Performances Analysis of a SCADA Architecture for Industrial Processes

Simona-Anda TCACIUC[1,2]
[1]Faculty of Electrical Engineering and Computer Science, Department of Computers
[2]Integrated Center for Research, Development and Innovation in Advanced Materials,
Nanotechnologies and Distributed Systems for Fabrication and Control (MANSiD)
Stefan cel Mare University of Suceava, Romania

***Abstract*—SCADA (Supervisory Control And Data Acquisition) systems are used to monitor and control various industrial processes, and have been continuously developed in order to incorporate the new technologies from software development and field busses areas. The middleware communication has the most relevant role in the development of such complex distributed systems as SCADA systems. These systems are very complex and must be reliable and predictable. Furthermore, their performance capabilities are very important. This paper presents a performance analysis of a SCADA system developed for Windows platform, including Windows Compact Embedded. The analysis is focused on the performance difference between computing systems based on Windows desktop and Windows CE operating systems. The utilization of the Windows CE is useful on the application with real-time requirements that cannot be achieved by the Windows desktop. Testing the application and analyzing the results led to the validation of the proposed SCADA system.**

Keywords—SCADA systems; middleware; data acquisition; data stream; distributed systems

I. Introduction

In the last decade, the SCADA systems that are found anywhere have been developing exponentially by including the new efficient and reliable technologies from the IT&C field. The SCADA applications are complex systems that acquired data from one or more local and remote processes. Furthermore, a SCADA system can control these processes by sending commands to the actuators depending on the type of process [1]. This system has the advantages of increases efficiency and profits and lowers costs.

A human operator can use this distributed system to remote monitor and control of the industrial processes, by using the HMI (Human Machine Interface) [2] provided by the SCADA system. This interface allows the handling of data related to that application (data acquired from the processes and the commands sent to the processes).

The most common areas of applicability for SCADA systems are the following: telecommunication systems [3], [4], power distribution systems [5], energy transmission systems [6], oil industry [7], gas and natural gas extraction, ore extraction systems, storage systems [8], irrigation systems [9], water distribution systems [10], sewerage systems, parameters measuring equipment for fluids, hydroelectric systems [11], monitoring systems for a production line or an entire company [12], monitoring systems for a building or a building complex, etc.

In this paper, it is analyzed a proposed new architecture of a SCADA system. The analysis focuses on the performance achieved with different data volumes and different operating systems.

Furthermore, this paper is organized as follows: Section II presents the most important specifications launched for the industrial field and the main manufacturers for the SCADA applications; Section III describes the architecture of the SCADA system; Section IV focuses on several considerations regarding the implementation; Section V highlights the experimental results obtained, whereas Section VI finalized with conclusions.

II. Related Work

Over the years, a wide variety of applications have been developed to achieve the data exchange between two devices or between a device and a software application using the client-server or peer-to-peer paradigm (M2M – Machine to Machine and D2D – Device to Device). These include: COM / DCOM (Component Object Model/Distributed Component Object Model) -based classic applications, applications based on the .NET platform, CORBA (Common Object Request Broker Architecture) - based applications, Web Services applications, Adaptive Communication Environment (ACE) applications, or applications based on other technologies.

In turn, the applications using the specifications launched by the OPC Foundation have a unified view [13]. These specifications are the following: OPC (Object Linking and Embedding for Process Control) Data Access, OPC Alarm and Events, OPC Historical Data Access, OPC Unified Architecture, OPC .NET (previously called OPC eXpress Interface - XI) [14]. Depending on the type of architecture they use, these specifications have been divided into: OPC classic – based on DCOM (it includes: OPC Data Access, OPC Alarm and Events, OPC Historical Data Access), OPC Unified Architecture and OPC .NET [15].

Due to this unified vision, the OPC specifications have become widely known and used by most of the production systems as the following: MES (Manufacturing Execution System) systems, HMI/SCADA systems and ERP (Enterprise Resource Planning) systems.

Currently, SCADA applications used in industry are being produced by many manufacturers, such as the following: Siemens, ABB, Emerson, Rockwell Software, Schneider Electric, Matrikon, B-Scada, Iconics, Intellution, Indusoft, Mitsubishi Electric, Yokogawa Electric, Honeywell, WonderWare, Omron, Citect, GE/Fanuc, USDATA, National Instruments, and Think & Do. However, each of the application developed by the manufacturers mentioned above, is targeted for certain issues and for some devices.

III. Architecture of the Proposed SCADA System

In order to implement a monitoring and control system of some industrial processes, the main functions that a SCADA application must consider are the following:

1) Data acquisition **–** It refers generally to acquiring data from a process, either as inputs or outputs, or even data that change during the process.

2) Communication/Data transfer over the network – It is related to the “transport” of data towards the Master station and vice versa (from the Master station towards the process). An important role is played by the communication network that must be well developed, flexible and easy to remodel.

3) Data presentation - It can be seen by the human user through a graphic interface located on the Master server. It presents an overview of the monitored process, it alerts the human operator if certain limits are exceeded, it processes the data collected from the process, it details the information at the command of the user, and keeps record of all the logs.

4) System Control - Represents the manual or automated configuration of the system, depending on the parameters and events that have been generated. This function can only be performed with the help of the human operator (the user), and the communication environment between the operator and the system is represented precisely by the interface of the SCADA system. This interface displays real-time process data, and the system is controlled by entering commands or messages for the process.

All these four important functions of a control and monitoring system intertwine at the most important level that is **middleware** level. In fact, this is where the **data exchange** is performed.

Starting from the proposed architecture presented by the Gaitan et al. in [16] the system is an industrial process to monitoring and control, designed and developed to meet the requirements of a complex system called the Metropolitan Heterogeneous System for Monitoring Utility-Specific Data (SMEDU).

The architecture of this system is presented in Fig. 1 and contains two software modules, namely: the client module and the server module; the middleware bus connects the two modules [17], [18].

The client module is an executable software module, called MCIP (Monitoring and Control of Industrial Processes) [19], after the role it has to fulfill within the proposed SCADA system. This is in fact a standard communication interface through which the user (the manager of an enterprise or a production line) can create and organize all the objects needed in the managed industrial process. The MCIP application presents two working modes:

- The first is the working mode, enabling the implementation and modification of the project. This working mode is performed when the application is unplugged from the servers. Basically, in the editing mode, the user can add as many graphic objects as he needs in the process, connecting them to the real devices to be monitored. Also, he can set certain features, and has the possibility to change or delete objects. The types of objects used are: graphic objects for displaying texts or images, template objects, control objects and middleware interfacing objects, through which the human operator can access the values in the process.
- The second working mode is the execution mode. In this case, the application executes what has been designed in the editing mode.

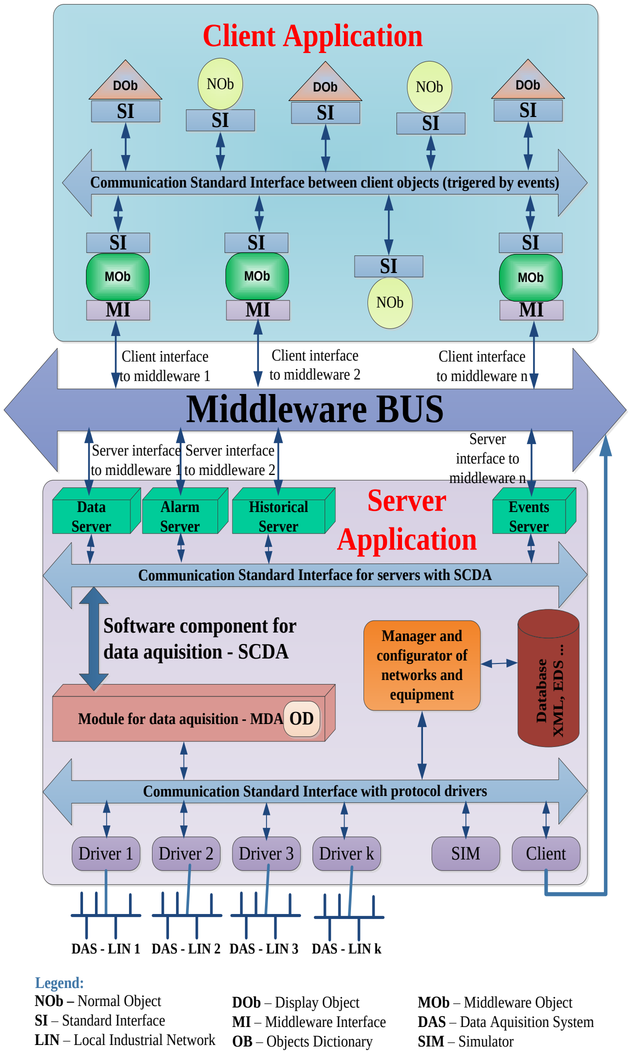
Client Application
DOb
NOb
DOb
NOb
DOb
SI
SI
SI
SI
SI
Communication Standard Interface between client objects (trigered by events)
SI
SI
SI
SI
MOb
MOb
MOb
MI
MI
MI
NOb
Client interface to middleware 1
Client interface to middleware 2
Client interface to middleware n
Middleware BUS
Server interface to middleware 1
Server interface to middleware 2
Server interface to middleware n
Data Server
Alarm Server
Historical Server
Server Application
Events Server
Communication Standard Interface for servers with SCDA
Software component for data aquisition - SCDA
Manager and configurator of networks and equipment
Database XML, EDS ...
Module for data aquisition - MDA
OD
Communication Standard Interface with protocol drivers
Driver 1
Driver 2
Driver 3
Driver k
SIM
Client
DAS - LIN 1
DAS - LIN 2
DAS - LIN 3
DAS - LIN k
Legend:
NOb – Normal Object
SI – Standard Interface
LIN – Local Industrial Network
DOb – Display Object
MI – Middleware Interface
OB – Objects Dictionary
MOb – Middleware Object
DAS – Data Aquisition System
SIM – Simulator

Fig. 1. Architecture for real-time heterogeneous distributed systems oriented SCADA applications [16].

The middleware bus is a software bus connecting the client mode to the server mode. Here, we can find some middleware technologies from the ones mentioned above, namely: COM/DCOM, CORBA, .NET technology, Web Services, but also others. These technologies must ensure the client-server communication in all possible cases.

The server module is also an executable software module that brings together, through standard communication interface, a variety of servers with the data acquisition software component, the object dictionary, the network manager, the database and the device drivers.

Within the server module, there may be several types of servers that use different specifications, but also serve a wide range of activities. For example, the most common types of servers are: data access servers, alarms and events servers and history servers.

The data dictionary, which can be found at the level of the data acquisition software, is nothing more than a hierarchical organization of the process data. Its most important feature is the uniformity of the data presentation, namely, it remains the same for any type of server used.

The server application also includes a standard interface for connecting to the database, as well as to the network manager.

IV. The Implementation of the Proposed System

As mentioned in the previous sections, the proposed system has been designed according to the client-server model and contains two software modules, namely: the MCIP client and the gpDAServer [19] data server.

Each of these has been designed so that in the end it could be easily installed and used by any device in the industry. Therefore, in the end, the two software modules will be executable.

The server module was developed in C++, and the MCIP client was developed as an application in C # (using the .NET platform).

When implementing both modules, the following aspects have been considered:

- A middleware object can connect to a single server placed on the same computer as the client application or any other computer in the network that supports the same middleware technology. If we are dealing with different computers, we have two possible cases: the two devices are in a LAN (local connection) or the two devices are connected to the Internet (this time, the connection is remote).
- The data provided by a server can be retrieved by as many middleware objects need that data.
- A server can connect to as many local industrial networks as possible.
- At the level of the data dictionary, the most important thing is the uniformity of the data presentation. In other words, all servers must have the same data and in the same mode.

V. Experimental Results

In order to validate the proposed system, a total of four tests have been performed on two similar star architectures. The differences between the two test architectures are the type of system used (real time or not) and the implementation mode for each individual case.

The first architecture consists of 7 test stations and a Master station (8 PCs with Windows XP/7/8/10 operating systems) connected in a TRENDnet TEG224WS+ switch, with a 100Mbps Ethernet interface. This test architecture is presented in Fig. 2.

The second test architecture consists of 6 embedded eBox 2300 SX systems, 1 embedded PDX 089T system with Windows CE real-time operating system, a Master desktop PC (the same as in the first test architecture) and a Super Stack II Baseline 10/100 Switch. This test architecture is presented in Fig. 3.

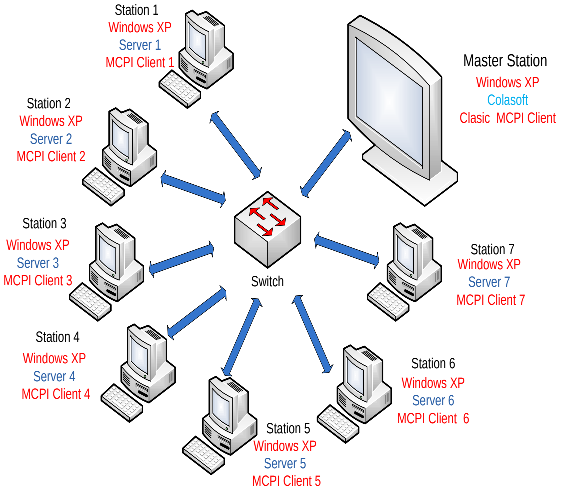

Fig. 2. Test architecture 1 [19].

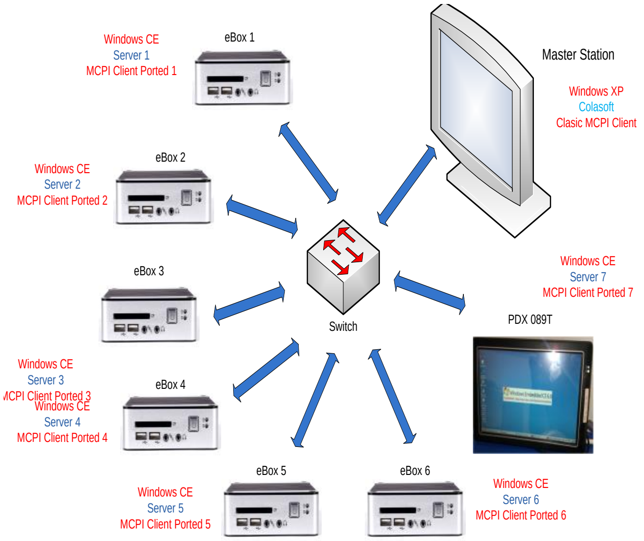

Fig. 3. Test architecture 2 [19].

In both cases, the Master station was used to monitor the flow of data sent to the local network. That is why the Colasoft Capsa Enterprise software (software developed for analyzing data flow) was installed on the Master station. It allows the interception and analysis of Ethernet frames in terms of transmission speed.

A MCIP client application and a server application have been installed on each of the other 7 systems (either the 7 PCs in the case of Test Architecture 1, or the 7 embedded eBOX or PDX embedded systems in the case of Test Architecture 2). These were launched in execution, and a process and the objects required for test have been added to each MCIP application. In this way, the client application has connected to the server through a OPC middleware object, thus managing to read or even write data. Through this object, it was possible to set an update rate required for reading and writing data.

In terms of transmission rates, the test range was of 100ms (0.1seconds) to 1000ms (1 second). Also, for each OPC object, the data transfer speed for 1 item, 5 items, 10 items and 15 items has been tested.

Fig. 4 shows the chart of experimental results obtained by testing the Test Architecture 1 at different refresh rates and for a different number of items in a group.

Similarly, tests have been performed for Test Architecture 2 and the experimental results are shown in Fig. 5.

In order to facilitate the analysis and interpretation of data obtained from the experimental tests, a comparison has been performed between the data resulted from the tests performed on the two operating systems (Windows XP and Windows Embedded CE), using identical parameters.

The analysis of the experimental results chart in Fig. 3 shows a tendency to decrease the data transfer speed once with the increase of the update rate.

The analysis of the experimental results chart in Fig. 4 also shows a tendency to decrease the data transfer speed once with the increase of the update rate.

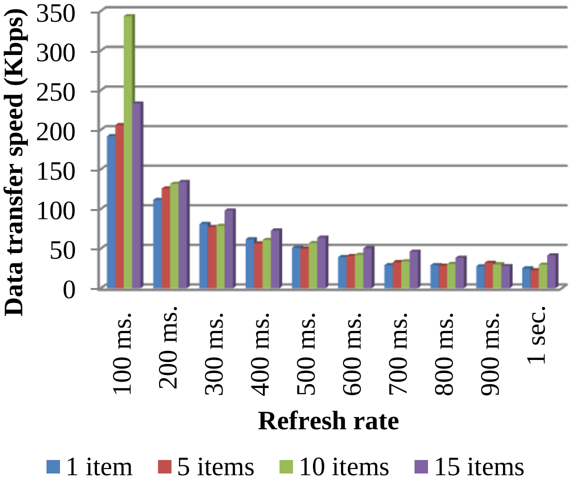

Fig. 4. Results diagram for test architecture 1.

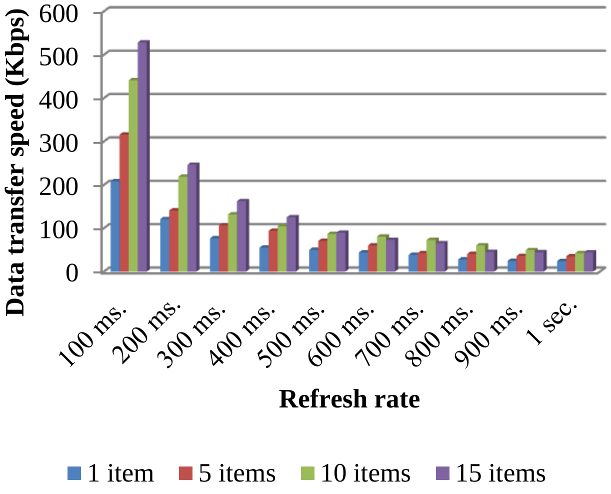

Fig. 5. Result diagram for test architecture 2.

Also, there is a minor difference between the data transfer speed of the two tested operating systems. This difference may be due to the fact that the server, at a certain point, might not perform exactly the update rate, because the CPU is overloaded. The update rate is performed on the server, because it calls a callback function in the MCIP client application.

Another explanation might be the implementation mode of the TCP/IP stack for the Windows Compact Embedded (CE) operating system. It can also be noticed that, at high update rates, the data transfer speeds in the case of Windows CE tests are close to the values obtained for Windows XP.

In conclusion, following the tests carried out on the two architectures and the analysis of the experimental results, the proposed model and solutions have been validated.

VI. Conclusions

The need to have more and more efficient and reliable systems led to the rapid development of SCADA systems. However, there is not yet an architecture pattern that can be used by all types of SCADA systems and that can meet all requirements.

The system proposed in this article attempts to satisfy these needs. This is why it has been designed to be easy to use and improvable by adding new components or features that might completely change the context.

This paper presented the performance analysis of a SCADA application that is executed on a computing device based on Windows desktop and Windows CE operating systems. We can conclude that the Windows CE can be used at the application with real-time requirements, which cannot be achieved by the computing systems based on Windows desktop.

As future work, we intend to develop the proposed system to approach other technologies. We also want to port the server application on a real-time system (Linux RTAI or Windows Compact Embedded).

REFERENCES

[1] P. Church, H. Mueller, C. Ryan, S. V. Gogouvitis, A. Goscinski and Z. Tari, “Migration of a SCADA system to IaaS clouds - a case study”, Journal of Cloud Computing: Advances, Systems and Application, DOI: 10.1186/s13677-017-0080-5, 2017.

[2] C. P. Carvajal, L. A. Solis, J. A. Tapia and V. H. Andaluz, “SCADA/HMI System for Learning Processes of Advanced Control Algorithms”, IT Convergence and Security 2017 (ICITS 2017), Lecture Notes in Electrical Engineering, vol 450. Springer, Singapore, DOI: 10.1007/978-981-10-6454-8, ISBN: 978-981-10-6453-1, 2017.

[3] S.M.Hosseini and M. Jahanian, “Enhancing security and risk control in SCADA systems”, International Conference on Engineering & Applied Sciences, Dubai, March 2016.

[4] R. Radvanovsky and J. Brodsky, “Handbook of SCADA Control Systems Security - Second Edition”, CRC Press - Taylor & Francis Group, ISBN-13:978-1-4987-1708-3, 2016.

[5] R. Jackson Tom and S. Sankaranarayanan, “IoT based SCADA integrated with Fog for power distribution automation”, The 12th Iberian Conference on Information Systems and Technologies (CISTI), DOI: 10.23919/CISTI.2017.7975732, Electronic ISBN: 978-9-8998-4347-9, 2017.

[6] S. Sebastio, G. D'Agostino and A. Scala, “Adopting the cloud to manage the electricity grid”, The IEEE International Energy Conference (ENERGYCON), DOI: 10.1109/ENERGYCON.2016.7513996, ISBN: 978-1-4673-8464-3, 2016.

[7] M. Liu, M. Yuan, F. Wang and C. Sun, “The Oil and Gas Pipiline Clouding SCADA System and Multiple Data Centers Storage System Design”, The International Conference on Manufacturing Construction and Energy Engineering (MCEE), ISBN: 978-1-60595-374-8, pp. 293-297, 2016.

[8] L. A. Sabri, S. A. Mohammed and M. I. Issa, “Design of SCADA System for Oil Pipeline Control Using LabVIEW”, Al-Nahrain Journal for Engineering Scinces (NJES), Vol. 20, No. 3, pp. 600-614, 2017.

[9] E. Zaev and D. Babunski, “SCADA system for real-time measuring and evaluation of river water quality”, The 5th Mediterranean Conference on Embedded Computing (MECO), Montenegro, DOI 10.1109/MECO.2016.7525708, ISBN: 978-1-5090-2223-6, 2016.

[10] C. M. Ahmed, V. R. Palleti and Aditya P. Mathur, “WADI: a water distribution testbed for research in the design of secure cyber physical systems”, Proceedings of the 3rd International Workshop on Cyber-Physical Systems for Smart Water Networks, ISBN: 978-1-4503-4975-8, pp. 25-28, 2017.

[11] F. M. Enescu and N. Bizon, “SCADA Applications for Electric Power System”, Reactive Power Control in AC Power Systems, Springer, ISBN: 978-3-319-51117-7, DOI: 10.1007/978-3-319-51118-4, pp.561-609, 2017.

[12] J. A. Ruiz Carmona, J. C. Munoz Benitez, J. L. Garcia-Gervacio, “SCADA system design: A proposal for optimizing a production line”, The International Conference on Electronics, Communications and Computers (CONIELECOMP), Mexico, DOI: 10.1109/ CONIELECOMP.2016.7438574, Electronic ISBN: 978-5090-0079-1, 2016.

[13] I. Gonzales, A. J. Calderon, A. J. Barragan and J. M. Andujar, “Integration of Sensors, Controllers and Instruments Using a Novel OPC Architecture”, MDPI-Sensors, No. 17(7), 1512, DOI:10.3390/s17071512, 2017.

[14] N.C. Gaitan, , “Real-time Acquisition of the Distributed Data by using an Intelligent System”, Electronics and Electrical Engineering - Kaunas Technologija, Vol. 8(104), 2010, pp. 13-18.

[15] M. F. Rohani, N. A. Ahmed, S. Sahibudding and S. M. Daud, “OPC Protocol Application for Real-Time Carbon Monitoring System for Industrial Environment”, International Journal of Electrical and Computer Engineering (IJECE), Vol. 7, No. 2, ISSN: 2088-8708, DOI: 11591/ijece.v7i2.pp1051-1059, pp. 1051-1059, 2017.

[16] N. C. Gaitan, V. G. Gaitan, S. G. Pentiuc, I. Ungurean and E. Dodiu, “Middleware Based Model of Heterogeneous Systems for SCADA Distributed Applications” in Advances in Electrical and Computer Engineering, vol. 10, No. 2, 2010, pp. 121–124.

[17] V. G. Gaitan, M. G. Danila, M. Robu and C. N. Gaitan, “MCIP - a HMI Application for Monitoring and Controlling the Industrial Processes”, The 9th International Conference on Development and Application Systems, ISSN 1844-5020, 2008, pp. 32-37.

[18] N. C. Gaitan, V. G. Gaitan and I. Ungurean, “An IoT Middleware Framework for Industrial Applications”, International Journal of Advanced Computer Science and Applications, vol. 7, Issue 9, 2016, pp. 31-41.

[19] S. A.Gherasim and V. G. Gaitan, “Performance Evaluation of a SCADA System for Monitoring and Controlling the Industrial Processes”, Buletinul Institutului Politehnic din Iasi, Automatic Control and Computer Science Section, vol. 1, 2013, pp. 9-20.