DUE TO THE PROLIFERATION OF INTERNET-OF-THINGS (IoT)-based technologies in the last several years, digital computing hardware and software technologies have seen massive performance improvement. Additionally, these technologies provide lower costs for comparatively higher computation and storage, more compact size hardware, and compatibility with a large selection of operating systems. Furthermore, communication protocols have increased the penetration of single-board computers in many consumer and industrial applications. This article presents the application of a state-of-the-art edge computing infrastructure to the electrical power distribution grid. Electrical power distribution is becoming increasingly complex with the large degree of integration of distributed energy resources (DERs). The distribution system also experiences many different undesired events, such as different types of temporary and permanent faults, loss of measurement data, and cyberattacks. This article highlights a small-scale experimental validation of edge computing in power distribution automation that can be used for classifying different faults, detecting anomalies in the grid, measurement data recovery, and other advanced analytics techniques.

Introduction
With a large number of parallel data sources becoming readily available in a smart grid, data fusion techniques that combine multiple data sources lie at the heart of smart grid platform integration.
Related to this concept, the authors developed intelligent applications that reside on edge computing device hardware in the distribution grid. The developed edge processor platform performs advisory control functions to assist in the larger goal of providing enhanced grid resilience and distributing intelligence and provides a faster response time to system anomalies.

**Edge Computing and Communications Basics**

Edge computing combines technologies related to data processing, communications, and analytics at the grid edge, which is defined as the distribution system between the substation and the end use customer sites. Edge computing devices (ECDs) are connected to grid field devices, such as pole top reclosers and switches, meters, line post sensors, and other field devices, via wired and wireless communications. Each ECD is also capable of communicating with its peers on the same distribution system, other edge processors, and providing communication redundancy and application coordination. Furthermore, each ECD is capable of communicating upstream to substation computers, utility control centers, and even the cloud. See Figure 1 for an example. Multiple communication media can be implemented using the ECDs, including Wi-Fi, cellular, and radio communication. Utility preference will determine which media are used for the field communications and for the upstream communications. The deployment of 5G and future advanced cellular platforms will only enhance the functionality that is capable of being deployed by ECDs.

The edge processor concept entails the merging of edge computing, ubiquitous communications, and advanced analytics. As shown in Figure 2, the architecture contains layers of communication devices and intelligent applications. At the outer layer, low-cost edge computing devices (LC-ECDs) gather data from the local field devices and communicate this data to the medium-cost edge computing devices (MC-ECDs) or to the high-cost edge computing devices (HC-ECDs). The MC-ECDs contain basic to slightly advanced applications and communicate and coordinate between each other and with the upstream HC-ECDs. The HC-ECDs contain advanced application and management functions and can communicate to the cloud if desired by system designers.

**Edge Computing Framework**

The proposed embedded framework for the edge processor concept is shown in Figure 3. The framework is designed to support two operating system variants (Windows/Linux) to accommodate different applications and commercialization options. Many communication libraries (industrial communication, web server, protection, and control proprietary software) are generally hosted in the Windows operating systems with separate
The framework is designed to support two operating system variants (Windows/Linux) to accommodate different applications and commercialization options.

development toward containerized solutions that are computer operating system independent. On the other hand, novel secure virtual private network (VPN) technologies (OpenVPN, wireguard, and so on), mesh wireless libraries, and machine learning applications are easiest to evaluate and implement in Linux. Container engines such as Docker or Qemu can then be installed to support containerized grid applications.

Some implementations of the ECDs or device network may have a hierarchical structure. A high-level block diagram of such implementation is shown in Figure 4. This figure shows a multilayer hierarchal architecture with different capability ECDs (HC-ECD, MC-ECD, LC-ECD). These device networks feature a central or main ECD with management capabilities for pushing applications to individual devices. A container registry, hosted either on the highest level ECD or in the cloud, provides image repositories for the applications hosted on each computing device. These repositories contain various tagged versions of container images and the lower level ECDs can pull down the image they require by requesting the device or using configuration-specific tags.

A management bus in this architecture implements control functions, and the container image pulls over secure protocols. The transfer of applications to specific devices is as simple as providing a descriptive text file for image dependencies to pull or for pulling the actual image itself from the higher level ECD.

The management bus features a broker communicating over publish-subscribe type message queuing protocols, such as message queuing telemetry transport (MQTT), advanced message queuing protocol (AMQP), and so on. Each application supports receiving message commands to start and stop application processes, receive configuration-descriptive files coordinating grouping with other devices, and zone configurations. This management interface also manages pushing updates to deployed applications, syncing databases at each individual device for distributed applications, and any updates to security or public key infrastructure (PKI) technologies for operational needs. The lower-level ECDs regularly check for updates to individual device configuration and zone configuration files exchanged over a JavaScript Object Notation (JSON) or equivalent file format. A separate communication bus provides protocols used for communication between applications on each device necessary for wider area or distributed applications. The applications of the communication bus could be sensor measurements, status messages, or other necessary data interchanges. This communication bus will also include modbus, distributed network protocol 3 (DNP3), and other established protocols used for distribution automation. A few examples of base applications for distribution automation are machine learning, data collection.
and fusion, database management, and human–machine interaction or web-server-hosted graphical viewing applications.

Figure 5 shows a high-level process and components for converting standalone grid applications (software algorithms) into containers and then deploying, managing, deleting, and updating these containers across many ECDs in the field application.

Several tools exist that can be used to accomplish the containerization and orchestration tasks. In this demonstration, Docker Engine was selected to containerize the application due to its simplicity in implementation. However, there are multiple tools for orchestration, namely Docker-Swarm, Kubernetes (also known as K8S), and K3S (a certified Kubernetes distribution for resource constrained or remote locations). Among these, K3S is the simplest and least resource-consuming platform for container orchestration, which is desired for LC-ECDs and MC-ECDs. Therefore, K3S was chosen to be the container orchestration platform in this demonstration.

**Experimental Prototype Design**

A fault detection, isolation, and restoration (FDIR) application scheme, as shown in Figure 6, was considered for validating the edge processor concept. Depending on the specific needs of the application, the edge computers can directly communicate with intelligent electronic devices (IEDs) and sensors as well as with other ECDs in the hierarchy. The logical system in this demonstration consists of two substations with a typical five-recloser loop, including four protection recloser
controllers, one grid-tie switch, multiple loads, voltage and current sensors, three ECDs, and two gateway devices. Gateway devices communicate with multiple ECDs for redundancy and backup. For example, recloser controller R2 can be controlled by both ECD1 and ECD3, and recloser controller R3 can be controlled by ECD2 and ECD3, and so on.

The simulation of the FDIR scheme under test was done in a JavaScript-based webserver tool called Node-RED. The Node-RED software allows creation of logical and communication nodes that can represent different components in the FDIR scheme and perform intended control operations. Another benefit of the Node-RED system is that it also has graphical user interface (GUI) tools that can be used for creating a very interactive simulation demonstration.

The FDIR algorithm is based on the state machine represented in Figure 7. The state machine runs inside the ECD3 or supervisor ECD. Each logical state represents the state of the electrical system. More logical states can be added in the future development of complex applications.

**Demonstration With Wireless Sensors and IED Devices**

A demonstration of the FDIR scheme using wireless line sensors and commercial protection and control products is discussed here. The demonstration consists of three edge computing (three HPE EL10) devices, as shown in Figure 8. The hardware setup uses actual recloser controllers (RER620) to emulate recloser R1 and grid tie GT1 from Figure 6. Other relay devices are modeled by simplified software switches in Node-RED. Wireless sensors were fabricated by using commercially available line sensors (voltage and current) and digital signal processor-based sensor data acquisition and wireless modules to connect and share data with respective ECDs. MQTT was used as the communication protocol to move the data and commands between ECD devices, whereas Modbus TCP protocol was used to communicate with IED RER620. However, other protocols are also supported (e.g., DNP3, IEC 61850).

In the setup of Figure 8, the actual electrical load is connected to both wireless sensors. The load voltage is standard 110 V 60 Hz and a few resistive loads were connected to draw sufficient current for the purpose of the demonstration. The trip threshold is set to 0.2 A in the supervisor ECD (ECD2), meaning when the current through any load exceeds 0.2 A the respective relay should trip.

**Figure 5.** Simplified process for building containerized grid application on edge computing platform.

**Figure 6.** Example FDIR scheme for validating proposed edge processor-based container orchestration.
The wireless sensors measure both voltage and current of their load and send that measurement data to the respective ECD over the wireless local network. As soon as the load current crosses the threshold, the supervisor takes an action and sends the trip command to the recloser protection relay. The relay opens and its status is sent back to the supervisor. Later, when the load current goes below 0.2 A, indicating that the fault is cleared, the relay then goes back to the closed position.

The GUI panel for one of the edge devices used in this demonstration is shown in Figure 9. One thing common in all GUI panels (across all ECDs) is a “system diagram” and fault status indicators. Apart from that, each edge GUI panel shows settings and statuses relevant to their location in the system. The GUI accepts certain inputs from the user to simulate different types of faults as well as displays system status, parameters, and waveforms. For example, ECD1 shows the status of Substation A breaker, Relay 1 and 2, and has GUI buttons with the capability to inject a simulated fault at locations Fault 1 and Fault 2. The demonstration shows the capability to simulate faults at three different locations in the two-feeder substation topology. During normal operation, simulated faults can be introduced at different locations in the system via GUI buttons for faults. Depending on the fault type, the algorithm in the ECD decides to open or close certain recloser devices. The system maintains the state as long as the fault exists. Upon removal of the simulated fault, the system is restored to normal operation. Prefault load on faulted circuits is used to check the load before restoration.

This simple demonstration verifies many different aspects of the edge computing platform, such as connecting to and collecting data from wireless sensors, connectivity with existing protection relay devices, as well as distributed computing and relaxed real-time control through the edge computing architecture.

**Demonstration With Software Containers and Orchestration**

A final demonstration for FDIR using container orchestration is discussed here. The demonstration consists of six ECDs (3 × EL10 and 3 × Raspberry Pi4) as shown in Figure 10. All the edge devices are connected over a wireless network that is also connected to the Internet via a router. The K3S master and agent tool are installed on respective ECDs. Together, the six ECDs form a K3S cluster. Once the master ECD detects all the agent nodes, the setup is ready to run the FDIR application. The container image for FDIR application is also developed on two different hardware platforms, i.e., ARM7 (Raspberry Pi4) and x86 (EL10). Both container image versions are then uploaded to the Docker-Hub public repository for test purposes. Each image is about 200 MB. The container orchestration tool, K3S, assures that the set performance metrics of the container cluster system are met all of the time, for example, start and stop of containers, updating to the latest version, deleting containers, creating more instances of a container in the same edge device, deploying a specific container on a specific edge device in the network, and so on.

In this demonstration, for simplicity, a command line interface is used to deploy and monitor the FDIR application. A configuration file is developed that defines the settings for each ECD. The configuration file includes the type of container image that should run on the target device(s), port mapping for user interface, number of...
instances of each application, and so on. After deploying the yaml file (a type of data serialization file) through the command line, the status of K3S cluster can be monitored. After deploying the yaml configuration file, all the ECDs download their respective (either ARM7 or x86) FDIR container image from the DockerHub file registry. The download can take several minutes, depending on the speed of Internet access. As soon as the images are downloaded, they are automatically deployed in the respective ECD. Then, based on the logic inside the container image, the Node-RED application is automatically started in all ECDs, and they each run specific parts of the entire image based on their identifications.

The ECD that runs the master node of K3S can be used to access the GUI panel for the FDIR demonstration. The demonstration has the capability to simulate faults at three different locations in the two-feeder substation topology. Depending on the type of fault simulation, the state machine algorithm discussed previously will execute a certain sequence of operations in order to first isolate the fault, then restore the power, and when the fault is cleared, go back to normal prefault operation.

**Applications of Interest**

The data fusion techniques deployed on the edge processor platform have addressed two main objectives: “Fixing problematic data,” which refers to the case when the data source is having quality issues such as inconsistency, imperfection, and so on, and “Extracting higher-level information” to obtain knowledge from multiple data sources. The following provides more information regarding four applications developed in the edge processor framework.

**Distribution Grid Event Classification**

Both machine-learning-based and domain-expertise-based methods were developed to help differentiate between permanent and temporary events observed on the grid. Examples of permanent failures include cable/conductor faults,

![Figure 9](image-url)

*Figure 9.* Graphical panel of an example Edge device showing near real-time system parameters/information.

Authorized licensed use limited to: IEEE Staff. Downloaded on September 07,2023 at 17:42:30 UTC from IEEE Xplore. Restrictions apply.
animal contacts, and equipment failures. Examples of temporary faults include vegetation management issues, lightning strikes, and switching transients.

**Synchrophasor Data Recovery for State Estimation**

Phasor measurement unit (PMU) data are sampled at synchronized instants, and measurements of nearby PMUs are correlated based on the power system topology. As a result, PMU data exhibit low-dimensional structure despite the high-dimensionality of raw data, as demonstrated by Patel et al. Specifically, the resulting matrix that contains measurements of nearby PMUs at multiple sampling instants is approximately low rank. Therefore, reconstructing missing PMU data can be formulated into a low-rank matrix completion problem as demonstrated by Candes et al. Different matrix completion methods such as nuclear, Hankel, and total variation norm minimization have been introduced and developed based on convex optimization technique to recover both randomly and temporally missing data with applications to synchrophasor (PMU) data recovery for state estimation.

**False Data Injection, Anomaly Detection, and Fault Location in Feeder Data**

False data injection attacks can be created to be unobservable by the commonly applied residual-based bad data detection. However, given the high sampling rate of PMUs, one would expect the PMU measurements to be highly correlated in time because the power system is unlikely to have dramatic changes in such a short time period under normal operating conditions. Therefore, a detector that takes into consideration the temporal correlations of the PMU measurements may be able to detect such “unobservable” false data injection attacks. A prediction method is developed based on the data completion techniques with applications to false data or anomaly detection in PMU data. A statistical inference method based on the random matrix theory technique has also been developed for anomaly detection and fault location based on the eigenvalue's distribution and eigenvectors of the sample covariance matrix following the Marcenko-Pastur law as demonstrated by author Qiu et al.

**Distribution Network Parameter Estimation**

An estimation tool based on the recursive least squares technique was developed for distribution network parameter estimation with applications to short-circuit detection, load flow management, dynamic rating, load modeling, fault detection, and network parameter estimation.

**Application Integration**

The data fusion and related applications were developed and tested on a PC but had not been implemented on an ECD. These applications were containerized like the FDIR application. The containers made it easy to deploy the applications across different hardware and operating systems. These were tested on ECDs to test and compare performance. The synchrophasor data recovery and false data injection applications were both originally developed in Matlab but were later translated to Python to simplify deployment. The two event classification applications were developed in Python and had no need to be translated. Each of the applications and their dependencies were packaged into a Docker image. A script was also written to build and run the applications initially. Another script was written to run each of these applications after the initialization script had stopped. These applications were then run on an MC-ECD and a HC-ECD.

The four applications were containerized and worked as containerized applications on the MC-ECD and HC-ECD. Each of the applications had results that were the same as when tested on a PC but had different execution timing. The MC-ECD had timing that was between five and ten times slower than the HC-ECD and six to ten times slower than the PC. The slower timing was anticipated due to hardware constraints and is acceptable. The HC-ECD had timing that was comparable to the PC, with timing that was in the range of 0.8–1.4 times the speed of the PC.

![Hardware setup for a demonstration with six ECD devices and containerized FDIR application.](image-url)
Hierarchical Grid Intelligence and Future Applications

While the edge computing concept presented in this article focuses on distributed intelligence, the concept of hierarchical grid intelligence will likely be the future of the grid’s protection, automation, and control architecture. The grid ECDs will provide distributed intelligence and fast response to time-critical grid issues (e.g., fault detection, isolation, and restoration). Communicating pertinent data to devices at the substation level (e.g., substation computers, high-end ECDs) will provide another layer of protection, automation, and control (PAC), and potentially a sanity check that the ECD-based control decisions made were correctly taken. Last, by interfacing and coordinating with the cloud and end customer application-based IoT systems, a comprehensive picture of the effects of grid events can be obtained. This level of communications enables the link to consumer and industrial and commercial sites from the utility and other systems. Advanced analytics are another application that can be applied at all levels of this framework to provide more hierarchical intelligence. Figure 11 shows this concept at a high level.

At the grid edge, basic to slightly advanced applications, such as fault detection, isolation, and restoration, can be deployed. The type of ECD (LC-ECD, MC-ECD, or HC-ECD) will determine the level of advanced application that can be deployed. At the substation and cloud levels, where computation power and capabilities are greater, more advanced applications can be deployed. For example, machine learning-based applications could be deployed at all levels, but the complexity of the technique and data models required will determine the deployment hardware platform. A simpler machine learning method could be deployed even on the LC-ECDs. A more complex method (e.g., deep/reinforcement learning based) would need to be deployed on a substation computer or even on the cloud. This flexible deployment architecture provides the basis for a future vision of a hierarchical grid intelligence framework, allowing for a dynamic protection, control, and automation system that can accommodate many new subsystems to the grid (e.g., DERs, energy storage).

Considering the architecture proposed previously, future applications to be deployed include site-specific (e.g., event analysis and detection) to broader system-level (e.g., DER monitoring and control) applications. Some general technologies, such as data fusion, machine learning/artificial intelligence, and 5G real-time communications that are being implemented in IoT-based solutions today will also have a significant impact on future product offerings. Data fusion

![Figure 11. Proposed edge computing architecture for power distribution applications.](image-url)
merges various data sources and types together to manipulate data and provide useful inputs to end applications. Machine learning techniques use supervised, unsupervised, and reinforcement learning methods to solve traditional and new distribution protection, automation, and control applications. The deployment of 5G and other advanced, real-time communication systems will enable a paradigm shift in distribution PAC functionality, allowing for the deployment of applications that could only be realized on paper 20 years ago.

“Virtualized” protection and control [aka “centralized protection and control” (CPC)] is another new technology that can leverage the edge computing and advanced communication infrastructure to provide enhanced hierarchical grid intelligence. A virtual platform entails the implementation of the PAC functions on a server (i.e., in software) that resides in the substation. The advantage of this platform is mainly the flexibility to provide enhanced PAC applications, analytics, and cybersecurity methods, on top of the existing virtualized PAC functions. For example, implementing PAC functions on a server instead of a substation computer allows for more advanced machine learning-based applications to be implemented on the platform. Furthermore, if 5G and IEC 61850 communications are utilized with the virtual platform, then real-time sampling at the field device level can be achieved, providing a comprehensive real-time view of the system, which leads to the implementation of more advanced and comprehensive PAC functions that span the distribution grid.

**Conclusions**

Based on the potential of edge computing and the results from experimental tests presented, it can be concluded that the proposed concept offers faster response to grid events than centralized or substation-based solutions, indicating that coordination with supervisory control and data acquisition, distribution management system and substation computers is feasible. Edge computing is a new wave of technology that can provide significant benefits to the utility distribution grid. Main applications of interest include distribution automation, overall grid management, advanced analytics, DER monitoring and control, and asset management. The merging of edge computing technology, ubiquitous communication medium and protocols, and advanced analytics will provide strong, distributed intelligence platforms, supporting the next generation of distribution automation and grid management products.

**Acknowledgments**

The information, data, or work presented herein was funded in part by the Advanced Research Projects Agency-Energy (ARPA-E), U.S. Department of Energy. The views and opinions of authors expressed herein do not necessarily state or reflect those of the U. S. Government or any agency thereof.

**For Further Reading**


**Biographies**

**James Stoupis** is with ABB U.S. Research Center, Raleigh, NC 27606 USA.

**Rostan Rodrigues** is with ABB U.S. Research Center, Raleigh, NC 27606 USA.

**Mohammad Razeghi-Jahromi** is with ABB U.S. Research Center, Raleigh, NC 27606 USA.

**Amanuel Melese** is with ABB U.S. Research Center, Raleigh, NC 27606 USA.

**Joemoan I. Xavier** is with ABB Distribution Solutions, Cary, NC 27511 USA.