Remote maintenance of control system performance over the internet

Published in Control Engineering Practice, 15(5), pp.533-544, 2007

Published in Control Engineering Practice, 15(5), pp.533-544, 2007

Remote Maintenance of Control System Performance over the Internet

Shuang Hua Yang*, Chengwei Dai, and Roger P. Knott

Computer Science Department, Holywell Park,

Loughborough University, Loughborough, Leicestershire LE11 3TU United Kingdom

Abstract: The Internet provides a significant benefit for the remote maintenance and fault diagnosis of various devices and plants. One such example is the UK's distributed aircraft maintenance environment (DAME) (www.cs.york.ac.uk/dame), which provides a genetic test bed for distributed diagnostics based on Grid-enabled technologies. This paper focuses on developing a systematic method for the design of such a remote maintenance systems specifically for process control systems. Design issues of Internet-based remote maintenance systems for process control such as that proposed here include control performance assessment, fault detection, control performance maintenance, and heterogeneous data transfer over the Internet. A back-end and front-end architecture is proposed, in which all the heavy calculations are carried out locally. Light data and the characteristics of any heavy data are sent to the front-end located on the remote server for consideration by remote experts. The remote maintenance system is illustrated by reference to the implementation in a process control rig.

Keywords: Remote maintenance; Internet; Control performance assessment; Process control; Fault compensation.

* Corresponding author: Prof. S H Yang, Professor of networks and control. Telephone: 0044 1509 635670, Fax: 0044 1509 635722. Email: s.h.yang@lboro.ac.uk

1. Introduction

Over the last two decades many thousand Advanced Process Control (APC) systems have been installed in process plants. The challenge is to maintain the optimum performance of existing APC systems. The motivation for building remote maintenance systems is to provide efficient support for these control systems, which are characterized as being distributed either within the process plants or, in some occasions, geographically. Remote maintenance systems, enables companies, with multiple sites in remote locations, to access, analyse, and react to information from the plant floor more quickly and efficiently than previously. It is also enables SME to delegate such maintenance to a service company or software supplier without the need for any internal experts. It can also virtually eliminate the need for any control software supplier's expert to conduct 'on-site maintenance.' Therefore both time and money can both be saved.

Past research on ScadaOnWeb (supervisory control and data acquisition on web) (ScadaOnWeb, 2002), remote access to industrial plants (Calvo, et al., 2006), design of Internet-based control (Yang et al., 2003; Yang et al., 2005), e-Diagnostics (Sematech, 2002), and performance monitoring (Huang & Shah, 1999; Harris et al., 1999) has done much to investigate the best methods to design an Internet-enabled system, to locally monitor the performance of processes and control systems, and to remotely access industrial plants and identify malfunction of equipments and auxiliary hardware. DAME (distributed aircraft maintenance environment) (DAME, 2002) demonstrates how Grid technology can facilitate the design and development of decision support systems for diagnosis and maintenance, in a situation where geographically distributed resources, actors and data are combined within a virtual organization. What it seems not yet been considered is the best means to maintain several Internet-enabled control system software components from a remote location. People still appear to know little about how to identify any control system software components which is in need of repaired or updating, how to maintain control system software with minimum disturbance to the controlled processes, and how to share the maintenance experience and reduce the costs between companies. People continue to see control system supplier's field service engineers wasting time and money on their worldwide travel in order to rectify a simple fault which occurred in their software product.

Much is now known about various components of the problem. The ScadaOnWeb system (ScadaOnWeb, 2002) targets a new standard and a generic architecture for handling numeric data on the web and enabling process control, monitoring, and optimisation via the web. Design of Internet-based control (Yang et al., 2003; Yang et al., 2005) focuses on dealing with architecture selection, Internet latency, multiple user access, and security and safety. Wireless and wired communications technologies have been used in remote health monitoring and control

of aircraft (Thompson, 2004). E-Diagnostics (Sematech, 2002) enables an authorized equipment supplier's field service person to access and diagnose equipments from outside. The Heath and Safety Executive (HSE, 2002) addresses the health and safety issues associated with the use of the Internet and Internet related technologies in and by industry, and also provided guidance (HSE, 1995) on the safe use of remote diagnostic equipment fitted to computer-controlled machinery and associated equipment. Various approaches to monitor process performance (Huang & Shah, 1999) are available. Performance monitoring for control systems (Harris, et al., 1999; Nougues et al., 2002) provides means to benchmark the performance of any system against the 'best in class' performance, and enables users to diagnose possible problems so that the appropriate corrective actions can be taken. A comprehensive overview of control performance assessment technology and industrial applications (Jelali, 2006) was recently made available.

This paper investigates design issues for Internet based maintenance systems for process control. These maintenance systems are divided to two parts, a back-end and a front-end. All the heavy calculations are carried out in the back-end that is located locally. All functions relevant to the use by remote operators are implemented in the front-end that is located remotely. Any remote maintenance system must have only one back-end, but can, if necessary, have multiple front-ends. The design issues of the system include performance assessment, fault detection, degraded control performance recovery, and data transfer over the Internet. A process control rig in our laboratory is used as a case study to implement and illustrate the proposed maintenance system.

The rest of this paper is organized as follows: In Section 2, the methodologies of control performance assessment, fault detection and compensator design are briefly described. In section 3 the structure of the remote maintenance system is presented. This consists of a back-end and a front-end system. The details of the back-end and the front-end system are presented in Sections 4 and 5 respectively. Section 6 discusses the implementation of the remote maintenance system for our process control rig. Finally, Section 7 contains the conclusions of this work,

2. Methodologies

2.1 Control Performance Monitoring

Control performance has been an active research field over several decades and usually refers to how well a control system and the process it controls work together in a single loop. A number of indexes have been used to

measure control performance. These include, amongst other, the Integral of Absolute Error (IAE), the Mean Squared Error (MSE). In this work, a control performance index is defined as the ratio between the actual cost functions of the control system and a desirable Linear Quadratic Gaussian (LQG) controller.

$$\eta = \frac{J_{LQG}}{J_{act}} \tag{1}$$

where J_{LQG} is the cost function of the LQG controller and J_{act} is the cost function of the actual controller.

The LQG controller is an unconstrained controller and provides the best achievable nominal performance among the class of all stabilizing controllers (Patwardhan et al., 2002). Any cost of the actual controller will be greater than the LQG's cost and the index takes a value between 0 and 1 ($0 \le \eta \le 1$). The index has a value of 1 at the perfect performance and 0 at the worst.

2.2 Fault Detection

The LQG controller is also utilised as a benchmark or a reference model for the target control system. The LQG controller provides a useful lower bound of the performance achievable by a linear controller when the actual control system is in a healthy state. If both the controllers are applied to the same process, discrepancies between the outputs of the actual controller and the LQG controller will be stable; any fault occurring in the actual controller will make the discrepancies vary significantly (Dai et al., 2004). A General Likelihood Ratio (GLR) test is employed to monitor the discrepancies in order to detect the faulty controller as early as possible. Providing the observed actual control signal $u_{act}(k)$, and the benchmark LQG control signal $u_{LQG}(k)$, the discrepancies between the two signals $\Delta u(k) = u_{act}(k) - u_{LQG}(k)$ where k is the current time instant, follows a Gaussian distribution with mean μ_0 and variance σ_0 as the number of observations becomes large (Calkins, 2005).

The measurement of $\Delta u(k)$ is denoted as $r(k)$ as below:

$$r(k) = \Delta u(k) + r_0(k)$$

where r_0 is a white noise following a Gaussian distribution with zero mean.

The GLR test is used to calculate the likelihood ratio s_k of $r(k)$ as:

$$s_k = \frac{\mu_1 - \mu_0}{\sigma_0^2}\left(r(k) - \frac{\mu_0 + \mu_1}{2}\right) \tag{2}$$

where μ_1 is the calculated mean of data in a moving observation window. Fault is identified using a predefined threshold λ as follows:

$s_k > \lambda$: a fault is detected;

$s_k \le \lambda$: No fault is detected and test is continued.

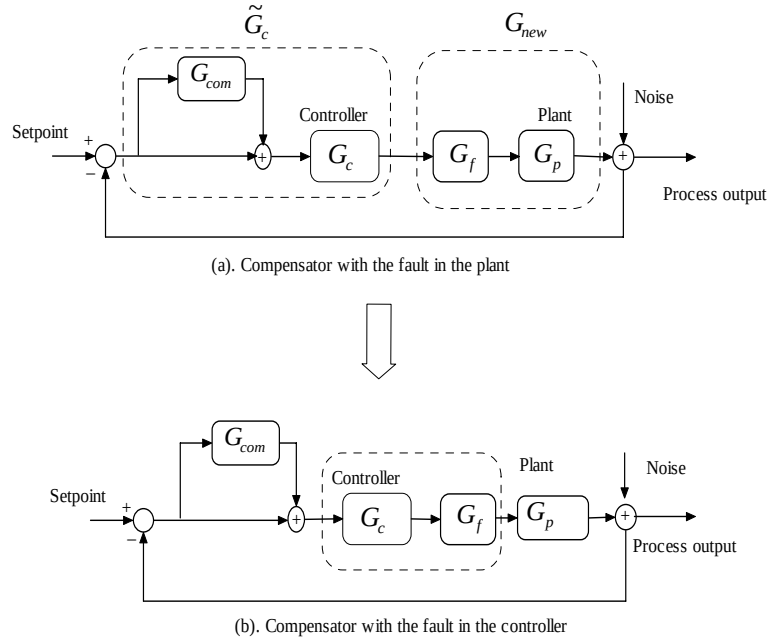

Fig. 1: Closed loop with a compensator (Dai and Yang, 2004).

2.3 Compensation Design

A compensator is designed to work together with the faulty controller to stabilize the control process and restore the degraded performance. Consider the feedback control system shown in Fig. 1(a), in which a model of the controller is $G_c(z)$, and $G_p(z)$ is a plant model. Assume that a fault occurs in the controller and the fault model is described by $G_f(z)$, $m(z)$ is the identified model for the normal open loop and $m_f(z)$ is the identified model for the faulty open loop.

$$\begin{cases} m(z) \approx G_p(z)G_c(z) \\ m_f(z) \approx G_p(z)G_f(z)G_c(z) \end{cases} \tag{3}$$

$G_f(z)$ is calculated as:

$$G_f(z) = \frac{m_f(z)}{m(z)} \tag{4}$$

When a controller failure happens, we can suppose the controller is still in a normal state, the fault part G_f is "shifted" to the plant as shown in Fig. 1(a) and the equivalent faulty plant model is:

$$G_{new}(z) = G_p(z)G_f(z) \tag{5}$$

A virtual controller $\tilde{G}_c(z)$ with a desirable control performance is designed for the equivalent faulty plant $G_{new}(z)$. The compensator $G_{com}(z)$ is designed for the equivalent faulty plant $G_{new}(z)$ to work with the original controller $G_c(z)$ to achieve a reference of control performance set by a virtual controller $\tilde{G}_c(z)$.

$$[1 + G_{com}(z)] \times G_c(z) = \tilde{G}_c(z) \tag{6}$$

Therefore, the model of the compensator can be given as follows:

$$G_{com}(z) = \frac{\tilde{G}_c(z) - G_c(z)}{G_c(z)} \tag{7}$$

Here $\tilde{G}_c(z)$ can be designed in terms of any existing controller design methods or be simply designed as a Proportional-Integral-Derivative (PID) controller. Fig. 1(b) shows that the compensator designed for the equivalent faulty plant $G_{new}(s)$ can be directly adopted in the faulty controller to maintain its performance.

3. Architecture of Remote Maintenance System

Internet time delay is one of the biggest obstacles in any Internet-based application, particularly when real-time bulk data transfer is involved. Remote maintenance systems need data that are collected from the real-time control process. Transferring data properly is important to ensure that any remote maintenance system responds in time. If all the calculations are carried out in remote sides, it may produce heavy Internet traffic and cause unacceptable time delay. In this section data is categorized into heavy data and light data based on the size of the data. The heavy data is processed on the local site and only the extracted characteristics of the heavy data, such as parameters of the identified models, is transferred to the remote side. Also a back-end system in the local site is designed to carry out all the heavy calculations and a front-end system in the remote side to allow experts to remotely maintain the control system.

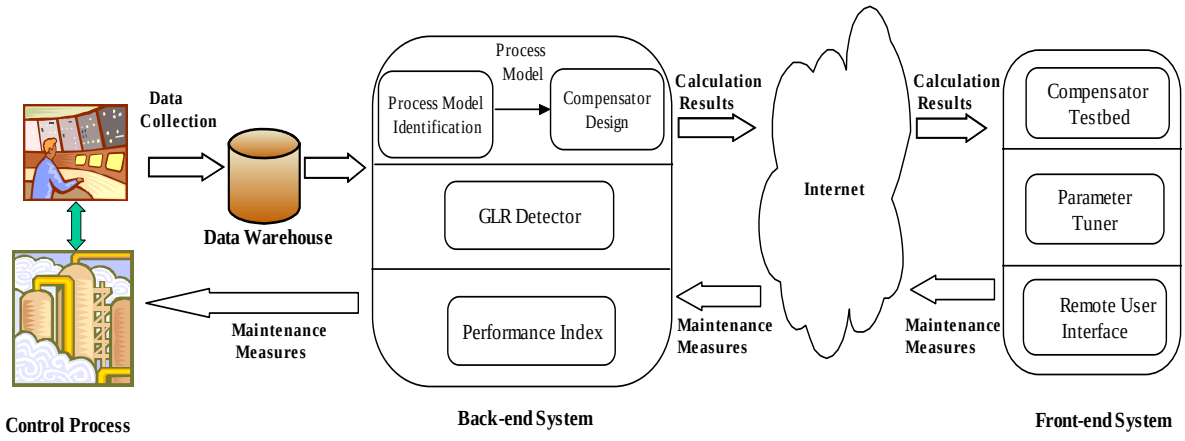

Fig. 2: Structure of the remote maintenance system.

3.1 Back-end and front-end systems

Fig. 2 shows the perspective view of the remote maintenance system. It consists of a local control system, a back-end system, and one or more front-end systems. Process data are collected and processed by the back-end system, in which the control performance index shown in Equation 1, the likelihood ratio shown in Equation 2 and the compensator model shown in Equation 7 are calculated. The back-end system also provides communication services. It uses the full power of object-oriented technology on distributed computing systems to transfer data objects between the remote front-end system and the local back-end system. The front-end system provides various services over the Internet to the remote users, including a compensator test bed, together with a platform for controller parameter tuning, control performance monitoring, and fault detection.

3.2 Data transfer over the Internet

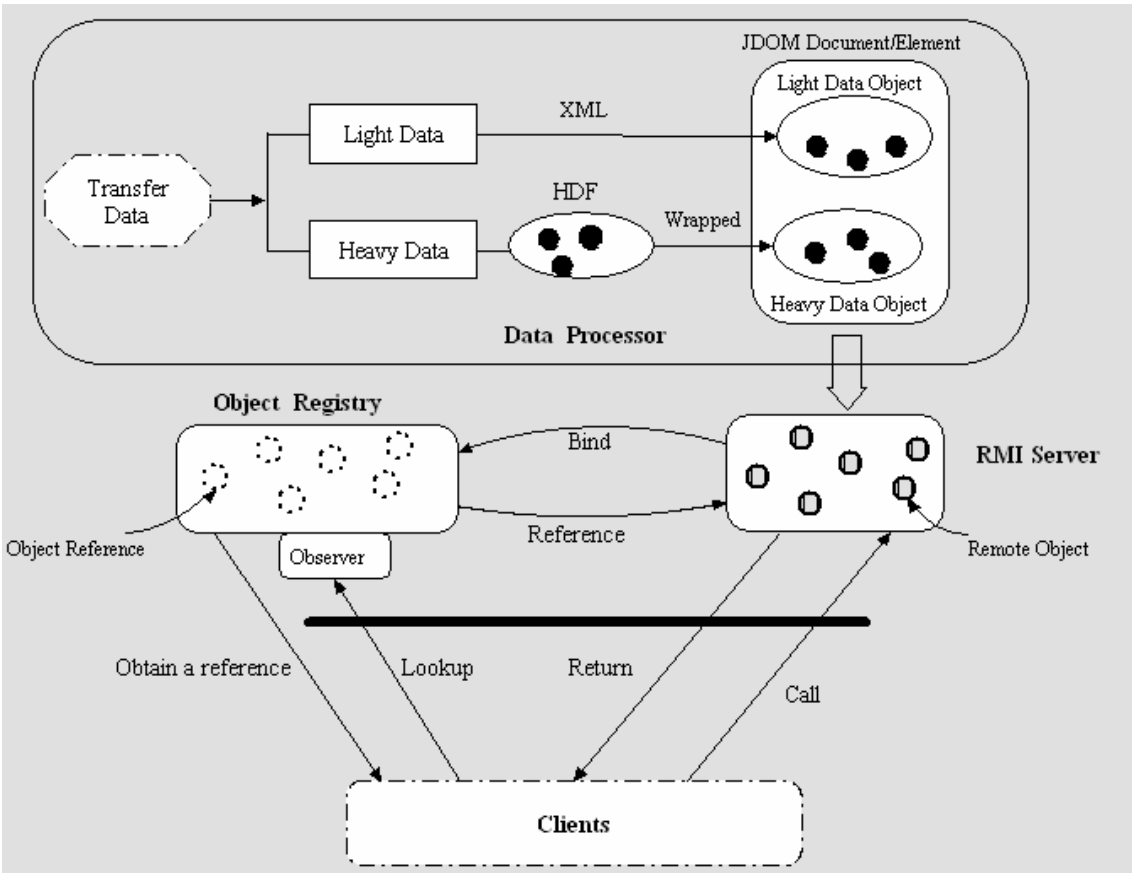

Fig. 3: Data transfer mechanism based on RMI.

Data collected from the control process is often heterogeneous and large. A single data point may include thousands of data values corresponding to different measured times, positions and variables. The heterogeneous data is categorized into light and heavy data based on the amount of the data. XML is a textual language which is quickly gaining popularity for data representation and exchange on the Web and it fast becoming an industry-standard for a system-independent way of representing data. XML is suitable for light data description and binding. Java RMI is a popular middleware platform for Internet based applications and enables remote communication between programs written in Java. RMI allows Java developers to invoke object methods, and execute them on remote Java Virtual Machines (JVMs). Under RMI, entire objects can be passed to and returned from a remote method as parameters.

Fig. 3 depicts the structure of the data transfer mechanism proposed in this study, which is built on the RMI infrastructure. The structure is composed of four basic elements: a RMI server, a RMI Object Registry, a data processor and remote clients. The RMI server provides the back-end communication services. The Object Registry plays the role of object management and naming service for the remote objects. The RMI transfer system can pass data objects as arguments and return values, using the full power of object-oriented technology on the distributed computer. The data processor's role is to collect and process the data from the actual sites.

According to the heterogeneous feature, the transfer data is firstly categorized into light and heavy data according to their physical meanings and actual requirements. The heavy data is organized in the HDF form. Both categories of the data are wrapped with XML and processed as remote objects in the Java environment. In detail, the Object Registry works together with the RMI server to achieve the object transfer in the RMI infrastructure. The RMI server creates remote objects and binds each instance of the remote objects with a name in the Object Registry. When a client invokes a method of a remote object, it first looks up the remote object within the Registry by the remote object name. If the remote object exists, the Registry will return a reference of the remote object to the client, which is used to make method invocations on the remote object. As long as there is a reference to the remote object, the object will remain reachable from the remote clients.

The sequence of data transfer in the RMI infrastructure is summarised as the following five stages:

Stage 1: Categorizing the heterogeneous data into light and heavy data according to its features.
Stage 2: Organizing the heavy data in the HDF format and wrapping both types of data with XML
Stage 3: Generating the stub and skeleton of remote objects.
Stage 4: Starting up the Object Registry and binding the remote objects in the Object Registry.
Stage 5: Transferring data by involving the RMI method every sampling interval with the data object as an argument.

In the remote front-end side, a query, which specified sensors, time period, variables, control loops, and/or data reduction (e.g. hourly averages, peaks) is sent to the local back-end side, the required subset of the data with a complete specification of the semantics will be returned to the front-end side as the result of this query. Consequently the required information of the remote maintenance system becomes available in the front-end side, and the remote maintenance can take place there.

4. Implementation of Back-end System

The back-end system shown in Fig. 2 can be further decomposed into that shown in Fig. 4. The data warehouse collects and stores the process variables from the control process. The data warehouse provides data query and retrieval services for the back-end system components. Each component is driven by the process variables retrieved from the data warehouse. The following steps are involved:
Step 1: The process model is identified based on the process variables stored in the data warehouse. A LQG controller is designed as a benchmark controller. The generated LQG control signal and LQG process output are utilized to compute the LQG cost function.

Step 2: The process variables retrieved from the data warehouse are used to compute the actual cost function. The values of the actual cost function and LQG control function are compared to get a performance index value according to Equation 1. A poor performance index value will trigger the GLR detector to work.

Step 3: The actual control signals are sent to the GLR detector. The GLR detector traces the discrepancies between the actual control signals retrieved from the data warehouse and the LQG control signals generated by the LQG controller. The likelihood ratio s_k value is computed according to Equation 2 and compared with the pre-defined threshold λ. The result of this comparison determines whether or not any compensation is required. In normal operation, the compensator is inactive. Once s_k becomes greater than the threshold λ, which indicates that the control system is unhealthy, the compensator will be designed and sent to the remote experts for testing and approval. The approved compensator will be returned and implemented in the local control system.

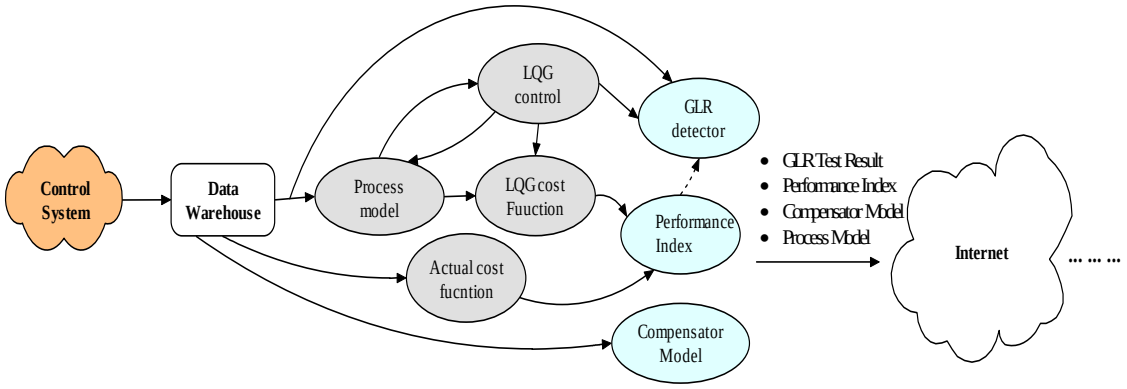

Fig. 4: Back-end system.

The design of the compensator involves the following three steps:

Step 1: Identifying the normal open loop model $m(z)$, the faulty open loop model $m_f(z)$, and the process model $G_p(z)$ respectively. Calculate the fault model $G_f(z)$ according to Equation 4.

Step 2: Designing a new virtual controller $\tilde{G}_c(z)$ based on the model of the faulty process $G_{new}(z) = G_p(z)G_f(z)$ with a satisfactory control performance.

Step 3: Calculating the compensator model based on Equation 7.

Components in the back-end system are required to co-operate with each other and perform their tasks concurrently. Multi-thread technologies are employed here to ensure that such cooperation occurs. Each component is implemented in conjunction with a Java thread, which allows the component programs to

efficiently perform their tasks independently while simultaneously sharing the latest process variables. The running threads in the components are established with different priorities and managed by a Java thread scheduler. The scheduler can suspend a component by making its thread sleep or kill a component by terminating its thread. It can also monitor all running threads and decide which threads should be running and which should be waiting, according to their priorities and total running time.

5. Implementation of Front-end System

The front-end system provides a test bed for testing and approval of the compensator and a platform for the remote tuning of controller parameters, control performance monitoring and fault detection. The platform is a web-based user interface which will be illustrated in our case study. This section focuses on the compensator testing and implementation.

5.1 Compensator Testing

The compensator is tested and tuned in the test bed as shown in Fig. 5 by remote experts after it is received by the front-end system. The test bed offline simulates the real implementation of the designed compensator. The compensator and the faulty open loop model m_f form a virtual control loop. The remote experts monitor the performance of the virtual control loop, tune the compensator parameters, and see how well the compensator works with m_f to trace the setpoint. A small and stable range of deviations between the set-point and the virtual process output indicates a healthy performance of the compensator. An unstable and large range of deviations indicates a demand for adjusting the compensator parameters. The parameters are manually adjusted until a satisfactory control performance of the virtual control loop is achieved. Only an expert can do the compensator testing and parameter tuning properly. For this reason the test bed for the compensator is placed in the remote front-end system.

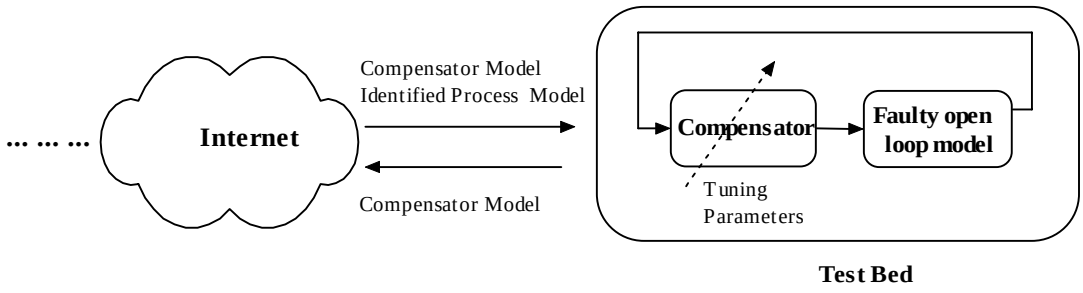

Fig. 5: Compensator testing in the remote side.

5.2 Compensator Implementation

The final compensator is sent back to the back-end system and installed in the local control process over the Internet to restore the degraded control performance. The compensator was designed and tested in the remote side within the remote maintenance system. There are two ways to download and implement the compensator from the remote side to the local side within the process plant: weak migration and strong migration, as shown in Fig. 6. Weak migration transfers only the parameters of the compensator from the remote side to the local side, whereas strong migration transfers both the parameters and the structure of the compensator. Strong migration is employed only if there is no compensator previously installed or the structure of the compensator has been significantly changed since implementation. Weak migration is employed to update the parameters of the compensator provided it has the same structure as the compensator tested. Both migrations are established on the basis of the Java RMI client/server *Remote* interface *java.rmi.Remote*. Weak migration transfers the parameters of the compensator from the remote side to the local side by passing them as the parameters of a remote method invoked. The strong migration, in contrast, transfers the code of the compensator from the remote side to the local side through the Java RMI interface. The *ClassLoad( )* method is used to search a suitable compensator from the remote front-end system and transfer the code to the local side.

The model of the designed compensator shown in Equation 7 can be re-written as the following general linear difference equation:

$$a_0 y(k) + a_1 y(k-1) + a_2 y(k-2) + .. + a_n y(k-n) = b_0 x(k) + b_1 x(k-1) + .. + b_n x(k-n) \quad (8)$$

where *y(k)* and *x(k)* are the output and input of the compensator at the instant *k* respectively. The parameters a_0, a_1, …, a_n, and b_0, b_1, …, b_n are the model coefficients obtained from the test bed. Weak migration will only transfer the parameters a_0, a_1, …, a_n, and b_0, b_1, …, b_n from the remote front-end system to the compensator frame located in the local back-end system. Strong migration will transfer the complete code of the compensator designed in the remote front-end system to the local side.

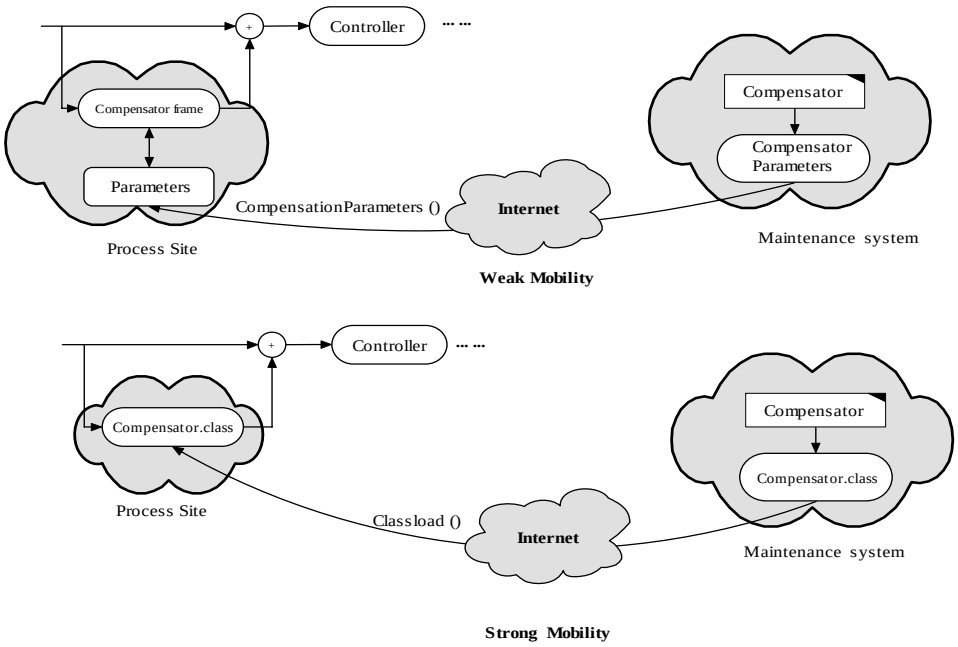

Fig. 6: Compensator implementation.

The migrated compensator in the local process side acts as a 'black box' and a serial link with the faulty controller as shown in Fig. 1. The difference of the process output and the setpoint is used as the input of the black box; the output generated by the compensator is added to the input of the fault controller to recover the degraded control performance. In the case of updating a compensator, the running compensator needs to be suspended, and a new compensator is then downloaded to the back-end system of the local side via weak or strong migration. A local operator is required to authorize the updating.

6. Case Study

To illustrate the proposed remote maintenance system, a Process Control Unit (PCU) in the Networks & Control Laboratory at Loughborough University has been chosen for the demonstration and evaluation.

6.1 System Description

The layout of the experimental system is illustrated in Fig. 7, which includes the PCU, a back-end system, and a number of identical front-end systems. The PCU consists of a water tank rig, an I/O interface, and a local control system. The water tank rig comprises a process tank, a sump, a pump, a cooler and a number of drain valves. The I/O interface manages the data acquisition and signal conversion from analogue to digital and from digital to analogue. The local control system measures the liquid level of the water tank and regulates the flow rate of the pump to maintain the liquid level of the water tank at a desired value. Fig. 8 gives the operation interface of the local control system. The controller parameters, setpoint, and sampling interval can be chosen by the local

operators in the operation interface. The local control system and the back-end system are physically located in the same location and connected with the front-end system via the Internet. Fig. 9 shows the interface of the remote front-end system. The left hand side of the interface is the available performance monitoring list, the available detection list, and the compensator test bed. All the available tools can be added into or removed from the right hand side windows. Fig. 9 illustrates the performance index defined in Equation 1, the GLR test defined in Equation 2 and the test bed window for the compensator defined in Equation 8. Other available performance monitoring tools and fault detection tools can be integrated into this front-end system. In Fig. 9 two traditional control performance assessment indexes, IAE and MSE, are also implemented the performance monitoring list. The compensator test bed is illustrated at the bottom part of Fig. 9. The remote experts can tune the compensator parameters designed by the back-end system and send the satisfactory one back to the back-end system through the weak or strong migration.

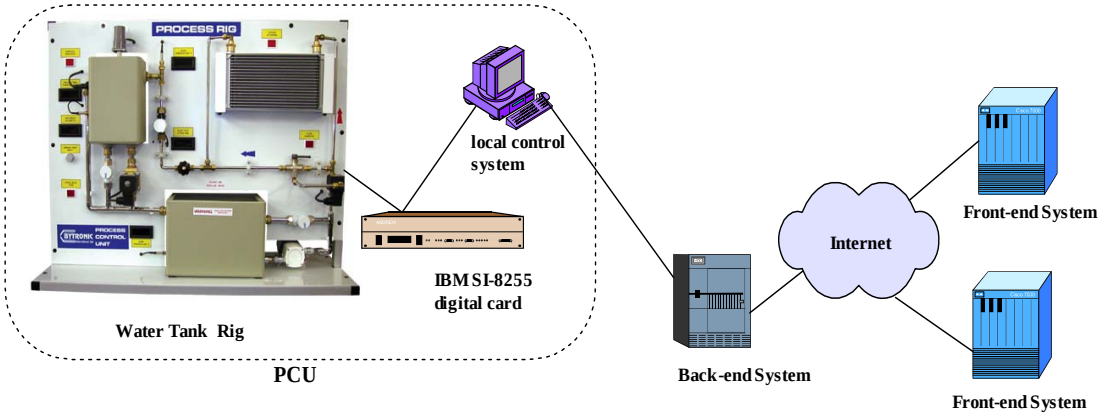

Fig. 7: Experimental system layout.

6.2 Remote Data Processing and Transfer

The light data collected from the water tank process and the identified model generated from the heavy data are processed and converted into a Java Document Object Model (JDOM), which is a Java representation of an XML document. JDOM provides a way of representing the document for easy and efficient reading, manipulating, and writing of XML in a Java environment. Table 1 is an example showing three parameters of a PID controller embedded in a XML file. Each of the PID parameters embedded in the XML file is a real time data point, which includes the value of the parameter and the corresponding sampling time. Other process data set take a similar format in their XML files. All required data are processed in the Java environment and output as JDOM Document objects. The JDOM objects are defined as the returned values of the data transfer methods. When the front-end system invokes a data transfer method of a remote object in the back-end system, the back-

end system sends the required data object back to the front-end system as a returned value of the data transfer method.

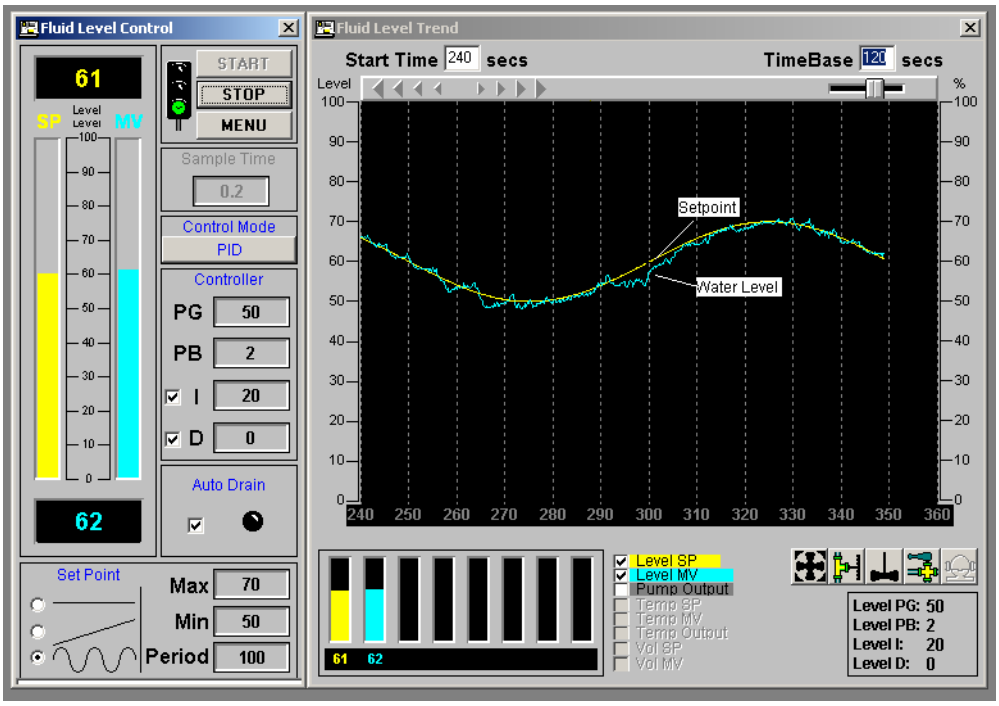

Fig. 8: Local control system.

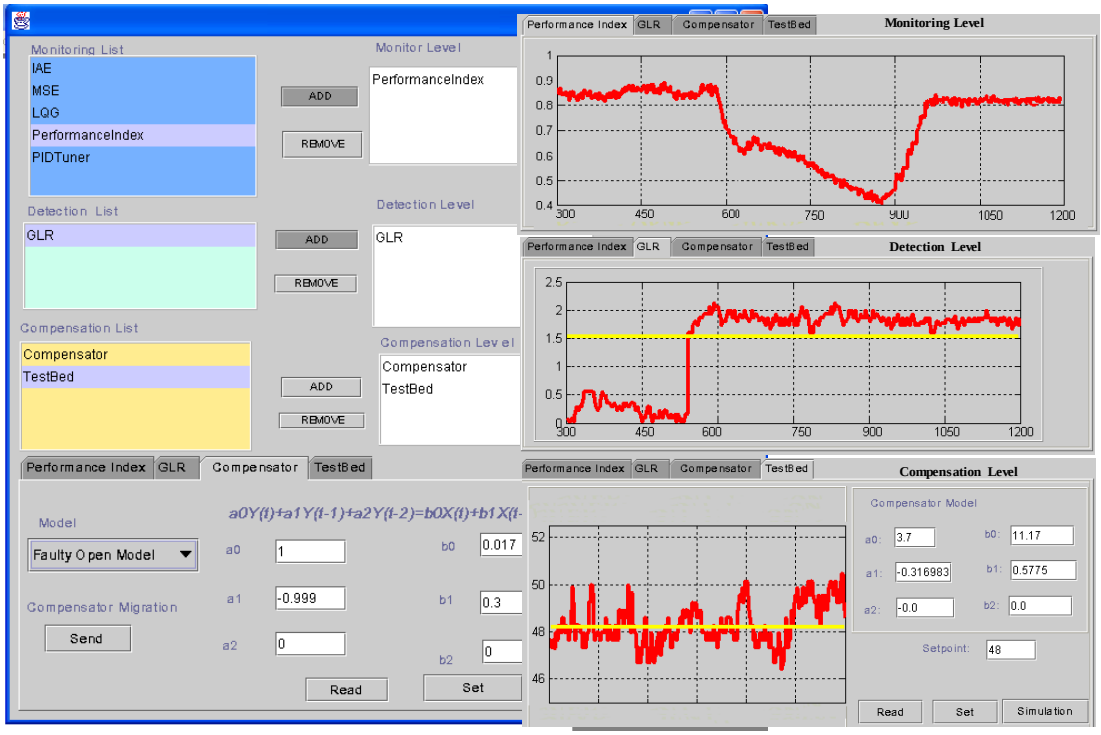

Fig. 9: User interface of the remote maintenance system.

Table 1: PID parameters embedded in a XML file.

```
<?xml version="1.0" encoding="UTF-8"?>
<Simulation>
 PID.xml <Sector>
                        <Parameters>
                                <PID_P>3</ PID_P >
                                <Sampling Time>11:34:47 25/03/2005</ Sampling Time >
                                <PID_I>20</ PID_I >
                                <Sampling Time>11:34:47 25/03/2005</ Sampling Time >
                               <PID_D>0</ PID_D >
                                <Sampling Time>11:34:47 25/03/2005</ Sampling Time >
                      </ Parameters >
PID.xml </Sector>
</Simulation>
```

6.3 Performance Monitoring and Fault Detection

The water tank model is obtained by injecting a step change in the inlet flow rate of the tank and measuring the response in the liquid level of the tank. The sampling interval is chosen as 1 second. The input of the model is the inlet flow rate of the tank, and the output is the liquid level. The identified discrete model of the tank is described as

$$G_p(z) = \frac{0.308z + 0.359}{z^2 - 0.4991z + 0.1} \tag{9}$$

where z denotes the z-transform. A MatLab identification toolbox is used in the model identification. A LQG benchmark controller is automatically designed in the back-end system for the control performance monitoring and fault detection purposes. The LQG control design is based on the Model Predictive Control (MPC) solution (Patwardhan et al., 2002). Both the prediction and control horizons in the MPC are set as 15 sampling intervals. The cost of the LQG benchmark controller working within the back-end system is compared with the actual control cost made by the actual PID controller; therefore the performance index is obtained. The differences between the benchmark controller and the actual PID controller are also used in the GLR fault detection.

The original PID controller with the Proportional-Integral-Derivative parameters K_P=10, K_I=0.1, K_D=0 is described as

$$G_c(z) = \frac{20.1z - 19.9}{2(z-1)} \tag{10}$$

In order to illustrate how well the remote maintenance system can identify the potential faults occurring in the local control system of the water tank process, a gain, K=0.1, is introduced in the output of the local PID controller at the instant 550 second. As the result, the output of the PID controller is reduced by 0.1 times afterwards. Therefore, the fault model is given as

$$G_f(z) = 0.1 \tag{11}$$

Fig. 10 shows the water level deviating away from the setpoint immediately after the fault is introduced. Fig. 11 illustrates the performance index significantly dropping from 0.85 at the normal operation to 0.42. The likelihood ratio generated by the GLR test jumps from 0.2 at the normal operation to 1.8 as shown in Fig. 12. A faulty state is detected.

6.4 Compensation in Action

The desirable virtual controller $\tilde{G}_c(z)$ for the faulty process is set as a PI controller as follows:

$$\tilde{G}_c(z) = \frac{405z - 25.6}{10z - 1} \tag{12}$$

Once the likelihood ratio generated by the GLR test goes over the pre-defined threshold (1.5 in this study) a compensator is automatically generated in the back-end system for the particular abnormal operation in order to achieve a reference of control performance set by the virtual controller shown in Equation 12. The compensator is designed in terms of the virtual controller $\tilde{G}_c(z)$ shown in Equation 12 and the original controller $G_c(z)$ shown in Equation 10 based on the compensator model shown in Equation 7.

$$G_{com}(z) = \frac{11.3z - 0.5775}{3.7z - 0.37} \tag{13}$$

The compensator is sent to the front-end system for testing, tuning and approval in the test bed. The approved compensator is expressed as a linear difference equation as follows:

$$3.7y(k) - 0.37y(k-1) = 11.3x(k) - 0.5775x(k-1) \tag{14}$$

where *y(k)* and *x(k)* are the output and input of the compensator at the instant *k*.

A general compensator structure shown in Equation 8 has been previously implemented in the local control system. After the compensator for the particular fault is approved in the test bed the weak migration is employed to remotely install the compensator shown in Equation 14 in the real process side. The parameters {(3.7, -0.37, 0), (11.3, -0.5775, 0)} are sent to the pre-defined compensator structure. The local operators put the compensator in action if they have been convinced by the simulation results that the compensation is necessary and safe. In our case study the compensator is put in action at the instant 880 second. Figs. 10 and 11 show that the liquid level of the water tank returns back to the desired setpoint and the performance index back to the normal value 0.81 immediately after the compensator being taken place. GLR test indicates the original controller is still in an unhealthy state, which will remind the local operator to update the controller once the plant is shutdown.

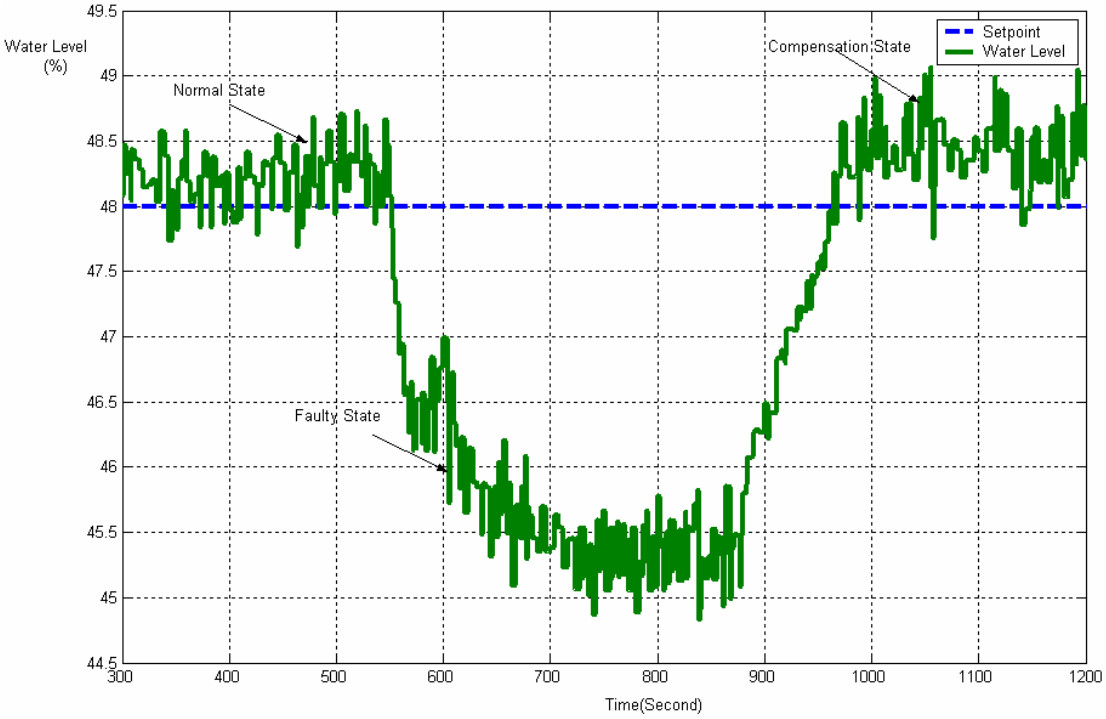

Fig. 10: Liquid level of the water tank.

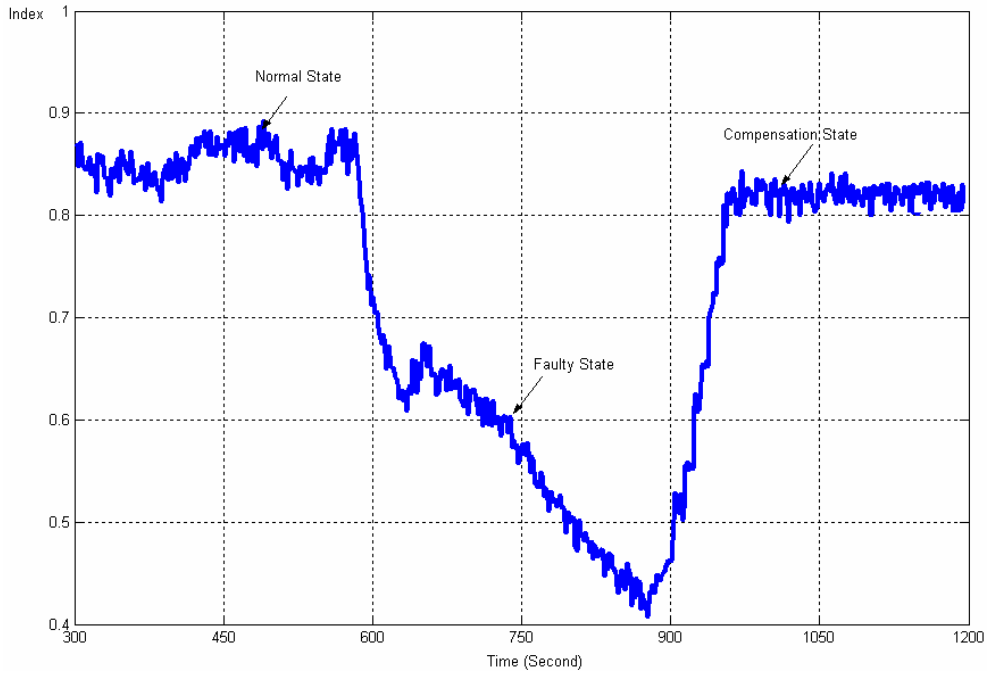

Fig. 11: Performance Index.

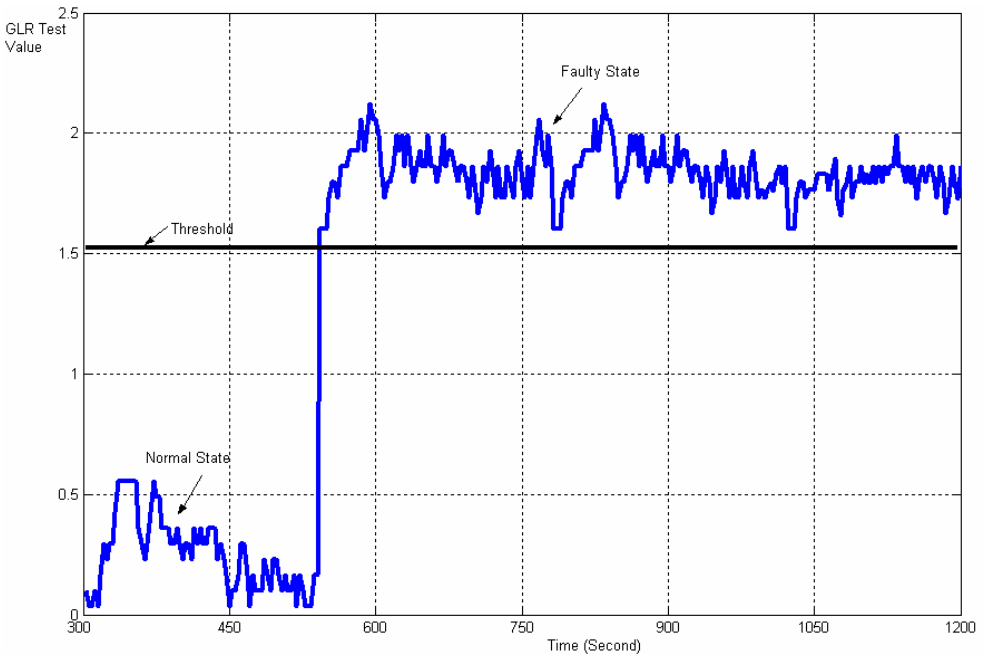

Fig. 12: GLR test.

7. Conclusions

This paper presents a design method for the remote monitoring and maintenance of control systems and demonstrates its implementation in our Process Control Unit. In order to avoid the influence of the Internet time delay on the remote operation, all the heavy calculations are carried out in the local back-end system, and only the light data and the parameters of the plant model are transferred to the remote side. The real-time requirements in the remote monitoring and maintenance system are not necessarily essential. Another feature of our method is to compensate the faulty controller rather than to replace it with a new one. It is obvious that updating a faulty controller might introduce a significant impact to the control system if it is in operation during the updating process. Appending a control signal produced by a compensator into the existing controller will greatly reduce the influence of this maintenance to the normal operation. The faulty controller can be updated offline after the process plant is shutdown.

Remote monitoring and maintenance of control system performance might be a good solution for process plant companies with multiple sites in remote locations (for example, power stations and wind electricity generation stations) in order to provide the central support for their geographically dispersed control systems. By using this remote monitoring and maintenance system control software suppliers can monitor and maintain their control software products remotely over the Internet. The need for any control software supplier's expert to conduct on-site maintenance will be eliminated. Therefore both time and money can be saved. Furthermore, small and mid-

sized companies without internal expertise to maintain their control systems, can now rely on the remote monitoring and maintenance system provided by a service company for their system maintenance. This represents a significant cost savings in terms of sharing specialized support staff with other partners. The generic principle of remote monitoring and maintenance can also be applied to fail-safe or backup health monitoring of critical systems.

Today the high-speed Ethernet, also a non-deterministic communication medium, is being adopted for process automation (Zhang, et al., 2001). Industries are beginning to implement networked control systems through this high-speed communication medium. Given the potential development of the next-generation Internet and other enhancements to the WWW infrastructure, the speed of the next-generation Internet might be sufficiently fast to be able to dramatically reduce the transmission delay and data loss. Therefore it is possible that Internet latency and data loss might become less important issues in future Internet applications. Safety and security issues will ,however, remain crucial because of the public nature of the Internet.

References

Calkins, K.G. (2005). Introduction to statistics. available at http://www.andrews.edu/~calkins/math/webtexts/stattit.htm

Calvo, I., Marcos, M., Orive, D., and Sarachaga, I. (2006). A methodology based on distributed object-oriented technologies for providing remote access to industrial plants, *Control Engineering Practice*, 14(8), 975-990.

DAME (2002). DAME Distributed Aircraft Maintenance Environment, available at http://www.cs.york.ac.uk/dame

Dai, C., & Yang, S.H. (2004). Maintaining control performance in faulty control systems**,** *IEEE International Conference on Systems, Man & Cybernetics*, W. Thissen, W., Wieringa, P., Pantic, M. and M. Ludema, M. (eds), The Hague, The Netherlands (pp. 5074-5078).

Dai, C., Yang, S.H., & Tan, L. (2004). An approach for controller fault detection, *in the Proceedings of the 5th World Congress on Intelligent Control and Automation*, Hangzhou, China (pp. 1637-1641).

Harris, T.J., Seppala, C.T., & Desborough, L.D. (1999). A review of performance monitoring and assessment techniques for un-variant and multivariate control systems, *Journal of Process Control*, 9(1), 1-17.

HSE. Safety implications of industrial users of Internet technology, *HSE Contract Research Report*, 408, 2002.

HSE. Safety in the remote diagnosis of manufacturing plant and equipment, *HSE Books*, 1995.

Huang, B., & Shah, S.L. Performance assessment of control loops: theory and applications, Springer Verlag, New York, NY; 1999.

Jelali, M. (2006). An overview of control performance assessment technology and industrial applications, *Control Engineering Practice*, 14(5), 441-466.

Nougues, A., Vadnais, P., & Snoeren, R. (2002). Performance monitoring for process control and optimization, *ESCAPE–12*, The Hague, Netherlands (pp. 733-738).

Patwardhan, R.S., Shah, S.L., & Qi, K.Z. (2002). Assessing the performance of model predictive controllers, *The Canadian Journal of Chemical Engineering*, 80, October, 954-966.

ScadaOnWeb (2002). http://www.scadaonweb.com.

Sematech (2002). E-diagnostics guidebook, http://www.sematech.org/public/docubase/techrpts.htm.

Thompson, H. A. (2004). Wireless and Internet communications technologies for monitoring and control, *Control Engineering Practice*, 12(6), 781-791.

Yang, S.H., Chen, X, Tan, L., & Yang, L. (2005). Time delay and data loss compensation for Internet-based process control systems, *Transactions of the Institute of Measurement and Control*, 27(2), 103-118.

Yang, S.H., Chen, X., & Alty, J.L. (2003). Design issues and implementation of internet based process control, *Control Engineering Practice*, 11(6), 709-720.

Zhang, W., Branicky, M. S., and Phillips, S. M. (2001), Stability of networked control systems, *IEEE Control Systems Magazine*, February, 84-99.

Published in Control Engineering Practice, 15(5), pp.533-544, 2007

Caption lists: