findings in [21] that some topologies showed strong increases in diversity for $k < 4$ whereas others showed strong increases for $k < 7$.

For *Gsford's* T_{dist} parameter, we ran experiments on the values $T_{dist} \in \{20, 30, 40, 50, 75100, 1000, 2500\}$. The *Gsford* heuristic performs better for smaller T_{dist} values, which intuitively makes sense as setting this value to 1 would essentially turn it into an approximation of the *Path Intersection* heuristic, which we show below performs the best. We settled on $T_{dist} = 30$ as performing generally well both in this current experimental setup as well as in some previous studies using different randomly generated topologies.

We also ran experiments on different disaster locations and on both the less redundant and more redundant topologies we created. The results described below hold across these different parameters as well.

Figure 6 shows the results comparing all the heuristics described in Section III-D2. Recall that the *Ideal* heuristic represents the upper bound on the delivery rate we can achieve with overlay routing. We see that *Gsford* and *Path Intersection* perform similarly, though the latter has a clear slight advantage. It is interesting to note that *Path Intersection* would be easier to implement in a real system as it does not need to know the physical locations of the routing components along a physical path. Because of the above two facts, we chose *Path Intersection* as our best non-optimal heuristic and use it in a few experiments described below. The *AreaDistance* heuristic does not appear to perform particularly well, especially as it is usually matched by the *Random* heuristic, which has the simplest implementation of all.

2) Comparing Other Parameters: We also ran experiments to explore how the multipath fanout and failure probability

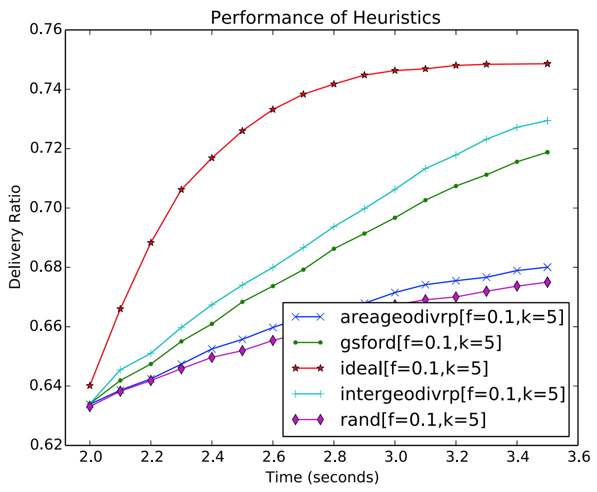

Fig. 6. The delivery rate for each of the heuristics.

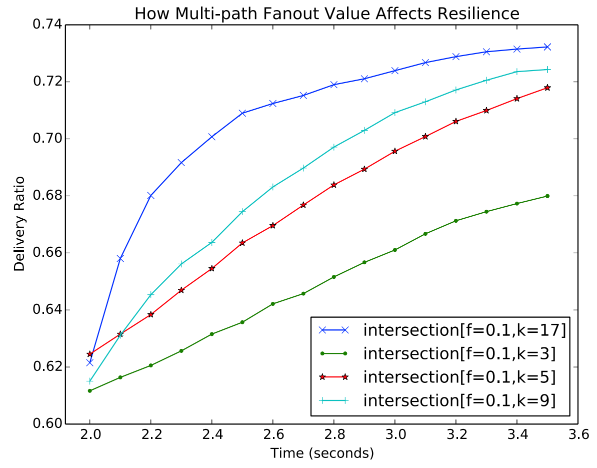

Fig. 7. The delivery rate for varying multipath fanout values.

affect the delivery rate.

Figure 7 shows the results for using $k \in \{3, 5, 9, 17\}$ with the *Path Intersection* heuristic. We see that using $k = 5$ achieves almost as high an improvement as with higher values, which appears to reproduce the results in [21]. When considering the detrimental effects of too many message copies flowing through an already-challenged network, we believe using a smaller value for k to be ideal.

Figure 8 shows the upper bounds (using the *Ideal* heuristic) on delivery rates for 1 and 2 servers and $f \in \{0.1, 0.2, 0.3, 0.5\}$. This demonstrates how impactful a slight increase in p(fail) can be on the network's performance. We see that the curves tighten with higher p(fail) values, indicating that the performance improvement becomes less during more failures as fewer working paths are available. When the percentage of network components failing goes beyond 25% it appears as though the delivery rate expected drops below acceptable values, even with the use of RONs.

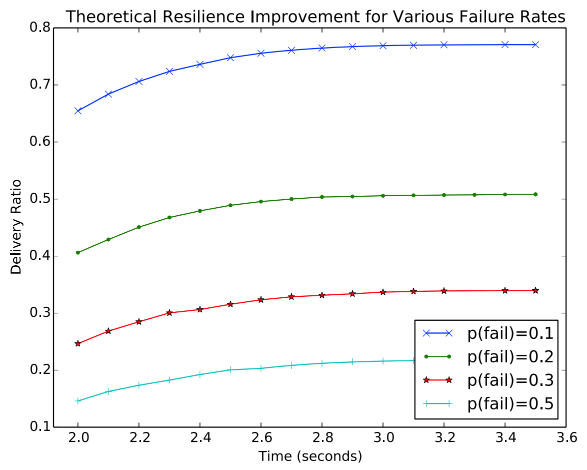

Fig. 8. The theoretically optimal delivery rate for even higher values of p(fail) .

3) Sharing Network Resources Between IoT Deployments: To study the possibility of both IoT deployments joining the same GeoCRON overlay and sharing network resources, we ran some experiments in which we applied three different treatments of which nodes participate in the overlay: only seismic, only water sensor basestations, and both types of nodes. The previously presented results above are from this last case where all IoT nodes actively participate. For this experiment, we isolated only the results of one network or another (i.e. only seismic nodes both in the case where they were the only ones participating and also when they could use the water sensor basestations as an overlay).

We found that combining the overlays did not actually have a significant impact, either negatively or positively, on the delivery ratio. We believe that this result is due to a combination of several factors. First, the density of the seismic sensor node deployment and the fact the overlay topology is fully connected in the simulations means that the seismic nodes are already capable of finding nearly all of the same paths using only seismic node peers and so the addition of using the basestations does not open up many further possibilities. Second, the structure of our network topology is such that the basestation nodes are already highly resilient and so do not benefit from using the seismic nodes as overlay peers. Because the basestations are connected to each other with long-range wireless links that do not fail when the failure model is applied, they can easily contact each other to route around failures. We actually noted that the use of GeoCRON resulted in achieving a 100% delivery ratio for most scenarios when p(fail) $= 0.1$. Therefore, we see that the application of GeoCRON does generalize across different types of IoT networks beyond the seismic sensing scenario. Because of these findings, we intend to repeat these experiments with different network topologies and scenarios in the future in order to determine whether these results generalize across further networks and IoT deployments.

V. Towards A Scalable Implementation

To begin testing GeoCRON in real-world settings, we created an initial prototype implementation. To leverage our existing deployments and infrastructure, we implemented it within our ongoing IoT effort: the Safe Community Awareness and Alerting Network (SCALE) [9].

A. *SCALE Overview*

SCALE (see Figure 9) aims to demonstrate the use of a middleware solution for IoT devices to sense the environment, locally analyze data for possible events of interest, upload events to a cloud data exchange, further analyze them for emergency event-detection in a cloud service, and alert residents and emergency dispatch using an Internet phone service. SCALE client devices are implemented using a Raspberry Pi in a box full of sensors (e.g. motion, seismic, gas) and associated networking equipment (Wi-Fi, Sigfox ultra-narrowband, etc.). The clients run an asynchronous Python-based middleware that manages interacting with the various sensors, local data processing, and reporting sensed events to the cloud data exchange through several different communications media and protocols (MQTT and HTTP). We implemented this middleware using abstract and concrete components loosely coupled around an internal publish-subscribe broker. In this manner, we can easily implement new modules that support different sensors, networks, and platforms, thereby improving the system's ability to handle heterogeneous devices.

B. *Extending SCALE with GeoCRON*

In our initial implementation, a SCALE client using the GeoCRON overlay feature will contact its SCALE server and request a list of geo-diverse peers. We implemented a location-sensing module to collect the geographic locations of SCALE nodes based on their IP address or a user-specified configuration. The SCALE client middleware contains a group of modules, referred to as *EventSinks*, that handle reporting sensed events to the data exchange using the appropriate networking technologies. The GeoCRON *EventSink* chooses a configurable number of overlay peers, packs necessary information (e.g. server IP address) into a Google protobuf [1], and transmits this header along with the sensed event to each chosen overlay peer over UDP. The overlay peers read the header info and forward the packet to the requested server.

To determine the overhead incurred by the overlay routing, we set up an experiment with SCALE devices. We configured a GeoCRON overlay on 6 Raspberry Pis to send sensor data (over Wi-Fi) directly to a laptop acting as the server as well as through each other. The server recorded the timestamps at which the different packets were received so we could determine the difference in latency from the direct message and the overlay message. We ran the experiment a number of times and varied the number of such messages sent (100 and 1000) and the number of hops in the overlay (1-5). The results (see Table V-B) indicate very low latency increases when using the overlay. The latency increase appears non-linear as we add more hops, but it has high variance, possibly due to effects of operating system scheduling. Future work will include setting up an experimental testbed for studying the resilience improvement of using our GeoCRON implementation in a real-world setting as well as repeating the above experiment with devices not on the same local area network.

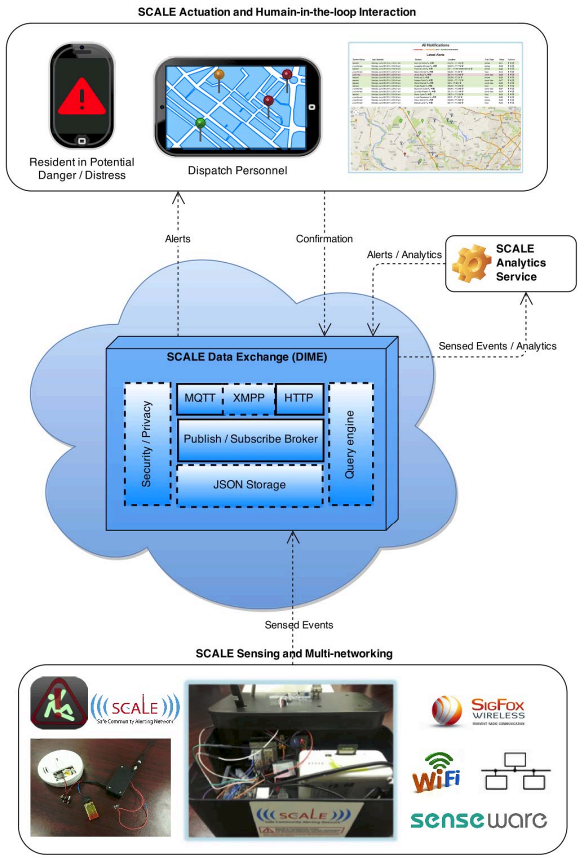
SCALE Actuation and Humain-in-the-loop Interaction
Resident in Potential Danger / Distress
Dispatch Personnel
Alerts
Confirmation
Alerts / Analytics
SCALE Analytics Service
Sensed Events / Analytics
SCALE Data Exchange (DIME)
Security / Privacy
MQTT
XMPP
HTTP
Publish / Subscribe Broker
JSON Storage
Query engine
Sensed Events
SCALE Sensing and Multi-networking
SCALE Safe Community Alerting Network
SigFox Wireless
WiFi
senseware

Fig. 9. The SCALE system.

# hops	min	max	mean	stdev
1	0.0002279	0.42008	0.02377	0.02418
2	0.0004408	0.35384	0.04049	0.03435
3	0.0004480	0.29621	0.05441	0.03706
4	0.0006402	0.34069	0.07287	0.04285
5	0.0004800	1.61702	0.12674	0.10012

TABLE I. Latency difference in seconds between overlay and direct packet for 100 messages.

VI. Conclusion and Future Work

In this paper, we discussed the concept of communications resilience in IoT deployments. To motivate our research, we discuss two IoT systems (seismic and water infrastructure sensing) and the design of a realistic network topology to support them. We proposed the use of *Geographically-Correlated Resilient Overlay Networks (GeoCRON)* for improving these systems' data delivery during a large-scale geo-correlated failure event (earthquake). This middleware is run on IoT systems where inexpensive devices deployed in communities communicate information with remote cloud platforms. We presented and evaluated (in simulations) several heuristics for choosing multiple geo-diverse overlay paths in IoT deployments with varying degrees of knowledge regarding the underlying network topology. This paper also discussed the design and implementation of an initial prototype system for GeoCRON in the context of the SCALE IoT platform.

As we continue exploring resilience issues in the SCALE system, we plan to address the following future work in the generic GeoCRON middleware:

- More realistically modeling correlated failures due to shared link bundles and cascading failures (e.g. power grid dependencies).
- Incorporating wireless ad-hoc networking so that physically hyper-close nodes can work closely on data delivery and event detection.
- Modeling and testing against dynamic failures in which failures happen over a period of time, rather than instantaneously, and also recover over time. The overlay heuristics and implementation should not negatively impact the network during recovery, and the peers should coordinate overlay maintenance so none become disconnected. Furthermore, overlay peers can exchange information about perceived failures in the network to improve this dynamic adaptation.
- Incorporating additional network technologies (e.g. cellular), infrastructures (e.g. transportation), and failure models (e.g. flood, fire).
- Studying different strategies for overlay usage according to the application requirements of different IoT systems (e.g. quality-of-service levels).
- Studying whether GeoCRON also generalizes to mobile nodes (e.g. smartphones).

Acknowledgment

This work was supported by National Science Foundation award nos. CNS 1143705, CNS 0958520 and CNS 1528995. The authors also thank Mani Chandy and Julian Bunn at Caltech for their discussions related to CSN and this work; Daniel Hoffman, Chief Innovation Officer of Montgomery County, MD, for organizing the SCALE project and team; Andrew Yang for his help with the initial GeoCRON prototype.

References

[1] Protocol Buffers - Google's data interchange format. https://github.com/google/protobuf.

[2] Community resilience planning guide for buildings and infrastructure systems. *National Institute of Standards and Technology*, 1, July 2015.

[3] Community Seismic Network. http://www.communityseismicnetwork.org/, May 2015.

[4] ns-3. http://www.nsnam.org/, May 2015.

[5] Sentilo. http://www.sentilo.io, Feb 2016.

[6] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient overlay networks. In *Proceedings of the eighteenth ACM symposium on Operating systems principles*, SOSP '01, pages 131–145, New York, NY, USA, 2001. ACM.

[7] K. E. Benson and N. Venkatasubramanian. Improving sensor data delivery during disaster scenarios with resilient overlay networks. In *2013 IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops 2013*, pages 547–552, 2013.

[8] E. Cochran, J. Lawrence, C. Christensen, and A. Chung. A novel strong-motion seismic network for community participation in earthquake monitoring. *Instrumentation Measurement Magazine, IEEE*, 12(6):8 – 15, Dec 2009.

[9] B. et al. Scale: Safe community awareness and alerting leveraging the internet of things. *Communications Magazine, IEEE*, 53(12):27–34, Dec 2015.

[10] N. Feamster, D. G. Andersen, H. Balakrishnan, and M. F. Kaashoek. Measuring the effects of internet path faults on reactive routing. *SIGMETRICS Perform. Eval. Rev.*, 31(1):126–137, Jun 2003.

[11] J. Han, D. Watson, and F. Jahanian. Topology aware overlay networks. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 4, pages 2554 – 2565 vol. 4, march 2005.

[12] A. F. Hansen, A. Kvalbein, T. Čičić, and S. Gjessing. Resilient routing layers for network disaster planning. In *Proceedings of the 4th international conference on Networking - Volume Part II*, ICN'05, pages 1097–1105. Springer-Verlag, 2005.

[13] J. Hartigan and M. Wong. Algorithm as 136: A k-means clustering algorithm. *Royal Statistical Society, Series C*, 28(1):100–108, 1979.

[14] K. Kim and N. Venkatasubramanian. Assessing the Impact of Geographically Correlated Failures on Overlay-Based Data Dissemination. In *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*, pages 1–5. IEEE, Dec. 2010.

[15] K. Kim, Y. Zhao, and N. Venkatasubramanian. GSFord: Towards a Reliable Geo-social Notification System. In *2012 IEEE 31st Symposium on Reliable Distributed Systems*, pages 267–272. IEEE, Oct. 2012.

[16] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian. Delayed internet routing convergence. *SIGCOMM Comput. Commun. Rev.*, 30(4):175–187, Aug 2000.

[17] L. Massoulie, A.-M. Kermarrec, and A. Ganesh. Network awareness and failure resilience in self-organizing overlay networks. In *Reliable Distributed Systems, 2003. Proceedings. 22nd International Symposium on*, pages 47 – 55, oct. 2003.

[18] S. Neumayer and E. Modiano. Network reliability with geographically correlated failures. In *INFOCOM, 2010 Proceedings IEEE*, pages 1 –9, march 2010.

[19] N. Proposal. Eager: Exploring resilience in smartcity water infrastructure. *Division of Computer and Network Systems*, June 2015.

[20] B. Quoitin, V. V. Schrieck, P. Francois, and O. Bonaventure. Igen: Generation of router-level internet topologies through network design heuristics. *in Proceedings of the 21st International Teletraffic Congress*, Sep. 2009.

[21] J. P. Rohrer, A. Jabbar, and J. P. G. Sterbenz. Path diversification for future internet end-to-end resilience and survivability. *Telecommunication Systems*, 56(1):49–67, Aug. 2014.

[22] L. A. Rossman. Epanet users manual. *United States Water Supply and Water Resources Division, National Risk Management Research Laboratory*, September 2000.

[23] A. I. T. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*, Middleware '01, pages 329–350, London, UK, UK, 2001. Springer-Verlag.

[24] J. Sterbenz, E. C andetinkaya, M. Hameed, A. Jabbar, and J. Rohrer. Modelling and analysis of network resilience. In *Communication Systems and Networks (COMSNETS), 2011 Third International Conference on*, pages 1 –10, jan. 2011.

[25] J. P. G. Sterbenz, D. Hutchison, E. K. Çetinkaya, A. Jabbar, J. P. Rohrer, M. Schöller, and P. Smith. Resilience and survivability in communication networks: Strategies, principles, and survey of disciplines. *Comput. Netw.*, 54(8):1245–1265, jun 2010.

[26] U. S. W. Supply and N. R. M. R. L. Water Resources Division. Epanet. *[Online] Available: http://www.epa.gov/nrmrl/wswrd/dw/epanet.html*, April 2009.

[27] J. Wu, Y. Zhang, Z. M. Mao, and K. G. Shin. Internet routing resilience to failures: analysis and implications. In *Proceedings of the 2007 ACM CoNEXT conference*, CoNEXT '07, pages 25:1–25:12, New York, NY, USA, 2007. ACM.