Fault Detection in IP-based Process Control Networks using Data Mining

Byungchul Park[1], Young J. Won[1], Hwanjo Yu[1], James Won-Ki Hong[1],
Hong-Sun Noh[2], and Jang Jin Lee[2]

[1]Dept. of Computer Science and Engineering, POSTECH, Korea
{fates, yjwon, hwanjoyu, jwkhong}@postech.ac.kr

[2]Electric & Control Maintenance Dept., POSCO, Korea
{ vishnu4, jangjin21}@posco.co.kr

Abstract— **Industrial process control IP networks support communications between process control applications and devices. Communication faults in any stage of these control networks can cause delays or even shutdown of the entire manufacturing process. The current process of detecting and diagnosing communication faults is mostly manual, cumbersome, and inefficient. Detecting early symptoms of potential problems is very important but automated solutions do not yet exist. Our research goal is to automate the process of detecting and diagnosing the communication faults as well as to prevent problems by detecting early symptoms of potential problems. To achieve our goal, we have first investigated real-world fault cases and summarized control network failures. We have also defined network metrics and their alarm conditions to detect early symptoms for communication failures between process control servers and devices. In particular, we leverage data mining techniques to train the system to learn the rules of network faults in control networks and our testing results show that these rules are very effective. In our earlier work, we presented a design of a process control network monitoring and fault diagnosis system. In this paper, we focus on how the fault detection part of this system can be improved using data mining techniques.**

Index Terms—**Process control network, network management, fault detection, fault diagnosis, machine learning, data mining.**

I. Introduction

Process control networks support communications of devices in a controlling or manufacturing process. There are several categories of control networks, namely Building Automation, Factory Automation, and Process Automation [1]. Control networks have many different roles and environments; however, they are typically deployed in mission critical operations which have to support secure and reliable communications. Thus, a minimum delay and guaranteed transmission of control messages are especially important QoS attributes for all existing communication sessions in process control networks. Most traditional process control networks have been deeply dependent on the proprietary protocols such as FOUNDATION fieldbus [2], PROFIBUS [3], MODBUS [4], and BACnet [5]. These control network technologies were developed separately from the relatively recent emergence of Ethernet and IP-based network technologies. Recently, these control network technologies have been using Ethernet (e.g., Industrial Ethernet) and IP for their low cost, high scalability, and easy maintenance purposes. When adopting IP-based control network technology, the control networks also encounter the common network problems that exist in IP networks.

This paper investigates process control networks in a steel manufacturing company, POSCO, the world's second largest steel manufacturer and operator of a number of plants world-wide. Their single operational site consists of more than 40 manufacturing plants, equally 40 process control networks where some are organized in a synchronous and sequential order. POSCO's networks are classified into the following: business (or office) networks or process control networks. A business network is similar to a typical IP-based data network consisting of IP routers and switches. On the other hand, a process control network is a combination of IP-based and non-IP based control technologies, so that different monitoring and maintenance techniques should be used [6].

In this paper, we work on applying data mining techniques to detect control network faults. Most of the existing IP network monitoring and diagnosis tools (such as Sniffer [15] and Wireshark [16]) do not yet have the capability to understand the faults in process control networks. These tools only can detect network-level errors. However, network-level errors and actual faults in process control networks are different. Network-level errors, such as packet retransmission and checksum errors, are the symptoms of control network faults but the definite faults and the network-level errors can occur frequently in normal IP networks depending on the network condition. Therefore, the primal requirement of fault detection system for process control networks is the capability to understand the correlation between network-level symptoms and actual faults.

To fulfill this requirement, we have developed an accurate but flexible control network fault detection engine which can continuously train itself using data mining

978-1-4244-3487-9/09/$25.00 ©

techniques to learn detection rules from network-level metrics and user interpretation about already-known control network faults. In our earlier work, we presented a design and implementation of a process control network monitoring and fault diagnosis system [7]. The fault diagnosis part of that system was very primitive using some basic and fixed rules. This new, intelligent engine has replaced the previous simple fault diagnosis part.

The organization of this paper is as follows. An overview of process control networks is given in Section II. Section III identifies the communication fault cases in a process control network. Section IV provides experimental results of network fault detection using data mining techniques. In Section V, we present a control network diagnosis system. Section VI addresses the related work. Finally, concluding remarks are given and possible future work is discussed.

II. Overview of Process Control Networks

In this section, we describe process control network elements and their roles. Our motivation for this research also explains the practical need for monitoring process control networks.

The process control network elements are operated in a hierarchical manner. The controller at the top level provokes lower level controlled devices to do specified actions. The following are brief descriptions of each element and its role.

- **Process Controller/Human Machine Interface (PC/HMI, abbreviated as PC in this paper):** This is a part of the software and hardware package provided by the PLC vendors. It is simply process control application software on a computer running UNIX or Windows that can remotely access PLC. Custom built or vendor provided server applications are placed in PC to communicate with PLCs. Its graphical user interface typically also provides real-time process monitoring results. Connections to PLC are often established over reliable TCP/IP.
- **Programmable Logic Controller (PLC):** It is a microprocessor computer for process control attached to a process control network. A complex sequence control of machinery (or low end controlled devices on factory assembly line) is handled by the custom built software programs running in PLC.
- **Controlled Devices:** These machines refer to sensors, actuators, motors, etc. Through the embedded interface, they receive the command signal from PLCs.

The architecture of POSCO's process control network is illustrated in Figure 1. PC/HMI and PLCs are connected to each other via Ethernet while controlled devices communicate with PLCs using proprietary protocols such as FOUNDATION fieldbus and PROFIBUS.

All devices in control networks operate synchronously or sequentially, therefore even a network fault can be fatal. In other words, a single failure of a control device can compel an entire manufacturing process to stop or delay and lead to a huge financial loss. Therefore, fast and accurate fault detection of control network faults is very important.

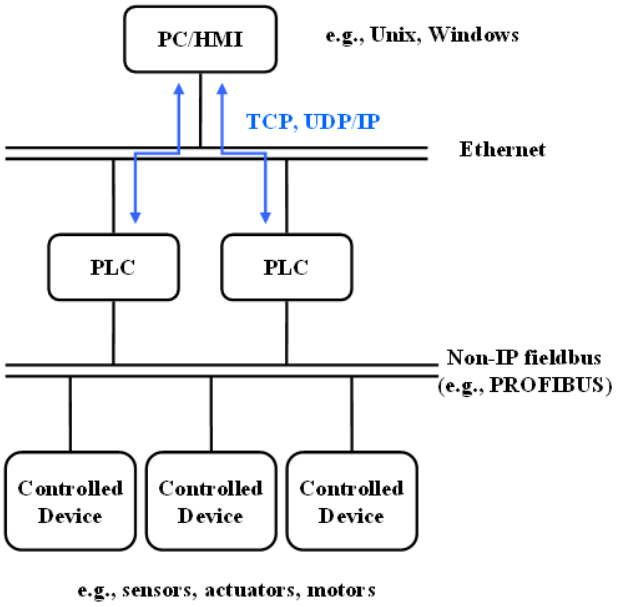

Figure 1. Architecture of POSCO's Process Control Network

III. Fault Detection and Diagnosis

In this section, we present the real-world fault cases in process control networks. In our previous research, we categorized the process control network specific failures and these are summarized in Table 1 [7].

Table 1. Classification of Process Control Network Failure Cases

Failure Type	Failure Causes
IP connectivity errors	Unstable transmission, low throughput, delay, network security threat, IP resource management
Network mis-configuration	Network topology loop, Non-optimal path (redundancy), duplex mismatch
Physical defects	Hardware malfunction, link corruption (cable damage), electrical noise, power outage, duplicate hardware addresses
Software defects	PLC programming bugs, device driver bugs, protocol unawareness

The existing IP network diagnosis tools (e.g., Sniffer [15], Wireshark [16]) do not have the capability to understand the failures in process control networks and to provide the causes of such failures. However, control network failures are accompanied by network-level errors as early symptoms. We have selected several IP network metrics and identified the conditions accordingly that best reflect any irregularity of communication in process control networks as listed in Table 2 [7]. These metrics themselves are not very unique, but they have not been properly analyzed by most commercial IP network diagnosis tools. A new set of monitoring categories and conditions is necessary because even a popular tool, like

Sniffer, could not fully detect the control network specific failures but could only generate false network alerts. The selected metrics can be measured using passive network monitoring techniques which do not interfere with network operations.

Table 2. Measurement Metrics and Alarm Conditions

Index	Network Metric	Alarm Condition
1	Collision frames	First appearance, or threshold-based
2	Jumbo (>= 1514 bytes) frames	First appearance
3	Runts (<= 64 bytes) frames	First appearance
4	CRC error frames	First appearance
5	IP/TCP checksum errors	First appearance
6	Fragment packets	Threshold-based
7	Retransmission packets	First appearance, or threshold-based
8	Packet inter-arrival time (ms)	Increase to the previous value
9	Throughput (bps)	Decrease, drop to 0, or pattern analysis over monitoring period
10	Packets per second (or packet burst)	Increase, decrease, drop to 0, or pattern analysis over monitoring period
11	Min/max/diff packet size (bytes)	Change in difference of max and min sizes over monitoring period
12	Min/max/diff TCP window size	Drop to 0, change in difference of max and min sizes over monitoring period
13	Out-of-order sequence packets	First appearance
14	Broadcast packets	Threshold-based
15	Unsupported protocol packets	Threshold-based

These metrics should be monitored for each flow, which refers to a bidirectional packet stream that shares the same header information, namely, a pair of MAC addresses, IP addresses, source/destination ports, and protocol. 'First appearance' in the alarm condition specifies the condition that an alarm should be issued whenever the corresponding metric increases. In other words, a single occurrence of such an abnormality has a high potential of causing communication failures. The rest of the conditions are against pre-configured threshold value and changes in the measured value over the monitoring period.

The metrics in index 8-12 in Table 2 have distinct conditions compared to those of IP data networks. In the first 24 hours of a rolling out process, the communications between PC and PLC tend to be continuous with a steady inter-packet generation time. They also yield very low bandwidth communications with a fixed packet size. The proposed alarm conditions in Table 2 indicate that any sudden changes (up or down) of value are considered abnormal in process control networks.

IV. Data Mining for Fault Detection

It is important for network administrators to recognize early symptoms of process network communication failures. A human operator must search through vast amounts of data to find anomalous sequences of network connections. Moreover, different control networks from different plants may have different traffic characteristics and fault cases. To support fault detection work, we need a system that can extract fault detection rules automatically from historical fault data. Thus, we employed decision trees to automatically generate rules that best reflect any irregularity of communication in process control networks for fault detection.

A. Methodology for Cause Analysis

Data mining technology can be considered as an inference engine for control network fault detection. Figure 2 illustrates three input variables to the inference engine for fault cause analysis. The inference engine keeps a list of measured metric values and possible causes from the fault history. Using network metrics and user's interpretation, the inference engine calculates the correlation between network metrics and actual network faults. Finally, this correlation is translated into fault detection rules to be applied in a fault detection system.

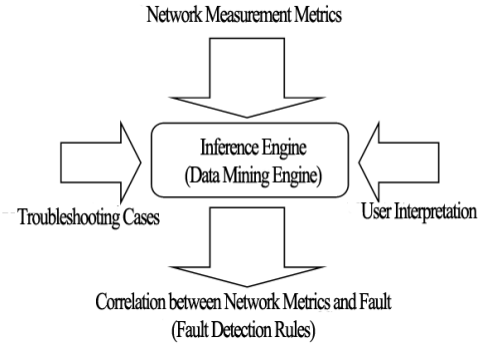

Figure 2. Conceptual View of Data Mining Engine

B. Decision Tree Algorithm

Decision trees are structures used to classify data with common attributes. Each decision tree represents a rule which categorizes data according to these attributes. A decision tree consists of internal nodes, leaf nodes, and edges. Each internal

node specifies an attribute by which the data is to be partitioned. Each edge is labeled according to a possible condition of the attribute in the parent node. Leaf nodes are labeled with a decision value for categorization of the data. Decision trees are very effective on unbalance classification problem (i.e., it has been shown to be effective if the abnormal class has few training examples) [8].

Figure 3 illustrates an example of a fault decision tree. In this example, source IP address and source port number label the nodes, fault and normal label the leaves, and the labeled arrows are the edges.

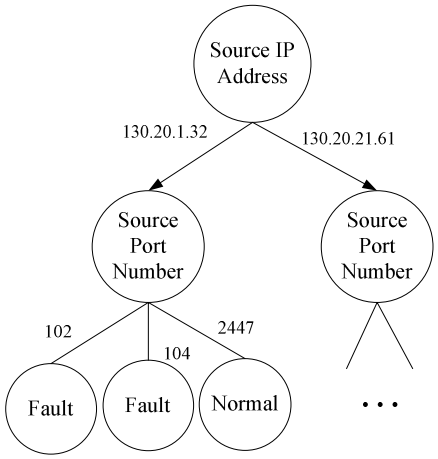

Figure 3. Example of a Fault Decision Tree

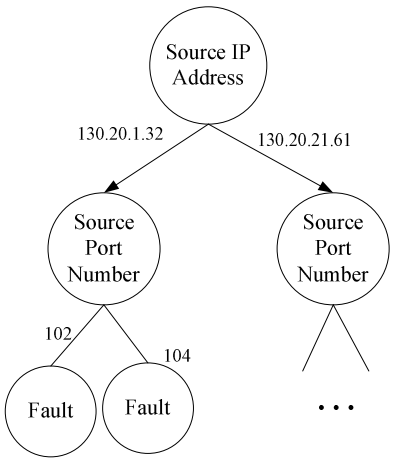

Figure 4. Example of a Pruned Decision Tree

We employed the C4.5 algorithm [9] to construct decision trees from training data. The C4.5 algorithm uses the information gain function to create efficient decision trees. Given a training data set, a list of parameters (features) describing each data instance and a set of categories to partition the data into; the information gain function determines which feature most accurately classifies the data. In most cases, pruning is used for decision trees to generalize the information learned; however, pruning can affect classification accuracy. Figure 4 shows an example of a pruned decision tree of Figure 3 after the pruning operation. All connections are assumed to be normal if they are not classified as network faults.

Table 3. Feature List

Index	Feature
1	Source IP address
2	Destination IP address
3	Source port number
4	Destination port number
5	Protocol
6	Total Bytes
7	Number of packets
8	Number of Jumbo packet
9	Number of Runts error
10	Number of fragmented packet
11	Number of IP checksum error
12	Number of TCP checksum error
13	Number of UDP checksum error
14	Number of retransmitted packet
15	Flow duration
16	Minimum packet size
17	Maximum packet size
18	Average packet size
19	Variance of packet size
20	Minimum packet inter-arrival time
21	Maximum packet inter-arrival time
22	Average packet inter-arrival time
23	Variance of packet inter-arrival time
24	Number of TCP SYN flag
25	Number of TCP FIN flag
26	Number of TCP ACK flag
27	Number of TCP RTS flag
28	Number of TCP URG flag
29	Number of TCP PSH flag
30	Number of TCP ECN flag

C. Data Collection and Data Processing

We used full packet trace collected from one of the production process control networks in POSCO. The collected packet trace covered a 5-day period and total traffic volume was about 75Gbytes. In order to describe communication pattern, we aggregated traffic data to a flow that is a collection of packets sharing identical 5-tuple (source IP, destination IP, source port, destination port, and protocol) information because individual packets can not describe the communication pattern between network devices. After aggregating traffic data into the flow, the features were extracted from the flow. Table 3 lists 30 features that are considered in this research. All network measurement metrics explained in the previous section and other metrics are included in the feature set. These features are related with statistical information of connection log and intrinsic to each connection.

We manually analyzed and labeled each flow in the data set. When used for training and testing, each flow is described as a row and labeled as either normal or fault. Each row contains all features listed in Table 3 to describe training or testing example. Table 4 describes the example data set used for the training and testing step. The data set used for training and testing contains 60,572 rows of normal flow and 3,058 rows of fault flow.

D. Evaluation Metric

To measure the accuracy of generated decision tree with regard to the network fault classification result, we used three measures that are widely used in data mining to evaluate the accuracy: precision, recall, and F1 metric [17]. The former two measures are functions calculated by the true positive (TP), false positive (FP), and false negative (FN). The number of correctly classified objects in a class is referred to as the TP. Any object that is not correctly classified in a class is considered a FP. Similarly, FN is the number of positive objects erroneously classified as other classes. *Precision* measures how many identified or classified objects were correct and it is the ratio of TP to the sum of TP and FP. *Recall* measures how many objects in a class are misclassified as other classes. It is defined as the ratio of TP to the sum of TP and FN. The last measure, *F1* is a harmonic mean of recall and precision and better reflects the effectiveness of the classifier. These measures can be calculated as follows:

- $precision = \frac{TP}{TP + FP}$
- $recall = \frac{TP}{TP + FN}$
- $F1 = \frac{2 \cdot recall \cdot precision}{recall + precision}$

E. Experimental Results

In fault detection, reducing the number of false negatives (i.e., classifying fault data as normal data) is a very critical problem. As described in Section II, one fault event can be fatal especially in a control network; therefore, we conducted several empirical studies to reduce the false negatives. Table 5 describes the result of each experiment. All evaluation metrics were the result of 3-fold cross validation.

Firstly, we used the standard C4.5 algorithm to create decision trees which classify connections based on the features listed in Table 3. In this experiment decision trees generalized information by using pruning during their construction.

All three evaluation metrics have shown very high accuracy (over 99%) because the communication in control networks is much simpler than those of normal Internet. However, there were 128 FNs in the result. We made an observation that might account for FNs. The observation was that IP addresses were not considered an import feature during the decision tree construction. One network connection consists of two network devices and a pair of source IP and destination IP addresses describe one network connection. The standard C4.5 algorithm; however, used source IP address and destination IP address as independent features. To solve this problem, two different approaches were used.

Table 4. Sample Data Set

Index	Source IP	Destination IP	Source Port	Destination Port	…	Class
1	10.1.1.76	224.0.6.127	1043	8044	…	Normal
2	10.1.1.87	244.0.6.17	1043	8044	…	Normal
3	10.1.1.88	10.255.255.255	138	138	…	Normal
4	10.1.1.88	224.0.6.127	1046	8044	…	Normal
5	130.20.21.32	130.20.21.67	7024	7024	…	Normal
6	130.20.21.52	130.20.21.142	102	2056	…	Normal
7	130.20.21.213	130.20.21.255	138	138	…	Fault
8	130.20.21.213	130.20.21.255	138	138	…	Fault
9	130.20.21.213	130.20.21.255	137	138	…	Fault
10	130.21.21.157	130.20.21.67	6257	4526	…	Fault
…	...	…	…	….	…	….
63630	130.21.21.157	130.1.21.255	138	138	…	Fault

Table 5. Experimental Results

Experiment	TP Rate	FP Rate	Precision	Recall	F-measure	Class	# of FN	# of FP
Standard C4.5	0.974	0	1	0.974	0.987	abnormal	128	0
	1	0.026	0.989	1	0.994	normal		
IP pair as a feature	0.974	0	1	0.992	0.995	abnormal	40	3
	1	0.008	0.997	1	0.998	normal		
IP pair as a feature without pruning	0.992	0	0.999	0.99	0.995	abnormal	39	7
	1	0.008	0.997	1	0.998	normal		
2-stage decision tree	0.995	0	1	0.995	0.997	abnormal	22	2
	1	0.005	0.998	1	0.999	normal		

- ***Pair of IP addresses***: a pair of source IP and destination IP addresses is used as a feature in the second experiment. The number of FNs was reduced to 40 and all of the three evaluation metrics were increased. We also conducted the same experiment without the pruning option. One FN was reduced in this experiment. Even if the same data set was used, pruning generalized the decision tree and reduced node in the tree and it affected the accuracy of classification result.
- ***Multi-stage classification***: In the other two experiments, 2-stage decision tree classification was used in order to use IP addresses as important features. We constructed decision trees only with IP addresses in the first stage of classification. This decision tree classifies the fault based on IP addresses so that we can primarily detect IP related faults. Second stage of classification used all 30 features to construct the decision tree. The result showed a decrease of FNs to 22.

The last 2-stage classification showed the best performance among all experiments in terms of all accuracy metrics and number of FNs and FPs. However, this

methodology also could not detect all the faults. We manually analyzed the cause of the other 22 FNs. Most of the erroneously classified objects were very unique fault cases. Since some fault cases appeared only once in the entire data set, those fault data were not properly trained in the context of cross-validation. To detect these faults, 22 decision tree independent rules are needed. Although using extra rules can cause over fitting, it is desirable in control network fault detection due to the characteristic of the traffic pattern. The process control network components generate network traffic with specified periods. It means that every fault can be detected if the cycle of the data is trained well.

From the decision tree, we generated classification rules of the form:

if <condition> then <action>

These rules were applied to real fault detection systems which will be described in the next section.

V. System Architecture and Implementation

This section provides the design of a fault detection and diagnosis system for process control networks using classification rules generated by the data mining engine which uses decision tree. Figure 5 illustrates high-level system architecture of the process control network traffic monitoring system. The proposed system supports network diagnosis capability and remote accessibility. It consists of one or more monitoring probes, collectors (DB servers), a presenter (Web user interface), and a data mining engine. This distributed system architecture allows us to design flexible monitoring systems where multiple links can be monitored with a single point of data representation [10].

The monitoring probe is attached to the target link and is controlled remotely from the Web server running in the presenter. Packet traces are forwarded to the probe via port mirroring or passive signal tapping. Packet Collector captures packets from the interface and passes on to Flow Generator and Packet Log. Fault Analyzer records the list of IP network oriented metrics listed in Table 3 and runs the validation check against the fault detection rules extracted from the Rule Extractor which converts the decision tree structure into detection rules. If any of the metrics violates the defined detection rules, the alarm generator sends an email or SMS message. The metrics should be monitored for each flow.

The Data Mining Engine generates fault detection rules which reflect the correlation between network-level error symptoms after receiving network metrics and user's interpretation about actual network faults. The network administrator can attach the explanation to the alarm log via User Interpretation interface. It is more of an intuitive opinion about cause analysis. This information is later referenced to find a mapping relationship between network metrics and previously known fault cases by generating a decision tree. Binary packet log files are stored for further offline analysis if necessary. Packet Decoder converts binary packets into Packet Details Markup Language (PDML) [11] instance on users' request.

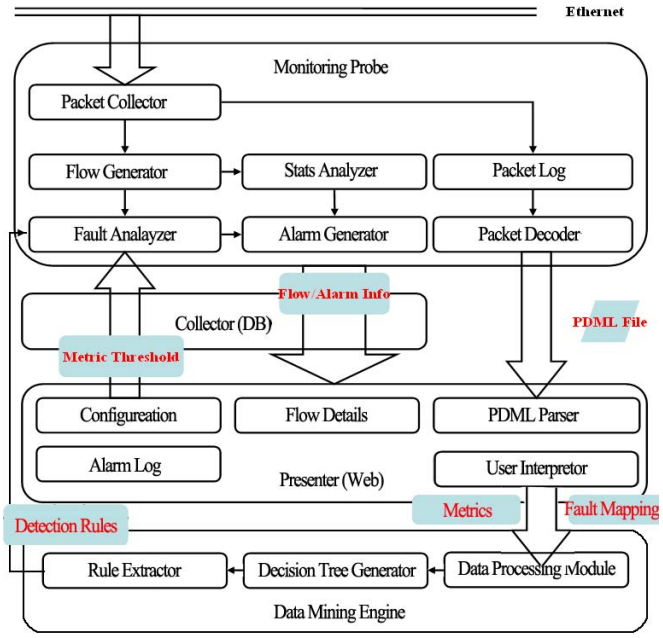

Figure 5. System Architecture

VI. Related Work

There has been a lot of research done on fault detection in IP networks. There is some good work in this direction using data mining techniques as well [12-15], but to the best of our knowledge there has been no effort to evaluate fault detection using data mining techniques for industrial process control networks. Various classification and clustering techniques have been tested to do fault detection in IP networks such as Support Vector Machines (SVMs) [18], which attempt to separate data into multiple classes (two in the basic case) through the use of a hyper-plane. Mukkamala et al., [14] used a more conventional SVM approach for fault detection. They used five SVMs, one to identify normal traffic, and one to identify each of the four types of malicious activity in the KDD Cup [19] dataset. Many researchers have tended to favor the use of SVMs. However, SVMs typically require training time that is quadratic in the number of training examples and SVMs are inefficient for the skewed data set.

Another very popular classification technique is decision trees. Frank [13] cites decision trees as a prime example of a classification method that is well suited for the intrusion detection domain, however he does not implement such a system. Apart from classification there are several other data mining approaches such as clustering [14] and fuzzy logic [12]. One of the recent works using the clustering technique compares various outlier detection algorithms to identify the faults in the IP network [14]. Most of this research used the DARPA data set [20] of the KDD cup data set for testing their technique. However, although there has been some research done in this area, still the traffic characteristics of control networks are different. There is a definite need to analyze the

faults in control networks, and tune the data mining techniques to fit the control network security systems.

In our previous research [7], we designed and implemented a remote fault detection and diagnosis system for process control networks, which has a multiple link monitoring applicability and flexible architecture. Although this system enhanced the fault detection capability greatly compared to other IP network monitoring tools, it required network operator's further analysis to determine the relation between network error symptoms and actual faults. In this paper, we added a data mining engine which can calculate the correlation between network error symptoms and actual faults in process control networks to improve our existing system.

VII. Concluding Remarks

Although there have been numerous studies on detecting network anomalies by investigating traffic from the Internet data plane, monitoring and analyzing process control networks has not been studied much thus far. The fault detection system presented in this paper provides a new systematic approach to detecting control network faults. However, existing IP network diagnosis tools (e.g., Sniffer [15], Wireshark [16]) do not yet have the capability to understand the failures in process control networks. Our fault detection system supports a very accurate but flexible process that continuously trains itself to learn detection rules from network metrics and user interpretation about already-known actual fault cases.

For future work, we plan to establish a prediction model using data mining techniques for actual network failures by matching the generated alarms with real network outage cases. We also plan to generate fault detection rules using data sets collected from other process control networks in POSCO.

Acknowledgment

This research was partly supported by This work was partly supported by the IT R&D program of MKE/IITA [2008-F-016-01, CASFI] and WCU (World Class University) program through the Korea Science and Engineering Foundation funded by the Ministry of Education, Science and Technology (Project No. R31-2008-000-10100-0)

References

[1] Nobuo Okabe. "Issues of Control Networks When Introducing IP," Proc. of Symposium on Applications and the Internet Workshops, Vol. 00, pp. 80-83, 2005.

[2] Fieldbus Foundation. FF-581-1.3, "FOUNDATION Specification: System Architecture," 2003.

[3] PROFIBUS International. IEC 61158, "Digital Data communication for Measurement and Control – Fieldbus for Use in Industrial Control Systems," 1999.

[4] MODBUS.ORG. "Modbus Application Protocol V1.0," 2002.

[5] ASHRAE, ANSI/ASHRAE Standard 135-1995. "BACnet A Data Communication Protocol for Building Automation and Control Networks," 1995.

[6] Feng-Li Lian, James R. Moyne, and Dawn M. Tilbury. "Performance Evaluation of Control Networks: Ethernet, ControlNet, and DeviceNet," IEEE Control System Magazine, 117(6), pp. 641-647, 2001.

[7] Young J. Won, Mi-Jung Choi, James W. Hong, Myung-Sup Kim, Hwa Won Hwang, Jun Hyub Lee, and Sung-Gyoo Lee, "Fault Detection and Diagnosis in IP-based Mission Critical Industrial Process Control Networks," IEEE Communications Magazine, Vol. 46, No. 5, May 2008, pp. 172-180.

[8] Francois Denis, "Pac learning from positive statistical queries," ALT 1998.

[9] J. Quinlan, "C4.5: Programs for Machine Learning," Morgan Kaufmann Publishers, 1993.

[10] Se-Hee Han, Myung-Sup Kim, Hong-Taek Ju and James W. Hong. "The Architecture of NG-MON: A Passive Network Monitoring System," 13th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, LNCS 2506, pp. 16-27, Montreal, Canada, October, 2002.

[11] Packet Details Markup Language Specifications. http://www.nbee.org/Docs/NetPDL/PDML.htm/.

[12] Dickerson, J. E. and J. A. Dickerson, "Fuzzy network profiling for intrusion detection," In Proc. of NAFIPS 19th International Conference of the North American Fuzzy Information Processing Society, Atlanta, pp. 301–306. North American Fuzzy Information Processing Society (NAFIPS), 2000.

[13] Frank, J. "Artificial intelligence and intrusion detection: Current and future directions," In Proc. of the 17th National Computer Security Conference, Baltimore, MD. National Institute of Standards and Technology (NIST), 1994.

[14] Mukkamala, S. and A. H. Sung, "Identifying significant features for network forensic analysis using artificial intelligent techniques," International Journal of Digital Evidence 1 (4), 1–17, 2003.

[15] Network General. http://www.networkgeneral.com/.

[16] Wireshark. http://www.wireshark.org/.

[17] Jiawei Han and Micheline Kamber, "Data Mining: Concept and Techniques, 2nd edition," Margan Kaufmann Publishers, 2006.

[18] N. Cristianini and J. Shawe-Taylor, "An introduction to support vector machines," Cambridge University Press, 2000.

[19] KDDCup. http://www.kdnuggets.com/datasets/kddcup.html.

[20] R. P. Lippmann, D. J. Fried, I. Graf, J. W. Haines, K., P. Kendall, D. McClung, D. Weber, S. E. Webster, D. Wyschogrod, R. K. Cunningham, and M. A. Zissman, "Evaluating Intrusion Detection Systems: The 1998 DARPA Off-line Intrusion Detection Evaluation," Proceedings DARPA Information Survivability Conference and Exposition (DISCEX) 2000, Vol 2, pp. 12-26, IEEE Computer Society Press, Los Alamitos, CA, 2000.