ASIS: Autonomous Seismic Imaging System With *In Situ* Data Analytics and Renewable Energy

Jose Clemente, Fangyu Li, Maria Valero, An Chen, and WenZhan Song

***Abstract*—Understanding earthquakes and volcano eruptions phenomena require the analysis of large amounts of subsurface data. Currently, data collected from these events need post-processing in an off-line mode due to hardware limitations and strong bandwidth constraints. Furthermore, analytics may take days, even weeks, in being processed after collection. In some cases, underground structures require previous studies and subsurface knowledge from experts to establish certain input parameters. By leveraging current IoT technologies, we introduce an autonomous seismic imaging system (ASIS), a real-time system for monitoring and generating analytics of seismic data based on a sensor network. ASIS processes real-time analytics near where the events are generated (fog computing) to mitigate bandwidth limitations. Calculations for earthquake detection, location, and magnitude are processed *in situ*. ASIS allows monitoring sensor status, visualizing the data of each sensor, and generating two dimensional/three dimensional (2-D/3-D) subsurface structures. The system is able to learn the underground structure by taking advantage of ambient noise data analysis, which significantly reduces the need of initial parameters. We incorporated a renewable energy source to extend sensor functionality, and we made the system self-healing and fault-tolerant. Indoor and outdoor evaluation of the system showed the error of earthquake detection is 13.7 ms and the spatial location accuracy is 5.1 ft.**

***Index Terms*—IoT, monitoring system, real-time analytics, seismic data.**

I. Introduction

SUBSURFACE monitoring is crucial to study natural disasters, such as earthquakes and volcano eruptions. In September 2017, a magnitude 7.1 earthquake killed at least 245 people in Mexico [1]. Modern monitoring systems and analysis of the subsurface may help to evaluate, learn, and predict these kinds of natural events. Furthermore, a system based on sensor networks that generates real-time alerts and analytics allows data processing near the location of the event, avoids long-latency periods during data processing, and enables recovering after hardware/energy problems without human intervention. For these reasons, surface monitoring is a desirable and valuable tool.

Currently, in earthquake and volcano monitoring, warning systems are commonly used for triggering alarms after event generation. These warning systems require real-time information to discriminate differences on arrival time of a signal and issue a warning when it detects a signal above a certain magnitude. The time, bandwidth, and energy used to send all raw data to a central place for processing may lead to a natural disaster due to outdated information. To have a better and efficient warning system, we need to understand the seismic phenomena by observing fault movements, tectonic plate movements, and subsurface velocities in real time. By leveraging current Cyber-Physical Systems (CPS) technologies defined as transformative technologies for managing interconnected systems between its physical assets and computational capabilities [2]—we introduce autonomous seismic imaging system (ASIS), a real-time system for monitoring and generating analytics of seismic data based on a sensor network. ASIS processes real-time analytics near events (fog computing) to mitigate bandwidth limitations. The innovations of this paper are described as follows: 1) An autonomous seismic imaging system. We incorporate a learning stage using ambient noise data analysis for estimating the reference initial velocity model that is used as input for complex analytics; for instance, the location of earthquake events; 2) an *in situ* data analytics procedure for earthquake detection, location, and magnitude calculation; 3) a robust storage strategy that includes data reduction, data compression, and data synchronization to mitigate the storage problems of seismic IoT devices; 4) a renewable energy scheme that allows sensor nodes to charge their batteries using solar panels and independently recover from an exhausted battery stage.

II. System Components and Architecture

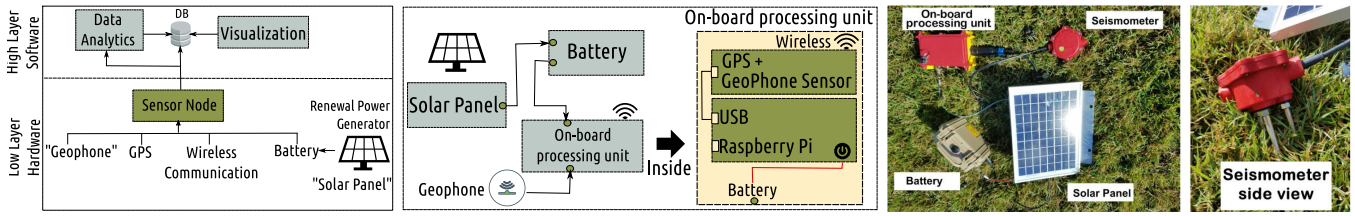
ASIS is a system composed of a suite of wireless sensor nodes integrated with data analytics capabilities and a monitor visualization tool. The system provides real-time analytics about geophysical events as well as raw data monitoring. The analytics are constantly processed by units inside the sensor network to reduce latency. Some analytics are processed by each node independently, and other specialized analytics are done by using a combination of own data and preprocessed data from other nodes inside the network. The preprocessed data are collected by *sink nodes*. ASIS architecture has been designed to be scalable, self-healing, and resilient. Our system can handle from four to N nodes.[1] Fig. 1(a) illustrates ASIS architecture.

Manuscript received August 22, 2018; revised April 6, 2019 and May 20, 2019; accepted May 25, 2019. Date of publication June 12, 2019; date of current version March 2, 2020. This work supported in part by Awards NSF-CNS-1066391, NSF-CNS-0914371, NSF-CPS-1135814, and NSF-CDI-1125165, and in part by Southern Company. *(Corresponding author: Jose Clemente.)*

The authors are with the Center for Cyber-Physical Systems, University of Georgia, Athens, GA 30602 USA (e-mail: jclementes@uga.edu; fangyu.li@uga.edu; maria.valero@uga.edu; an.chen25@uga.edu; wsong@uga.edu).

Digital Object Identifier 10.1109/JSYST.2019.2920073

[1]Many earthquake detection techniques require at least four sensors to detect earthquakes to avoid false alarms. Refer to [3].



(a) (b) (c) (d)

Fig. 1. (a) ASIS architecture. The two components, software, and hardware, divided in two layers. The low layer has the sensor hardware components, and the high layer has the software components. The two components are joined by the processing unit inside the sensor. (b) Hardware component structure. (c) Real sensor hardware. (d) Side view of the three-channel geophone.

TABLE I
SINGLE-BOARD COMPUTER SPECIFICATIONS

Raspberry Pi 3 Model B	
CPU	1.2GHz 64-bit quad-core ARMv8
Memory	1 GB SDRAM
USB 2.0 ports	4 (via the on-board 5-port USB hub)
On-board storage	32 Gb Micro SDHC
On-board network	10/100 Mbit/s Ethernet, 802.11n wireless, Bluetooth 4.1

1) Hardware Components: Nodes in the systems form a mesh network. Each node on the mesh has a global positioning system (GPS), a three-channel seismometer called geophone, a Raspberry Pi 3 board, a battery, and a solar panel as shown in Fig. 1(b). Some hardware components are housed in a waterproof box called R1 to protect them from the harsh environment. The low-power GPS provides the geo-location of the node and timestamps used for the system to collect, synchronize, and process data. The three-channel geophone detects the velocity of ground movements. Each channel records its own data respect to its axis X, Y, and Z, which correspond to directions North, East, and Depth. The geophone is low power. It consumes less than 240 mW per channel, and it is connected to the Raspberry Pi via USB port. It has 127 dB signal-to-noise ratio at 500 samples per second. It handles noise floor less than 0.2 μV rms. The single-board computer (Raspberry Pi) is the core of the system and is in charge of collecting and storing raw-data, processing data analytics, communicating with other units, and providing raw and processed data to the visualization tool. The main characteristics of the single-board are shown in Table I. A radio frequency XBee 802 module is connected to the single-board computer to perform data communication in areas over 100–1200 m. For smaller areas, the system communicates data using a WiFi mesh network. We also integrate a waterproof lithium battery 11 V and 99.9 Wh. The battery is connected to a 10 W solar panel to give the system renewable energy. This carbon-free energy solution is essential to slow climate change. Currently, researchers have joined forces in favor of this solution. Researchers in [4] are using energy-efficient devices (embedded devices) powered by solar framework system to provide their computing resources to end-users. Each unit consumes around 24% of the battery daily, which keeps the unit running for 4 d. Solar panels recharge the lithium batteries that power the units. Fig. 1(c) shows the custom design of one node, and Fig. 1(d) is a side view of the geophone.

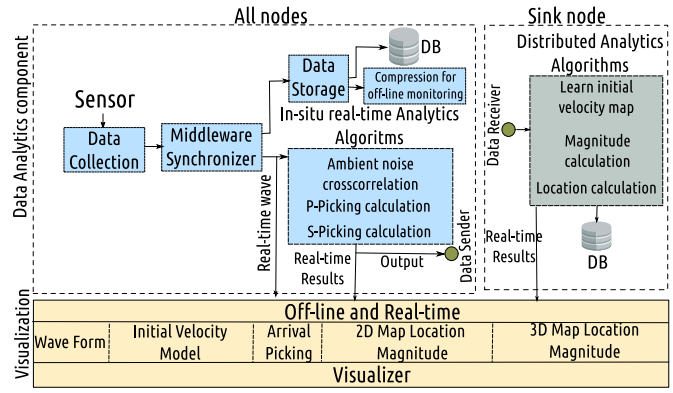


Fig. 2. Software design.

2) Software Components: We incorporate two software components into our system (See Fig. 2). The first component is the *Data analytics component* that process real-time data sensed by the geophone. There are several processes involved in computing the analytics, namely data collection, data synchronization, data storage, ambient noise cross-correlation, and arrival picking (P and S) calculation. All of them are running inside all nodes managing a low latency (milliseconds). Also, there are some distributed analytics constructed using nodes cooperation, such as event (earthquake) location and magnitude calculation that handle medium-low latency (less than four seconds). The analytics will be explained in detail in Section IV. The second component is the monitoring *visualization component*, which is used by scientists for visualizing real-time and off-line data.

III. DATA ACQUISITION AND STORAGE STRATEGY

In this section, we explain the data collection and data synchronization modules of different units via GPS. In addition, we explain the robust data storage method using data reduction and data compression techniques.

A. Data Collection

The first module of our system design is in charge of reading data sampling from the geophone sensor and synchronizing it with an accurate UTC timestamp from the GPS.

TABLE II
ENCAPSULATED INFORMATION ON THE SEISMIC PACKAGE

Name	Description
Sampling Rate	Data samples carried per second
Gain	Sensor sensitivity level
Power Status	Load level of the power supply
Temperature	Temperature in Celsius
Longitude	Geographical Longitude
Latitude	Geographical Latitude
Altitude	Geographical Altitude
GPS Quality	Connection quality with GPS satellite
GPS Satellite	Identifier of the connected satellite

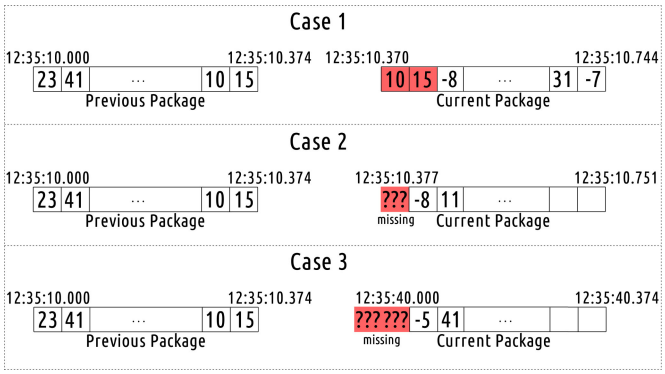


Fig. 3. Scenarios of data desynchronization.

For our experiments, we used a sampling rate of 500 Hz. For time synchronization the GPS provides an accuracy of 30 ns referenced to UTC time which adapts perfectly to the requirements of our system. Our seismic data is obtained in packets that are read one by one every 375 ms. Each data packet encapsulates the raw seismic data of the three-channels seismometer as well as useful information about the sensor and GPS as shown in Table II.

B. Middleware Data Synchronizer

Some factors as hardware speed, GPS connection quality, and GPS reconnection affect the accuracy of the data timer that is provided by the hardware. We present a solution when the data is not synchronized between nodes to avoid problems during the correlation of the signals. We notice that the data cannot be analyzed or saved in the database directly without making first an analysis between the first element of the current packet (FECP) and the timestamp of the last element of the previous packet (LEPP).

There are three scenarios of data desynchronization. The first case occurs when the sensor's clock is faster than 375 ms (packet time) making the FECP timestamp less than the LEPP timestamp and causing data duplication (See Fig. 3 top). In this case, problems usually appear in the last 10 ms of the packet. The second scenario (See Fig. 3 middle) occurs when the clock is slower than 375 milliseconds, which causes loss of some elements between LEPP timestamp and FECP timestamp. The last scenario occurs when there is a connection failure with the satellite and the difference between FECP and LEPP timestamps are greater than one second (See Fig. 3 bottom). In this case, the sensor buffer is drastically reduced.

Because a low latency approach is indispensable real-time systems, we designed a solution that allows analyzing part of the seismic packet instead of the whole packet. This guarantees that the *in situ analytics algorithms module* and/or the *data storage module* can use part of the package at the same time than the synchronizer is working. Then, the synchronizer only analyzes the last 10 ms (from the 375 packet ms), which represents less than 3% of the packet. Ten milliseconds were selected because, in the first two scenarios, the difference between the timestamps of the packets is less than that time.

Synchronization is handled by the synchronizer using a circular buffer. When a new data packet is read by the sensor, the position in the circular buffer is calculated using the timestamp of both previous and current packets. If the FECP is located next to the LEPP, there is no synchronization problem. The synchronizer sends the 375 ms of data to the other modules (10 ms from the previous package and 365 from the current package) and keeps 10 ms for analyzing the next package.

The first scenario of desynchronization is detected, when some element kept by the synchronizer are overwritten by the current package. In this case, the elements of the previous package are overwritten, and the synchronizer sends less than 375 ms of data to the other modules preserving the last 10 ms. The second scenario is presented if the incoming packet is located after the last timestamp but with some intermediate spaces. These intermediate spaces are replaced by zeros. In the same way, as in the previous cases, the synchronizer keeps the last 10 milliseconds and sends the rest of the elements to the other modules, which in this case is greater than 375 ms. In the last scenario, we consider that the GPS functionality may fail due to a problem with the line of sight between satellites [5], and the data is not read during that time. The synchronizer waits at most 475 ms for the next package. If in this time the new package has not been obtained, the synchronizer sends the previous 10 ms that are in its buffer to the other modules. Then, it clears the buffer and waits for the new packet that will be placed at the beginning of the buffer.

C. Data Storage

ASIS also stores raw data for posterior analysis and/or visualization. Every node has a MySQL database. We chose MySQL because it offers on-demand scalability, high performance, comprehensive transactional support, and it is open source. The data is stored in blocks. The system adopts 9 s as block size since this number is a multiple of the sensor clock (375 ms). However, due to the flexible design of our table structure, any number of seconds or milliseconds can be selected as the block size. The system gathers the raw data points and a timestamp for each one of these points.

Then, we incorporate a method that uses data reduction and data compression to meet storage limitations of our nodes. For data reduction, we focus on the timestamps. Instead, to store one timestamp for each point of the block, we only keep the start time (startTime) and the end time (endTime) of the block. Then, we store the frequency (sampling rate) that is used later for calculating the removed timestamps when needed. This implies that the larger the data block, the greater the data reduction.

Field	Type	Null	Key	Default	Extra
startTime	double	NO	PRI	NULL	
endTime	double	YES		NULL	
samplingRate	smallint(2)	YES		NULL	
dataType	char(3)	YES		NULL	
traceBuf	blob	YES		NULL	

Fig. 4. System table structure in the database.

For compression, we focus now on the raw data points. Rather than storing the data points, we applied a well-known compression function, compress(), that comes from the *zlib* library, which is later serialized and stored (traceBuf). One additional remark is that we store each type of data we are saving (dataType), which indicates if the raw data is unsigned integer, integer, long, double, etc. This makes our database structure more flexible. Fig. 4 shows the structure of the tables inside our database.

IV. Learning Stage and Data Analytic

ASIS nodes can perform real-time analytics individually (*in situ*) and collectively (distributed). To locate earthquakes, nodes have to: 1) learn the initial velocity model of the field; 2) estimate the arrival time of the earthquake signal; and 3) locate the event and estimate its magnitude. We develop different algorithms for performing these analytics. The initial velocity model is needed for estimating the arrival time (arrival picking) and estimating the location and magnitude of the earthquake. We use Ambient Noise Tomography (ANT) [6] as the technique to calculate the initial velocity model. In this calculation, nodes work both individually and collectively. Later, each node performs an algorithm to automatically detect the events and calculates its arrival picking time (individually). When an arrival picking time is calculated, it is sent to the sink node (collectively). Sink node receives and processes in real time these arrival times to estimate the earthquake location and its magnitude.

A. Learning Initial Velocity Model

We use ambient noise algorithms to estimate the reference initial velocity model that is used as an input in our event location and magnitude analytics. To obtain this model, we first set up the nodes in the field and let them run for about 30 min. During this time, they are able to figure out the subsurface velocities and construct the velocity model. This feature adds a novel characteristic to our seismic system because it can be used in any field without previous knowledge of the subsurface structures. The work-flow to construct the velocity model is shown in Fig. 5. We first read the synchronized data after t time. We preprocess the signal to remove wild noise and instrumental irregularities as proposed in [7]. Then, we transmit the preprocessed and compressed data to neighbor sensor nodes. Once every node receives its neighbor data, we perform the cross-correlation between the signals. The cross-correlation helps to estimate the delay time between two particular station locations (s_1 and s_2), and it is given by

$$C_\Gamma(\tau, s_1, s_2) = \frac{1}{\Gamma}\int_0^\Gamma u(t, s_1)u(t+\tau, s_2)dt, \qquad (1)$$

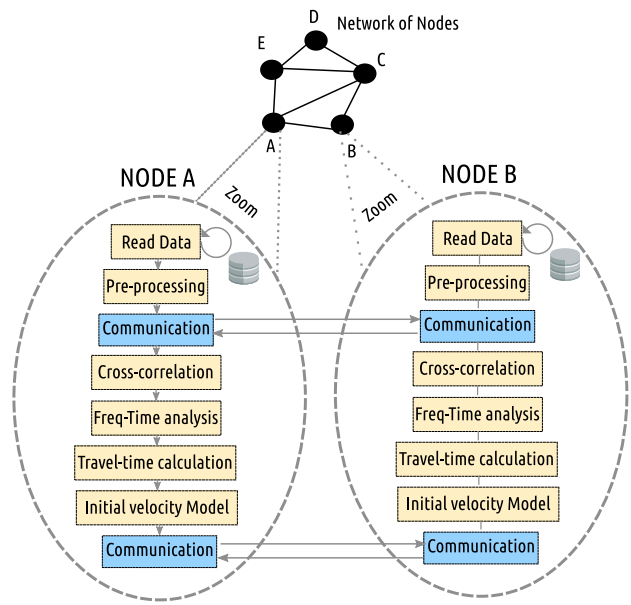


Fig. 5. Workflow for learning initial velocity model.

where Γ is the total time recorded (usually a window is used), u is the wave signal, and τ is the displacement. After cross-correlation, the phase velocities can be estimated as a function of the period (1/frequency) by using traditional frequency-time analysis. The method we use is based on a traditional version of frequency time analysis described in detail in [8]. We then construct the initial velocity model by using the travel time estimations and a technique called Eikonal Tomography[9]. We use the distributed version of Eikonal Tomography explained in detail in [10], and the inversion of s-wave velocities [11]. Finally, the referent velocity model is communicating with all the nodes in the network and can be used as the initial velocity model of other analytics.

B. Arrival Picking

Considering the complexity and efficiency, we adopt the arrival-time picking algorithm, STA/LTA method on a single component of the seismic records [12]. The STA (short time average) is sensitive to rapid fluctuations in the amplitude of the time series, whereas the LTA (long time average) provides information about the background noise [13], [14]. The arrivals are picked on the maximum value of the derivative function of the STA/LTA ratio curve. The STA and LTA at the ith time sample are commonly defined as:

$$\mathrm{STA}_i = \frac{1}{ns}\sum_{j=i-ns}^{i}\mathrm{CF}_j, \quad \mathrm{LTA}_i = \frac{1}{nl}\sum_{j=i-nl}^{i}\mathrm{CF}_j \qquad (2)$$

where, i, j are the sample number, ns and nl represent the short and long window lengths. CF stands for characterization function which characterizes waveform properties [15]. In our application, we use the 2nd order energy function to highlight amplitude changes.

C. Event Location and Magnitude

Based on the reference velocity model obtained from the workflow shown in Fig. 5, we can get an initial microseismic hypocenter as well as the origin time (x_0, y_0, z_0, t_0). Because of the difference between the estimated velocity model and the real velocity model, the arrivals from the initial hypocenter will show a time difference with the recorded data. The travel-time residuals should be a linear function of the hypocentral distance correction. The calculated arrival times at station i are $t_i^c = T(x_0, y_0, z_0, x_i, y_i, z_i) + t_0$ and the travel-time residuals r_i are $r_i = t_i^o t_i^c$. The corrections Δx, Δy, Δz, and Δt should be as small as possible. Using a Taylor series expansion, the residual can be expressed as:

$$r_i = \frac{\delta T}{\delta x_i}\Delta x + \frac{\delta T}{\delta y_i}\Delta y + \frac{\delta T}{\delta z_i}\Delta z + \Delta t. \tag{3}$$

Geiger [16] invented and applied an iterative inversion method so-called Geiger method to reach a minimum residual point to obtain the hyper center. After getting the microseismic location, we can estimate its magnitude. The standard body-wave magnitude formula is calculated as:

$$m = \log_{10}\frac{A}{T} + Q(D, h) \tag{4}$$

where A is the amplitude of ground motion recorded by the geophones, T is the corresponding period (in seconds); and $Q(D, h)$ is a correction factor that is a function of distance D (between epicenter and station) and the earthquake depth h.

V. System Visualization Tool

An essential part of a monitoring system is the visualization tool. Our system has a visualization tool that was designed for monitoring both raw data and the results of the analytics. Also, our system has two modes of visualization, one in real time (*on-line mode*) and another mode that allows selecting data and results previously stored in the database called *off-line mode*. An added value to our visualization tool is that the waveform can also be heard.

A. On-Line Mode

Using this mode, the experts can monitor the units that are in the deployment. They can observe in real time and in parallel any option that the tool offers such as waveform from any channel, arrival picking, and the analytics results (location and magnitude) in a 2-D map and/or 3-D structure.

1) Waveform and Arrival Picking: After selecting the unit and channel, our visualization tool shows the raw data waveform that is read by the sensor in real time. The new window that is created has a one second update timer, which is approximately the time that takes the system to synchronize and send the data to the front end. Several channels can be monitored at the same time. Additionally, the system stores the travel time pickings results for *P*-wave and *S*-wave. Travel time pickings can also be monitored in real time. Real-time waveform and arrival picking are shown in Fig. 6.

2) Location and Magnitude on 2-D Maps and 3-D Structures: As mentioned in Section III, location and magnitude analytics

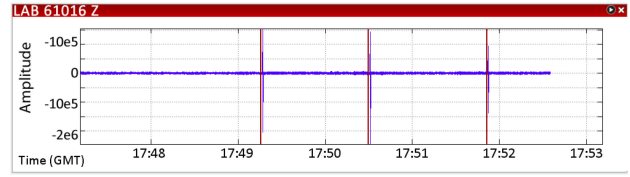


Fig. 6. Real-time arrival picking. The top window shows the real time waveform from data obtained by the sensors.

(a) (b)

Fig. 7. Event location and magnitude. The images show event time and magnitude. The green dot represents the unit location and the read circle/sphere represents the event magnitude. (a) 2-D map. (b) 3-D structure.

are processed in a sink node at the edge of the network. Therefore, to visualize these results, the sink unit should be selected.

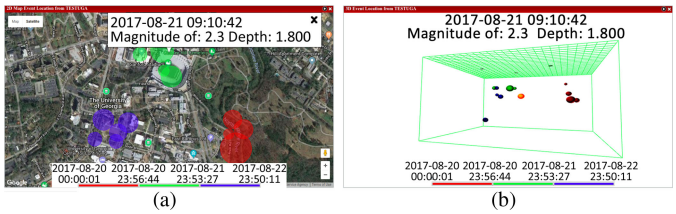
There are two types of views where events are displayed, 2-D maps or 3-D structures. The 2-D map was developed with the Google Maps API. This view allows to see the estimated location where the event occurred, as well as the magnitude represented by the circle size shown in Fig. 7(a). However, Google Maps does not allow visualization inside the subsurface of the earth. Then we use a 3-D structure to observe the depth where the events happened. It was developed using *3d.js library*. The communication with the back end is done through *JSON* packages where the latitude, longitude, depth, and magnitude of the event are received. An example of 3-D visualization is shown in Fig. 7(b). The front end is updated only when a new event is generated and processed by the back end. These views were developed using *JavaScript* to avoid the need to update the whole screen.

3) Unit Status: The unit status monitoring is needed for supervising the sensor units, especially when they are deployed in an uncontrolled or harsh environment. The module monitors three main measurements: i) the battery level, which indicates in real time if there are irregularities in the battery draining; ii) the quality of the connection with the satellite. This is an important measure because the whole system depends on the satellite synchronization; and iii) the temperature of the sensor unit.

4) Wave Sound: The wave sound can be useful for earthquake/volcano/tremor prediction analysis. We include this novel characteristic to our visualization tool by allowing users to hear in real time the sound of a specific waveform. The data is translated from raw data to white noise, where the frequency and intensity of the noise changes with respect to the amplitude of the wave. Variations in the wave sound may potentially indicate an event that can be detected by hearing.

B. Off-Line Mode

All options presented in the on-line mode are also found in the off-line mode. This mode offers additional features for data selection and visualization. The main difference between this



(a) (b)

Fig. 8. Location for different stages. The colors represent the different time periods. (a) 2-D map. (b) 3-D structure.

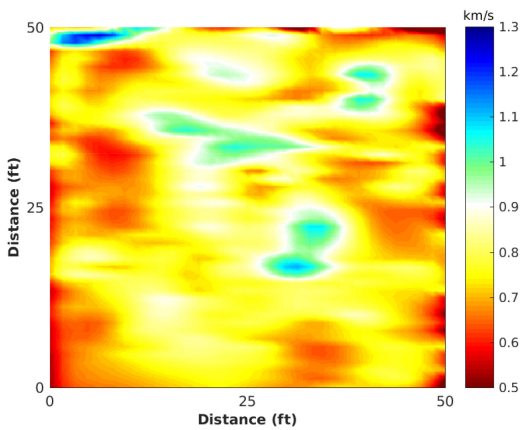


Fig. 9. Initial velocity model. Outdoor deployment.

mode and the on-line mode is the way in which the information is read and transmitted. In off-line mode the information is read directly from the database that is located inside the unit, and the viewing windows have disabled the option of real time communication with the unit.

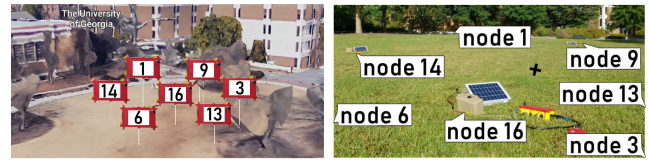
1) Data Selection: There are two methods of data selection. The first method called *manual selection* allows users to visualize data in a specific range of time defined by the user. In this case, the user should know the time range (stage) to visualize. The second method is *interactive selection*, which allows users to directly click on the waveforms to zoom in results.

2) Location and Magnitude on Different Stages: Analytics results that are stored in the sink unit can also be viewed using off-line mode. Users can use any of the two data selection methods to generate a stage (range of time) of interest. The results can be seen on 2-D maps as shown in Fig. 8(a) or 3-D maps like the example shown in Fig. 8(b), where the different stages are represented with colors.

VI. Evaluation and Validation

A. Outdoor Deployment

1) Initial Velocity Model: To validate our system, an outdoor test was conducted. We deployed 6 edge units and a sink node in a circular area with a diameter of 50 ft inside the University of Georgia campus. We applied ambient noise algorithms to estimate the reference initial velocity model of the outdoor deployment as explained in Section IV-A. Note in Fig. 9 that the subsurface velocity is kind of stable around 0.8 m/s.



(a) (b)

Fig. 10. Outdoor deployment location.

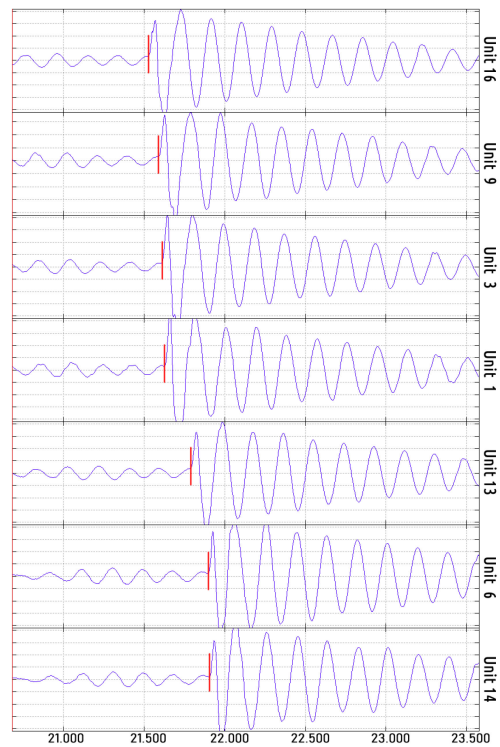


Fig. 11. Outdoor deployment arrival picking results.

The advantage of having this information is to avoid the assumption of constant velocity of other works, which makes the location estimation more stable.

2) Data Analytics: We use hammer shock on the ground to generate events in a specific source location. Using this technique it is easy to validate the arrival picking and the event location because the location of the event source is controlled. The sink node was located in the middle of the area. Fig. 10(a) shows the real location of the units in our deployment.

The stations were monitored to verify whether they were working correctly. Then, we created 20 events in different parts of the field including inside and outside of the deployment area. In total, 140 events were generated and received for the sink unit. One location source is used for presenting results, and it is represented with a cross shown in Fig. 10(b).

The recorded signal and the picking arrival time from one hammer shock event captured for all units are shown in Fig. 11. This image presents in ascending order the arrival time pickings of the events. According to this order, the closest and farthest units from the source are units 16 and 14, respectively. This result matches with the distances between the source and these units in the real deployment. We compared manually selection

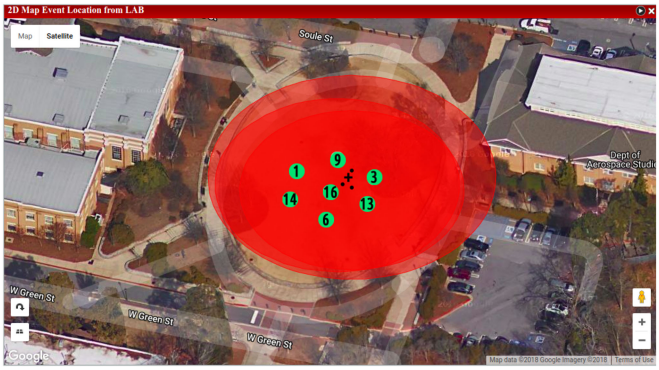


Fig. 12. Location and magnitude result from hammer test. The cross and black dots represent the real location of the hammer shock and the detected event location, respectively.

TABLE III
DATA STORAGE COMPARISON

Raw Data Storing			
Field	# of Elements	Size (bytes)	Total of Bytes
timeStamp	4500	8	36000
Data	4500	4	18000
		Total Storage	54000
Proposed Method			
Field	# of Elements	Size (bytes)	Total of Bytes
startTime	1	8	8
endTime	1	8	8
samplingRate	1	2	2
dataType	1	3	3
traceBuf[2]	1	9360	9360
		Total Storage	9381
Storage Improvement			82.62%

[2]compressed data block.

of picking arrival time with our automatic system and tense the mean absolute error. We found our system has a difference of 13.7 ms in arrival time picking regarding the manual selection, which is an acceptable error.

Finally, Fig. 12 confirms the event location accuracy because all events were detected in the experiment area. Location evaluation also was compared with the real location of the hammer shock. The mean square error shows a difference of 5.1 ft between the real spatial location and the calculation of our system.

B. Storage Reduction

We use 9 sec block to analyze the storage reduction provided by our method. First, by making the storage in blocks, our method reduces the number of timeStamps. The reduction in terms of space is 36 K which represents 66% of data size. Second, by applying the data compression our method reduces an additional 16% of space. Table III shows a comparison between saving raw data without any reduction or compression and our proposed method. Note that if we do not apply any technique for a block size of 9 sec, we have to store 54 K of data (18 K of data and 36 K of timestamps). In our proposed method, we store a start time (8 bytes), an end time (8 bytes), a sampling rate value (2 bytes), an indicator for the type of data (3 bytes), and a block of data after compression. The total storage improvement after data reduction and data compression is 82.6%.

C. Stress Test for Renewal Energy

A full charge lithium battery keeps the system running just four days. Solar panels are included in hardware design in order to maximize the system life. Solar panels recharge the lithium batteries, but it is affected by environmental factors such as rainy or very cloudy days. A stress test was performed on our lab rooftop to study system performance, units behavior, and power sources. The test was executed during a period of 30 d. Fig. 13(a) shows battery levels of four deployed units. Also, the weather of each day is displayed in the upper part of the figure. The test revealed two important facts. First, on sunny days the solar panel is able to recharge the battery in an average of 32%. It is enough for keeping a unit working for one and a half days. Second, during the worst load periods, between January 8 and January 12, the solar panels were able to extend the battery life from 4 to 5 days. It means the load average was around 8% daily. It is reflected in Fig. 13(b). However, it was not enough for unit 10 which ran out of power for a period of four hours. This unit returned to its normal function without any human intervention the next day thanks to the hardware and software design. This shows the self-healing feature of our system. Regarding the system, no failure was detected in the execution of the analytics algorithms.

VII. RELATED WORKS

Several research areas have used wireless sensor networks for monitoring. A cooperative sensing framework for environmental monitoring was presented on [17]. The map of an unknown scalar field is built based on the data sensed by a mobile sensor network. The information from sensors is fused using a distributed consensus algorithm. They applied a simulation to validate the proposed algorithm, and it over-performs the normal cooperative sensing in term of sensing performance. Authors in [18] designed and developed a wireless sensor network for habitat and environmental monitoring. In collaboration of biologists at the College of the Atlantic, they collected data at the James San Jacinto Mountains for studying the relation between animal behavior and weather. In 2007, system architecture and hardware-software organization for personal health monitoring was presented [19]. Several key technical issues such as sensor node hardware architecture, software architecture, network time synchronization, and energy conservation were addressed in this research. The first system for seismic monitoring using wireless sensor network was addressed for a group of Harvard researchers. They deployed four nodes on the Tungurahua Volcano in Ecuador, and they successfully collected three days of acoustic data [20]. On the other hand, our laboratory has been pioneer in developing applications based on wireless sensor networks for seismology purposes [21]. We have designed decentralized algorithms for executing *in situ* analytics instead of collecting data to the central place for postprocessing [22], [23]. Platforms for testing and performing analytics as part of *Fog Computing* have been developed in our previous research [24], [25].

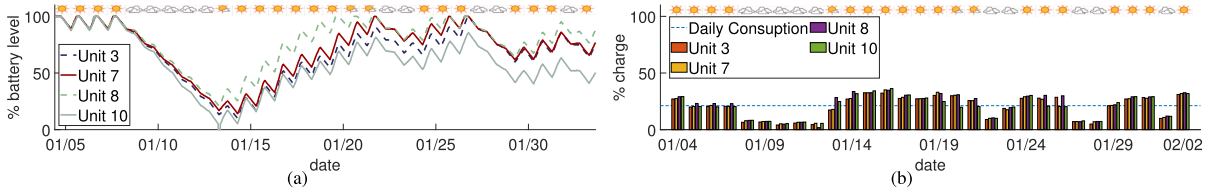


Fig. 13. (a) Battery level of the rooftop deployed units during a thirty days test. Top of the image: Weather forecast for each day. (b) Daily battery load during the stress test. The image shows the daily charge level of each battery, and the power consumption average of each unit. Top of the image: Weather forecast for each day.

Also, we have worked in the area of imaging for monitoring and mapping subsurface geophysical structures [10].

VIII. Conclusion

We presented an autonomous seismic imaging system for real-time monitoring and data analytics. IoT *in situ* computing was a key component for the design and implementation of the system. By processing analytics near where the data is generated and using collaborative data processing between different sensors, we meet energy and bandwidth sensor constraints and tense results without the need of sending all data to a central place. We presented a scheme for data collection, synchronization, and reduction that adapts storage needs to our IoT sensor devices. We also showed our system is capable of recovering from exhausted battery stages by using solar panels and self-healing services. Most importantly, we incorporated an autonomous learning stage using ambient noise data analysis for estimating the reference initial velocity model that is used as input for complex analytics. Indoor and outdoor tests showed our system generates results closer to manual interpretations and provides a powerful tool for subsurface monitoring.

References

[1] N. Chavez, S. Almasy, R. Sanchez, and D. Simon, "Central Mexico earthquake kills more than 200, topples buildings," 2017. [Online]. Available: https://www.cnn.com/2017/09/19/americas/mexico-earthquake/index.html

[2] J. Lee, B. Bagheri, and H.-A. Kao, "A cyber-physical systems architecture for industry 4.0-based manufacturing systems," *Manuf. Lett.*, vol. 3, pp. 18–23, 2015.

[3] R. Tan, G. Xing, J. Chen, W.-Z. Song, and R. Huang, "Fusion-based volcanic earthquake detection and timing in wireless sensor networks," *ACM Trans. Sensor Netw. (TOSN)*, vol. 9, no. 2, pp. 17:1–17:25, 2013.

[4] C.-M. Cheng, S.-L. Tsao, and P.-Y. Lin, "SEEDS: a solar-based energy-efficient distributed server farm," *IEEE Trans. Syst., Man, Cybern.: Syst.*, vol. 45, no. 1, pp. 143–156, Jan. 2015.

[5] R. G. D'Eon, R. Serrouya, G. Smith, and C. O. Kochanny, "GPS radiotelemetry error and bias in mountainous terrain," *Wildlife Soc. Bulletin*, vol. 30, no. 2, pp. 430–439, 2002.

[6] F.-C. Lin, M. P. Moschetti, and M. H. Ritzwoller, "Surface wave tomography of the western united states from ambient seismic noise: Rayleigh and love wave phase velocity maps," *Geophysical J. Int.*, vol. 173, no. 1, pp. 281–298, 2008.

[7] G. D. Bensen *et al.*, "Processing seismic ambient noise data to obtain reliable broad-band surface wave dispersion measurements," *Geophysical J. Int.*, vol. 169, no. 3, pp. 1239–1260, 2007.

[8] A. L. Levshin and M. H. Ritzwoller, "Automated detection, extraction, and measurement of regional surface waves," in *Monitoring Comprehensive Nuclear-Test-Ban Treaty: Surface Waves*. Switzerland: Springer, 2001, pp. 1531–1545.

[9] F.-C. Lin, M. H. Ritzwoller, and R. Snieder, "Eikonal tomography: Surface wave tomography by phase front tracking across a regional broad-band seismic array," *Geophysical J. Int.*, vol. 177, no. 3, pp. 1091–1110, 2009.

[10] M. Valero, G. Kamath, J. Clemente, F.-C. Lin, Y. Xie, and W. Song, "Real-time ambient noise subsurface imaging in distributed sensor networks," in *Proc. 3rd IEEE Int. Conf. Smart Comput.*, Hong Kong, China, 2017.

[11] J. Xia, R. D. Miller, and C. B. Park, "Estimation of near-surface shear-wave velocity by inversion of Rayleigh waves," *Geophysics*, vol. 64, no. 3, pp. 691–700, 1999.

[12] J. Akram and D. W. Eaton, "A review and appraisal of arrival-time picking methods for downhole microseismic data arrival-time picking methods," *Geophysics*, vol. 81, no. 2, pp. KS71–KS91, 2016.

[13] S. R. Taylor, S. J. Arrowsmith, and D. N. Anderson, "Detection of short time transients from spectrograms using scan statistics," *Bulletin Seismological Soc. Amer.*, vol. 100, no. 5A, pp. 1940–1951, 2010.

[14] F. Li and W. Song, "Automatic arrival identification system for real-time microseismic event location," in *Proc. Soc. Explor. Geophysicists Tech. Program Expanded Abstracts*, 2017, pp. 2934–2939.

[15] J. Wong, L. Han, J. Bancroft, and R. Stewart, "Automatic time-picking of first arrivals on noisy microseismic data," *CSEG. 0 0.2 0.4 0.6 0.8*, vol. 1, no. 1.2, pp. 1–4, 2009.

[16] L. Geiger, "Probability method for the determination of earthquake epicenters from the arrival time only," *Bull. St. Louis Univ*, vol. 8, no. 1, pp. 56–71, 1912.

[17] H. M. La, W. Sheng, and J. Chen, "Cooperative and active sensing in mobile sensor networks for scalar field mapping," *IEEE Trans. Syst., Man, Cybern.: Syst.*, vol. 45, no. 1, pp. 1–12, Jan. 2015, doi: 10.1109/TSMC.2014.2318282.

[18] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *Proc. 1st ACM Int. Workshop Wireless Sensor Netw. Appl.* Atlanta, Georgia, USA, 2002, pp. 88–97.

[19] A. Milenković, C. Otto, and E. Jovanov, "Wireless sensor networks for personal health monitoring: Issues and an implementation," *Comput. Commun.*, vol. 29, no. 13–14, pp. 2521–2533, 2006.

[20] G. Werner-Allen, J. Johnson, M. Ruiz, J. Lees, and M. Welsh, "Monitoring volcanic eruptions with a wireless sensor network," *Second Eur. Workshop Wireless Sensor Netw.*, Jan. 2005.

[21] G. Kamath, L. Shi, W.-Z. Song, and J. Lees, "Distributed travel-time seismic tomography in large-scale sensor networks," *J. Parallel Distrib. Comput.*, vol. 89, pp. 50–64, Mar. 2016, doi: 10.1016/j.jpdc.2015.12.002.

[22] L. Zhao, W.-Z. Song, X. Ye, and Y. Gu, "Asynchronous broadcast-based decentralized learning in sensor networks," *Int. J. Parallel, Emergent Distrib. Syst.*, 2017.

[23] L. Zhao and W. Song, "Decentralized consensus in distributed networks," *Int. J. Parallel, Emergent, Distrib. Syst.*, vol. 33, no. 6, pp. 550–569, 2018.

[24] J. Clemente, M. Valero, J. Mohammadpour, X. Li, and W. Song, "Fog computing middleware for distributed cooperative data analytics," in *Proc. IEEE World Fog Congress 2017*, Santa Clara, CA, USA, 2017.

[25] G. Kamath, P. Agnihotri, M. Valero, K. Sarker, and W.-Z. Song, "Pushing analytics to the edge," in *Proc. IEEE Global Commun. Conf.: Sel. Areas Commun.: Internet Things*, Washington, USA, 2016.