Distributed Application Architecture and LinkNet Topology Processor for Distribution Networks Using the Common Information Model

ALEXANDER A. ANDERSON\(^1\), (Member, IEEE), ROBIN PODMORE\(^2\), (Life Fellow, IEEE), POORVA SHARMA\(^3\), (Member, IEEE), ANDREW P. REIMAN\(^1\), (Member, IEEE), ROHIT A. JINSIWALE\(^1\), (Member, IEEE), CRAIG H. ALLWARDT\(^1\), (Member, IEEE), AND GARY D. BLACK\(^3\)

\(^1\)Pacific Northwest National Laboratory, Energy Infrastructure and Buildings Division, Richland, WA 99352, USA
\(^2\)Incremental Systems Corporation, Issaquah, WA 98027, USA
\(^3\)Pacific Northwest National Laboratory, Advanced Computing Math and Data Division, Richland, WA 99352, USA

Corresponding author: Alexander A. Anderson (alexander.a.anderson@ieee.org)

This material is based on work supported by the U.S. Department of Energy (DOE), Advanced Grid Research Program, as part of the Grid Modernization Laboratory Consortium initiative. Pacific Northwest National Laboratory is operated for DOE by the Battelle Memorial Institute under Contract DE-AC05-76RL01830.

ABSTRACT Ongoing decarbonization efforts and increasing penetrations of distributed energy resources (DERs) may soon disrupt current operational paradigms, requiring a shift from centralized control and optimization to hierarchical and distributed applications. To support this transition, a distributed application architecture for communication and control of distribution network assets is introduced. The architecture defines a structure of communication and coordination layers based on the Laminar Coordination Framework. The challenge of defining distributed control areas is resolved through definition of switch-delimited topological areas based on the physical topology of the feeder instead of communications network layers. Implementation of the architecture in standards-based platforms is enabled by a robust graph-based topology processor leveraging the class structure of the Common Information Model (CIM), which differs significantly from traditional bus-branch representations by explicitly modeling the nodes and terminals of every type of power system equipment. The topology processor creates a set of LinkNet linked list structures for indexing the nodes and terminals of all CIM class object instances and mapping of the feeder topology in real-time. The data structures and algorithm are computationally lightweight, with performance tests presented for several IEEE distribution test systems.

INDEX TERMS Active distribution networks, common information model, distributed computing, micro-grids, network topology, smart grids.

I. INTRODUCTION

To reach decarbonization goals, a significantly higher level of renewables generation will be required. Many of these renewable resources will appear as distributed energy resources (DERs) connected throughout the distribution network. Furthermore, increasing penetration levels of large-scale renewables, customer rooftop solar, electric vehicles, battery storage, and other distributed energy resources will likely disrupt current operational paradigms [1], [2]. As a result, a new generation of advanced power applications will be needed to ensure reliable, resilient, robust, safe, and economic operations [3]. This new generation of applications [4], [5] will need to operate in a data-rich environment, leveraging multiple sources of data not available to current energy management system (EMS) and advanced distribution management system (ADMS) applications.

To meet the challenges of operations in a data-rich environment with high DER penetration, new advanced power applications will likely need to operate in a distributed...
manner [6], [7]. Introduction of a distributed architecture provides several benefits including increased resilience, scalability, responsiveness, and objective convergence [8]. Distributed applications can continue to operate during partial communications outages, fragmented grid conditions, and incremental restoration of the distribution network. Additionally, distributed architectures enable grid-edge agents to participate in and receive benefit from coordination and control, promoting democratization of the power system operations. Further, entities can limit information shared across ownership boundaries and topological areas to provide compartmentalization of data and control.

This paper introduces a novel approach to enabling distributed architectures by addressing three key prerequisites:

- A method to divide a distribution feeder into distinct areas of responsibility
- A consensus-based vocabulary so that any application, service, or software platform can interpret the exchanged data without any custom adapters or data translation
- Accurate knowledge of the network topology in real-time using a service that is sufficiently lightweight to run on substation-level hardware with limited computing resources

The first requirement of creating distinct distributed areas is fulfilled in this work by the creation of static compartmentalized regions of the feeder [6] based on the network topology and the concept of switch-delimited topological areas. A robust, flexible, scalable distributed application architecture based on the Laminar Coordination Framework for power systems [9], [10]. The architecture presented was developed in close collaboration with a working group composed of key stakeholders from utilities, industry, ADMS vendors, and researchers from universities and national laboratories.

The second requirement is satisfied through use of the Common Information Model (CIM) for representation of the power system network model and all messages exchanged between distributed agents [11]. Traditionally, use of the CIM introduces a significant amount of computational overhead due to the object-oriented representation and the fact that information associated with a single bus is spread across a multitude of separate classes and attributes [12]. This paper provides the first detailed explanation of representation of common distribution feeder equipment configurations understandable to a power application developer without extensive knowledge of UML modeling and data science.

The third requirement is satisfied by the introduction of a robust, lightweight topology processor leveraging the node-terminal structure of CIM objects. Several CIM-based topology processing algorithms are available in the literature for transmission [12], [13], [14], [15], [16], [17] transmission [12], [13], [14], [15], [16], [17] and balanced distribution networks [18], [19], [20], [21], [22], [18], [19], [20], [21], [22]. However, none of the existing methodologies explicitly handle split-phase transformers and unbalanced two-phase low-voltage secondary networks commonly found in North America. Additionally, none of the previous works provide a method to partition a feeder into discrete topological areas for compartmentalization of distributed applications. This paper also introduces the use of LinkNet linked lists to create a graph structure based on the classes and associations of the Common Information Model. The data structure enables rapid topology processing to provide not only the switch-delimited topological areas of the introduced distributed architecture, but also real-time information about the topology of feeders and intentional islands for both meshed and radial unbalanced networks.

The remainder of this paper is organized as follows. Section 2 introduces the distributed application architecture. Section 3 delineates representation of distribution system equipment using CIM. Section 4 presents the LinkNet linked list graph structure. Section 5 outlines the topology processing algorithm for identifying feeders, islands, and switch-delimited topological areas. Section 6 describes implementation in the GridAPPS-D platform. Section 7 concludes.

II. DISTRIBUTED APPLICATION ARCHITECTURE
A. PRINCIPLES OF GRID ARCHITECTURE

The concept of grid architecture is derived from the systems engineering domain and extends control theory, network theory, and systems architecture to define the structure and interactions of key stakeholders in planning, operations, and regulation of power systems [9], [23]. More broadly, a system architecture defines the structure, behavior, limits, and interactions within a system. Some of the principles of a good architecture are that it needs to meet the needs of stakeholders, be cognizant of the global system when optimizing subsystems, and divide the system such that component responsibilities do not overlap or conflict [9]. Additionally, the architecture must recognize that power systems are not simply an electric circuit, but rather a network of structures involving control, coordination, communications, sensing, and data management.

Future grid operations will likely include a combination of centralized, distributed, and decentralized operations with coordination across utility and non-utility participants. Focus here will be given to distributed systems, in which the separate (often geographically dispersed) elements of a system cooperate to solve a common problem. The Laminar Coordination Framework presented by [9] is adapted to enable local optimization schemes within a larger global coordination framework, with key characteristics including:

- Boundary deference, where system and organization boundaries are explicitly defined and recognized
- Control federation so that competing or conflicting control objectives are resolved without ambiguity
- Control disaggregation, where control commands are decomposed for local consumption
- Scalability of coordination data traffic with increasingly large power system networks
• Local “selfish” optimization, where local agents seek to achieve local objectives while coordinating to create a globally optimal solution.

B. DEFINITION OF PHYSICAL LEVEL LAYERS

Utility distribution networks are defined in terms of individual feeders, which may be topologically radial or meshed. Each feeder is associated with a single substation from which it is normally energized (with supply from alternative substations defined operationally as an abnormal switching configuration). Within a single distribution feeder, one possible implementation of the distributed architecture is to define three layers at the physical level for sorting and aggregating equipment, customers, and measurement devices. To implement the concept of boundary deference, each physical layer is defined in terms of particular set of equipment boundaries that do not change, except for addition of new equipment or cuts/jumpers made by line crews.

The boundaries for the highest-level layer of this implementation, the Feeder Area, are the substation high-voltage transformer on one side and first downstream switching device on the other side. This layer contains all medium voltage (MV) substation equipment and sensors, as well as all switching devices within the feeder. Physical assets are categorized as addressable equipment capable of receiving control messages (including reclosers, sectionalizers, voltage regulators, and capacitor banks) and unaddressable equipment (such as bus bars, transformer tanks, and sensors).

The next level of this implementation is the Switch Area, which contains all equipment in a section of feeder that is bounded by a set of one or more switching devices. Examples of Switch Areas include:

• A microgrid connected through a single point of common coupling (PCC)
• A section of feeder between an upstream recloser and a downstream sectionalizer
• A single-phase lateral downstream of a line fuse

Addressable equipment contained in a Switch Area include poletop regulators, poletop capacitors, and DERs connected at the MV level. Unaddressable equipment contained at this level include overhead lines, underground cables, customer secondary transformers, and measurement devices connected to MV equipment.

The lowest-level layer of this implementation is the Secondary Area, which is bounded by the secondary service transformer and contains all low voltage (LV) equipment served by that transformer. Examples of addressable secondary equipment include smart inverters, distributed generators, energy storage systems, advanced metering infrastructure (AMI) assets, and customer devices controllable through demand response initiatives. Triplex service lines, passive loads, sensors, and all other low-voltage equipment not capable of receiving control commands are categorized as unaddressable equipment.

In Figure 1, a small section of the 9500 Node Test System [24], [25] is illustrated with representative examples of a Feeder Area, Switch-delimited Area, and Secondary Area.

C. DEFINITION OF COMMUNICATION LEVEL LAYERS

Communications, control, and coordination within the introduced distributed application architecture are divided into three levels of abstraction. The lowest level is communications infrastructure, which is composed of hardware and frequently bandwidth-constrained in power distribution networks.

The middle layer consists of operations technology (OT) message buses, which serve as a virtual partition to provide compartmentalization of data and control to enable control federation and control disaggregation. The distributed message buses are based on the concept of the CIM Enterprise Service Bus defined in the IEC 61968-1 standard [26]. Within CIM interface reference model, there exists a shared message bus across which all CIM-compliant applications and services communicate using standards-based messages exchanged through query-response, publish-subscribe, or other communication paradigms. The concept of the shared message bus is expanded in a distributed manner to mirror the set of physical layers within the distribution network, as shown in
A. A. Anderson et al.: Distributed Application Architecture and LinkNet Topology Processor

The highest level of abstraction contains distributed applications and distributed agents, which are created in software to perform optimization, control, and coordination tasks for a specific distributed area.

D. MAPPING OF DISTRIBUTED AGENTS TO PHYSICAL LEVEL LAYERS

To provide consistent boundary deference between the communications layer and the physical layer, message buses are defined for each distributed area within the distribution network. The message bus for each area is responsible for handling all measurement data streams, control messages, and generic queries made by all distributed agents within that area.

At the highest level is the Distribution System message bus, which handles messages exchanged between feeders and across the entire service territory of a utility. This layer also hosts the global Coordinating Agent, which manages coordination between feeder-level distributed agents.

At the second level is the Feeder message bus, which handles coordination messages between distributed agents inside lower-level Switch-delimited Areas. It also distributes control commands made to controllable switches in the feeder and substation equipment, as well as aggregated measurements from all devices within the feeder.

At the third level is the Switch Area message bus, which handles messages between devices, agents, and distributed applications within a single Switch-delimited Area. This includes control commands to poletop equipment and addressable devices connected at the MV level. Message buses at this level also deliver coordination messages between distributed agents in contained Secondary Areas, as well as measurements for poletop sensors and aggregated data from lower-level measurement devices.

Finally, at the fourth level is the Secondary Area message bus, which handles control and communication between low-voltage devices downstream of a secondary transformer.

One of the key enablers of data compartmentalization is that a device within a particular distributed area can only publish and subscribe to the single message bus for that area. It cannot directly send messages to equipment outside its area boundary, nor cannot it obtain measurements from sensors in adjacent or higher-level areas. Coordination messages between individual areas are passed to message bus at the next higher level, which then directs the message to its correct destination.

E. CONTEXT MANAGER AND FIELD BUS MANAGER

Coordination of messages and mapping of distributed applications to the correct message bus is performed by the Context Manager and Field Bus Manager. Communications and data exchange between the managers and distributed applications are shown for two possible configurations in Figure 3. Reference implementation of these managers in the GridAPPS-D platform is outlined in [7].

The responsibility of the Context Manager is to obtain topology information from the CIM topology processor and create unique identifiers for the Feeder message bus, Switch Area message buses, and Secondary Area message buses. The unique identifiers for each area as based on the master resource identifier (mRID) of the feeder are subsequently used to map each distributed agent to the correct message bus. The Context Manager also maps sensors and metering devices to each topological area to ensure that measurement data streams are published onto the correct message bus for that distributed area. It also responsible for passing control commands issued by centralized apps running in a utility.

Figure 2 and discussed in the next subsection. The distributed message buses are virtual constructs in software and may persist through partial communication outages. Security and permissions are supported through the definition of the Context Manager and Field Bus Manager [7].

The distributed agents communicate, receive measurements, and publish control messages through the single message bus serving their distributed topological area.
control room to the correct field bus (which in turn, publishes it to the distributed device that is subscribed to its local message bus). New distributed agents that are added to local area are registered by the Context Manager and allowed to communicate on the correct field bus.

The Field Bus Manager provides application programming interface (API) functions and has full knowledge of the equipment and measurements within its distributed switch area from the CIM topology processor. Requests for network information and feeder model data from distributed applications are received through the Context Manager, which triggers the Field Bus Manager to obtain the desired set of CIM class instances and attribute values. Additional interfaces provided by the Field Bus Manager allow distributed applications to

- read latest measurement data for monitoring
- send control commands to change device attributes
- provide connected device list and location context for situational awareness and asset management

III. THE COMMON INFORMATION MODEL

The Common Information Model (CIM) [27] is a Unified Modeling Language (UML) specification used to describe an electrical network and the various equipment used on the network. CIM is widely used for data exchange of bulk transmission power systems and is now beginning to find increasing use for distribution modeling and analysis. CIM is also a canonical information model that provides a formal representation of objects, their attributes, the relationships between them, and the operations that can be performed on them. It is a technology-agnostic specification model for describing the properties of physical power system equipment, power flow data, and messages that can be serialized and exchanged between various platforms and applications. It serves as a semantic reference model describing the majority of utility business objects, power system network model components, physical asset data, operations procedures, and transactive market mechanisms [11].

CIM is freely available to use under an Apache 2.0 license and is maintained by the CIM Users Group (CIMug), which was formed in 2005 as a subgroup of the UCA International Users Group (UCAiug). The UCAiug collaborates with the IEC and other standards communities for the development of technical and informative specifications. The CIM is divided into three separate information models that cover different aspects, centered around a single unified model for the entire power system network. Each portion of the information model is aligned with the series of technical standards published by IEC:

- The IEC 61970 information model describes electrical connectivity and characteristics of the power system network model, including generation, loads, topology, overhead lines, underground cables, transformers, voltage control equipment, etc.
- The IEC 61968 model describes asset information (such as transformer impedances) and utility business functions, such as asset health monitoring, work orders, clearances, system planning, and customer metering/billing.
- The IEC 62325 model focuses on electricity market processes, such as transmission capacity allocation, load forecasting, generation bidding, and clearing of market auctions.

CIM classes and their attributes can be distinguished through a capitalization convention that classes and packages use InitialCapitals, while attributes and enumerations use camelCase. This paper will use consolas font to highlight CIM class and attributes.

A. EQUIPMENT CONNECTIVITY REPRESENTATION

Unlike the bus-branch representation of most power flow solvers, power system network data specified using CIM explicitly describes the equipment connectivity and topology through a set of classes, including ConnectivityNode, TopologicalNode, and ACDCTerminal. Although this representation seems more complex and is sometimes reduced [17], the set of classes create an inherent node-edge graph structure with the ConnectivityNode objects forming the nodes and the Terminal objects forming the edges.

However, before introducing the graph structure, it is important to discuss the topological configuration of common types of power systems equipment found in distribution networks (such as split-phase customer transformers) and how
such equipment types are represented using the Common
Information Model.

1) ONE-TERMENAL EQUIPMENT TYPES
The first group of equipment to be discussed are shunt
elements (i.e. devices drawing or injecting real or reac-
tor power into the network). All of these devices are
modeled using a single Terminal object associated with
the ConnectivityNode to which it is connected. The
first category are generation sources, which may be repre-
sented using the SynchronousMachine class for conven-
tional generators, AsynchronousMachine for Type I/II
wind turbines, and PowerElectronicsConnection
for inverter-based generation and energy storage. The
next group is shunt capacitors and reactors, which are
both represented using the ShuntCompensator class.
The LinearShuntCompensator class is used for
banks of reactive equipment with equal admittance val-
ues in each section. Both transmission-level loads and
individual household customers are represented using the
EnergyConsumer class and are connected by a single
Terminal to the associated ConnectivityNode. Load
modeling will be discussed more in part D of this section.
Finally, sub-transmission buses (mathematically modeled as
infinite buses or swing buses) or other current injections are
modeled as an EnergySource object with one terminal
associated with the source node. In distribution networks, the
EnergySource is typically associated with the high-side
node of the substation transformer or one node of the feeder
breaker.

2) TWO-TERMENAL EQUIPMENT TYPES
The majority of switch and branch objects (except trans-
formers) are represented as two-terminal objects with a
single ConnectivityNode and single Terminal at
either end, as shown in Figure 5. This representation is
used for overhead lines and underground cables (modeled as
ACLineSegment objects) as well as switches defined
using the Breaker, Recloser, Sectionaliser,
LoadBreakSwitch, and Fuse classes. Likewise, series
capacitors and reactors (which are typically only found in
transmission networks) are also two-terminal elements, mod-
eled using the SeriesCompensator class.

3) REPRESENTATION OF TRANSFORMERS
The CIM uses two classes to describe transformers. The
first, PowerTransformer, is typically used to describe
three-phase transformers used in substations and balanced
networks. The second, TransformerTank, describes the
physical tank containing the windings and is used to represent
single-phase equipment, such as voltage regulators, customer
service transformers, or any other case where each phase
winding is physically contained within a separate tank.
The physical connection point of each winding is repre-
sented by a TransformerEnd object, which is associated
with a Terminal and then with a ConnectivityNode.
Common transformer configurations found in North Amer-
ican distribution systems are depicted in Figure 6. Two-winding transformers commonly used for substi-
ations and three-phase loads are represented with a single
ConnectivityNode and single Terminal at each wind-
ing. Three-winding substation transformers are represented in
a similar manner, with a total of three Terminal objects and
three ConnectivityNode objects defined to represent
connectivity of other equipment to the three transformer
terminals with three different voltage ratings.

Representation of split-phase customer secondary trans-
former commonly used in North America was introduced in [28] and uses a total of three Terminal objects but
only two ConnectivityNode objects, as depicted in
Figure 6(c). The high-side winding is typically connected
only to a single conductor phase and is represented by a
single ConnectivityNode and single Terminal. The
two low-side windings are phased 180° apart to create a line-
to-line voltage (typically 240V) of twice the line-to-neutral
(typically 120V). Customers are then served from two-phase
secondary triplex lines running from the transformer to the
meter. Each transformer low-side winding is assigned a Ter-
minal object, but both Terminal objects are associated with
only a single ConnectivityNode because they represent
individual phases at the same nominal voltage.

The last common transformer configuration is that of the
single-phase voltage regulator, which are typically arranged
in banks of three to provide independent regulation of
all three phases Each voltage regulator is modeled as
A separate TransformerTank with a single Terminal on each winding. The three separate A-, B-, and C-phase Terminal objects from each regulator winding are then associated with a single ConnectivityNode, as shown in Figure 6(d).

At the topological level, CIM does not distinguish between the winding configurations (wye-wye, wye-delta, etc.) of the transformer. Description of the windings on each end of the transformer is handled by the connectionKind attribute of the TransformerEnd object, rather than by any attributes of the nodes or terminals to which the transformer is connected.

4) REPRESENTATION OF CUSTOMER LOADS

Customer loads can be represented at three levels of detail depending on the requirements of the modeling effort. The simplest method places an aggregated EnergyConsumer object at the terminal where the high-side of the service transformer would be connected, as depicted in Figure 7(a). A single Terminal object is from the EnergyConsumer is associated with the ConnectivityNode to which it is connected. The next level of detail includes the customer transformer as a simple two-winding PowerTransformer object with a single Terminal associated with the Energy-Consumer object and the ConnectivityNode to which the transformer low-side winding is connected, as depicted in Figure 7(b). The first two representations can be used for both single-phase and three-phase loads.

The most detailed representation models the split-phase secondary transformer, two-phase triplex lines, and individual unbalanced single-phase customer loads. For the sake of visual simplicity, this configuration is depicted for only a single customer in Figure 7(c), but multiple customers are often connected to the low-side of the transformer, each with a length of overhead or underground secondary triplex line. As discussed earlier, the split-phase TransformerTank is topologically represented by three Terminal and two ConnectivityNode objects; the triplex ALineSegment uses two Terminal objects. The EnergyConsumer uses a single Terminal connected to the ConnectivityNode at the end of the triplex line.

IV. LINKNET GRAPH STRUCTURE

LinkNet is a lightweight graph structure that uses a set of linked lists for representing power system models first introduced in [29] and [30] to process the IEEE 118 bus model on an IBM 360/44 mainframe with 128 kB of RAM. It also forms the core of the OpenPA open-source power application development environment for transmission grid simulation [31], [32]. The low computational burden and inherent alignment with the CIM topology structure make LinkNet an excellent candidate for topology processing on local distributed agents running on microcontrollers or other hardware with limited resources. Several additions and modifications to the original LinkNet structure are introduced in this section to adapt it for distribution networks with new equipment types, such as grid-forming inverters.

LinkNet uses a set of linked lists to represent the topology and connectivity of equipment within the power system network as a graph network. The nodes and edges originally described in [29] and [30] directly correspond to the ConnectivityNode and Terminal objects in CIM, and thus the CIM object names will be used throughout this paper.

The topology of power system network is represented by three linked lists whose indices are used to map the connectivity of equipment. The linked lists track the relationships between the ConnectivityNode and Terminal objects in three different directions. The lists are created the first time the power system network model is parsed using database queries for line, transformer, and shunt equipment objects. The equipment objects do not need to be parsed in any specific order. The ordering of equipment and their connectivity is handled by the indexing of the linked lists.

The graph structure can be implemented as lists, vectors, arrays, dictionaries, or JSON strings depending on the programming language selected. For clarity, the term “structure” will be used to describe each linked list without usage of any language-specific terminology.

A modified IEEE 13 node model is depicted in Figure 8 with a meshed topology and additional substation source. This model will be used as a running example for explaining the LinkNet graph structure. Each electrical bus in the feeder is associated with a ConnectivityNode. While each branch object is parsed in the manner described in Section VII below, the ConnectivityNode objects are assigned sequential index values of 1 through 24 and the Terminal objects are assigned sequential index values of 1 through 50.

A. “LAST” STRUCTURE

The “last” structure contains the set of mappings between a ConnectivityNode and the last Terminal object associated with that ConnectivityNode. The length of the “last” structure is equal to the total number of
LinkNet Graph Structure:

\[ \text{last}(CN) = T \rightarrow \text{Last parsed Terminal that is associated with a ConnectivityNode} \]

\[ \text{next}(T) = T \rightarrow \text{Next Terminal connected to the same ConnectivityNode} \]

\[ \text{far}(T) = CN \rightarrow \text{ConnectivityNode at the far end of branch from the Terminal} \]

**Example Topology Sweep using LinkNet:**

Start at bus 633 / node CN2

\[ \text{last}(CN2) = T43 \rightarrow \text{go to Terminal T43} \]

\[ \text{next}(T43) = T23 \rightarrow \text{far}(T23) = CN23 \rightarrow \text{adj. to bus 634} \]

\[ \text{next}(T23) = T2 \rightarrow \text{far}(T2) = CN14 \rightarrow \text{adj. to bus 635} \]

\[ \text{next}(T2) = 0 \rightarrow \text{end of trace} \rightarrow \text{no more adjacent buses} \]
ConnectivityNode objects in the network. In certain programming languages, the structure may be pre-allocated to provide improved performance.

The index of each entry in the “last” structure corresponds to the order in which ConnectivityNode objects were parsed. Thus, the first entry in the “last” structure corresponds to the first ConnectivityNode parsed, the second entry in the “last” corresponds to the second ConnectivityNode parsed, and so on. Note that the first ConnectivityNode parsed does not have to be the slack bus, substation head, or any other specific node. Typically, it is just the first node of the first branch object returned by the database query, and a result can be anywhere in the network.

The value of each entry in the “last” structure is a numerical value that specifies the index of the last Terminal object associated with that ConnectivityNode. The values of the structure entries are updated recursively as each branch or shunt device is identified. Thus, when a new device is discovered and parsed from the database query with the \( n^{th} \) new Terminal, the value of the “last” entry for that ConnectivityNode is updated with the numerical value \( n \). The value of the “last” entry is also updated when merging TopologicalNodes associated with a closed switch.

For example, consider bus 645 in Figure 8 corresponding to ConnectivityNode CN4. It has two Terminal objects connected to two ALineSegment objects from T5 to T6 and from T7 to T8. The last Terminal object that was parsed from the database query was T7, and so last (CN4) = T7.

B. “next” STRUCTURE

The “next” structure contains the set of mappings between a Terminal object and the next Terminal object associated with the same ConnectivityNode. The length of the “next” structure is equal to the total number of Terminal objects in the network. For an arbitrary power system network with \( t_1 \) shunt devices, \( t_2 \) two-terminal branches, and \( t_3 \) three-terminal devices (such as three-winding and split-phase customers transformers), the “next” structure can be pre-allocated with an approximate length of \( t_1 + 2t_2 + 3t_3 \) entries.

The index of each entry in the “next” structure corresponds to the order in which each new Terminal object was parsed. For example, the 20\(^{th} \) Terminal identified from the database query would correspond to the 20\(^{th} \) index position in the “next” structure. The value of that entry corresponds to the index value of the next Terminal.

Consider the modified IEEE 13 bus test feeder depicted in Figure 8. Bus 633 corresponds to ConnectivityNode CN2, which is associated with Terminals numbered T43, T34, and T32. The associated entries in the “next” structure would be next (T43) = 23, next (23) = 2, and next (2) = 0. The null value is used to indicate that there are no more Terminal objects associated with that ConnectivityNode.

C. “far” STRUCTURE

1) MULTI-TERMINAL BRANCH OBJECTS

The “far” structure contains the set of mappings between a Terminal and the opposite ConnectivityNode on the far side of a branch. It provides a method for traversing the power system network by moving from the Terminal of one ConnectivityNode to the next adjacent ConnectivityNode. The length of the “far” structure is the same as the “next” structure and is equal to the total number of Terminal objects in the network.

The “far” structure uses the same indexing as the “next” structure such that the \( n^{th} \) Terminal parsed corresponds to the \( n^{th} \) index position in the structure. The value of each entry corresponds to the index position of the opposite ConnectivityNode on the other side of the branch.

If there are multiple parallel branches between the two ConnectivityNodes (e.g., parallel phase conductors or a bank of single-phase voltage regulators), then multiple Terminal objects will have the same “far” value. In Figure 8, voltage regulator REG2 in the bottom of the feeder contains three Terminal objects associated with each ConnectivityNode for each single-phase TransformerTank. All three terminals have the same “far” value, with far (T38) = far (T40) = far (T42) = CN20. Likewise, on the opposite side of the voltage regulator bank, far (T37) = far (T39) = far (T41) = CN21.

Split-phase customer transformers have a similar duplication of far values, but only on the low-side. The high-side of the transformer has only a single winding and Terminal. The low-side of the transformer has two phase windings with two Terminal objects associated with a single ConnectivityNode. For the transformer named TPOLETOP in Figure 8, far (T45) = CN24 on the high-side, and far (T46) = far (T47) = CN3 on the low-side.

2) ONE-TERMINAL GENERATOR AND INVERTER OBJECTS

The “far” structure can also be used to index generator, inverter, and other one-terminal devices. Although indexing of these devices is optional, it is frequently useful to include the associated Terminals in the LinkNet graph structure. This enables these objects to be used as the root node for building island spanning trees using the process to be described in Section VII.

Unlike branch objects (where “far” corresponds to the opposite end of the branch), on-terminals are mapped back to the same ConnectivityNode to which they are connected. In Figure 8, the model contains one SynchronousMachine at Bus 636 and two PowerElectronicsConnection inverter objects at buses 634 and HOUSE. The “far” structure is used to refer to the Terminal objects of each one-terminal device back to the associated ConnectivityNode to which it is connected. As a result, far (T48) = CN15, far (T49) = CN24, and far (T50) = CN23.
D. "NODE" AND "TERM" STRUCTURES

Two additional structures may be required if a Python dictionary or similar data format is used to organize the "last", "next", and "far" structures using the unique mRIDs of each ConnectivityNode and Terminal object. These dictionaries are formatted as key-value pairs with the mRID as the key and the numerical node index as the value, such as node["CN56 - mRID"] = 56 and term["T23 - mRID"] = 23. These structures are created dynamically during the process of parsing the power system model from the database query.

V. LINKNET TOPOLOGY PROCESSOR

A. STRUCTURE AND CIM INTEGRATION

The LinkNet topology processor algorithm is designed to serve an underlying core service within a CIM-compliant, standards-based platform environment, such as the GridAPPS-D platform [3], [28], [33]. The structure of the topology processor service is depicted in Figure 9 and will be explained in detail in the subsections below.

Implementation within such a CIM-compliant platform reduces the number of directions in which data needs to be exchanged by enabling shared services to be utilized by all the integrated applications. This approach not only eliminates the need for multiple application to build their own topology processor, but also reduces the number of interfaces between individual application and shared services. Each application only needs a single interface with the standards-based platform and is then able to pass standardized API messages to request the desired information from the topology processor.

B. CREATION OF LINKED LISTS

The first portion of the topology processor parses query results for the power system model and builds the set of linked lists to form the LinkNet graph structure. The database queries are structured to return the mRIDs of each equipment object as well as those of associated ConnectivityNode and Terminal objects. Usage of TopologicalNode objects is optional. No other information from the CIM is required to build the LinkNet graph structure and network topology.

The process for building the initial set of linked lists is depicted in Figure 10. This process is implemented as a single short script that is agnostic to the type of power system equipment or class of CIM object. All common classes (such as ALineSegment, PowerTransformer, TransformerTank, SynchronousMachine, etc.) can be handled with same indexing scheme. Power system equipment objects are parsed sequentially with no requirement for the query results to be returned in any particular order.

For each new equipment definition, the topology processor first checks if the ConnectivityNode objects have been parsed and assigned an index value in the "node" structure.
If not, the mRID is recorded and the corresponding entry in the “last” structure is set to zero indicating that no terminals have yet been assigned to that ConnectivityNode. Next, each terminal associated that piece of equipment is parsed and added to the “term” structure.

Creation of the linked list structure does not depend on the order in which various types of equipment are parsed. Each object is processed as shown in Figure 10. For branch objects, the indexing process does not depend on either the order in which CIM objects are returned or the from-to ordering of nodes/terminals at either end of a branch. Each Terminal is indexed sequentially and mapped to the correct ConnectivityNode using the “last”, “next”, and “far” structures. Differentiation between single-terminal (shunt) devices and multi-terminal (branch) devices is handled by the “far” structure as described earlier. Only the minimum set of classes and objects necessary for the particular application need to be indexed. Thus, if a particular application was not concerned by the mapping of individual EnergyConsumer loads within the feeder, then it is not necessary to add the terminals of those objects to the LinkNet graph structure.

It should be noted that none of the switching devices of any CIM class (including Breaker, Fuse, LoadBreakSwitch, Recloser, Sectionaliser, and Switch) are indexed in the linked list structure. When building the “last”, “next”, and “far” structures, all switches are treated as open. Closed switches are then processed by merging the TopologicalNode objects and updating the “last” and “next” values for the ConnectivityNode objects on either side of the switch, as described below.

C. CREATION OF DISTRIBUTED TOPOLOGICAL AREAS

To support identification of distributed control areas for use by the Context Manager, the topology processor builds a set of breadth-first spanning trees to map all ConnectivityNode objects in each switch-delimited area. The algorithm iterates through the complete list of all switch objects of all CIM classes and builds a spanning tree from the second Terminal of each switch using the algorithm shown in Figure 11. The topology is built without merging any closed switches so that each switch-delimited area appears as a topological island.

The spanning tree algorithm uses a breadth-first approach using the “last”, “next”, and “far” structures to move from a given ConnectivityNode (indexed using the FirstNode counter) to all adjacent nodes. The last node that was added to the spanning tree is indexed using the LastNode counter. When the two indices are equal, all nodes within a particular topological island have been parsed, and the spanning tree is complete.

The algorithm uses an outer loop (inside the dashed box of Figure 11) uses the “last” structure to obtain the index of last Terminal associated the current ConnectivityNode. That Terminal index value is passed to the inner loop (inside the dotted box of Figure 11) which first uses the “far” structure to obtain the adjacent ConnectivityNode at the far end of branch, which is added to the spanning tree if not already there. The “next” structure provides the index of the next Terminal connected to the starting node. The inner loop continues until the “next” structure reaches an index of 0, indicating there are no more branches and triggering the outer loop to move to the next ConnectivityNode in the tree. The approach is identical to the graphical trace that was illustrated earlier in Figure 8, in which all adjacent nodes to bus 633 were identified by iterating through the table of “last”, “next”, and “far” values for a modified IEEE 13 node model.

D. MERGING OF CLOSED SWITCHES

To identify all the feeders and topological islands in a topological island at a given time, it is first necessary to identify all closed switches and update the linked list structure. After all branch and shunt elements (excluding switches of all class types) have been indexed in the linked list structure, the topology processor parses the positions of all switch objects using the logic shown in Figure 12.
FIGURE 12. Process for merging topological nodes and updated linked list indices for closed switches in a distribution feeder.

Instead of indexing switches in a manner similar to other branches, the “next” structure of the ConnectivityNode objects on either side of the switch are combined. Likewise, the “last” structure is updated such that both nodes point to the same Terminal (which is selected as the one with the higher index value). As a result, the full set of Terminal objects associated with both nodes can be traced from either side of the switch. The result was illustrated graphically earlier in Figure 8 such that both last(CN2) = T43 and last(CN22) = T43. Likewise, next(T43) = T23 reaching back across the closed fuse to complete the full set of Terminal objects associated with both nodes.

E. CREATION OF FEEDER AND ISLAND TREES

Identification of individual topologically separate feeders and islands within a network model is performed using the same breadth-first spanning tree algorithm described earlier. Grid-connected distribution feeders are identified by taking the low-side bus of the substation transformer as root node to build each spanning tree. Islanded microgrids are identified by taking the node of distributed generators and grid-forming inverters as the root for building each spanning tree. If the feeder topology is meshed, with a path between different substations (a common intermediate topology during load transfer actions), the set of meshed nodes are listed as energized from both substations.

VI. IMPLEMENTATION IN GRIDAPPS-D PLATFORM

A. GRIDAPPS-D PLATFORM AND ARCHITECTURE

The GridAPPS-D Platform was developed as a first reference implementation of an open-source data-rich platform for distribution system data integration based on CIM representation of data models, programming interfaces, and data exchange interfaces [3], [28], [33]. The GridAPPS-D platform provides industry, researchers, and utilities a well-documented roadmap to developing CIM-compliant applications, services, and data integration bridges to demonstrate new grid functionalities and use cases. CIM-compliant messaging is utilized between all grid devices in the field, distributed applications in utility systems, and centralized applications that are integrated within the platform environment. Through use of the CIM and other industry standards (such as DNP3, IEEE 2030.5, and OpenFMB), it is intended that the application development environment is independent of any specific proprietary data formats and capable of interacting with existing vendor ADMS environments. It should be noted that the GridAPPS-D platform is not intended to serve as a replacement for industry ADMS tools or as a production-grade control room software suite, but rather as a flexible, open-source sandbox for co-simulation and development of new tools and concepts that are not feasible without a data-rich environment and standards-based interfaces between portable applications and reusable services.

The GridAPPS-D Platform architecture (illustrated in Figure 13) provides a layered structure for data integration based on the concept of the enterprise message bus introduced in IEC std 61968-1:2020 [26] and communication paradigms described in IEC std. 61968-100:2022 [34]. Data from multiple disparate sources (such as SCADA measurements, weather info, and smart meter data) are aggregated and published to a CIM-based message bus. These data streams are mapped to the terminals of associated power system equipment through a set of unique measurement mRIDs defined for each measurement ingested by the platform. The ingested measurements, power system network model, and associations are contained in a set of databases structured around the unique mRIDs and directional relationships between the set of CIM classes and attributes used. CIM-compliant advanced power applications can then pass request/reply queries through the message bus to the CIM-based data stores and shared services to obtain network model information, historical data, snapshot power flow results, and other static data. Applications can also subscribe to streaming data, which are published in real time as CIM-compliant messages on the message bus. Shared services and applications within the platform can publish real-time information (such as the current network topology or latest Y-bus matrix) or publish CIM-compliant equipment control commands through the message bus. These messages can then be converted to the format of any specific communications protocol (such as DNP3, IEC61850, IEEE 1547, etc.) and sent to physical equipment controllers.
### TABLE 1. Performance benchmark for IEEE distribution test systems.

| Process               | CIM Class            | IEEE 13 Node | IEEE 123 Node | IEEE 8500 Node | IEEE 9500 Node |
|-----------------------|----------------------|--------------|---------------|----------------|----------------|
| Database Query        | ConnectivityNode     | 16           | 130           | 4876           | 5294           |
| Database Query        | Terminal/Measurement | 109          | 782           | 19509          | 22278          |
| LinkNet Indexing      | ALineSegment         | 11           | 118           | 3661           | 3909           |
| LinkNet Indexing      | TransformerTank      | 5            | 6             | 1189           | 1293           |
| LinkNet Indexing      | PowerElectronicsConnection | 0          | 0             | 0              | 180            |
| Switch Merge          | LoadBreakSwitch      | 1            | 8             | 38             | 107            |
| Feeder Tree #1        | ConnectivityNode     | 16           | 130           | 4875           | 573            |
| Feeder Tree #2        | ConnectivityNode     |              |               |                |                |
| Feeder Tree #3        | ConnectivityNode     |              |               |                |                |

#### B. REAL-TIME TOPOLOGY SERVICE

The LinkNet topology processor algorithm described in the previous section is implemented as a real-time ADMS service in the GridAPPS-D platform to provide topology information based on real-time switch positions from streaming simulation data or SCADA measurements. During initialization, the service performs the required database queries and builds the initial LinkNet structure. The service subscribes to the Feeder message bus and receives timestamped position measurements for all switch objects in the feeder. If the service detects that a switch has opened or closed, the topology processor updates the linked list indices using the process illustrated previously in Figure 12. The updated linked lists are used to create a new set of feeder and island spanning trees, which are published onto the service output topic.

#### C. BACKGROUND TOPOLOGY Daemon

A parallel implementation of the topology processor algorithm is created in the platform as a background daemon that is used to inform the Context Manager and Field Bus Manager during initialization and assignment of distributed agents to the correct set of topological areas. The daemon communicates using a query-response paradigm using API calls that return:

- the set of switch-delimited topological areas
- the set of feeder and island trees for a historical timestamp (from the Timeseries Database or specified data historian)
- the set of feeder and island trees for the normally open/closed switch positions specified in the CIM model

The topology processor queries the CIM network model database, creates the LinkNet structure, and generates the requested response, which is returned via the API as a JavaScript Object Notation (JSON) message.

#### D. PRELIMINARY BENCHMARKING

To represent the limited hardware capabilities of most substation devices, the topology processor was benchmarked on a 2016 Intel NUC single-board computer with an i5-7260U CPU with two cores at 2.20 GHz and 8 GB of RAM. As the first CIM-based topology processor designed to handle...
distributed applications for unbalanced distribution networks, there are no other directly comparable open-source topology processing algorithms against which to compare performance (to the best knowledge of the authors). Average performance statistics are summarized in Table 1 for the IEEE 13, 123, 8500, and 9500 node distribution test systems. The full Python application code for both the real-time service and background daemon are available under a BSD2 software license [35].

VII. CONCLUSION

This work presented a novel distributed application architecture that satisfies the set of key requirements to provide boundary deference, control federation, control disaggregation, and scalable communication for coordination of individual distributed agents. The architecture defined a set of static switch-delimited topological areas that are used to divide the physical layer into a Feeder Area, Switch Area, and Secondary Area. Each area has concrete boundaries formed by the substation transformer, switching devices, and customer secondary transformers. The architecture defined a one-to-one mapping between the software-defined partitions in the communication layer and switch-delimited topological areas of the physical layer. Each topological area was associated with a message bus that handles control, measurement, coordination, and any other communication messages that are needed by distributed agents within a topological area. Data and control compartmentalization was achieved by allowing distributed agents to publish and subscribe only to the local message bus to which they are assigned. The introduced architecture was validated through extensive engagement of key stakeholders from utilities, industry, ADMS vendors, universities, and national laboratories.

In support of the distributed architecture, a robust, lightweight, scalable topology processor was introduced that leverages the structure of the Common Information Model for unbalanced distribution networks. The algorithm was based on the LinkNet data structure for power system models, which uses a set of linked lists to create a graph representation of the distribution feeder. The data structures and methods for traversing the network model were presented in detail, along with an example implementation in the GridAPPS-D platform. Performance benchmarks were presented on the several test feeders using a single-board computer.

The distributed architecture and topology processor presented in this work are currently being extended to provide coordination, control, and data integration across the transmission-distribution boundary. In addition to providing the consensus-based vocabulary needed for data integration (as discusses within this work), the CIM also provides the only consistent representation of power systems equipment within the bulk transmission system and distribution system. Increasing penetration of DERs and distribution-connected renewables are beginning to disrupt transmission operations [1], with an emerging need for layered coordination. The distributed message buses presented in this work can be conceptually extended in an upward direction, with definition of sub-transmission, transmission, and global bulk electric system message buses. Future work will focus on identifying a method for defining transmission-level distributed areas in a consistent manner similar to the switch-delimited topological areas for distribution feeders described in this paper. Current definitions of transmission operator, balancing authority, and reliability coordinator boundaries are often arbitrary and related to utility service territories, rather than network topology. A consistent, topology-based definition of distributed control areas with associated layered message buses and distributed agents will enable a new generation of applications and controls crossing the transmission-distribution boundary. A pilot demonstration extending the layered structure introduced in this paper is planned for an adaptive under-frequency load shedding optimization coordinator for the state of Vermont [36].

REFERENCES

[1] Reliability and Market Considerations for a Grid in Transition, NY-ISO, Ronkonkoma, NY, USA, 2019.
[2] S. Vadari, Electric System Operations: Evolving to the Modern Grid, Norwood, MA, USA: Artech House, 2020.
[3] R. B. Melton, K. P. Schneider, and S. Vadari, “GridAPPS-D™: a distribution management platform to develop applications for rural electric utilities,” in Proc. IEEE Rural Electric Power Conf. (REPC), Apr. 2019, pp. 13–17.
[4] S. Poudel, M. Mukherjee, and A. P. Reiman, “A fairness-based distributed energy coordination for voltage regulation in distribution systems,” in Proc. IEEE Green Technol. Conf. (GreenTech), Mar. 2022, pp. 45–50.
[5] F. B. dos Reis, A. P. Reiman, and G. D. Black, “Multi-agent energy state estimation for distribution systems,” in Proc. IEEE Power Energy Soc. Gen. Meeting (PEGSM), Denver, CO, USA, Jul. 2022, pp. 1–5.
[6] R. Sadran, S. Poudel, A. Dubey, and K. P. Schneider, “Layered coordination architecture for resilient restoration of power distribution systems,” IEEE Trans. Ind. Informat., early access, May 24, 2022, doi: 10.1109/TII.2022.3177464.
[7] P. Sharma, A. Riepnieks, A. Reiman, A. Singh, R. Melton, J. Ogule, and C. Allwardt, “Model-based interface design for smart field-device integration,” in Proc. IEEE Rural Electr. Power Conf. (REPC), Savannah, GA, USA, Apr. 2022, pp. 31–37.
[8] J. P. Ogle, R. B. Melton, K. P. Schneider, and R. Jinsiwale, “Enhancing responsiveness and resilience with distributed applications in the grid,” in Proc. IEEE Rural Electr. Power Conf. (REPC), Apr. 2021, pp. 15–20.
[9] J. D. Taft and A. Becker-Dippman, “Grid architecture,” Pacific Northwest Nat. Lab., Richland, WA, USA, Tech. Rep. PNNL-24044, 2015.
[10] J. D. Taft, “Comparative architecture analysis: Using laminar structure to unify multiple grid architectures,” PNNL, Richland, WA, USA, Tech. Rep. PNNL-26089, 2016.
[11] A. A. Anderson, E. G. Stephan, and T. E. McDermott, “Enabling data exchange and data integration with the common information model,” Pacific Northwest Nat. Lab., Richland, WA, USA, Tech. Rep. PNNL-32679, 2022.
[12] A. Schumlin, H. K. Cakmak, K.-U. Stucky, C. Duepmeier, and V. Hagenmeyer, “Semantic CIM2Matpower: Ontology-based network topology processor and converter for CIM node-breaker models,” in Proc. IEEE 7th Int. Conf. Smart Energy Grid Eng. (SEGE), Oshawa, ON, Canada, Aug. 2019, pp. 31–36.
[13] R. Podmore, M. Goodrich, D. Becker, P. Hirsch, S. Lee, and M. Robinson, “Building a topology estimator for large inter-regional networks,” in Proc. 33rd Annu. Hawaii Int. Conf. Syst. Sci., 2000, p. 4011.
[14] D. S. Popovic, E. Varga, and Z. Perlic, “Extension of the common information model with a catalog of topologies,” IEEE Trans. Power Sys., vol. 22, no. 2, pp. 770–777, May 2007.
[15] Y. Pradkeel, P. Seshuraju, S. A. Khaparde, and R. K. Joshi, “CIM-based connectivity model for bus-branch topology extraction and exchange,” IEEE Trans. Smart Grid, vol. 2, no. 2, pp. 244–253, Jun. 2011.
[16] S. Shukla and M. G. Yao, “Efficient method for extracting network information from the CIM asset model,” in *Proc. IEEE Power Energy Soc. Innov. Smart Grid Technol. Conf. (ISGT)*, Minneapolis, MN, USA, Sep. 2016, pp. 1–5.

[17] Z. Zhou, C. Yuan, Z. Yao, J. Dai, G. Liu, R. Dai, Z. Wang, and G. M. Huang, “CIM/E oriented graph database model architecture and parallel network topology processing,” in *Proc. IEEE Power Energy Soc. Gen. Meeting (PESGM)*, Portland, OR, USA, Aug. 2018, pp. 1–5.

[18] J. Huang, X. Xu, X. Li, and C. Wang, “Island partition in active distribution network based on CIM,” in *Proc. Int. Conf. Electron. Ind. Autom. (IEA)*, 2017, pp. 96–90.

[19] I. L. Lendak, A. M. Erdeljan, and D. S. Popovic, “Algorithm for cataloging topologies in the common information model (CIM),” *Comput. Math. Appl.*, vol. 61, no. 3, pp. 715–721, Feb. 2011.

[20] X. Liu, S. Qiau, and X. Xu, “Research on topology analysis of uncertain distribution network,” in *Proc. China Int. Conf. Electr. Distrib. (CICED)*, Shenzhen, China, Sep. 2014, pp. 1547–1550.

[21] A. Miranda, P. Arboleya, L. Suarez, and J. M. Carou, “A common information model integration in a graph database for LV terminal distribution networks with PLC-based smart meters,” in *Proc. IEEE Madrid PowerTech*, Madrid, Spain, Jun. 2021, pp. 1–6.

[22] K. Shahid, E. Schiavone, D. Drenjanac, R. P. Baklund, R. L. Olsen, and H.-P. Schwefel, “Extraction of CIM-based distribution grid topology information for observability,” in *Proc. 15th Eur. Dependable Comput. Conf. (EDCC)*, Naples, Italy, Sep. 2019, pp. 165–170.

[23] J. D. Taft, “Grid architecture 2,” Pacific Northwest Nat. Lab., Richland, WA, USA, Tech. Rep. PNL-24044-2, 2016.

[24] A. A. Anderson, S. Vadari, J. Barr, S. Poudel, A. Dubey, T. McDermott, and R. Podmore, “Introducing the 9500 node test system to support advanced power applications,” Pacific Northwest Nat. Lab., Richland, WA, USA, Tech. Rep. PNL-33471, 2022.

[25] IEEE PES Test Feeders Working Group (Aug. 2022). *Temporary Review Page for the IEEE 9500 Node System*. Accessed: Sep. 2022. [Online]. Available: https://cmte.ieee.org/pes-testfeeders/temporary/

[26] *Application Integration at Electric Utilities—System Interfaces for Distribution Management—Part 1: Interface Architecture and General Recommendations*, Standard IEC 61968-1:2020, Apr. 2020. Accessed: Jun. 2022. [Online]. Available: https://webstore.iec.ch/publication/32542

[27] CIM Users Group. (Dec. 2021). *Current CIM Model Drafts*. Accessed: Jan. 2022. [Online]. Available: https://cimug.ncia.org/CIM%20Releases/Forms/AllItems.aspx

[28] R. B. Melton, K. P. Schneider, E. Lightner, T. E. Mcdermott, P. Sharma, Y. Zhang, F. Ding, S. Vadari, R. Podmore, A. Dubey, R. W. Wies, and E. G. Stephan, “Leveraging standards to create an open platform for the development of advanced distribution applications,” *IEEE Access*, vol. 6, pp. 37361–37370, 2018.

[29] P. M. Cashin, M. R. Mayson, and R. Podmore, “LINKNET—A structure for computer representation and solution of network problems,” *Austral. Comput. J.*, vol. 3, no. 3, pp. 106–113, 1971.

[30] (1971). *Digital Computer Analysis of Power System Networks*. Accessed: Jun. 2022. [Online]. Available: https://it.canterbury.ac.nz/bitstream/handle/10092/6164/podmore_thesis.pdf

[31] Z. Chu, A. Pincetti, R. Kaviani, R. Khodadadeh, X. Li, J. Zhang, K. Saikumar, M. Sahaei-Ardakani, C. Mosier, R. Podmore, K. Hedman, O. Kosut, and L. Sankar, “A verifiable framework for cyber-physical attacks and countermeasures in a resilient electric power grid,” 2021, arXiv:2104.13908.

[32] PowerData Corporation. (2022). *OpenPA Overview*. [Online]. Available: https://powderdata.github.io/com.powderdata.openpa/

[33] A. Anderson, J. Barr, S. Vadari, and A. Dubey, “Real-time distribution simulation and application development for power systems education,” in *Proc. IEEE Power Energy Soc. Gen. Meeting (PESGM)*, Denver, CO, USA, Jul. 2022, pp. 1–5.

[34] *Application Integration at Electric Utilities*. Standard IEC 61968-100:2022, Feb. 2022. Accessed: Jun. 2022. [Online]. Available: https://webstore.iec.ch/publication/67766

[35] Pacific Northwest National Laboratory. (2022). *GridAPP-D Toolbox Topology Processor*. [Online]. Available: https://github.com/GridAPPD/gridappd-topology-processor

[36] A. Anderson, A. Bretas, D. Wang, V. Orestis, J. Carrara, and J. Pew, “Distributed load shedding application architecture and bi-level predictive estimator algorithm,” in *Proc. IEEE PES General Meeting*, 2023, pp. 1–5.
ROHIT A. JINSIWALE (Member, IEEE) received the bachelor’s degree in electrical engineering from the Sardar Patel College of Engineering, Mumbai, India, and the M.S. and Ph.D. degrees in electrical and computer engineering from the Georgia Institute of Technology, Atlanta, GA, USA.

He is currently a Research Engineer with the Pacific Northwest National Laboratory, Richland, WA, USA. He is also working with the Seattle Research Center. His current research interests include decentralized control methods for smart grids, working on high fidelity real-time simulators, equity-aware planning structures, and resilience evaluation schemes.

CRAIG H. ALLWARDT (Member, IEEE) received the Bachelor of Science degree in computer science from Washington State University, in 2007.

He is currently a Software Engineer with the Energy and Environment Directorate, Pacific Northwest National Laboratory, Richland, WA, USA. He is also a member of the Electricity Infrastructure and Buildings Division, Controls Team. He has more than 20 years of experience in the design and implementation of software systems. He specializes in developing new software tools in the field of knowledge management, distributed systems, integration systems, and messaging middleware. His research interests include developing software platforms for data management and data integration in power grid and building energy research.

GARY D. BLACK received the Bachelor of Science degree in computer science from Washington State University, in 1989. He is currently a Software Engineer with the Physical and Computational Sciences Directorate, Pacific Northwest National Laboratory, Richland, WA, USA. He is also a member of the Computational and Data Engineering Group, Scientific Software Engineering Team. He has more than 30 years of experience in the design and implementation of software systems that support simulation and modeling in various scientific application domains. His research interests include subsurface transport modeling, computational chemistry, carbon capture and sequestration, building energy, and power grid simulation.

* * *

120780

VOLUME 10, 2022