Dependable Peer-to-Peer SCADA Architecture

Mihály Sági

University of Novi Sad, Faculty of Technical Sciences, Trg Dositeja Obradovića 6, 21000 Novi Sad, Serbia, sagi@uns.ac.rs

Ervin Varga

University of Novi Sad, Faculty of Technical Sciences, Trg Dositeja Obradovića 6, 21000 Novi Sad, Serbia, evarga@uns.ac.rs, e.varga@ieee.org

Abstract: SCADA solutions are under a high flux to shift their focus from process control of a limited set of industrial plants to the control of large-scale system of systems. This is in par with the recent proliferation of ubiquitous/pervasive computing paradigm mostly embodied as Internet of Things (IoT). In a traditional setup, a whole system is only partially covered by SCADA data points; therefore, complex simulation is required to fit the missing measurements, hence, buttress decision support scenarios. This usually entails a fully integrated and centralized approach, where SCADA infrastructure needs to hold and distribute data, both collected and calculated. It leads to the increase of load on a supporting real-time database, which hosts millions of data points. It is a challenge that a traditional SCADA design (based on a shared memory database and competing processes), cannot fulfill in real-time. This paper proposes an alternative approach of an architecture and basic functionality of a SCADA system. The proposed architecture targets distributed SCADA systems that can be used to supervise and control large-scale distributed industrial or infrastructural systems. Strategic data organization and segmentation are introduced, so that the acquired data can be efficiently distributed throughout the system. The proposed architecture pushes forward a peer-to-peer node structuring scheme, where an autonomous node supervises and controls only subsets of the system. Nodes collaborate to establish a unified view of the entire system. The proof of the concept implementation has proven to be able to manage significantly more data points in a distributed fashion than a centralized variant.

Keywords: SCADA; smart city; distributed system; peer-to-peer architecture; smart grid

1 Problem Statements and Objectives

As industrial processes became more complex and computing power became cheaper and more robust, these processes started to be supervised and controlled by programmable logic controllers (PLC) to induce more precision and reliability into the process itself. With the growth of the industry and the complexity within the industrial processes, the need for a larger scale process supervision and control was needed, so SCADA systems were developed as a universal mean of access local control modules such as PLCs. They soon became the most commonly used industrial control systems. After the "automation revolution" in industrial systems, SCADAs started being applied to infrastructural management systems as well (e.g. electricity, gas, water, waste-water). Since infrastructure systems are of a more complex nature than industrial systems, the move to this field introduced new challenges into SCADA development. This paper will give a brief historic overview of currently available solutions and will propose a solution for the newly introduced challenges.

1.1 Current Work

SCADA systems were initially used in industrial processes that were located in a single processing plant with well defined geographical boundaries. The geographical and industrial process based limitations had many conveniences such as limited number of sensors and actuators, no or slow expension of number of sensors (telemetered data) over time. These systems usually had only a few operator terminals that showed the overall state of the plant to the operators. Decision support systems in manufacturing process were rare. [1] [2]

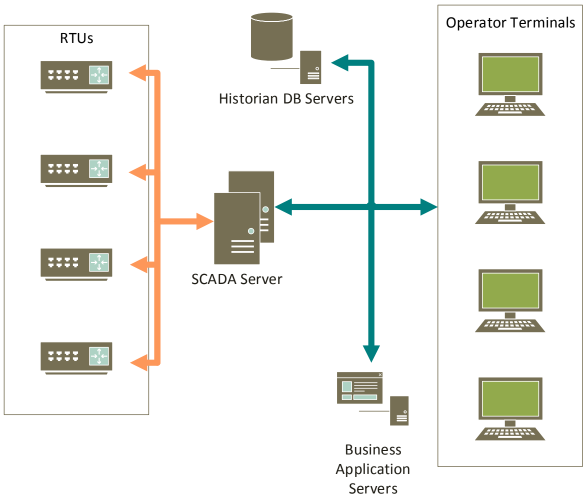

Figure 1. Building Blocks of a Traditional Clustered SCADA System

One of the most popular SCADA designs was building multiple processes that execute segments of the overall SCADA functionality (data processing, communication, supervisory control, automated control procedures, historization, decision support, etc.) (Figure 1). Such designs provided easy integration with the SCADA real-time database and simple extension of basic SCADA functionalities driven requirements specific for the controlled industrial process. Traces of such an architecture are still found in some of the high-end SCADA solutions available on the market.

Current developments in supervised systems generally go towards further merging and increase in size, which has to be followed by the provision of adequate control systems, both in size and acceptable performance. In addition, control requirements are far more demanding than before, in order to run technological processes in more precise and safer manner.

This is particularly obvious in infrastructural SCADA applications (from power distribution, through oil, gas and water transport and distribution systems to even smart cities), that requires integrated control system with acquisition of process data that is placed to the centralized real-time database with several million data points. [3]

In these systems, the decision support is integrated with the SCADA so the real-time database stores telemetered SCADA data together with derived values (e.g. calculations, estimations, predictions, simulations, etc.). The SCADA is expected to feed data to decision support components, treat the collected derived data in the same manner as the telemetered and to distribute all data to client (operator or business) applications. In addition, the scale of such systems introduces a need for multiple client application. Even though contemporary computers scaled through time, the bottleneck when applying traditional SCADA architectures to industrial systems is the communication infrastructures, that is far more complex and expensive to change or upgrade. Additionally, infrastructural systems usually are geographically scattered, they have a large number of sensing and actuating points (up to several million) and the configuration is changing (mostly growing) over time. Such systems also rely on complex decision support and tens of operator and maintenance terminals. [4]

The challenges coming from infrastructural systems were the main driving force of the design of a new SCADA architecture. However, with the advances in both hardware and software over time, there are additional requirements important for its applicability of contemporary SCADA systems: extensive and elaborate SCADA model, reusability, extensiblity, easy adaptation to user demands, cross-platform and secured operation. [5] [6] [7]

Contemporary SCADA systems on the modern market need to support the handling of millions of data points (part telemetered, part derived) in a networked, real-time environment. Data acquisition and primary data processing is always done in a centralized fashion due to architectural limitations, and then the concentrated values in the real-time database counting several million data points are distributed to achieve high-availability. The complex operations in the decision support systems are also performed where the data acquisition is taking place. An example is, an infrastructural system where SCADAs are being introduced are Distribution Management System implementations for power distributions, where it is a usual market requirement that the SCADA handles more than 10 million data points in real-time. To achieve high availability, multiple copies of the SCADA software is running and data replication mechanism is used. Due to the size of the database and the need that the data needs to be transferred as quickly as possible to the backup SCADA system, the communication subsystem must be optimized to the highest level. [7] [16]

Other contemporary approaches include WEB based SCADA systems. [8] Such systems leverage the network infrastructure already available and while they are easier to implement than “traditional” SCADA systems, the required level of security and real-time operation capability are not achievable for big systems under supervision. [9] [10]

1.2 Proposal

Core requirements for a SCADA system (regardless if industrial or infrastructure) are real-time operation, reliability and high availability. The minimum response time is defined by the supervised system and is measured in seconds. Other requirements that are mandatory for SCADA systems are reliability availability as for most of the real-time systems.

The architecture presented in this paper relies on a distributed storage system that provides data distribution with configurable bandwidth and performance ratio where the communication between SCADA nodes can be optimized per needs. [14] The data distribution is built into the real-time database enabling the distribution of telemetry and decision support subroutines.

The paper describes the architecture and basic functionality of a SCADA system developed in line with this concept, providing efficient real-time execution of complex supervisory and control procedures in a distributed environment.

The proposed architecture can be classified as a third generation SCADA architecture. The SCADA core is based on an earlier study [11], where a monolithic architecture is used. Basic principles are inherited and the architecture is upgraded to adopt the “modern”, networked nature.

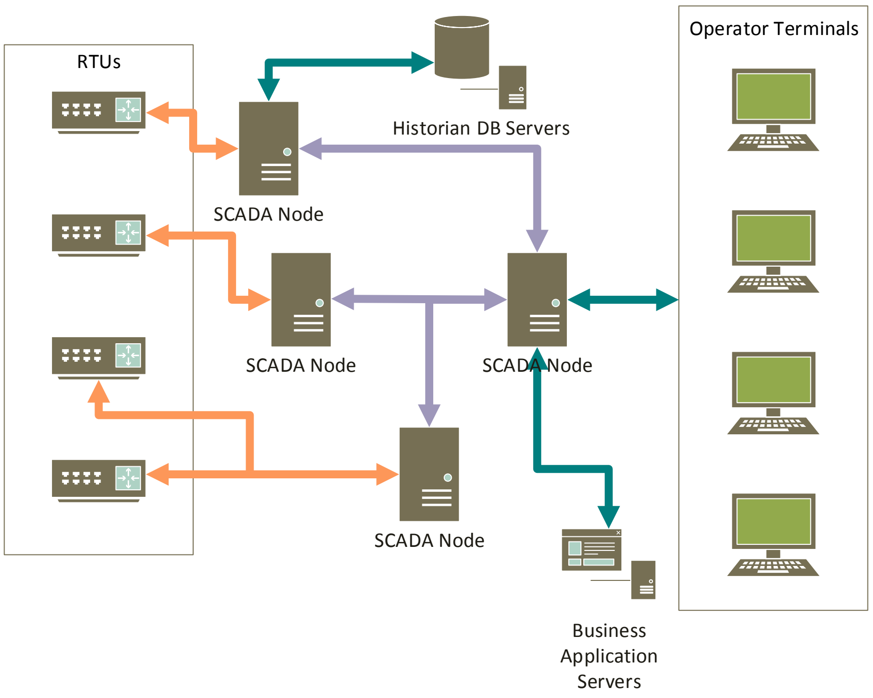

Figure 2
Proposed Peer-to-Peer Structured SCADA System

Since the proposed architecture is considered to be of a generic purpose, so it can drive all infrastructural decision support systems found in a smart city, the most complex and response critical infrastructure is taken as a target. The new architecture must be fit to feed a power system. Based on the research of Kang L. and Yang L. [4] the acquisition and control tasks are being parallelized throughout homogenous SCADA nodes.

Dietrich B. et al. [12] confirmed the usability of the object-oriented paradigm as a tool to reach full extensibility and customizability even in real-time application, thus it is adopted as the core concept of the new SCADA.

Ramdan F [8] and Qiang Z [13] show, that multi tier SCADA systems can be designed and preserve the full functionality of conventional SCADAs.

2 System Architecture

Taken into consideration the distributed nature of most common infrastructural management systems, a target SCADA should be distributed with homogenous nodes that can execute field communication and that can share data with all other "interested" nodes.

Each node is responsible for handling operations regarding the subset of telemetry equipment it is directly connected to through a computer network (e.g. ethernet)All other operations are ignored by the operation logic of the note, but is stored and distributed for redundancy purposes.

Client terminals are also connected to nodes. Each client receives data and is able to execute commands for its area of responsibility (AOR) that is configurable Commands and telemetry data that is outside of the nodes AOR is stored for redundancy purposes.

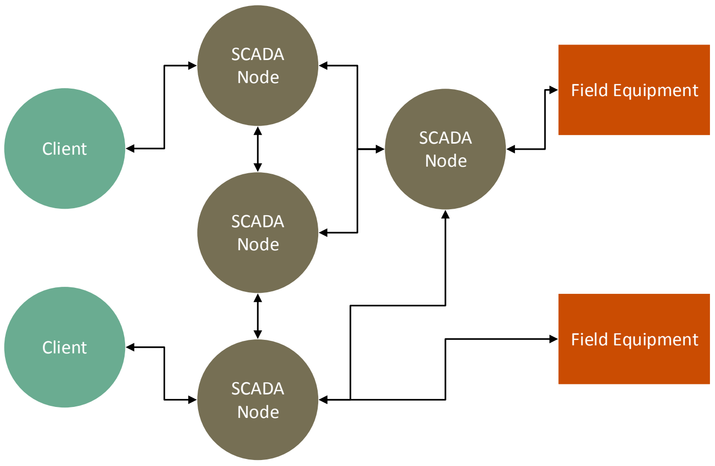

Figure 3
Simplifyed System Architecture

With such an architecture, the traditional monolithic SCADA is broken up into SCADA nodes. Since clients are not aware of the communication between these nodes, they are not aware of infrastructure or hierarchy change within thin the SCADA itself (Figure 3).

Data processing and conversion from the inner model to an industry specific model is also done in a distributed fashion. Once the data enters the system from the field equipment, it is immediately processed and then distributed. To allow the distribution of this task once the configuration changes and new field equipment is added to the system, the node that is least loaded will connect to the new equipment. On a node failure, all connected external devices are re-routed to other, healthy nodes. Load for each of the nodes is calculated based on the number of expected value changes in one second incoming from the connected field equipment.

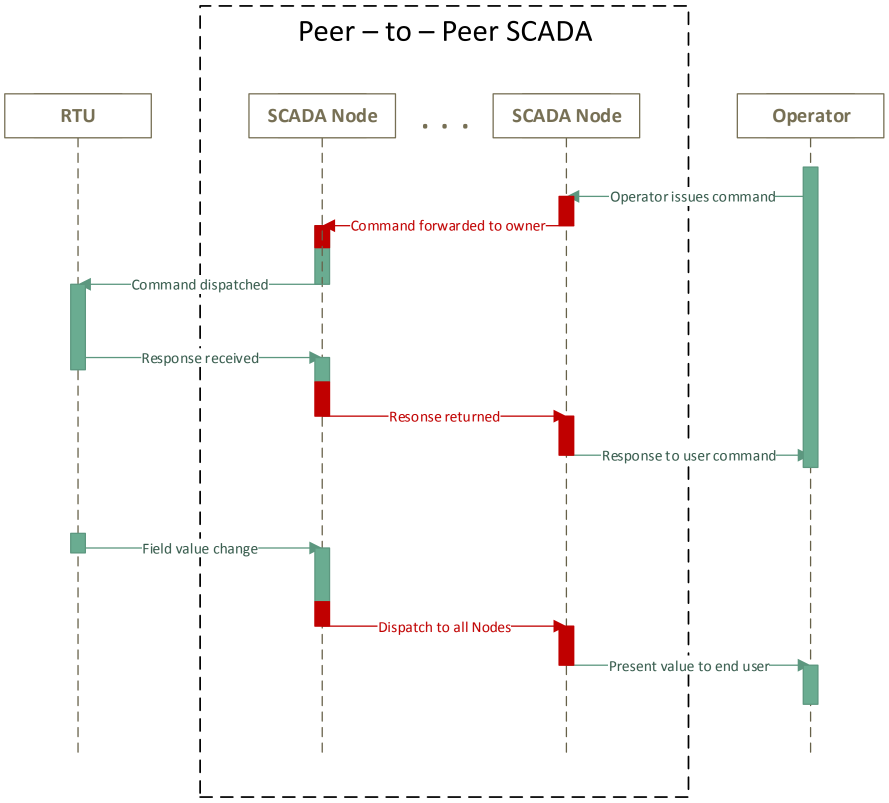

Figure 4
Sequence diagram of basic SCADA operations

The figure above (Figure 4) shows the sequence of major SCADA use cases: supervisory control and data acquisition. The sequence for both operations is decomposed on the domain specific part (colored green) that needs to be done in every SCADA system, no matter what architecture it has. These operations are: primary (front-end) and secondary (back-end) processing and data, and input validations. The architecture specific part of the sequence diagram (colored red) represents the additional steps (time) that is the overhead the distributed architecture includes.

During a configuration change it is of high importance that the entire cluster works on the same configuration. Configuration changes are distributed as changes in a distributed version control system ensuring the fulfillment of the above-mentioned requirement. [15]

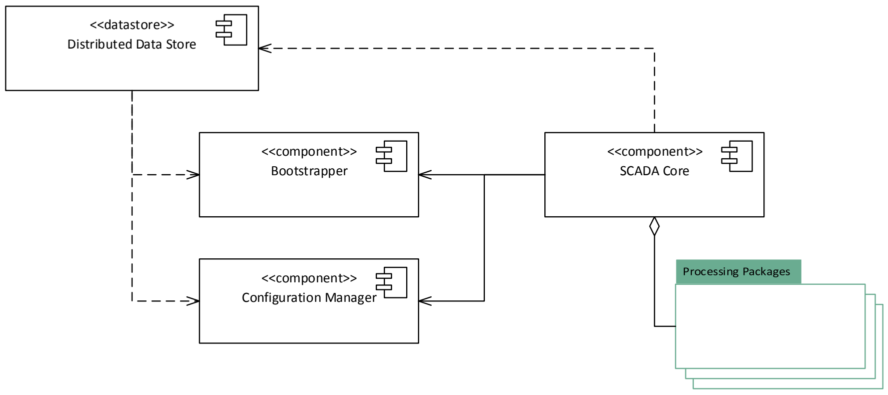

Figure 5
Components of a SCADA Node

A key required feature for the distributed data stores analyzed for this paper was an embedded VCS (*Versioning Control System*) support that allows versioning of the system configuration. The *Configuration Manager* component controls the configuration version using the VCS interface of the DDS.

The VCS that is embedded into the distributed data storage that the SCADA nodes are built on top of, enables that the configuration can be changed from any of the nodes the configuration manager client software connects to, as long as the operator using the software has sufficient privileges for the AOR the configuration change is intended for. Once the operator changes the configuration on one node, the changes are propagated through the DDS and trigger the configuration manager component to apply the changes to the SCADA core.

The internal SCADA model is designed to support one real-time, and multiple history/simulation data contexts and are accessible to all authorized users. Each context supports loading of different configuration version, current or any valid previous configurations.

Each node in the system, in spite of different roles it may play, always executes a single software executable, which holds the real-time database and performs the data acquisition, control and communication tasks. Since decision support routines need to be handled in the same (real-time) manner as telemetry, there is no need to differentiate them. A node is based on a common software infrastructure implemented within a core library, and a set of additional dynamic libraries implementing process variables, various protocols, etc. that are distributed with the configuration (Figure 5). Each library implements or extends an entity (class) using the predefined API. Using OOP paradigm, it is easy to extend or replace any of the classes, or add completely new.

Addressing entities within the system uses a simple, but effective schema where four logical sections of one unique global key are assigned to each entity: context, node and variable names is unique, providing a primary key to each for every configurable component in the system:

- *Context* – unique identifier of the context where the entity is available. Default 0 is the identifyer of the real-time context. When a real-time entity is copied to a simulation context only this portion of the primary key is changed.
- *Node* – unique identifier of the node owning the entity. Only nodes with this ID are making true changes in the entity (from the field or decision support routines). The rest of the nodes get these entities using data distribution.
- *Type*– entity type set from all the available types in the system configuration. The system has a pre-defined set of entity types, but can be extended to achieve customizability.
- *Sequence* – a simple ordinal number within the owning node.

To run the solution only two libraries are needed besides the executable – *Bootstrapper* and *Core*. With highly modular architecture, the rest of the libraries providing support for various industrial protocols, decision support or customizability are not the crucial. Thus, the system can function without them in a “store-and-forward” mode when only stores partial configuration and real-time data for redundancy purposes. [14]

Bootstrapper is the coordinator of the local node (initialization / de-initialization, configuration loading / unloading, starting / stopping routines, etc.). It is composed of two essential components:

- *Communication manager* – interface to the SCADA administrator to reach the configuration manager to take actions on a node.
- *Configuration manager* – runs both the initialization and de-initialization process of the local SCADA node core on configuration version change.

The core of the system is a communication engine that does the telemetry data acquisition and command dispatching based on the node configuration. All data reaching a node is treated the same way regardless if it is acquired directly from an RTU or through standardized industrial protocol or is distributed by the DDS itself. It also provides a means for local and remote user interfaces for system supervision and control.

4 Proof of the Concept Implementation

Key difference in the currently described architecture, compared to the traditional solutions, is the introduction of a high-level program model of the physical process based on catalogues and process variables (Figure 6). [16]

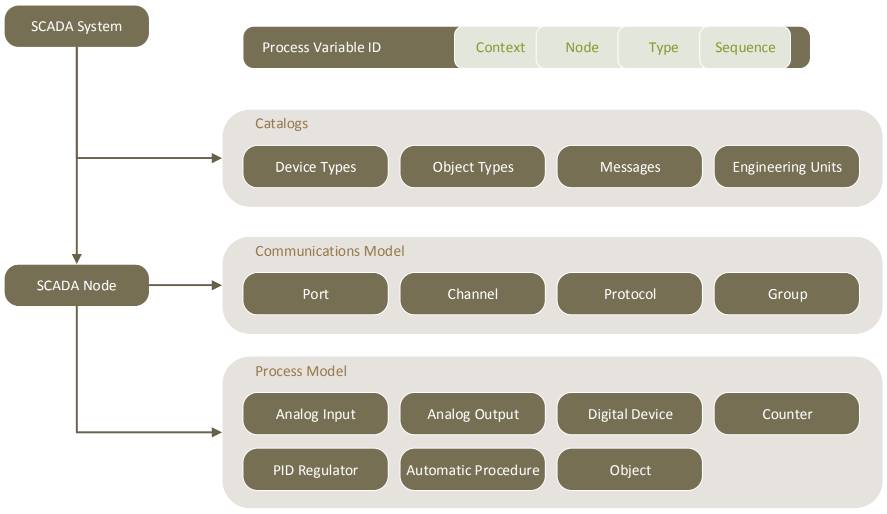

Figure 6
Process model and real-time database organization

Catalogs define set of application-specific values and data-types, referenced later by configured entities. As an example, digital devices catalog defines types of used control modules (e.g. on-off valve with two limit switches), including legal set of states and commands associated to the module.

Process variables as described in the process model section on Figure 6, are divided into levels by their complexity, represent various measuring and actuating equipment and attributes in the physical process itself. The current state or value of the process variable is always expressed in its engineering unit (EU) value, or through the associated state/command pair if the variable is digital in its nature. There are conversion capabilities provided so that external values can be converted to the internal data types.

To achieve extensibility and customizability, each type of process variables is, following OOP paradigm, providing a convenient way to extend or modify original definition or functionality according to the application-specific requirements.

4.1 Tools Used to Build the Solution

Real-time software is often built in C++ to leverage the performance this programming language provides and to us the object-oriented paradigm for easier implementation of the high level design. As one of the requirements set for the solution is cross-platform operation, the aforementioned programming language is also fits that need.

For the distributed storage system, two solutions were analyzed, the IBM Cloud Object Storage and IPFS [15]. After a detailed analysis of both solutions, IPFS was chosen as a platform to build the SCADA system on top of. [17]

To unify the C++ libraries that Visual Studio C++ compiler and G++ provide, Boost library is used.

Conclusions and Future Work

This paper analyzes the bottlenecks of centralized SCADA architectures when used on large scale CIS. In addition, it presents a possible architecture of a distributed industrial grade SCADA system that fulfills the demands of a smart city and/or smart grid, such as distribution of the data, processing and control, system scalability, real time efficiency and cross-platform operation.

The proposed architecture enables homogenization of the SCADA nodes that can be arbitrarily extended and configured due to the high modularity of the new concept. With these characteristics, both SCADA operators and business information system applications are enabled to connect to any node and acquire the data they demand, or control the process supervised by the node.

This architecture also provides the capability of having an arbitrary number of nodes in a system to share processing and storage of data. With a distributed data store as the basis of a SCADA system, requirements for high availability can be reached without additional mechanisms (e.g. replication). Both configuration and real-time data is shared across the system as soon as changes occur.

Configuration versioning as an integral part of the majority of distributed data stores, adds value to this solution since without a need of additional applications or components, version control of the configuration is achieved.

Further research can be done on an efficient way of managing the configuration of a distributed SCADA system using the proposed architecture and also measuring the performance and scalability of the solution as well as the minimum system requirements.

Some of the additional topics that need to be answered before reaching a commercially viable architecture are: separation of configuration and real-time data, determining the impact of persisting all changes in a normal work of the system.

As presented on Figure 3 and mentioned in Section 5, the current work relies on 3rd party implementation of distributed data storage system with proven history. This DDS is not optimized for real-time operation and some SCADA specific needs, once a stable system is built on top of this library, additional improvements will be needed on the DDS itself to minimize the overhead that the proposed architecture introduces to the SCADA system.

References

[1] D. Bailey and E. Wright, "Practical SCADA for Industry", Elsevier, 2003

[2] S. Lishev, R. Popov and A. Georgiev, "Laboratory SCADA Systems – the State of Art and the Challenges," Balkan Journal of Electrical & Computer Engineering, Vol. 3, No. 3, p. 164, 2015

[3] J. M. Black, C. Rawie and M. Mattson, "High-Performance SCADA System Provides an Integrated Information System," in Water Environment Federation, Anaheim, California, USA, 2000

[4] L. Kang and L. Yang, "A Distributed and Parallell Computing Framwork for SCADA Application in Power System," in International Conference on Electrical and Control Engineering, Tuxtla Gutierrez, Mexico, 2010

[5] O. Rysavy, J. Rab, P. Halfar and M. Sveda, "A Formal Authorization Framework for Networked SCADA Systems," in 19th IEEE International Conference and Workshops on Engineering of Computer-Based Systems, Novi Sad, Serbia, 2012

[6] l. Yang, X. Geng and X. Cao, "A Supervisory Control and Data Acquisition Network Security Attack Recognition Method Based on Multi-Agent," Journal of Computational and Theoretical Nanoscience, Vol. 13, No. 4, pp. 2504-2511, 2016

[7] S. Ju, J. Lee, J. Park and J. Lee, "Secure Concept of SCADA Communication for Offshore Wind Energy," in Advances in Parallel and Distributed Computing and Ubiquitous Services, Springer, 2016, pp. 91-97

[8] R. Fan, L. Cheded and O. Toker, "Designing a SCADA system powered by Java and XML," Computing and Control Engineering, Vol. 16, No. 5, pp. 31-39, 2005

[9] D. Li, Y. Serizawa and M. Kiuchi, "Concept design for a Web-based supervisory control and data-acquisition (SCADA) system," IEEE/PES Transmission and Distribution Conference and Exhibition, Vol. 1, pp. 32-36, 2002

[10] M. Shahidehpour and Y. Wang, Control in Electric Power Systems, Hoboken, New Jersey: Wiley-Interscience, 2003

[11] B. Atlagic, D. Kukolj, V. Kovacevic and M. Popovic, "Application development environment of an integrated SCADA system," in The IEEE Region 8 EUROCON 2003. Computer as a Tool, Ljubljana, 2003

[12] D. Beck, H. Brand, C. Karagiannis and C. Rauth, "A new approach to object oriented programming for real-time targets," in 14th IEEE-NPSS Real Time Conference, Stockholm, Sweden, 2005

[13] Z. Qiang and C. Danyan, "The Research to Power SCADA Based on J2EE Framework," in WASE International Conference on Information Engineering, Taiyuan, China, 2009

[14] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright and K. Ramchandran, "Network Coding for Distributed Storage Systems" in IEEE Transactions On Information Theory, Vol. 56, No. 9, pp. 4539-4551, 2010

[15] J. Benet, "IPFS - Content Addressed, Versioned, P2P File System (DRAFT 3)", Whitepaper, https://ipfs.io

[16] S. McCrady, "Designing SCADA Application Software", Elsevier, 2013

[17] A. Patil et al, "Cloud Object Storage as a Service: IBM Cloud Object Storage from Theory to Practice", IBM Corp, 2017