A Framework for Communication and Provisioning in an Intelligent Secondary Substation

Stephan Cejka, Alexander Hanzlik, Andreas Plank
Siemens AG, Corporate Technology
Email: [stephan.cejka, alexander.hanzlik, andreasplank]@siemens.com

***Abstract*—Gridlink provides a communication infrastructure for the implementation of distributed control systems in Java. It is a completely decentralized solution where the communication partners dynamically form a cluster of known instances during execution. Gridlink uses a distributed event bus based on an asynchronous communication model. A typical Gridlink system is built from a set of modules that execute a distributed application and that communicate with each other by exchanging messages. We present a smart grid use case dealing with the detection and handling of voltage band violations in low voltage networks deployed in secondary substation nodes.**

I. Introduction

Conventional electric power grids traditionally follow a centralized approach for power generation, transmission and distribution with few producing power plants in the high voltage grid and many consumers (households) in the low voltage grid. The on-going trend for decentral power management and renewable resources drives forward the evolution of so-called Smart Grids that integrate additional power sources, e.g., photovoltaic plants into the medium and low voltage power grid. Their integration increases power management complexity and imposes additional efforts for maintaining grid stability, primarily regarding load balancing and adjustment.

An important component in the electric power grid is the *secondary substation node* (SSN) [1], [2] that connects local commercial and residential users to the power grid transmission network. The task of the SSN is the distribution of electric power within a bounded voltage level to customers. Traditionally, the enforcement in case of problems is done by reinforcing the power line or replacing the transformer. Common SSNs are isolated devices in terms of communication; the interaction with and maintenance of them require physical presence of servicing personnel on site. New approaches suggest using active voltage regulation measures, e.g., on-load-tap-changer transformers [3]. Industrial research in automating low voltage grids is ongoing and products are under development (e.g., [4]–[6]).

To ease interaction and to reduce maintenance efforts and costs it is reasonable to provide access to the SSN from a remote control center over a communication network. We propose an architecture for an *intelligent SSN* (iSSN) [7] that comprises a set of interacting software modules executing a distributed application and communicating with each other. A typical iSSN application operates in an environment having various sensors (like meters) that transmit measurement data to processing modules for computation and generation of control data. Among actor modules may be a storage module for the persistence of measurement data, an analysis module operating on storage data to detect anomalies in the grid, a dashboard module providing visualization of the current grid state to the operator or modules that control peripheral components. Furthermore, a module for the detection of future grid problems based on current data and forecasts is in development. Once such problems are identified, an auction module could be initiated to call flexibilities from smart buildings [8] as shown in a proof-of-concept [9]. Such a message based control system produces considerable amounts of data the management and timely distribution of which requires an efficient, resilient, modular and scalable communication infrastructure [7]. *Gridlink* provides such an infrastructure based on an embedded message bus and designed for implementation of distributed control applications. Communication between modules is asynchronous and completely decentralized. The failure of a module does not prevent the remaining modules from communicating.

The rest of this paper is structured as follows: In Section II we introduce Gridlink - its features, terminology and architecture. Section III deals with application provisioning. In Section IV we provide a smart grid application case study and experimental results. Section V concludes the paper with the lessons learned and with an outlook to the future work.

II. Gridlink

The Gridlink architecture is based on the *vert.x*[1] distributed event bus used for communication between modules. Discovery of other modules is achieved by forming a *Hazelcast*[2] cluster, which in turn uses multicast messaging per default [10]. In addition to an earlier platform (e.g., [11]), Gridlink improves modularity with new functions like the service registry and provisioning features, which constitutes also the added value of Gridlink to vert.x. An exhaustive evaluation with comparable approaches, however, is out of scope of this

[1] http://vertx.io
[2] http://hazelcast.com

978-1-5090-1314-2/16/$31.00

WiP paper that demonstrates the framework next to the current project's use case. Related work will be thoroughly considered when these findings lead to practical applications in the field. Various aspects, like security, which are – without doubt – important, are therefore also not part of this contribution. Gridlink's API provides classes for common functionalities such as configuration file handling, Gridlink group forming and the registry.

Gridlink allows distributed execution of an application using modules, each of which providing a certain functionality. A module is a Java program that

1) implements a dedicated functionality,
2) takes over one or more roles,
3) is addressable and reachable over the event bus via one (or more) role address(es), and
4) provides functions to other modules.

These modules utilize the Gridlink core API and use common or self-defined services. Messages are transmitted to a module by specifying its destination role. This role has to be claimed first by the intended receiver module, e.g., as shown in Fig. 1 a storage module may take over the role storage, by which it is reachable by other modules. The same role can be registered and used at various receiver handlers. Modules that have not been registered to any role can serve as message producers only (e.g., a simple data generator) but are not addressable and thus cannot react to any messages from other modules. Each role can provide handlers for serving one or more services (e.g., a service handler for adding a measurement to the data storage). Therefore, a role can be seen as a container that groups related services. The module which implements the required service executes the service function and sends a reply to the requesting module, if applicable. Message handling is based on an asynchronous communication model, which allows for an efficient and scalable implementation and concurrent transmission of large amounts of data over the event bus. Assignment of services to modules is transparent. The requester of a service shall not need to know which module implements which service. To gain access to a service, only the role and the service name have to be known. Fig. 1 shows a module *StorageModule* registered for role *storage* (and an implicit *shutdown* role as described later). The role *storage* provides several services (e.g., *createDataPoint*).

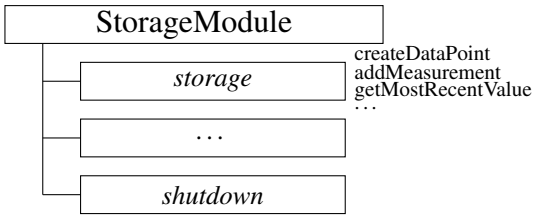

Fig. 1. Modules, roles and services

The Gridlink design contains no single point of failure since there is no central component like a server, and modules can be dynamically added or removed. Thus, the failure or removal of nodes does not prevent the remaining nodes from working and communicating with each other. Naturally, this is not valid whenever an implementation is dependent on messages from or to components that are not available (plug & automate paradigm). The only way to overcome the problem of failure of essential components is replication by running the same module on different hardware components which is supported by the Gridlink architecture by design.

A. Communication

Communication between modules is handled by messages transmitted on the Gridlink bus. All messages consist of a type (e.g., indicating the request for adding a measurement to the storage) and an optional payload (e.g., a measurement entry) and are transmitted as JSON messages to the specified recipient. Gridlink supports two types of communication:

1) `send` transmits messages to a module holding the specified recipient role. Optionally a reply can be issued to the sender, vital, e.g., when requesting data from the storage. Note that if more than one module are registered for the same role, only one of them is chosen for delivery using round-robin fashion.
2) `publish` transmits messages to all modules registered for the specified recipient topic (e.g., to implement observer functionality).

Gridlink does not require specific message formats, as implementations of requests, replies and events are use-case specific. From a semantic point of view, events can be seen as notifications, while requests indicate that the receiver is expected to do something. In most cases, a reply from the receiver of a request is expected. Messages are always transmitted to the addressed receiver(s) only, optimizing event bus bandwidth utilization.

B. Registry

Gridlink provides a distributed membership service available at each module, called the registry. It contains a list of all modules currently active and reachable. For each module, a list of roles implemented by a module is provided and finally for each role the list of services it provides. Observers can be registered to get informed whenever a node, module or role is added to or removed from the bus.

C. Groups

Per default, a Gridlink bus (or cluster) is formed of all modules running in the same subnet using the multicast discovery of Hazelcast. However, for some applications it may be necessary to separate modules and form groups to prevent knowledge of and communication with group-external modules. This is achieved by using a unique multicast address for each group.

D. Gateway modules

Some applications require communication with components outside a Gridlink system. This is done using gateway modules. A gateway module is an interface for communication

between the Gridlink bus and some external component using a dedicated communication protocol. Currently, Gridlink provides gateway modules for REST and XMPP protocol handling.

III. Application provisioning

This term comprises typical software maintenance tasks like installation (provisioning of new software), update (replacement of existing software by another version), configuration (parameterization of software during execution), start, stop, removal and status information of modules. In a native Gridlink system, these tasks are in the responsibility of the user who performs these actions using a local or a remote access tool. Modules serviced directly by the user are called *unmananged modules* (see Fig. 2). To perform application provisioning, the user has to interact directly with each single module.

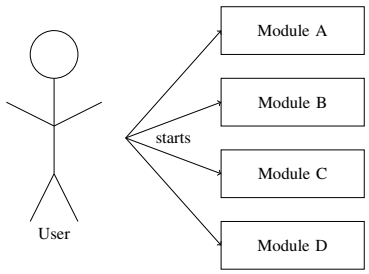

Fig. 2. Unmanaged (standalone) module

To make provisioning easier, an *AppManager* is added who is responsible for the mentioned provisioning tasks. The AppManager is a Gridlink module, therefore it is able to communicate with other modules over the Gridlink event bus. Additionally, the AppManager is a gateway that makes the Gridlink system accessible from outside. The AppManager is an *unmanaged* module, since it has to be started by the user. Modules serviced by the AppManager are called *managed modules* (see Fig. 3). Once an AppManager is available in a Gridlink system, application provisioning can be done also from remote sites. A failure of the AppManager crashes the provisioning feature, but does not interfere with the operation of other modules. All provisioning tasks are transparent to the modules. The following paragraphs describes main provisioning tasks in more detail.

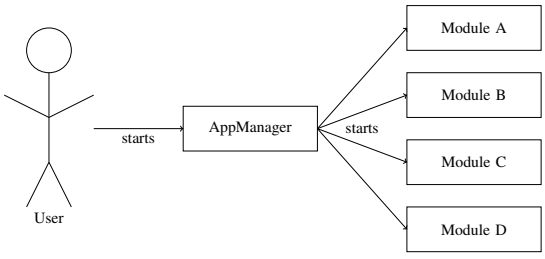

Fig. 3. Managed module

A. Module Installation

Module Installation refers to a functionality upgrade by starting a new module. The installation of a module is triggered from a remote site, e.g., by receiving an according XMPP message. Modules are initially stored in an app store containing all dependencies necessary to run the module. The AppManager downloads the module archive, extracts it and launches it.

B. Module Update

Module Update refers to the replacement of a running module with another version. A shutdown request is issued to the module's implicit shutdown role (c.f. Fig. 1). It stops operation and optionally stores its state containing all information required to continue properly after the update as defined by the module developer in a file. There may be situations where a module cannot immediately shut down, e.g., because it is currently performing some critical action that cannot be interrupted right away. In this case, the module refuses to shut down and replies accordingly to the AppManager. The AppManager reports an update failure to the back office and indicates that a manual intervention is required. In case of a success, the remainder of the update process is similar to the installation process.

C. Configuration Update

Configuration Update refers to a modification of operation parameters. It is a less severe intervention since it does not necessarily require a module restart. The form and use of configuration properties are in the sole responsibility of the module developer and may, e.g., contain a list of roles the module needs to communicate with. When triggering a configuration update, the AppManager downloads the new configuration file to the module working directory. The module gets a notification when the file was changed. It is up to the module to decide whether and how to handle this update notification.

IV. Gridlink and provisioning on the iSSN

The use case deals with the detection and proper handling of voltage band violations in low voltage networks. Modules involved are shown in Fig. 4.

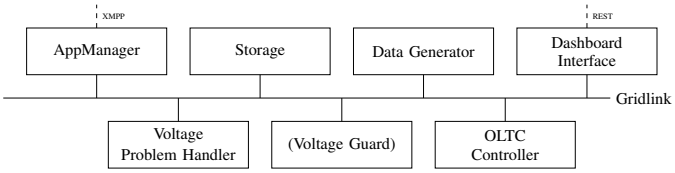

Fig. 4. Modules in this scenario

The *AppManager* module handles the software provisioning tasks described earlier. For remote communication it provides an XMPP connection. All other modules may communicate with remote partners over the AppManager, which makes it a *gateway* module.

For providing their functionality such as monitoring and analyses, other modules require access to historical and current time-series data. The *Storage* module is used for permanent storage of measured voltage values and meta data. It utilizes a Java-based embedded data store for persistence of time series and meta data, shown to be superior to state-of-the-art off-the-shelf solutions with respect to data retrieval time and required storage size [12]. Frequent readouts, immutability and statistical indicators, vital for this use case are optimally supported.

The *Data generator* module generates measurement values using predefined profiles simulating houses' power consumption during the day as well as power production in case they are equipped with a photovoltaic (PV) installation.

The *Voltage Problem Handler* module is a monitoring module that detects problematic voltages in the network below 220 V or above 240 V; upon detection of such a value the operator is notified such that proper countermeasures can be taken. This voltage band was assumed for demonstration scenarios identifying and detecting problems earlier than specified in EN 50160 [13]. Power quality criteria limits in EN 50160 would allow 95 % of all 10 min average values within a week between 207 V and 253 V (U_n $\pm$10 %).

The *Dashboard interface* module is a *gateway* module providing a REST interface. A web based dashboard connects to this REST interface to provide a view on current and historical values of sensors and the transformer.

The *On-Load-Tap-Changer (OLTC) controller* module is a *gateway* module to transmit tap change requests to the transformer [3]. The effect is an increase of the voltage level by 2.5 V on tap up, a decrease on tap down respectively.

The *Voltage Guard* module is an analysis module that is not started in the beginning, but installed by the AppManager on demand (e.g., after the problem handler has indicated a problematic voltage value). If voltages below 225 V or above 235 V are detected at any data point in the network, the module may decide to request the OLTC controller to change its tap position [3]. An increasing number of tap changes has a negative impact on the lifetime of the transformer; therefore the number of tap changes should be kept at a minimum level [14]. To avoid permanent tapping, the lazy algorithm in use aims at not changing the tap position immediately when hitting the barrier, but only when the amount of cumulative voltage violations over time exceeds a defined integration threshold value (ITV, given in [Vs]) [14], [15].

Fig. 5 shows the principle of this algorithm:

1) When leaving the acceptable voltage band the module starts the calculation of the voltage-time area by integration of the observed deviations from the respective voltage band barrier.
2) If this area exceeds the maximum ITV value, a change of the tap position is initiated and the ITV is reset to zero. Tap changes are avoided if they would lead to a voltage band violation at any other data point in the grid.

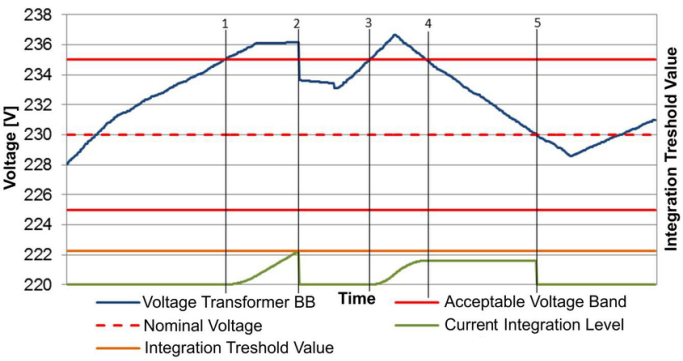

Fig. 5. Integration Threshold Value (ITV) algorithm [15]

3) The calculation is restarted when leaving the acceptable voltage band once again.
4) The calculation is paused when reaching the acceptable voltage band once again.
5) The voltage-time area value is reset when crossing the nominal voltage value.

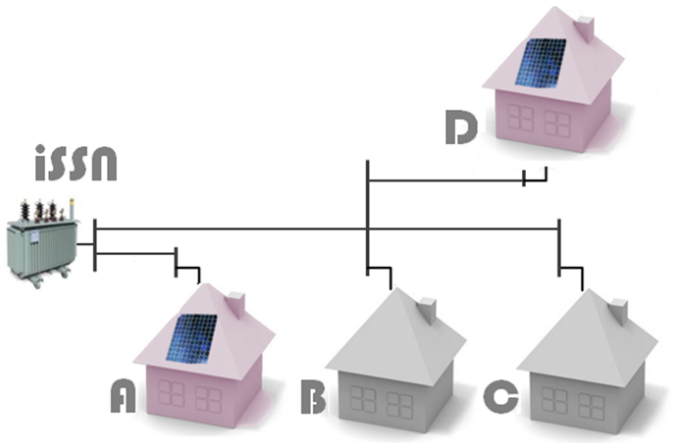

Fig. 6. Low voltage grid scenario (adapted from [9])

The simulation scenario comprises four apartment houses A, B, C and D (Fig. 6), characterized by the same standard load profile H0 for households but differing in their peak load. Houses in group A and D have an attached photovoltaic power plant that produces energy between sunrise (05:00) and sunset (20:30). An additional load (e.g., charging of a battery) is active between 05:00 and 07:15 in house group C. At 19:15, the *Voltage Problem Handler* module detects that voltage drops below 220 V and notifies the operator (Fig. 7).

To react on voltage band violations, the operator decides to install the *Voltage Guard* module that issues commands to the OLTC to tap up/down if values below 225 V and above 235 V are detected. By its installation, the number of values in the voltage band is increased (Fig. 8) such that the percentage of values between 225 V and 235 V increase from 73.16 % to 91.05 %. As values in the allowed range between 220 V

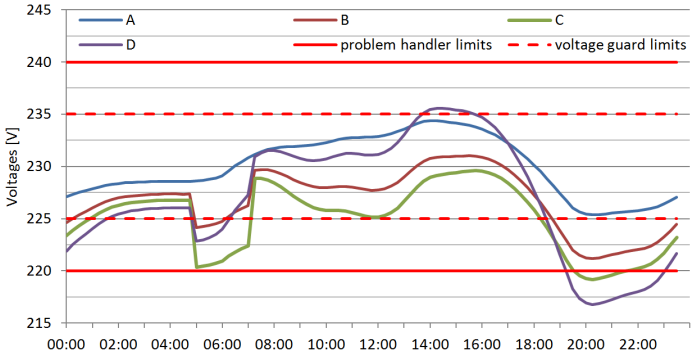

Fig. 7. Voltage curves before installation of Voltage Guard

and 240 V increase from 93.68 % to 100 %, all values remain within the barriers. The setting of the ITV has a significant impact on the number of tap changes: the higher the ITV value, the more severe the observed voltage violations before a reaction [15]. This simulation uses a moderate threshold value of 7.5 Vs, resulting in 8 tap changes on this day, which is an acceptable value recalling the negative impact of the number of tap changes on the transformer's lifetime [14].

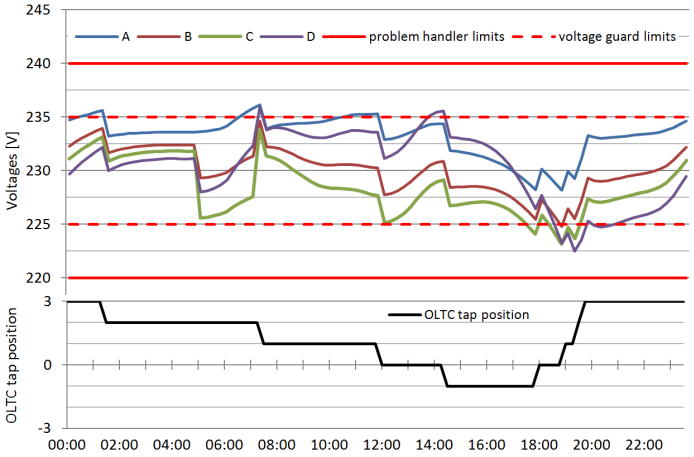

Fig. 8. Voltage curves and OLTC tap position with Voltage Guard installed

V. Conclusion and Future Work

We introduced the Gridlink middleware which provides a framework for the development of distributed control systems in Java. The Gridlink architecture and the software provisioning feature was described in detail. Further, we presented a smart grid application case study implemented using the Gridlink middleware. This case study served as a proof-of-concept to demonstrate the fitness of the Gridlink middleware for the implementation of distributed smart grid applications.

Currently, the AppManager can only service modules started on the same physical machine. In the future, it shall also be able to service modules on remote machines by implementing a multi-host-concept. We will replace the prototype data generator with an IEC 60870 gateway module that is able to receive and process real measurement data from smart meters. Gridlink's practicability is currently being investigated in the course of field tests. We are about to shift the use of the Gridlink from laboratory [16] to real environments [7] such as various Austrian Smart Grid model regions, among them Seestadt Aspern which is currently one of the biggest urban development projects in Europe.

Acknowledgements

The presented work conducted in the "iNIS" project is funded and supported by the Austrian Ministry for Transport, Innovation and Technology (BMVIT) and the Austrian Research Promotion Agency (FFG).

We thank Tobias Gawron-Deutsch for helpful comments.

References

[1] M. Alberto, R. Soriano, J. Gtz, R. Mosshammer, N. Espejo, F. Lemnager, and R. Bachiller, "OpenNode: A smart secondary substation node and its integration in a distribution grid of the future," in *Computer Science and Information Systems (FedCSIS), 2012 Federated Conference on*, pp. 1277–1284, Sept 2012.

[2] R. Soriano, M. Alberto, J. Collazo, I. Gonzalez, F. Kupzog, L. Moreno, A. Lugmaier, and J. Lorenzo, "OpenNode. Open Architecture for Secondary Nodes of the Electricity Smartgrid," in *21st International Conference on Electricity Distribution (CIRED)*, Jun 2011. paper 770.

[3] A. Einfalt, F. Zeilinger, R. Schwalbe, B. Bletterie, and S. Kadam, "Controlling active low voltage distribution grids with minimum efforts on costs and engineering," in *Industrial Electronics Society, IECON 2013 - 39th Annual Conference of the IEEE*, pp. 7456–7461, Nov 2013.

[4] Y. Chollot, P. Deschamps, A. Jourdan, and S. Mishra, "New approach to regulate low voltage distribution network," in *23rd International Conference on Electricity Distribution (CIRED)*, Jun 2015. paper 1145.

[5] M. Mangani, F. Kienzle, M. Eisenreich, Y. Farhat Quinones, R. Bacher, and A. Brenzikofer, "GridBox: An Open Platform for Monitoring and Active Control of Distribution Grids," in *23rd International Conference on Electricity Distribution (CIRED)*, Jun 2015. paper 1070.

[6] S. Russwurm, "Web of Systems for a digital world," Dec 2015. Keynote at the Internet of Things World Forum 2015, Dubai; http://www.siemens.com/press/pool/de/events/2015/corporate/2015-12-internet-of-things/presentation-iot-russwurm_web-of-systems-e.pdf [Online; accessed 14-April-2016].

[7] M. Faschang, M. Stefan, F. Kupzog, A. Einfalt, and S. Cejka, "'iSSN Application Frame' – a flexible and performant framework hosting smart grid applications," in *CIRED Workshop 2016*, Jun 2016. paper 255.

[8] T. Gawron-Deutsch, F. Kupzog, and A. Einfalt, "Integration of energy market and distribution grid operation by means of a flexibility operator," *e & i Elektrotechnik und Informationstechnik*, vol. 131, no. 3, pp. 91–98, 2014.

[9] T. Gawron-Deutsch, S. Cejka, A. Einfalt, and D. Lechner, "Proof-of-Concept for market based grid quality assurance," in *23rd International Conference on Electricity Distribution (CIRED)*, Jun 2015. paper 1495.

[10] "Hazelcast FAQ." http://docs.hazelcast.org/docs/3.5/manual/html/faq.html. [Online; accessed 14-April-2016].

[11] M. Faschang, F. Kupzog, R. Mosshammer, and A. Einfalt, "Rapid control prototyping platform for networked smart grid systems," in *Industrial Electronics Society, IECON 2013 - 39th Annual Conference of the IEEE*, pp. 8172–8176, Nov 2013.

[12] S. Cejka, R. Mosshammer, and A. Einfalt, "Java embedded storage for time series and meta data in Smart Grids," in *2015 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pp. 434–439, Nov 2015.

[13] CENELEC, "EN 50160:2010 - Voltage characteristics of electricity supplied by public electricity networks," Mar 2011.

[14] A. Plank, F. Zeilinger, and A. Einfalt, "Untersuchung der Effektivität von Regelkonzepten in Verteilnetzen," in *14. Symposium Energieinnovation, Graz, Austria*, Feb 2016.

[15] F. Zeilinger, A. Einfalt, K. Diwold, A. Plank, and A. Lugmaier, "Influence of Different Framework Conditions on the Effectiveness of Control Concepts in Distribution Grids," in *CIRED Workshop 2016*, Jun 2016. paper 259.

[16] M. Stifter, M. Windhab, M. Zahedi, and A. Frischenschlager, "Smart Meter Test Stand for requirement analysis of advanced smart meter applications," in *Smart Electric Distribution Systems and Technologies (EDST), 2015 International Symposium on*, pp. 547–552, Sept 2015.