Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

A Dynamic Connection Scheme for User Interface of Process Control System in Offshore Plant

Daekeun Moon*, Hagbae Kim, Jinho Park* and Dongho Park***

** Electro-Mechanical Research Institute, Hyundai Heavy Industries Co., Ltd., Ulsan, KOREA (Tel: +82-31-289-5240; e-mail: {dkmoon, jkan, dhpark}@hhi.co.kr).*
*** Electrical and Electronic Engineering Department, Yonsei University, Seoul, KOREA (e-mail: hbkim@yonsei.ac.kr)*

Abstract: In offshore plant, the process control system manages process variables and interfaces with all plant utilities. It consists of workstations, controllers and various field interface devices. Plant operators deal with a large variety of process variables using the Human-Machine Interface application (HMI) in operator stations, which is a software program to interact with plant operators and to communicate with the system server. In large-scale processes, the process control system has not only multiple system servers for system availability but also many operator stations for plant operators to manage easily the process. The main focus of our work is in a connection scheme of the HMI for linking to one of system servers without additional configuration works. The proposed scheme is adopted the fundamental concepts of autonomic computing and supports fault-tolerant and load-balanced features because it provides a dynamic connection according to the status of system servers. We implement the module for the proposed scheme as agent between the system server and the HMI, and then incorporate it into the process control system. Finally, the case study shows that the proposed scheme can provide reliable connection with system servers and efficient load status of system servers.

1. INTRODUCTION

Process control systems that monitor, diagnose and control process variables such as pressure, flow and temperature have been implemented for various processes. Especially, in offshore plant, the process is a typical offshore application with oil-gas-water separation, processing and treatment (Gobrick and Legge 1994). Additionally, all plant utilities are controlled from the process control system. The process control system is based on distributed control system hardware and has multiple interfaces to cooperate with other control systems.

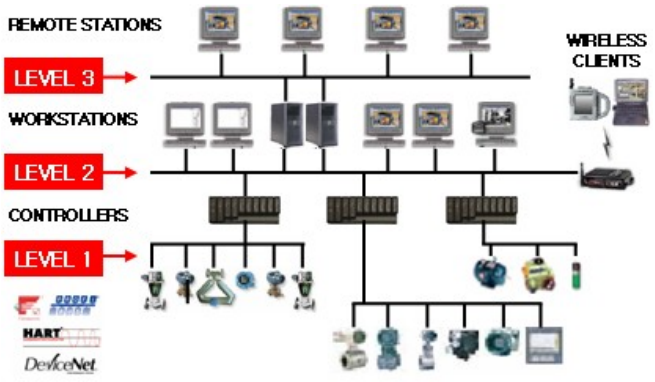


Fig. 1. Process Control System

In general, process control systems consist of workstations, controllers and various field interface devices. They have three network layers (Nixon et al. 2005) as shown in Figure 1. Level one is between the field devices and the controllers and runs the process. It is related to a direct control and adaptation, and has tight real-time requirement of high predictability and reliability. Level two is among the controllers and the workstations. It supports user interaction that includes configuration, control and monitoring according to the control strategies and coordination. It has less timing requirement than level one, but still requires good reliability. Level three is from the workstations to the outside world. It is the gateway of the control system to other corporate systems such as accounting, inventory and management decision systems. In the paper, we focus especially on level two and three in the interface viewpoint among workstations.

Plant operators deal with a large variety of process variables generated by the process control systems like the tasks of monitoring a process and assessing its current state, detecting and diagnosing any abnormal behavior, and taking appropriate control actions. Therefore, to assist the plant operators, process operational information must be presented in a manner that reflects the important underlying trends or events in the process through user-friendly interface applications provided by workstations. The type of workstations can be classified four categories: engineering station, system server, operator station and application-specific station. Engineering stations are for configuring system structure, I/O points and control algorithms. System servers are the key stations of workstation-level system in run-time environments. They collect all information from controllers, provide it to other stations and process various system or user commands. Operator stations display the information of the plant using a Human-Machine Interface application (HMI), which is a software program to interact

with plant operators and to communicate with the system server. The HMI receives different types of information from the system server and presents process-critical information in a timely manner for plant operators. Application-specific stations are defined by their main functions such as history management of process data, gateway for system extension, interface with other systems, and so on.

In large-scale processes, the process control system has not only multiple system servers for system availability but also many operator stations for plant operators to manage easily the process. In general, each operator station requires additional configuration works to connect with the system server. However, the additional configuration works involving human intervention have limitations on the accuracy and efficiency of the system configuration. With regard to all system problems, about 40% are attributable to errors made by human (Park et al. 2005). In addition, the usage of wireless devices by the rapid growth of wireless technology such as a handheld computer motivates us to devise a new scheme that needs no additional configuration work.

The main focus of our work is in a connection scheme of the HMI for linking to one of system servers without additional configuration works. The proposed scheme is adopted the fundamental concepts of autonomic computing (Kephart and Chess 2003) and supports fault-tolerant and load-balanced features because it provides a dynamic connection according to the status of system servers. It consists of system server discovery and selection algorithm parts. The system server discovery part is enabled to search system servers in the process control system when the HMI is initialized, and the selection algorithm part is used to determine the system server that can be provide the optimal connectivity. We implement the module for the proposed scheme as agent between the system server and the HMI, which provides not only process data but also alarm and event list in a timely manner, and then incorporate it into the process control system. Finally, the case study shows that the proposed scheme can provide reliable connection with system servers and efficient load status of system servers.

The rest of the paper is organized as follows. Section 2 briefly reviews the concept of autonomic computing and the related works. Section 3 gives an overview of the proposed dynamic connection scheme. Section 4 explains how it is implemented in the process control system. Section 5 presents the case study through an implemented prototype. Finally, section 6 concludes the paper and outlines some future research directions.

2. AUTONOMIC COMPUTING

Modern process control systems have the difficulties of their further development because of the complexity, heterogeneity and uncertainty. The autonomic computing means embedding intelligent control into the system infrastructure itself in order to automate everything in a self-managing manner. The goal of self-management is not only to free system administrators from the details of system operation and maintenance but also to provide users with a machine that runs at peak performance. It has features of self-configuration, self-healing, self-optimizing and self-protecting (Kephart and Chess 2003).

Self-configuration enables systems to set their own configurations and to adapt to changing conditions by adjusting them. It is important because installing and configuring process control systems is time-consuming and error-prone even for experts. Self-healing has the functions that detect, diagnose and repair failures in software and hardware. It makes systems more reliable through initiating corrective actions for eliminating failures automatically. Self-Optimizing is the ability to make systems more efficient in performance. It is accomplished by monitoring the state of systems and tuning their parameters. Self-protecting provides the right information to users based on the user's role and pre-established policies. It makes systems less vulnerable to unauthorized access and use, malicious attacks and cascading failures. Table 1 describes features for autonomic computing with the limitation of current computing.

Table 1. Features for autonomic computing

Limitation of current computing	Autonomic computing
Installation and configuration work is time-consuming and error-prone.	Self-configuration enables systems to set their own configurations and to adapt to changing conditions.
Problems in large, complex system are not easy to manage manually.	Self-healing makes systems more reliable through initiating corrective actions for eliminating failures automatically.
Systems have many configurations and tuning parameters, and their number increases with each release.	Self-Optimizing makes systems more efficient in performance by monitoring the state of systems and tuning their parameters.
Detection of and recovery from attacks and cascading failures is manual.	Self-protecting provides the right information to users and makes systems less vulnerable to attacks and cascading failures.

The concept of autonomic computing has been applied in various fields. Trumler et al. (2003) proposed a Smart Doorplate based on autonomic computing for a distributed system, which satisfies the demands for self-configuration, self-healing, context awareness and anticipatory. It configures and heals itself during runtime by exchanging messages over the peer-to-peer network. Whiteson and Stone (2004) investigated an adaptive network routing and scheduling toward autonomic computing. They proposed learning-based methods for addressing the problems of packet routing and CPU scheduling in computer networks. The results show clearly that machine learning methods offer a significant advantage in self-optimizing the performance of complicated networks. Chang et al. (2004) provided a reinforcement learning approach in the mobilized ad hoc network by applying autonomic computing principles, which is an adaptive algorithm for controlling movement and routing for ad hoc networks through self-optimizing and adaptive learning (self-configuration). Shen et al. (2005) proposed a framework for self-management in hybrid

wireless network based on autonomic computing principles. They outlined the autonomic computing background and its requirements for self-configuration and self-optimization as a first step toward autonomic communications.

3. DYNAMIC CONNECTION SCHEME

A dynamic connection scheme consists of two parts: system server discovery and selection algorithm parts. The system server discovery part is enabled to search system servers in the process control system when the HMI is initialized, and the selection algorithm part is used to determine the system server that can be provide the optimal connectivity.

The system server discovery part uses a broadcast mechanism; in other words, it starts to broadcast a system server request message (SSREQ) to system servers. When each system server receives the SSREQ, it returns a system server reply message (SSREP). The system server discovery part collects the SSREP and demands the optimal system server to the selection algorithm part. Then, it is completed by establishing the connection with the system server chosen by the selection algorithm part.

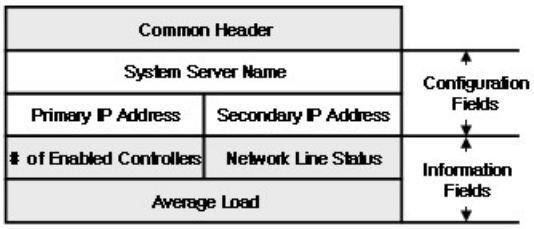


Fig. 2. System Server Reply Message (SSREP)

The SSREP has the configuration fields and the information fields as shown in Figure 2. The configuration fields include name, primary IP address and secondary IP address of the system server. The information fields include the number of enabled controllers, the average load and the network line status. The number of enabled controllers is related to the area of the plant that the system server can monitor and control. The more enabled controllers are connected to the system server, the larger area of the plant can be managed. The difference of the number of enabled controllers can be caused by the instability or the partial failure of the network. The average load of the system server is defined by the interface rate of the clients per a minute and is proportional to the clients connected to the system server. If the average load is high, the system server can be considered as a relatively overloaded status. The network line status is a factor that affects to the probability of network failure in the future.

The selection algorithm part uses the SSREP collected in the system server discovery part. The selection algorithm is based on the information fields of the SSREP as follows. First, it compares the number of enabled controllers of the SSREP, and then selects the system server with the largest number. Second, if the number of system servers with the largest number is more than one, it compares the average load among them, and then selects the system server with the lowest load. Third, if the number of system servers with the lowest load is also more than one, it compares the network line status among them, and then selects the system server with the best status. Last, if there are system servers whose all conditions are equal, it selects one randomly. The pseudo code of the selection algorithm is shown in Figure 3.

```
read the SSREP;
compare the number of enabled controllers;
collect the replies with the largest number;
if the number of collected replies is one
  select the system server from it;
else
  compare the average load for collected replies;
  collect the replies with the lowest average load;
  if the number of collected replies is one
    select the system server from it;
  else
    compare the network line status for collected replies;
    collect the replies with the best network line status;
    if the number of collected replies is one
      select the system server from it;
    else
      select the system randomly among collected replies;
    endif
  endif
endif
```

Fig. 3. Selection Algorithm

The proposed scheme is adopted the concepts of autonomic computing, which is computing systems that can manage themselves given high-level objectives from administrators. It searches system servers and chooses the optimal one based on the status of them. This operation supports self-configuration of operator stations and self-optimization of system servers because it offers a dynamic connection in a load-balanced manner. In addition, when the connection failure occurs, the proposed scheme detects it, clears the previous connection information, and makes a new connection automatically. This is a fault-tolerant feature of the scheme supporting self-healing. Finally, self-protection is considered in implementation phase through the blocking strategy that provides services if and only if a connection is determined completely.

4. DEVELOPMENT OF USER-INTERFACE AGENT

A software component is considered an agent if at least the characteristics autonomy, intelligence, capability of taking action, communication and mobility are fulfilled, although different levels of development are possible for each (Meissner and Hensel 2005). An agent can execute pre-processing and independent actions that must benefit the user to finish the assigned goals.

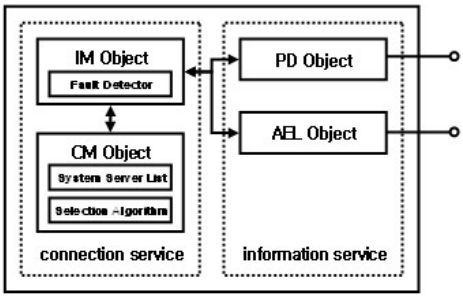


Fig. 4. Architecture of User-Interface Agent

The user-interface agent proposed in the paper is a module that connects the system server and the HMI, which provides not only process data but also alarm and event list in a timely manner. The roles of the user-interface agent are categorized into two kinds of services: connection service and information service. Figure 4 shows the architecture of the user-interface agent.

The connection service consists of Connection Manager (CM) object and Interface Manager (IM) object. The CM object is responsible for communicating with the system server practically. Also, it has the functions that can detect faults according to changing the network conditions. The IM object is the key object for our dynamic connection scheme. It manages the system server list through the system server discovery procedure and includes the selection algorithm.

The information service is supported by Process Data (PD) object and Alarm/Event List (AEL) object. The PD object provides the information of the process control system to the HMI so that plant operators can monitor and control the plant. The information is based on process data, system status data and system option data. Process data has the parameters such as type, description, value and function. System status data is divided into workstation-level, controller-level and device-level. System option data includes user setting values for operating the process control system. In addition, the PD object is the intermediate object for sending user commands to the system server in order to control the process and/or change the information manually. The AEL object provides both the current and the historical alarm/event lists in the process control system. Furthermore, it offers the SOE (Sequence of Event) list. These lists can be sorted by the selected field and classified by the predefined alarm/event group.

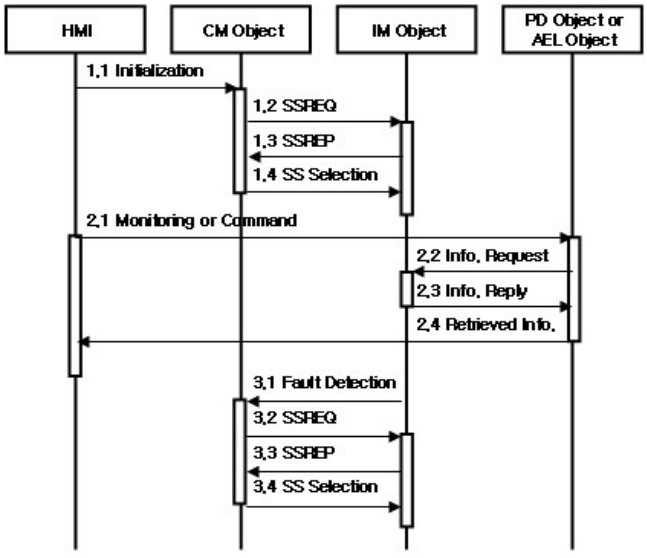


Fig. 5. Message Sequence Diagram

When the HMI is initialized, the CM object is initiated by requesting the system server discovery to the IM object. Then, the IM object broadcasts the SSREQ and waits the SSREP from system servers for a pre-defined time. At this time, the PD object and the AEL object block the requests of the HMI for getting information. After a waiting time, the IM object transfers the received SSREP to the CM object. The CM object chooses an optimal system server through the selection algorithm and inform the result to the IM object, the PD object and the AEL object. Finally, the PD object and the AEL object release the blocking condition of the requests of the HMI and provide information service. The recovery procedure for repairing the connection failure is the same as this procedure. Figure 5 shows the message sequence diagram of the procedure explained above.

5. CASE STUDY

Our user-interface agent with a dynamic connection scheme has been applied to HiMAX-2000PLUS®, which is the process control system developed by Hyundai Heavy Industries Co., Ltd, and its basic functions are tested. This section describes the remarkable features of our agent through the case study. This case study concentrates on fault-tolerant and load-balanced features and is compared with OPC (OLE for Process Control). OPC is a technical standard for data exchanging not only between the hardware drives and its application programs but also among different systems (OPC Foundation n.d.).

We considered the process control system that has three system servers connected with some operator workstations individually. In addition, we assumed that the conditions of each system server are same if they are not mentioned explicitly. The notation used in this case study is denoted as follows; CN_i is the connected controller number of i^{th} system server, AL_i is the average load of i^{th} system server, and OL_j is the added load to the system server by j^{th} operator workstation. Therefore,

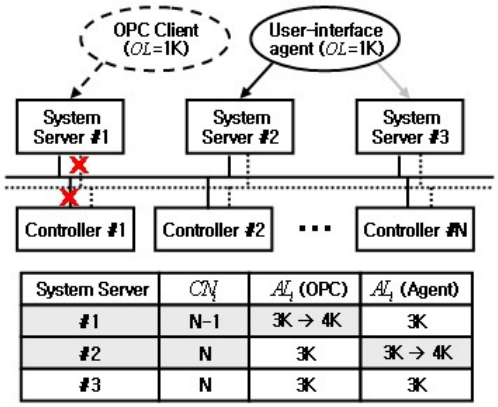
$$AL_i = \sum_{j=1}^{n} OL_j$$

where, n is the number of operator workstations connected with i^{th} system server.

5.1 Scenario A: Different Connected Controller Number

First, we consider a scenario that system servers have different connected controller number as shown in Figure 6; where the number of controllers is N, the initial AL_i of each system server is 3K, and the OL by new connection is 1K. This situation can occur due to instability of system network. Though this is a situation to appear rarely, it may affect the system seriously. In this scenario, if new OPC client is configured to connect with system server #1, its HMI will not display process data provided by controller #1 because system server #1 can not communicate with controller #1. However, our user-interface agent tries to connect with either system server #2 or #3 through the selection algorithm. Therefore, its HMI will display all process data of the plant. This means that our agent can provide the reliable

information of the plant in presence of system servers with different connected controller number.

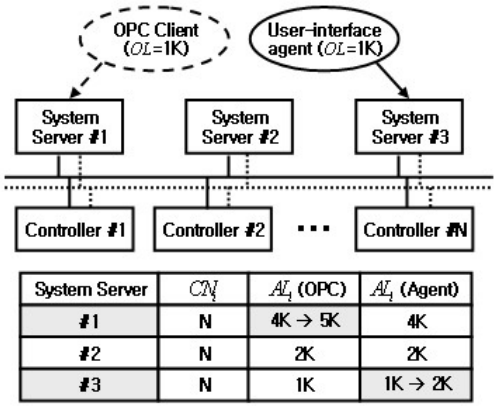


System Server	CN_i	AL_i (OPC)	AL_i (Agent)
#1	N-1	3K → 4K	3K
#2	N	3K	3K → 4K
#3	N	3K	3K

Fig. 6. Scenario A

5.2 Scenario B: Different Average Load

Second, we consider a scenario that system servers have different average load as shown in Figure 7; where the number of controllers is *N*, the CN_i of each system server is *N*, the initial AL_i of system servers are 4K, 2K and 1K respectively, and the OL by new connection is 1K. This situation can frequently occur according to the system conditions. In this scenario, if new OPC client is configured to connect with system server #1, system server #1 will be overloaded against others. However, our user-interface agent will connect with system server #3 because it had the lowest average load. Then, system servers will maintain the balanced state of system servers. This means that our agent can present the load-balanced feature of system servers in the situation that system servers have different average load.



System Server	CN_i	AL_i (OPC)	AL_i (Agent)
#1	N	4K → 5K	4K
#2	N	2K	2K
#3	N	1K	1K → 2K

Fig. 7. Scenario B

5.3 Scenario C: System Server Failure

Finally, we consider that system server #1 connected with OPC client or our user-interface agent failed as shown in Figure 8. In this scenario, if OPC client is configured to connect with system server #1 only, its HMI will not display any process data because it can not maintain the connection with the system server any more. However, our user-interface agent will try to connect with either system server #2 or #3 through re-initialization procedure for new connection. Then, its HMI will continue to display all process data of the plant. This means that our agent can present high availability because it has a fault-tolerant feature that can continue to monitor and control the plant in spite of the failure of any system server without additional configuration works.

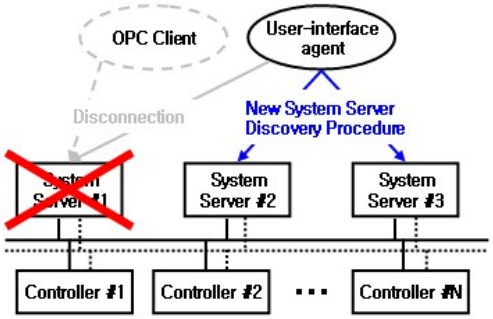


Fig. 8. Scenario C

In addition, we can consider that OPC client is predefined to connect with system servers in the order. In this case, OPC client can continue to provide process data to its HMI through reconnection with other system server. Nevertheless, our agent has the performance that is equal or superior to that of OPC client because it covers the situation such as scenario A and B.

6. CONCLUSIONS

This paper proposed a dynamic connection scheme of the HMI for linking to one of system servers in the process control system. Because it is based on autonomic computing principles, it configures and heals itself during runtime by exchanging simple messages without additional configuration works. Next, we implemented the user-interface agent for our scheme between the system server and the HMI, and incorporated it into the process control system. Finally, we proved that our scheme can improve both the availability of the HMI and the stability of the system server through the case study.

For future work, we will verify the usefulness of our scheme in the practical plant. In addition, we need to take security issues of the system into consideration. Finally, we will expand our concern about autonomic computing to controllers and devices in the process control system.

REFERENCES

Chang, Y., T. Ho, and L.P. Kaelbling (2004). Mobilized ad-hoc networks: A reinforcement learning approach. *Proc. International Conference on Autonomic Computing (ICAC)*, pp.240-247.

Gobrick, R. and N. Legge (1994). HlBERNlA - THE NEXT GENERATION OF OFFSHORE PLATFORM CONTROL SYSTEMS. Petroleum and Chemical Industry Conference (PCIC), pp.197-204.

Kephart, J.O. and D.M. Chess (2003). The Vision of Autonomic Computing. *Computer*. **Vol.36**, **No.1**, pp.41-50.

Meissner, K. and H. Hensel (2005). Agent Based Assistance- and Support in Process Control Systems. *18th International Conference on Systems Engineering (ICSEng)*, pp.159-163.

Nixon, M., R. Shepard, A.K. Mok, B. Bennett, and D. Chen (2005). Process control adopts wireless. *InTech*.

OPC Foundation, <www.opcfoundation.org>

Park, J., G. Yoo and E. Lee (2005). Proactive Self-Healing System based on Multi-Agent Technologies. *Proc. 3rd ACIS International Conference on Software Engineering Research, Management and Applications (SERA)*, pp.256-263.

Shen, C., D. Pesch, and J. Irvine (2005). A Framework for Self-Management of Hybrid Wireless Networks Using Autonomic Computing Principles. *Proc. the 3rd Annual Communication Networks and Services Research Conference (CNSR)*, pp.261-266.

Trumler, W., F. Bagci, J. Petzold, and T. Ungerer (2003). Smart Doorplates - Toward an Autonomic Computing System. *Autonomic Computing Workshop*, pp.42-47.

Whiteson, S. and P. Stone (2004). Towards Autonomic Computing: Adaptive Network Routing and Scheduling. *Proc. International Conference on Autonomic Computing (ICAC)*, pp.286-287.