박사학위논문

Fault Detection, Diagnosis, and Prediction for IP-based Industrial Control Networks

원 영 준 (元 永 晙)

전자컴퓨터공학부 (컴퓨터공학전공)

포항공과대학교 대학원

2009

IP 기반 공정 제어네트워크에서의 장애 탐지, 진단, 및 예측

Fault Detection, Diagnosis, and Prediction for IP-based Industrial Control Networks

Fault Detection, Diagnosis, and Prediction for IP-based Industrial Control Networks

By

Young Joon Won
Division of Electrical and Computer Engineering
(Computer Science and Engineering)
Pohang University of Science and Technology

A thesis submitted to the faculty of Pohang University of Science and Technology in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Division of Electrical and Computer Engineering.

Pohang, Korea
November, 2009

Approved by

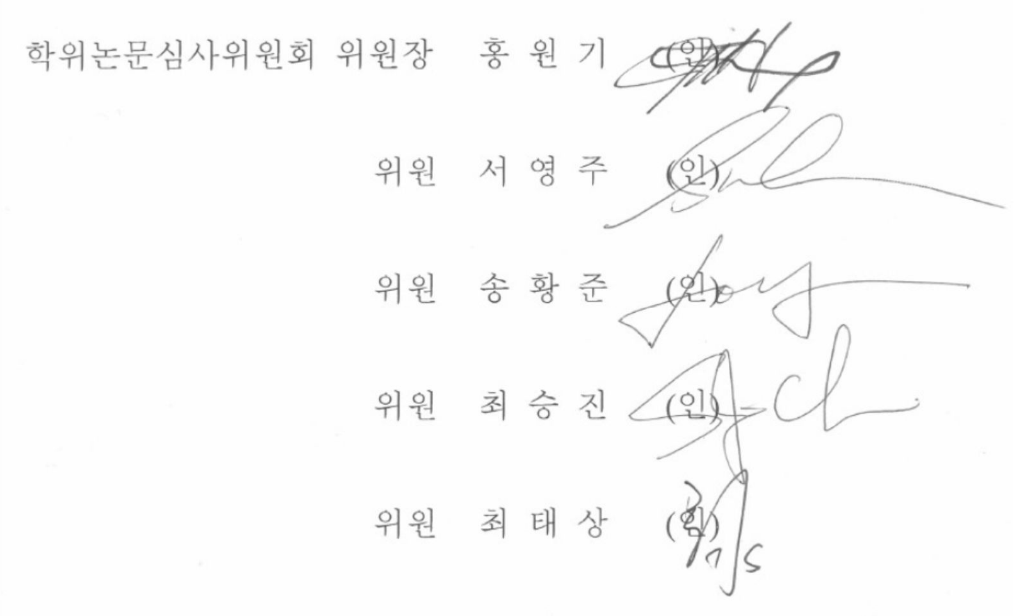
Major Advisor: James Won-Ki Hong

IP 기반 공정 제어네트워크에서의 장애 탐지, 진단 및 예측

원 영 준

위 논문은 포항공과대학교 대학원 박사 학위논문으로 학위논문 심사위원회를 통과하였음을 인정합니다.

2009년 11월 6일



학위논문심사위원회 위원장 홍 원 기 (인)

위원 서 영 주 (인)

위원 송 황 준 (인)

위원 최 승 진 (인)

위원 최 태 상 (인)

DECE
20063292

Young J. Won, "Fault Detection, Diagnosis, and Prediction for IP-Based Industrial Control Networks", Division of Electrical and Computer Engineering (Computer Science and Engineering), 2009, 105P, Advisor: James W. Hong, Text in English.

ABSTRACT

Mission-critical Industrial Control Networks (ICN) support secure and reliable communications of devices in process control or manufacturing environments. ICN has been heavily dependent on the proprietary protocols from various vendor-specific technologies. Recently, however, many of these proprietary protocols are being migrated to IP networks in order to consolidate various different types of networks into a single common network. This process has been undertaken to simplify the network operation, administration and maintenance and to reduce operational expenses and capital expenditures. Despite the wide deployment of ICN, most operators have very little knowledge on how to manage their ICN reliably and securely. This is mainly due to the operators' unfamiliarity with the various faults that occur on ICN and their defensive network maintenance strategy. The current process of detecting and diagnosing faults is mostly manual and thus the operators generally detect problems only after noticeable malfunctioning has occurred. In addition, the existing IP diagnosis techniques have not been able to fully handle fault symptoms and mainly focus on network diagnostics rather than process or device diagnostics. This thesis presents that the absence of advanced fault diagnosis techniques leads to the development of new methodologies which are suitable for ICN. We describe unique traffic characteristics and categorize the faults of ICN. We also propose novel fault diagnosis, prediction, and adaptive decision methodologies, and verify them with real-world ICN data from POSCO. Finally, this thesis proposes fault diagnosis system architecture for ICN. Our experience in developing the fault diagnosis system provides a firm guideline to understand the fault management mechanisms in a large ICN. Overall, this thesis presents a complete cycle of handling faults in IP networks within the scope of ICN: Monitoring, analysis, and prediction.

Contents

List of Figures

List of Tables

1 Introduction

This chapter provides a brief introduction to Industrial Control Networks (ICN). The motivation and problems of current ICN fault diagnosis approaches are illustrated in the outline of our solution approach.

1.1. Background

Communication within a complex automated manufacturing system, typically conveying belt lines in a factory, has been heavily dependent on the proprietary protocols from various vendor-specific technologies. The integration of the devices and programming tools from multiple vendors in a single network was almost impossible prior to the introduction of IP technology; in fact, the total solution from one single vendor was preferable for network designers and maintenance personnel to handle the process control network requirements. Traditional process control network technologies (e.g., FOUNDATION [1], PROFIBUS [2] MODBUS [3], BACnet [4]) were developed separately from the relatively recent emergence of Ethernet/IP-based network technologies. Recently, newer versions of these technologies have adopted Ethernet (e.g., Industrial Ethernet) and IP for reasons of low cost, high scalability and interoperability, and easy maintenance. Their place in the market is still limited because the decision of moving toward an all-IP manufacturing environment is beyond the technical superiority of Ethernet and IP. Replacing existing devices, including numerous low-end controlled machineries (e.g., sensors, actuators, motors, etc.), requires a huge investment.

Mission-critical Industrial Control Networks (ICN) support secure and reliable communications of devices in a process controlling or manufacturing environment. ICNs have been heavily dependent on the proprietary protocols from various vendor-specific technologies. Recently, however, many of proprietary protocols are being migrated to IP networks in order to consolidate various different types of networks into a single common network. This process is undertaken to simplify the network operation, administration and maintenance, and to reduce operational expenses and capital expenditures. Despite the wide deployment of ICN, most operators have very little knowledge on how to manage their ICN reliably and securely. This is mainly due to the operators' unfamiliarity with the various faults that occur on ICN and defensive network maintenance strategy. The current process of detecting and diagnosing faults is mostly manual and thus the operators generally do not detect problems until after noticeable malfunctioning has occurred. In addition, the existing IP diagnosis techniques have not been able to fully handle fault symptoms and mainly focus on network diagnostics rather than process or device diagnostics.

We collected and analyzed fault maintenance cases from real-world ICN systems in POSCO, the world's 2nd largest iron and steel manufacturer [24]. This thesis proposes monitoring and diagnostic techniques across all the OSI model layers for Ethernet/IP-based ICN systems. We also present how our proposed fault symptoms and network monitoring techniques are applied to the diagnosis of process control operations and related machineries.

1.2. Problem Statements

Communication failures reflect the status of machineries and their operation correctness. Communication failures in any stage of ICN can be fatal. Since all the machineries operate within a tight boundary for control delay and order, there is a possibility that a single communication failure to a device may delay or even force a shutdown of an entire plant. For instance, a steel manufacturing plant involves a series of continuous processes which are dependent on one another for the final product. It is crucial for the engineers to detect early symptoms of unstable network communications between control entities, analyze the cause, and take an appropriate countermeasure prior to the failure event. Therefore, the ICN must sustain robust communications between the control unit (e.g., process controller) and low-end devices (e.g., sensors, actuators, motors). It is deployed in mission critical operations which require a maximum level of network stability. Network stability is often described with several categories of network performance QoS metrics, referring throughput, delay, and loss measurements in packet exchange networks. The question arises whether these network performance metrics are sufficient to run valuable diagnostics [5] of ICN components and their communications. Any abnormality decision with respect to the typical IP traffic behavior does not necessarily coincide with ICN fault cases. A precise and specific diagnostic technique for ICN is required in order to remove uncertainty in detecting a lead to trouble cases.

In this section, the research problems and goals are briefly listed. This thesis tries to answer the following key questions.

- What is the importance of ICN and what is unique about it?
- Are the existing fault detection and analysis techniques in the conventional IP networks sufficient for diagnosing the ICN specific faults?
- What features should be investigated in order to identify or represent the ICN fault cases?
- What methodologies should be developed for accurate, scalable, and adaptive fault diagnosis system for ICN?
- How can we predict the fault occurrences in advance to prevent possible failures of the network? What is the next research step towards more advanced fault diagnosis technique?
- How does the diagnostic system help reduce OPEX cost and minimize network stoppage periods in mission-critical networks?

1.3. Research Approach

In this section, we explain the solutions to the problems mentioned in section 3 and the research methodologies. We collect and analyze fault maintenance cases from the real-world ICN in POSCO. We propose monitoring and diagnostic techniques across all the OSI model layers for ICN. We also present how our proposed fault symptoms and monitoring techniques are applied to the diagnosis of process control operations and related machineries.

Pertaining to these questions, the following items are the goals of this thesis.

- This thesis will provide an overview of the existing fault diagnosis techniques and previous traffic analysis results in ICN.
- This thesis will investigate the traffic characteristics and monitoring features for faults using real-world ICN traffic traces and fault cases.
- This thesis will propose fault diagnosis system architecture for ICN which targets remote, scalable, and flexible fault detection and analysis.
- This thesis will propose fault diagnosis and prediction methodologies in the part of the proposed system.
- This thesis will validate the proposed methodologies by implementing a prototype system and applying it to a real-world ICN environment, POSCO.
- This thesis will provide a guideline for handling faults in network by sharing valuable experience of a complete network management cycle: Monitoring, analysis, and prediction.

This thesis first investigates the operational and maintenance needs of real-world process control networks at POSCO. Their single operational site consists of more than 40 manufacturing plants with each plant having its own process control network. We selectively choose a number of process control plants which are particularly vulnerable to network outage and have previously experienced unstable network conditions. This thesis presents that the absence of advanced fault diagnosis techniques leads to the development of new methodologies which are suitable for ICN. We describe unique traffic characteristics and categorize the faults of ICN. We also propose novel fault diagnosis, prediction, and adaptive decision methodologies and verify them with real-world ICN data from POSCO. Finally, this thesis proposes fault diagnosis system architecture for ICN. Our experience in developing the fault diagnosis system provides a firm guideline to understanding the fault management mechanisms in a large ICN.

1.4. Outline of this Dissertation

The organization of this thesis is as follows. Chapter 2 describes the issues of running ICN and the previous analysis results of ICN. In Chapter 3, we introduce communication models of ICN and possible monitoring domains for fault detection. We also illustrate some observation and discuss ICN traffic characteristics from the real-world measurement site. Chapter 4 investigates the ICN fault cases and introduces the proposed fault diagnosis techniques. We describe the design and implementation details of our ICN fault diagnosis system in Chapter 5. Discussions on fault prediction, the adaptive decision engine, and the self-governing diagnosis system are drawn in Section 6. It also provides the evaluation details of the proposed diagnosis techniques. Finally, Chapter 7 concludes the thesis with summary, contributions, and possible future work.

2 Related Work

In this chapter, we introduce the current issues of managing and designing ICN. The related research on ICN traffic analysis is also given. Furthermore, we provide an overview of the existing network traffic monitoring and analysis techniques and alarm management schemes in the telecommunication industry. By investigating previous work, we can determine the ICN specific requirements for fault diagnosis.

2.1 Issues of ICN

Over the years, communication involving complex ICN has been heavily dependent on the proprietary protocols from various vendor-specific technologies. Integration of the machineries and programming tools from multiple vendors in a single network was almost impossible prior to the introduction of Ethernet/IP technologies. Until recently, and still in many cases, the total solution from a single vendor was the preferable choice for network designers and maintenance personnel for handling the process control requirements. The majority of current ICN in the world are now combinations of Ethernet/IP and Fieldbus [6] control technologies. Industrial Ethernet alternatives (e.g., Modbus TCP/IP, RAPIEnet, EtherNet/IP, PROFINET, Ethercat, TCnet, and Vnet/IP) are proposed to replace the old-fashion device-to-device wiring Fieldbus technologies. The growing need for monitoring Ethernet/IP-based ICN systems requires a careful study of traffic characteristics, models, and fault detection techniques. We also encounter more problems which reflect vulnerabilities of the IP networks in general, such as Internet and enterprise networks.

Some researchers have discussed the important issues in moving towards the IP-based ICN [7][8]. The roles and environments of ICN are distinguishable depending on the final products; however, they all are deployed in mission critical and non-fault tolerant operations. Thus, minimum delay and guaranteed transmission are particularly important QoS attributes for every existing communication session in ICN [9]. Secondly, while we are in the process of moving slowly toward IP networks, we encounter the problems reflecting vulnerabilities of IP networks in general. As mentioned, the majority of the current ICN are mixtures of IP-based and non-IP based control technologies. This necessitates therefore different monitoring and maintenance techniques [8]. The growing need for monitoring Ethernet/IP-based ICN systems requires a careful study of traffic characteristics, models, and fault detection techniques.

2.2 ICN Traffic Analysis

The guideline for running ICN has taken primitive and defensive approaches: Isolation and Prevention. Isolating ICN from other enterprise networks is one way to protect the serenity of the remote process control. Prevention refers to the strict control access of unproven or troublesome devices to the ICN. In addition, the e-diagnostics guideline [10] for semiconductor factory communication suggests the level 0-3 steps of network maintenance: Access and remote collaboration, Collection and control, Analysis, and Prediction. The principles behind this guideline are network/machinery fault monitoring and analysis techniques.

The traffic characteristics [11][12][13][14] have been studied to understand the traffic nature of ICN. Jasperneite et al. [15] showed that factory communication systems follow an on/off traffic model which can be applied to recognize any abnormality of machine or network malfunctioning from inconsistent traffic occurrence. Some QoS requirements [9][16][17][18][19][20] across the physical, link, network, and transport layers in the OSI reference model [21] have been proposed to help the diagnostic decisions in ICN systems. These requirements were illustrated with tolerable boundary figures for utilization, throughput, time delay, loss, and error rate violation. The existing tools (e.g., Sniffer, Observer, Wireshark), which were originally designed for the IP data networks, can be used to analyze these lower OSI-layer QoS violations. However, their capability to detect faults was very limited due to the different nature of the traffic in comparison to ICN.

Industrial Ethernet applicability was evaluated via simulation [15][18][22][23]. The communication delay of less than 10 ms between the devices was considered as suitable for robust ICN systems. Yet, the delay measure is not the only indicator for quality Ethernet-based ICN communications.

2.3 Network Traffic Monitoring and Analysis

Network traffic monitoring and analysis has been considered as a challenging and extra burdensome task by the network management community [53]. Due to the introduction of multi-gigabit networks (e.g., OC-48, OC-768), network administrators can no longer rely on off-the-shelf monitoring equipment anymore but costly and high-performance specialized network interface cards (NIC). In addition, the SNMP community continues to focus on network device management rather than on traffic monitoring, and the only RFC for traffic monitoring remains RMON-2 [53] in 1997.

The performance degradation of off-the-shelf products in traffic monitoring is mainly caused by NIC performance and memory speed limitations [54]. The monitoring specialized cards from Napatech [55], Inveatech [56], and Endace [57] are equipped with network process unit (NPU) to offload CPU overhead in the computer. At multi-gigabit speeds, software applications cannot cope with the volume of data and the depth of analysis required; thus, the ASIC and FPGA-powered network cards, such as P-series from Force 10 [58], make it possible to have onboard packet filtering or analyzing (e.g., deep packet inspection for real-time worm detection). Finally, multicore network processors, such as the Intel IXP family [59], Cavium Networks [60], and Tilera [61], facilitate ASCI and FPGA approaches for coping with high packet processing speeds and application programmability beyond 10 Gbps.

The sampling scheme is an alternative approach to monitoring high-speed network links without reliance on costly devices. The IETF working group psamp [62] is chartered to define a standard set of capabilities for network elements to sample subsets of packets by statistical and other methods. C. Estan, et al. [63] also propose a sampling mechanism to select heavy flows using minimal memory and processing power.

Table I describes a selection of well-known network monitoring tools [80] and experimental analysis tools in the community. We select these applications according to their respective popularity in order to benchmark basic functionalities for the traffic monitoring system. Table II presents off-line traffic analysis and modeling tools. This thesis relies on some of these tools to analyze traffic characteristics of ICN.

Table I. Popular network monitoring tools – Benchmark selection for functionalities

Name	**Functionalities and available URL**	**Open Source?**
Ntop	IP traffic usage monitor, Web-based UI NetFlow [64] and sFlow [65] compatible http://www.ntop.org/	Yes
Ngrep	Packet (regular, hex) matching & Display Ethernet, PPP, SLIP, FDDI, Token Ring http://www.packetfactory.net/projects/ngrep/	Yes

EtherApe	Display network activity graphically (Ethernet, FDDI, Token Ring, ISDN, PPP, SLIP) UNIX, Non-commercial http://etherape.sourceforge.net/	Yes
Nagios	Network monitoring at host level Notification function via email, pager http://www.nagios.org/	Yes
Argus	Monitoring IP flow metrics, connectivity, capacity, demand, loss, delay, etc. on a per transaction basis http://www.qosient.com/argus/	Yes
CoralReef	Real-time traffic monitoring, analysis, http://www.caida.org/tools/measurement/coralreef/	Yes
Dsniff	Packet sniffer http://www.monkey.org/~dugsong/dsniff/	Yes
Tele Traffic Tapper	Real-time, graphical, and remote traffic monitoring http://www.csl.sony.co.jp/~kjc/software.html#ttt	Yes
Microsoft Network Monitor	Real-time and post-capture modes of network data analysis (MSDN Package)	No
Ettercap	Terminal-based network sniffer/interceptor/logger for Ethernet LANs, dissection of ciphered protocols (e.g. ssh, https), OS fingerprinting http://ettercap.sourceforge.net/	Yes
ManageEngineTM NetFlow Analyzer	Netflow collector and analyzer http://manageengine.adventnet.com/	No
Wireshark	Network protocol analyzer, dissection of hundreds of protocols and media types, reconstruction of a TCP Session http://www.wireshark.org/	Yes

Kismet	802.11 wireless network detection, sniffer, and IDS http://www.kismetwireless.net/	Yes
Tcpdump	Classic packet sniffer,WinDump for Windows http://www.tcpdump.org/ http://windump.polito.it/	Yes
SolarWinds	Network discovery, monitoring, attack Password cracker function (Router, SNMP) http://www.solarwinds.net/	No
NetScout Sniffer	Real-time traffic monitor, troubleshoot, analyze, alarm report (10/100 Ethernet, Gigabit, ATM, PoS) http://www.networkgeneral.com/	No
OmniPeek	Real-time traffic monitor and analysis (graphical representation, statistical report) http://www.omnipeek.com/	No
FlowScan	NetFlow analysis using CAIDA's cflowd flow tool [66]	Yes
Nfsen	Web-based netflow analysis http://nfsen.sourceforge.net/	Yes

Table II. Traffic analysis and modeling tools

Name	**Functionalities and available URL**
TCP Replay	Replays a captured packet file on an interface http://tcpreplay.synfin.net/trac/

capinfo	Displaying statistics – timestamp, packet size http://www.ethereal.com/docs/man-pages/capinfo.1.html
tcpstat	Displaying statistics – bandwidth, # of packets, pps, STD packet size, and etc. http://www.frenchfries.net/paul/tcpstat/
tcpflow	Reconstruct TCP connections (flows), retransmission, out-of-order delivery, fragments analysis http://www.circlemud.org/~jelson/software/tcpflow/
tcptrace	Retransmission, RTT, Window size, throughput analysis http://jarok.cs.ohiou.edu/software/tcptrace/
FFT-FGN-C	Synthesizing a type of self-similar process http://ita.ee.lbl.gov/html/contrib/fft_fgn_c.html
Sanitize	Filtering packet contents (e.g. TCP SYN/FIN/RST packet) http://ita.ee.lbl.gov/html/contrib/sanitize.html
tcpreduce	Display TCP connections http://ita.ee.lbl.gov/html/contrib/tcp-reduce.html
tcpdpriv	Eliminating confidential information from packets http://ita.ee.lbl.gov/html/contrib/tcpdpriv.html
tcpslice	Merging of captured packet files http://sourceforge.net/projects/tcpslice/
tracelook	Graphical representation of TCP connections http://ita.ee.lbl.gov/html/contrib/tracelook.html

2.4 Alarm Management

In general, alarm refers to an event or a phenomenon that raises the awareness of the administrator to a potential threat. Alarm management requires a clear understanding of the scope of alarm in accordance to the purpose of the system being managed. Stanton [67] defines alarm in the following frameworks: Stimuli-based, message-, and response-based models. The stimuli-based model refers to an alarm as a particular state of the resource, such as network devices and network process. In the message-based model, an alarm is simply a notification process from the resource to the administrator. The corresponding resource itself would not be aware of state changes without a third party interpretation of such phenomenon. The response-based model considers an alarm with the interpretation by a human or management process in the network. This model has associated uncertainties resulting from unclear correlations between causes and events, and requires the filtering of meaningful alarms from the flood of alarms.

Alarm interface refers to the delivery mechanism and content of an alarm-triggering event. Various standards have been developed in the telecommunication industry [68]. X.733 [69], defined by ITU-T, is a de facto standard for alarm interfaces. The model is notification-focused, and defines the actual alarm state change messages. 3GPP [70] defines alarm using a state-based definition rather than a notification based definition. It supports administrative actions for acknowledgement and alarm comments. IETF Alarm MIB introduces an 'alarm model'. It is capable of mapping native SNMP notifications and object states to alarm states, but not all notifications report alarm state changes. TM-Forum defines a broad set of interfaces including different attempts, such as MTOSI [71] and OSS/J [72]. Section 4 will provide further details of the alarm definition used in this thesis and its relation to the fault and failure.

3 Communications in ICN

This chapter introduces ICN communication models and possible monitoring domains for fault detection. It also illustrates some observations and traffic characteristics from the real-world measurement site.

3.1 Overall Architecture

Typical process control networks are illustrated in Figure 1. The process control network components are organized in a hierarchical fashion where the controller at the top triggers corresponding actions in one or more controlled devices. The followings are brief descriptions of each element and its role.

- **Process Controller (PC):** This is a part of the software and hardware package provided by the Programmable Logic Controller (PLC) vendors. It is the process control software on a computer running UNIX or Windows that can remotely access PLCs. Custom-built or vendor provided server applications are placed in a PC to communicate with PLCs. It also communicates with the machines running Human Machine Interface (HMI) solutions that provide graphical presentation of real-time process status monitoring.
- **Programmable Logic Controller (PLC):** This 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 lines) is handled by the custom-built software programs running in PLC.
- **Controlled Devices:** These machineries refer to sensors, actuators, motors, etc. They receive the command signals from PLCs via the embedded interface and perform various tasks.

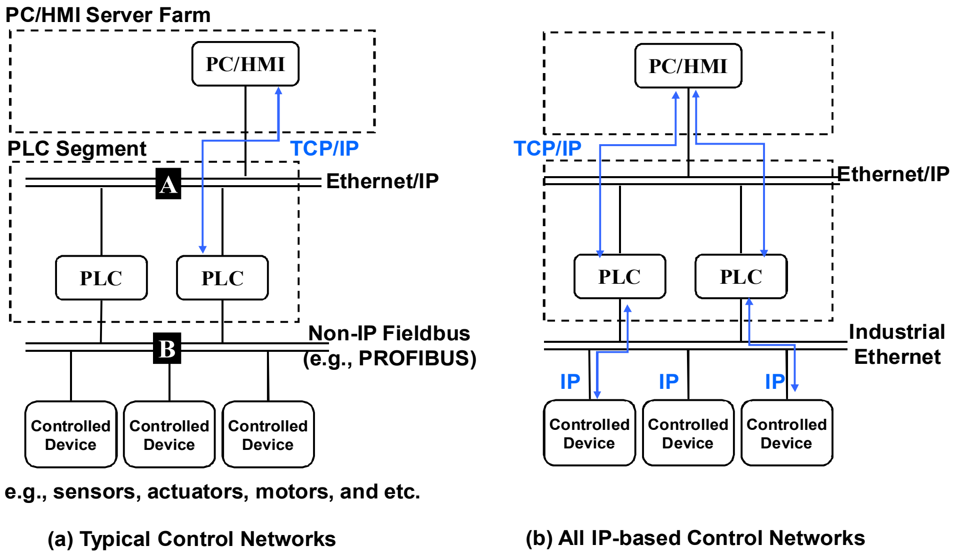


Figure 1. Simplified View of a) typical; b) all-IP-based ICNs

Since the late 1980s, communications between these three entities are handled by Fieldbus technology, which is suitable for control in a distributed environment. It is more of a vendor-driven technology and relies on proprietary protocols. The widely known PLC manufacturers (e.g., Siemens, ABB Ltd., Mitsubishi Electric) have been actively involved in deploying their choice of Fieldbus technologies in their solutions which left little room for the actual users, like POSCO, to customize their purchase for their own operational and maintenance needs. A few problems have been observed when maintaining the vendor specific solutions: high hardware cost (e.g., EIA-485 network interface), PC and PLC programming difficulty, and a lack of experienced engineers. These problems occur because the details of PLC technology are undisclosed to the users and the underlying mechanisms for communication are difficult to understand.

More difficulty arises when merging multi-vendor devices into a single network. Communication between different vendor products is often unreliable; consequently, the Ethernet and IP technologies were considered as an alternative option to give more flexibility, scalability and lower cost in designing the process control networks. The communication over IP networks was more convenient in terms of understanding the communication mechanisms and network programming for the engineers.

Figure 1 (a) shows the current status of process control networks, which are combination of IP-based and non-IP based control technologies. Figure 1 (b) illustrates the future of all IP-based process control network environments where the communications between PLC and controlled devices are established over Industrial Ethernet [25] and IP. Adapting to such a scheme requires more IP addresses for every controlled device and reprogramming of existing PLCs. Yet, most companies are reluctant to pursue this newer architectural model due to the uncertainty in stability and security of IP communication in a harsh manufacturing environment. This thesis focuses on a wide deployment of choice in ICN as illustrated in Figure 1 (a).

The ICN and its system components are organized in a hierarchical layering architecture. Table III illustrates the categories of ICN layering architectures. The Computer Integrated Manufacturing (CIM) model [11] consists of five layers. The simplified model [23] aggregates the CIM model into three layers: plant, cell, and device levels. Each connection point between the layers is a possible monitoring location.

Table III. Layering Architecture in ICN

CIM model layer	**Simplified model layer**	**Protocol in use**	**Control Entity**
Corporate management level	Plant level	Ethernet/IP	PC, HMI
Plant management level			
Supervisory level			

Cell control level	Cell level	Ethernet/IP+ Fieldbus	PLC
Device and sensor-actuator level	Device level	Fieldbus	Sensors, Actuators, Motors

3.2 Monitoring Domains

There are two different types of monitoring techniques: Active and passive measurements. The active measurement involves the periodic or request-based polling of certain control entities or communication status information. The snapshot of such information (sampling) may not be able to capture the moment of fault occurrence because the continuous monitoring of multiple devices or communication sessions is almost impossible using the active approach. It also adds an extra traffic burden to the existing ICN which may interfere with process operations. The delay and loss metrics are often sampled by the active method. Passive measurement involves collecting various process message formats encapsulated in packets or tokens. Router/switch or network link tapping devices at the monitoring point can forward the traffic for further analysis. Continuous monitoring without any network interference is possible; however, the large number of packet comparisons hinder real-time data analysis.

Two passive monitoring domains exist in ICN: PC/HMI to PLC network and PLC to low-end device network. Figure 1 illustrates the topological view of ICN and two monitoring points, A and B. This thesis focuses on monitoring the first half (A) of the ICN model which targets the PC/HMI to PLC communication session over Ethernet/IP. Although many IP network diagnosis tools are available, they often cannot detect control network failure cases due to a significant difference in traffic nature as well as distinct communication fault characteristics. No IP network diagnosis tools have satisfyingly supported process control network specifics which must reflect the status of device operation along with network conditions. Simultaneous monitoring of point A and B is desirable; however, in the upcoming section, we show how our application-layer QoS monitoring technique can discover the fault without monitoring point B. Since the second half (B) is also slowly moving toward Ethernet-based communication, the techniques developed in this work can be directly applied.

3.3 Empirical Analysis: Traffic Characteristics

The experimental data set we used was collected at the process control networks of POSCO, the world's 2nd largest iron and steel manufacturer. It operates a number of plants world-wide, and a single operational site consists of more than 40 manufacturing plants, with corresponding process control networks where they are organized in a synchronous and sequential order. Each process control network is a group of edge network segments. In simpler terms, a number of networks are working together to interconnect the machineries running continuously on a conveyor belt.

Our assumption is that no other data traffic is injected into the monitoring PLC networks. This was achieved by the complete isolation of the PLC network from the Internet or any other enterprise networks. The private IPs and dedicated Ethernet links were assigned to PLC devices.

3.3.1 Traffic Summary

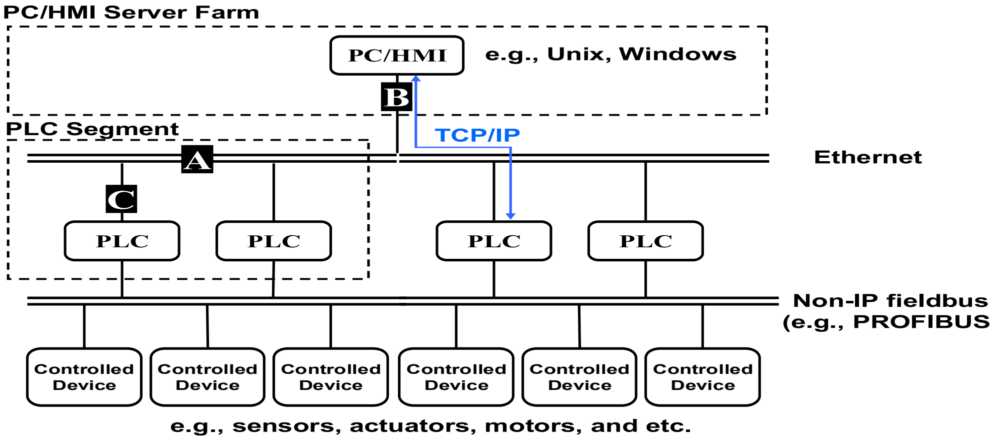


Figure 2. Measurement points – A (edge network), B (backbone network), and C (edge network)

We collected the traffic traces from various points of the process control networks using the standard libpcap [26]. Points A, B, and C in Figure 2 refer to the top of PLC segment, the process control backbone network, and the nearby end-host (PLC device), respectively. These representative locations were carefully selected to provide a precise and unbiased snapshot of the network.

Table IV illustrates the traffic summary of the collected traces. We analyzed a week-long trace at one of the edge segments, a typical working-hour trace at the plant backbone, and short (e.g., 5 minutes) traffic traces at the end-host segment – Segment A, B, and C, respectively. The actual name of each process segment is undisclosed due to security reasons. Most traffic is exchanged using TCP and the average utilizations yield a very low percentile in the 100 Mbps physical links environment. We observed that the total traffic volume of Segment B is almost same as those of Segment A regardless of its exceeding packet counts (e.g. triple). It implies a low yielding traffic volume and a major occupancy of light-size packets. Interestingly, in Segment B, only 533 flows are TCP flows and responsible for 121M packets. In fact, a small number of identical sessions with fixed amounts of hosts (PLC and PC) are continuously observed and generate the traffic.

Table IV. Summary of the datasets

Data Set	Date	Duration	Byte	Packets	Flows	TCP	Util.	Location
Segment-A	2006-09-29	170 hrs (7 days)	63.5 GB	542 M	48 K	98 %	1 %	Edge
Segment-B	2007-02-27	10 hrs (12:00-20:00)	74 GB	122 M	25 K	99 %	19 %	Backbone
Segment-C	2006-05-11	5 mins (13:15-13:19)	22 MB	84 K	48 K	99 %	0.57 %	Edge

Figure 3 (a) illustrates a long-term traffic pattern of a particular PLC segment – Segment A. The graph shows that bandwidth consumption is not bound to the time-of-day effect in the typical IP data networks where more traffic is generated during the day or working-hours. Instead, bandwidth usage is very steady and predictable throughout the course of the monitoring period. The sudden drops in the graph, the shade regions indicate an instant shutdown of the process due to scheduled or unscheduled maintenance purpose. Bandwidth consumption is strictly proportional to the number of devices in the network. Thus, network planning for control IP networks can rely on a very precise projection of the bandwidth growth model which is almost impossible in other types of IP networks.

The Segment B traces are a collection of multiple instances of the Segment A level of traffic. Figure 3 (b) shows a short-term bandwidth measurement, but with much larger traffic volume at the backbone. Its pattern closely coincides with the behavior exhibited in Figure 3 (a). Indeed, the microscopic view of traffic behavior can reflect the long-term behavior without loss of generality in control IP networks. This is because there are fixed patterns evident in all sessions occupying the control IP networks. More details will be covered in the following sections.

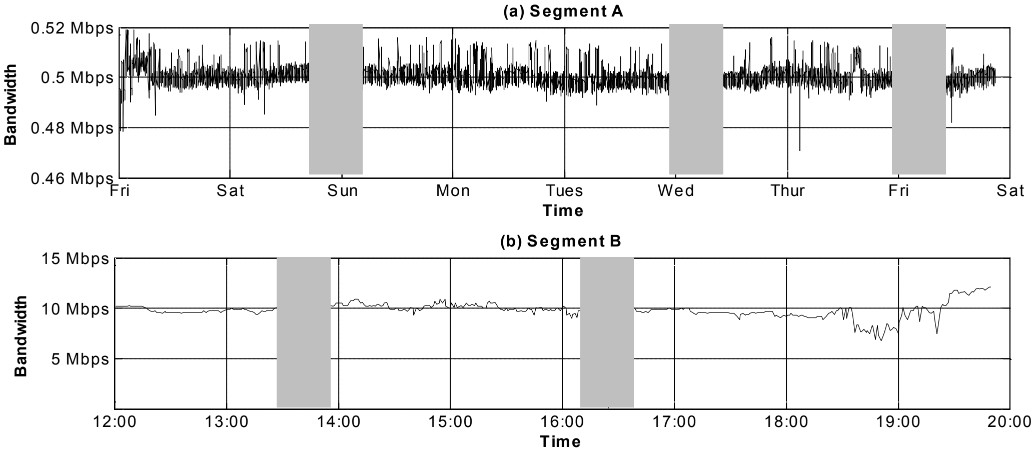


Figure 3. Traffic volume - a) segment A; b) segment B

3.3.2 Traffic Cycle

Figure 4 illustrates the four representative packet arrival patterns in PC-PLC sessions. The upper plane (above 0 on y-axis) of the graph indicates a unidirectional packet size arrival sequence according to the packet inter-arrival time. The lower plane is a packet size arrival sequence of the corresponding reverse transmission. Thus, a single graph represents two bidirectional flows. The packet inter-arrival time is measured in the time granularity of milliseconds. The dense region of the graph implies that the packet inter-arrival gap is reduced. In all four graphs, we can see the distribution of packet sizes and can also observe a unique and regular cycle of dense and sparse region occurrences over a short time period. For example, Figure 4 (a) shows an approximate 12-second cycle of two dense regions followed by a sparse region in both upper and lower planes. The lower plane exhibits a regular cycle of 1000 byte packet transmissions every 1.5

seconds. In a similar fashion, the rest of the graphs can be expressed as traffic pattern candidates for general PLC-PC sessions. All the sessions in our measurements belong to one of the pattern shapes shown in the graphs of Figure 4. Figure 4 (b), (c), and (d) illustrate the session traffic patterns for different PLC vendor solutions or sessions involving possibly in different processes.

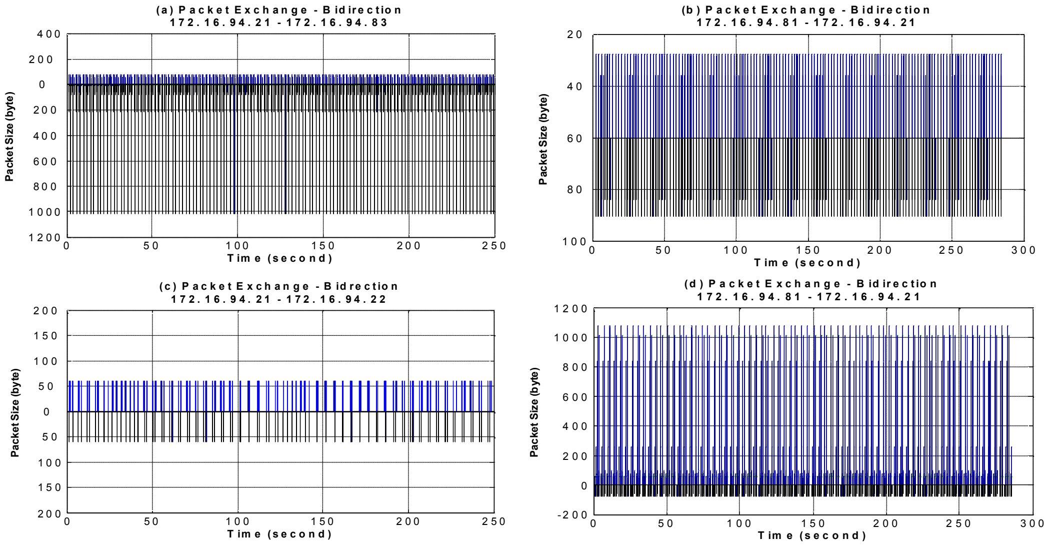


Figure 4. Bidirectional packet-size arrival sequence patterns of PC-PLC sessions

The average inter-arrival time of these four sessions ranges from 120 ms to 1.5 s. Note that, the selected sessions were operational without any performance irregularities at the time of the monitoring period. A matter of hundreds of millisecond or above may not be acceptable a set of values for a packet delay in the IP data networks. It is a unique characteristic where relatively longer packet delays are acceptable.

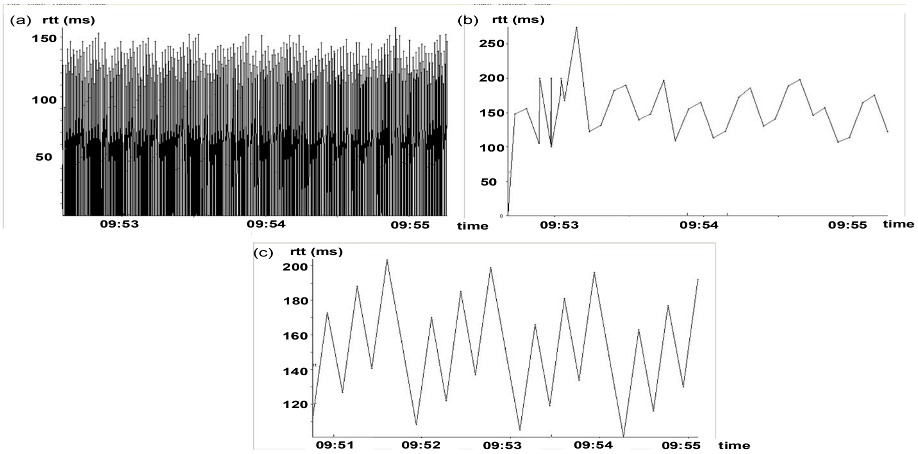


Figure 5. RTT measurement of PC-PLC sessions

Figure 5 illustrates the three sample RTT measurement graphs over the monitoring periods. The RTT values of each session are measured using tcptrace [27]. These PLC-PC sessions show clear periodic patterns which distinguish them from normal IP sessions. Their maximum RTT values also range from 150 ~ 250 msec, coinciding with the packet inter-arrival values of the sessions in Figure 4. The fixed packet arrival sequence is again apparent through the recurring shape of the graph.

It is important to recognize such pattern information in on-going sessions when detecting traffic anomalies and malfunctioning devices. This can be used as a guideline to determine 'irregularity' from the previously known communication patterns and avoid the ambiguous definitions of anomaly in control IP networks.

3.3.3 Traffic Symmetry

Figure 6 shows the symmetric behavior in terms of the number of packets being exchanged in a session. The upper plane (above 0) of the graph indicates packets per second (PPS) counts over the monitoring period in one direction. The lower plane shows the corresponding packet counts in reverse direction. All the graphs show almost identical packet transmission symmetry where one side shows slightly more packet counts than the other. These periodic packet generation patterns imply a simple request-response behavior between PC and PLC which follows a similar trail of HTTP behavior [28]. However, unlike the HTTP traffic, the request object (or service) and the corresponding reply are very much fixed in size and repetitiveness. The average PPS count of the sessions in control IP networks is below 10 PPS which is low compared to the Internet traffic.

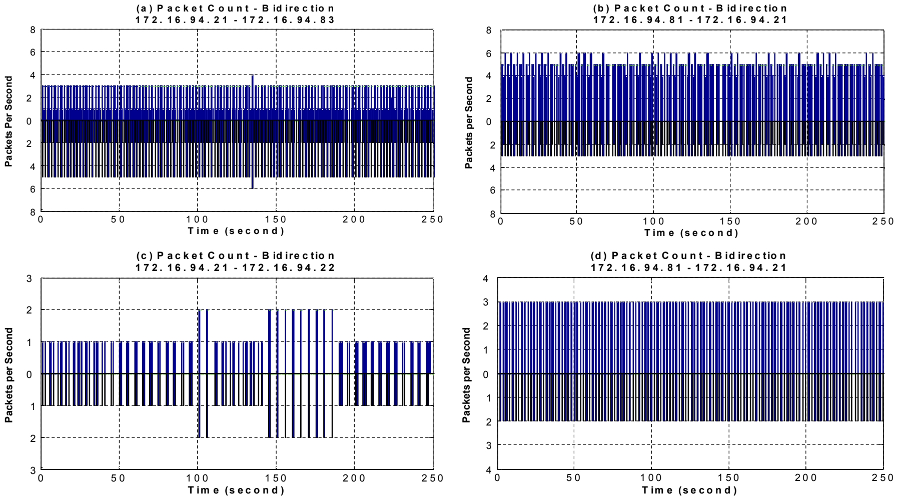


Figure 6. Bidirectional packets per second counts in PC-PLC sessions

3.3.4 Packet Size Distribution

The purpose of transmitting packets in ICN can be classified into the following: Signaling purpose for PLC operations, HMI display purpose, and management purpose (e.g., SNMP). At the time of data collection, we were assured that no management traffic data was being injected into the network. Figure 7 (a) and (c) indicate the packet size distribution of packets intended for signaling purposes. These figures show that over 90% packets are less than 100 bytes in size. In fact, their sizes range from 60 to 80 bytes which is just sufficient for containing TCP header information with a very few bytes for the actual payload. This explains the low bandwidth consumption in ICN.

Figure 7 (b) shows the increase of average packet size compared to Figure 7 (a) and (c) where they are collected at near the edge PLC segment. At Segment B, we collected the HMI request/reply packets along with the PLC packets. The packet size distribution shows two distinguishable characteristics depending on where we collect the data – backbone or edge.

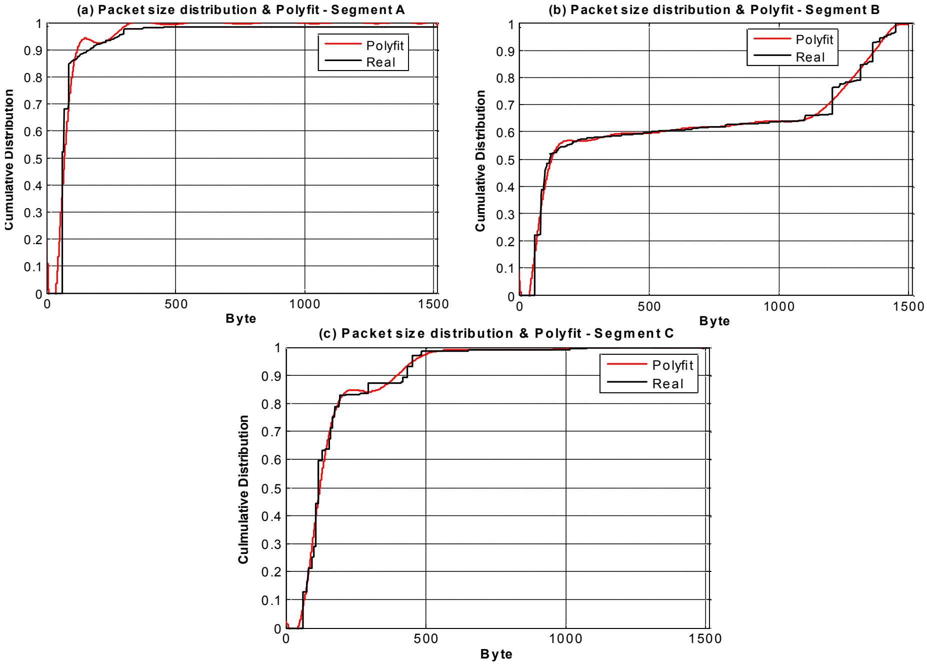


Figure 7. Packet size distribution – a) segment A; b) segment B; c) segment C

3.3.5 Session Length Distribution

There are two concrete session length patterns: short session length at the backbone (PC) and long session length near the end host (PLC). Figure 8 (a) illustrates the session length distribution at Segment B. More than 90% of the total flows terminates in less than 10 seconds. Note that the active timeout value for flow export is set to 300 seconds which was determined by trial and error. This implies that the number of short periodical sessions outnumbers that of the long continuous sessions. The communications with the centralized PC servers are periodical, so that the flows from the identical hosts appear to be separate. The communication period is relatively long, at least more than 300 seconds, for some PLC devices. Continuous PLC communications exist at the edge of the network and are a mixture of PC-PLC and PLC-PLC. The PLC-PLC sessions often stay within the local segment. Figure 8 (b) indicates that the session tends to stay connected during the entire monitoring period, about 250 seconds. The session length distribution from another edge, Segment C, shows a relatively longer session length than those in the backbone. However, it shows slightly uniform distribution of session length since the transmission intervals may vary due to different type of process.

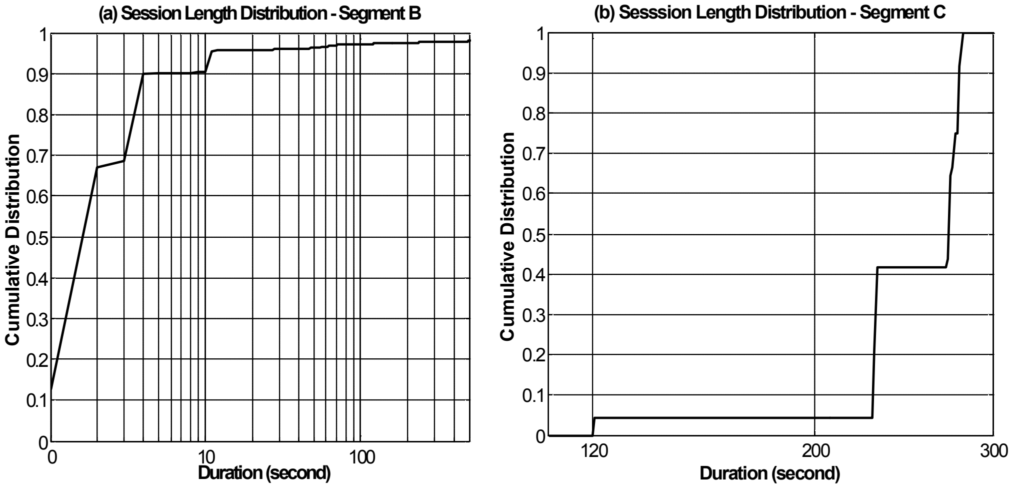


Figure 8. Session length distribution - segment B & C

It is generally believed that the longer and continuous sessions occupy the ICN because its operations often involve continuous and repetitive tasks over long hours (e.g., production lines built on conveyor belts.) Overall, it follows a typical action-trigger behavior; we observe lighter commands (e.g., small packets) from the top of the hierarchy and corresponding actions at the bottom which trigger a complex sequence of communications.

3.3.6 Packet Reordering

We measure the following two categories to detect any packet reordering in a session: out-of-sequence packets and retransmission packets. These refers to any out-of-order packet delivery. Figure 9 illustrates the ratio of the flows experiencing the out-of-sequence and retransmission packets. In the Segment B traces, we observe the total 21,539 TCP flows, and 92% of them experience one or more retransmission packets while online. In comparison to the rest of the traces, the backbone traces have show signs of improper packet delivery. Despite the high retransmission ratio, the process networks operated without problem during the time of data collection. For future work, it is worthwhile to investigate whether such a characteristic is bound to this particular case.

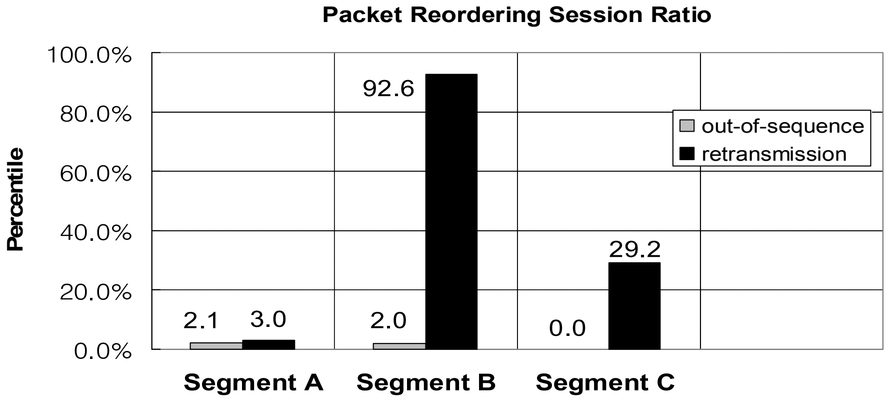


Figure 9. TCP sessions experiencing out-of-sequence and retransmission

Despite widespread deployment in the real world, the area of monitoring and analyzing process control IP networks has not yet received very much from the network measurement and management research community. Therefore, we know little about the traffic behaviors of process control IP networks. These networks are deployed in mission critical operations which entail a maximum level of network stability. Understanding the traffic behavior helps us to operate fault-tolerant control IP networks where the cost of network malfunctioning is far more severe within standard IP data networks.

In summary, we have presented a measurement study of the traffic traces from the real industrial process control IP networks. The traces for analysis are carefully selected to provide a precise snapshot of the network from different perspectives, the traces from the backbone, the edge network, and the end host. We have summarized the following unique characteristics.

- Low-yielding and steady bandwidth usage regardless of monitoring periods
- Periodic traffic cycle in terms of packet arrival sequence and inter-arrival time
- Traffic symmetry
- Occurrence of small signaling packets
- Session length distribution patterns
- High packet reordering ratios at the backbone

Based on the preliminary analysis of traffic characteristics, we plan to develop a control IP network traffic model which can be used for network planning and anomaly detection. The comparison study between the process control networks and the ordinary IP networks will be also helpful to identify and formulate the systematical differences of process control networks.

4 Diagnostics

Network diagnostics do not fully reflect the quality of ICN systems. However, much of the focus in related works was on the fault cases below the transport-layer characteristics in the OSI reference model. Diagnostic network operation usually refers to the communication of sensory information as necessary to deduce the health of a system; this is differentiated from network diagnostics [19]. The status of process and device operation in ICN is difficult to recognize from network diagnostics. This chapter presents the real-world ICN fault cases, fault symptoms, and our diagnosis approach.

4.1 Definition of Alarm, Fault, and Failures

Most of the telecommunication industry standards in Section 2.4 infer alarm as a symptom rather than an actual fault. In X.733, the fault initiates error and that error triggers the alarm eventually. It is a sequential one-way triggering mechanism from fault to alarm. In the DMTF CIM event model, event, indication, and alert are treated in the same level which is opposite to the sequential alarm triggering. In 3GPP and IETF, alarm is simply the undesired state in the management process. With these various alarm concepts in our hands, the relation between alarm and fault should be considered prior to handling fault in any kind of network operation.

In this work, we make a clear distinct between alarm, fault, and failure of an ICN according to the seriousness of the abnormality. First, an alarm is a notification of a specific event, which may or may not be a communication fault [79] (i.e., and interruption of communication service). It is invoked by recognizing early symptoms of communication difficulties, which describe the violation of one or more defined requirements. The actual devices may not even experience communication difficulties at all due to the error recovery schemes; thus, there may be no damage caused by the alarm-triggering event. Second, fault refers to the physical and/or algorithmic cause of the communication malfunction or degradation. Not every fault or alarm is critical to the quality and performance of the manufacturing process; hence, filtering of faults and alarms is required. Third, a failure defines a complete stoppage of a step in the manufacturing process, or the entire manufacturing process, caused by one or more faults. Critical faults and alarms are indicative of process failure.

4.2 Fault Classification

We have analyzed the troubleshooting history of ICN fault cases reported by the administrators at POSCO from 2005 to 2007. By analyzing the previous communication failure cases, we categorize the ICN specific failures. Note that these cases are identified intuitively by the network administrators after the communication failures have occurred. The existing IP network diagnosis tools (e.g., Sniffer, Wireshark) do not yet have the capability to understand the failures in process control networks and to provide the causes of such failures. The possible fault cases are as follows:

- **Ethernet duplex mismatch:** In the auto-configuration enabled environment, two end Ethernet devices may disagree about their duplex (half or full) settings after negotiation. Mismatch can increase the frame loss rate and add extra delays due to collision frames. This problem is not much of an issue in IP data networks which consist of high end computers and switching devices. While some embedded interface modules in PLCs and controlled devices are incomplete against the up-to-date Ethernet standard, the duplex negotiation in control networks often assigns an incorrect setting. A manual setting for every link is a time consuming procedure, but has been the only solution available to the administrators for now.

- **PLC programming bugs:** Poor PLC programming causes communication failures due to the lack of understanding and experience in socket programming. There have been a number of reports that might infer PLC programming bugs, such as unexpected packet occurrences (e.g., irregular keep-alive packets), disordered packet sequences, and unusual TCP window sizes. Reprogramming at either or both sides of server and client applications are needed to fix the problem.
- **Device driver bug:** Device drivers in PLCs and controlled devices may face interoperability problems between various devices from different vendors. The robustness of a device is also crucial for continuous operation. For example, if out-of-range signals in the physical link are received, then it may force the shutdown of the devices. It is difficult to manipulate the vendor-specific drivers in the embedded hardware, so that the replacement with a newer part for the device is needed.
- **Link corruptions:** These refer to physical damage to cables, such as cable cut, dust on fiber interface, and more. There are more chances of these incidents because process control networks are typically located in a hostile environment, such as in a factory. Detecting the damage spots is also challenging. We can only speculate the cable damage from measuring a few network metrics: invalid frame size, frame collision, CRC error, inconsistent throughput, etc.
- **Protocol unawareness:** If a particular device encounters unrecognizable protocols, it malfunctions and most likely causes a stoppage. It is important to prevent the unsupported protocol traffic, which is notified in the vendor’s manual, from floating in the control networks. For example, when Simple Network Management Protocol (SNMP) traffic was

received in an old PLC, the system went down and it had to reboot for recovery.

- **Bandwidth shortage:** The bandwidth is occupied by unwanted traffic, such as those caused by Internet worms. For example, an excessive amount of ARP packets, a so called ARP storm, was observed in process control networks. The required bandwidth for PLC communication could not be provided by the network.
- **Electrical noise:** Unstable transmission occurs due to signal interference. Network links, especially coaxial cables, and devices, nearby high voltage machinery, can suffer from noise interference.
- **Power outage:** Power supply of devices is malfunctioning. This problem is related to harsh conditions in process control networks, such as heat, moisture conditions, etc.
- **Misconfiguration:** Traffic is routed in non-optimal paths or even stays in loop due to incorrect routing table entries. This results in packet delays and loss.
- **Damages to router/switch interface:** This is a typical hardware failure. It is somewhat difficult to detect because the whole connection to a certain area of network can be lost simultaneously.

We have categorized the ICN failure types as communication errors, network misconfiguration, physical defects, and software defects. Although all these failure types can also occur in ordinary IP networks, particularly the last type (i.e., software defects), is more common in ICN. In fact, failures from the last type are due to the fundamental differences of network components (e.g., PLC vs. general computer).

Table V. Classification of ICN failure cases

Failure Type	Failure Causes
IP connectivity errors	Unstable transmission, low throughput, delay, network security threat, IP resource management
Network misconfiguration	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

Table VI presents the list of causes and related network phenomena followed by the faults. The severity of the failures are undisclosed. There is no one-to-one mapping of relations between the fault case and phenomenon. For example, the sudden increase in frame collisions has been observed in the case of Ethernet duplex mismatch, PLC programming bugs, device driver bugs, and etc. Frame collision cannot directly identify any one of these causes. It simply raises a wakeup call for the engineers to be aware

Table VI. List of fault causes and corresponding phenomenon

Fault Causes	Phenomenon
Ethernet duplex mismatch	• Frame collision • Unauthorized IP access • Irregular communication termination • Communication delay and loss • Unexpected shutdown of device • Unordered packet sequence arrival
PLC programming bugs	
Device driver bugs	
Link corruption	
Damage to Interface	
Harsh condition to H/W (Dust in optical device, current short	

due to metal particles, high/low temperature, moisture)	• Unordered message sequence arrival • Message corruption • Broadcasting packet flooding • Low throughput • Duplicate packet arrival
Duplicate H/W or IP addresses	
Protocol unawareness	
Bandwidth shortage	
Electrical noise	
Power outage	
Mis-configuration	
Virus, Worms	

4.3 Early Symptoms for Fault

Flow is an aggregation of the sequence of packets sharing the same meta-data information between two devices in the ICN systems. It is the basic monitoring unit for Ethernet/IP traffic. The previous analysis categories [29] in IP data networks are meaningless in ICN simply because the traffic behavior is not as interesting as that in the Internet. ICN link utilization is usually low as observed in Section 3.3 and consists of fixed traffic sources where it simplifies the traffic characteristics. It is noticeable that the distribution of packet inter-arrival times and of packet size variability follows multimodal distribution [12]. The interrupted Poisson process describes on/off behavior of ICN system devices which are easily patternized in cyclic and event triggering traffic occurrences. In fact, such simplicity of the ICN traffic is advantageous in detecting abnormality compared to sophisticated Internet traffic.

Figure 10 presents the abnormality phenomenon in two different sets of PC/HMI to PLC communications. Graph (a) and (b) specify the packet exchange patterns between HMI and PLC; while graph (c) and (d) are between PC and PLC. The upper plane (above 0 on y-axis) of the graphs indicates a unidirectional packet arrival sequence with respect to its arrival timestamp. The lower plane indicates the arrival sequence in reverse direction. Thus, both planes represent the bidirectional flow. HMI shows a graphical representation of PLC status and process in progress; thus, fast update message delivery to HMI is critical. The small gaps in graph (a) and (b) infer the unknown cause of communication stoppage for the duration of 10 seconds. We observe that neither side initiates any packet simultaneously. It is not clear if such a phenomenon is triggered by a network or device fault. However, one side simply just does not respond to the other side in graph (c) and (d). In this case, the PLC device (or low-end control device) did not respond for some reason [30]. The severity of these two discontinuity cases was actually low; no serious malfunction process was witnessed at the time of measurement. The traffic generation also went back to normal after minor delay.

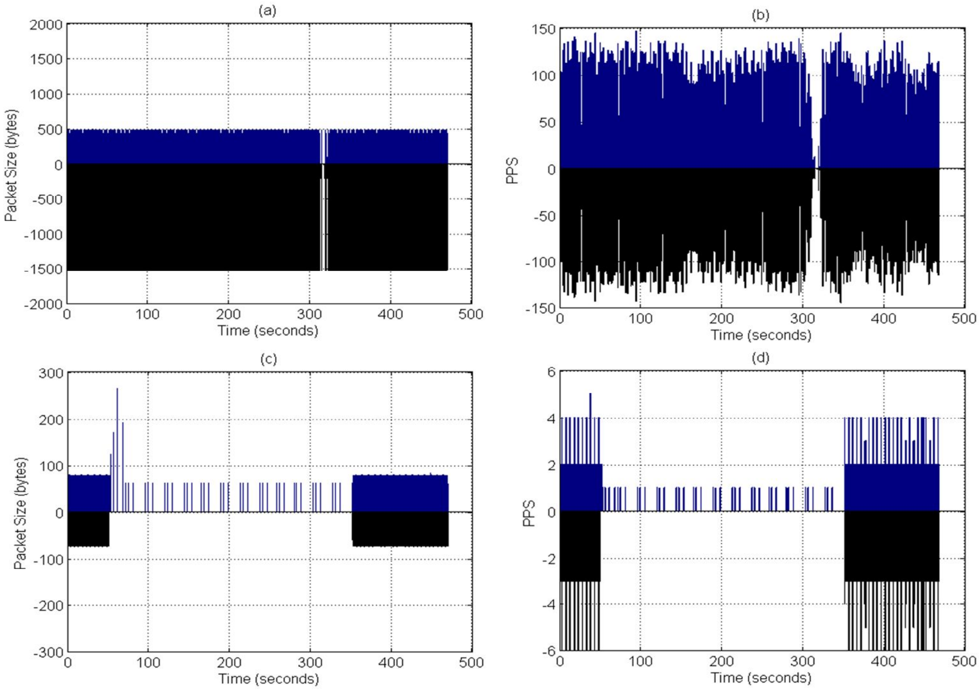


Figure 10. Traffic exchange patterns in packet size arrivals and packets per second (PPS) in bidirectional flows. HMI to PLC - (a) packet size distribution, (b) PPS; PC to PLC - (c) packet size distribution, (d) PPS

Based on the real-world phenomenon of faults, we have identified measurement categories to describe early symptoms that may lead to failure. Table VII illustrates ICN specific QoS metrics to be monitored and their corresponding OSI layers. The violation of metrics including those from the physical to transport layers can be determined against user defined threshold values [9]. Measuring application-layer QoS metrics needs a more sophisticated approach than packet header analysis. More details will be given in the next subsection.

Table VII. ICN QoS metrics and corresponding OSI layers

ICN QoS Metrics	OSI Layer
Collision frame	Physical
CRC error frame	
Dropped frame	
Jumbo frame occurrence	Data link
Runts frame occurrence	
IP checksum error	Network
Fragment packet	
Packet overflooding	
Packet inter-arrival time variation	
Packets per second variation	
Packet size variation	
Throughput variation	
TCP/UDP checksum error	Transport
TCP window size drop	
TCP packet sequence violation	
TCP retransmission packet occurrence	
Unsupported protocol packet occurrence	Transport or Application
Rule sequence violation	

Rule inter-arrival cycle violation	Application
Policy cycle violation	

It is important for network administrators to recognize early symptoms of process network communication failures. We need to revisit some IP network metrics that have been overlooked for some time because we rarely notice them any more in today's IP data communication networks. Thus, we have selected several IP network oriented metrics and identified the alarm conditions (thresholds) accordingly that best reflect any irregularity of communication in process control networks (see Table VIII). The metrics themselves are not very unique, but they have not been properly analyzed in many IP network diagnosis tools. A new set of monitoring categories and conditions is necessary because even a popular tool, like Sniffer, cannot fully detect the control network specific failures but generate false network alerts. The selected metrics can be measured using passive monitoring techniques which do not interfere with network operations.

Table VIII. ICN specific measurement metrics and alarm conditions

Index	Network Metrics	Alarm Conditions
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

The existing IP network diagnosis tools (e.g., Sniffer, Wireshark) 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 VIII. 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.

4.4 Policy-based Fault Detection

This section explains our policy-based detection method and the required technique for policy matching. We need to consider areas beyond the transport-layer faults because the QoS metrics up to the transport-layer (Table VII) do not directly correlate to process or device diagnostics.

4.4.1 Deep Packet Inspection: Signature

A signature is a portion of payload data that is static and distinguishable for applications, which can be described as a sequence of strings or hex values. Such an approach is not new to the Internet worm research community (e.g., intrusion detection systems); however, their focus is limited to identifying the security threatening traffic only. Meanwhile, we focus on the application identification among innocuous traffic which widens the target traffic to be identified. Signatures have been manually extracted by the network administrators or security experts. This requires preceding protocol semantic analysis or empirical packet payload inspection for pattern recognition. Human decision in such a process causes slow response times in dealing with new applications.

A signature is a portion of payload data that is static and distinguishable for applications, which can be described as a sequence of strings or hex values. Signature-based traffic identification has been widely adapted for intrusion detection systems to early detect and block worms [31][32][33][34]. These studies describe the development of automatic worm signature generation systems, which generate the signatures of anomaly traffic, such as port scanning and worm infection. It is their first priority to acknowledge even a slight possibility of vulnerability in the worm body. In this domain, a particular single byte or hexa-code representation is sometimes acceptable in order to recognize an incident like buffer overflow. Another difference to our study is that common vulnerability signatures, again like buffer overflow, can be sharable between the worms while the signatures for normal applications must be unique. In fact, their definition of worm signatures is somewhat looser than ours because their goal is to discover as many flows as possible which are related to any kind of worm traffic behavior. Short string or hex sequences may not be specific enough to reduce the false positives for innocuous traffic identification.

In general, worm signature generation studies rely on a few variations of the sliding-window algorithm, each variation with its own particular break point. Scheirer et al. [34] explained the types of available sliding-window algorithms and the selection of break points, which is the hex values of the instruction codes corresponding to specific actions within the repertoire of the worm's common behavior. The algorithms are called, Fixed Partition Sliding Window Scheme (FPSW), Variable-length Partition Sliding Window Scheme (VPSW), and Variable-length Partition with multiple breakmarks (VPMB). The window slides across the multiple byte streams (or packet payloads) until it finds matching sequences. The problem occurs when applying these algorithms to normal traffic; there are no such break points in the payload due to the differences in traffic nature. We cannot determine where to stop, and we are unable to ascertain an appropriate comparison window size to start with. Therefore, it is difficult to apply the sliding window algorithm using break points to general Internet applications such as P2P. This fact distinguishes our work from previous worm signature generation research.

In some limited instances, the signature can be extracted from the encrypted traffic. Ehlert et al [35] detected the hexadecimal patterns in the Skype (v.1.5, v.2.0) packet traces during the initial communication setup phase. Studies by Bernaille et al. [36] indicated early signature signs, carried over an encrypted SSL connection, for mixed application traffic. Thus, finding a pattern in a packet's payload is a valid approach even for encrypted traffic especially during the early stage of connection.

4.4.2 Approach & Detection Rule

We need to consider areas above the transport-layer faults because the QoS metrics up to the transport-layer (Table VII) do not directly correlate to process or device diagnostics. The network performance QoS violations are fundamentally dependent on the irregular pattern in the multi-modal distribution of ICN traffic.

Process malfunctioning caused by factors other than the network instability cannot be recognized without application-layer analysis.

Packet-level abnormalities, such as packet loss, delay, corruption, and reordering, are compromised by the error recovery scheme in each OSI layer, which are designed for generic network communication functionality, as opposed to devices using the network (which is the case in ICNs). It is important to understand that such abnormalities may influence the process of messages in and out of the PLC devices. For example, the packet loss measurement by random sampling does not accurately reflect the loss of a critical message. The corresponding message could have been delivered correctly by the retransmission scheme at the transport layer. The possible mapping categories for application-layer QoS violation cases are as follows:

1) *Packet reordering due to PLC message sequence violation*

2) *Packet delay/loss due to PLC cycle (timing) violation*

3) *Packet delay due to message delay*

4) *Packet loss due to message loss*

5) *Malformed or error packet due to message corruption*

We propose a policy-based fault detection technique along with a signature matching technique to solve the above problems. The signature matching technique requires deep packet inspection (DPI), which refers to the examination of a packet's payload data for a fixed pattern of information. We defined a policy-based approach to monitor application-layer message patterns. Each policy is of the generic event-condition-action form, where an event triggers the evaluation of a condition. If the condition evaluates to true, then a set of actions are applied. Table IX illustrates the attributes of policy and their descriptions. Events indicate a message of interest; conditions allow us to specify the arrival and timing sequence of certain packets that carry important PLC messages; actions define how to correct problems, and may call additional policy rules to implement more complex behavior. Hence, any loss, delay, and unordered message arrival causing possible failures can be detected. A similar rule-based monitoring technique has been used in Intrusion Protection Systems (e.g., Snort [39], Bro [40]) detecting worm/virus packets. Figure 11 illustrates a ‘PLC_reply_error’ policy example with two consecutive rules where two packets containing ‘0x6000’ and ‘0xe000’ signatures are being exchanged. Packets falling into these two rules must appear in order within 500 ms apart. The total policy cycle should not exceed 2000 ms in order to avoid the fault.

Table IX. Policy details

Attributes	Description	Type
Policy name	Fault Case	String
Policy cycle	Maximum PLC cycle time (ms) for message exchange. It must not exceed the total sum of next sequence arrival time in each rule.	Integer
Source/Destination MAC address	Ethernet hardware addresses (hex) of two communication devices	String
Source/Destination IP address	IP addresses of two communication devices	String
Source/Destination Port	Port number	Integer
Signature	A pattern of hexadecimal digits or specific strings that are preset in the packet’s payload. It refers to a portion of PLC message or protocol (e.g., 0x6000 at offset 0)	Hex or String
Offset	Position of signature in packet’s payload	Integer
Word size	Byte size of signature	Integer

Rule sequence index	Arrival order among the rules	Integer
Rule inter-arrival time	Expected inter-arrival time till the same rule occurs again	Integer
Rule next sequence arrival cycle	Expected inter-arrival time between the current rule and upcoming rule	Integer

```
<policy-table>
   <policy name="PLC_reply_error">
           <policy-list cycletime="2000">
           <rule
                      src_mac="08007023420e" dst_mac="aa0004003250"
                      src_ip="130.30.141.53" dst_ip="130.30.10.41"
                      src_port="1026" dst_port="8453"
                      signature="6000" offset="0" word_size="2"
                      rule_sequence="1"
                      rule_intertime="1500"
                      rule_nexttime="500" />
           <rule
                      src_mac="08007023420e" dst_mac="aa0004003250"
                      src_ip="130.30.141.53" dst_ip="130.30.10.41"
                      src_port="1026" dst_port="8453"
                      signature="e000" offset="0" word_size="2"
                      rule_sequence="2"
                      rule_intertime="1000"
                      rule_nexttime="500" />
           </policy-list>
</policy-table>
```

Figure 11. Policy example using XML expression

Algorithm 1. Policy violation check using signature matching

```
procedure Policy_Violation (Packet$_A$)
    Flow_Pool {$F_1$…$F_x$}
    Policy_Pool {$P_1$…$P_y$}
    $F_1$ ← Iterate, matching Flow for Packet$_A$
    $P_1$ ← Iterate, matching Policy for $F_1$
```

```
i = 0
while j from 0 to |$P_1$| and i++ do      // # of rules ($R_j$) in $P_1$
    if KR ($R_j$→signature, |$R_j$->signature|, $Packet_A$,$R_j$->offset)    == TRUE, then
        if $F_1$ != FIRST, then
            if i != 1, then break, end if
            $F_1$→rule_sequence = FIRST
            $F_1$→next_rule_arrival_time =$R_j$→ next_rule_arrival_time
      else
            if $F_1$→rule_sequence >= j, then
                if $F_1$→rule_sequence == |$P_1$|, then
                    if ($F_1$→rule_arrival_time – $Packet_A$→arrival_time) > 0, then
                         alert $F_1$ rule inter-arrival violation alarm, end if
                    if ($Packet_A$ →arrival_time – $P_1$->cycle_time) > 0, then
                         alert $F_1$ policy cycle violation alarm, end if
                    $F_1$→rule_sequence = FIRST
                    $F_1$->next_rule_arrival_time = $R_j$→next_rule_arrival_time
                end if
            else
                alert $F_1$ rule sequence violation alarm, end if
     end if
     if $F_1$→rule_sequence+1 == j, then
          alert $F_1$ rule inter-arrival violation alarm, end if
     end if, end while

procedure KR (char *x, int x_size, char *y, int y_size,     int offset)
  int d, hx, hy, i, j;
  while i, d from 1 to i<x_size and i++ do
       d = d<<1, end while //left shift
  while i, hx, hy from 0 to i<x_size and i++ do
       hx = (hx<<1) + x[i]
```

```
35:         hy = (hy<<1) + y[i], end while
36:     //search for signature
37:     while j from 0 to j <= y_size - x_size do
38:         if (hx == hy) and (*memcmp (x, y+j, x_size) == 0), then
38:         if offset < 0, then return TRUE, end if
39:         if j <= offset, then return TRUE, end if
40:         hy = *REHASH (y[j], y[j+x_size], hy), end while
41:     return FALSE
*REHASH(a, b, h) ((((h) - (a)*d) << 1) + (b))
*memcmp: memory_compare()
```

Algorithm 1 presents a pseudo code representation of policy violation checks for message cycle timing (line 17, 19, 27) and message exchange behavior (line 24). Every incoming packet will iterate to find its flow entry (Fx) and corresponding set of policy rules (Py). If there is a matching policy, the packet will be inspected for the signatures in the rules (Rj) of P1. The Karp-Rabin algorithm [41] is used for string matching between the payload and signature (line 8). Its expected runtime is linear, O (x_size+y_size) which are the byte lengths of signature and payload being compared, respectively.

4.4.3 Modeling ICN Application-layer Fault

In order to model the ICN application-layer faults, we adapt packet or token-based network models that are designed to represent network stability in contention-based transmission schemes. Network-induced delay and packet drop models are constructed to determine the stable state of control networks. Here, we map the packet or token to the actual message model, so that we can discard any noise from the retransmission and other recovery schemes. The state changes regarding the transmission sequence between the control entity and lower-end devices (e.g. actuators, sensors) can be described as the following:

$$kh, (k+1)h, (k+2)h, \ldots, (k+n)h \quad (1)$$

where k is a sample transmission sequence, which infers the message delivery sequence between the control entity and devices. The corresponding models by [23] refer to the packet sequence, where the contention-based transmission is applied to all layers in ICN and its systems. This may not be true in the scope of this thesis, since our monitoring domain imposes bus sharing. h is a monitoring sample period in seconds.

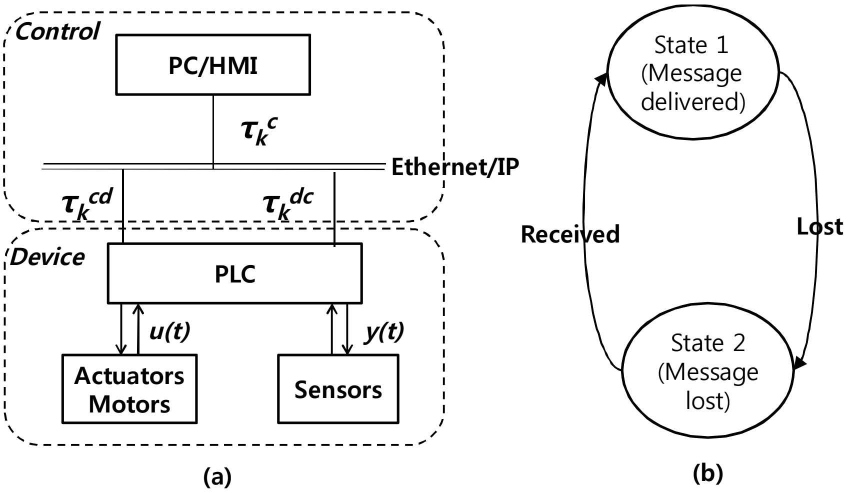


Figure 12. (a) ICN message delay model; (b) ICN message dropout state model

$$\tau = \tau_k^{cd} + \tau_k^{dc} + \tau^{c} \quad (2)$$

The ICN fault model incorporates continuous and discrete dynamics; signal (or message) transfer and state of ICN operational status, respectively. Figure 12 (a) illustrates the message delay model of ICN. There are three types of delays: Network delay from control to device (τ_k^{cd}), network delay from device to control (τ_k^{dc}), and processing delay at PC/HMI (τ^c). The total delay (τ) is the sum of these three delays as in (2) and must stay within a tolerable transmission interval. Any violation of this tolerable boundary is indicated by cycle violation categories in the policy. The following model is adapted from [77] to represent the theoretical range of delay in an ICN:

$$\max\left\{\frac{1}{2}h - \frac{1}{K}, 0\right\} < \tau < \min\left\{\frac{1}{K}, h\right\} \tag{3}$$

or

$$\max\left\{\frac{1}{2}h - \frac{1}{Kh}, 0\right\} < \frac{\tau}{h} < \min\left\{\frac{1}{Kh}, 1\right\} \tag{4}$$

h refers to the sampling rate and K is a multidimensional state matrix for ICN. Formulas (3) and (4) show the relationship between the monitor sampling rate and network delay. For example, in our case, we set h as 10 µs since it is the minimum time granularity for a packet timestamp. K is a 1x1 matrix representing the malfunctioning ratio of a single ICN fault, which implies either a stable or unstable state of the ICN (e.g., K=[0.4]).

In a similar way, the ICN message drop can be modeled using a packet-drop model [77] as follows:

$$\text{State 1: } \bar{x}(kh) = x(kh) \tag{5}$$

$$\text{State 2: } \bar{x}(kh) = \bar{x}((k-1)h) \tag{6}$$

For a certain rate of message loss in a continuous time frame, the state transition (or switch) can happen as in Figure 12 (b). We further investigate on such rate patterns by using a statistical prediction approach in Section 6.

5 Architecture of Fault Diagnosis System

In this chapter, we describe the design and implementation details of the ICN fault diagnosis system using our proposed methods.

5.1 Requirements

The following are the requirements we have considered in designing our diagnosis system.

- **Distributed monitoring/analysis architecture:** With a single off-the-shelf PC system, it is hard to monitor and analyze all the packets on a remote and multi-gigabit network link. It is required to divide the monitoring task into several functional units and distribute the processing loads. We consider a pipeline method to distribute either the collected traces or analysis functionalities across multiple monitoring entities (e.g., server/probe). In addition, each entity should provide remote accessibility for the network administrators.
- **Lossless packet capture:** It is preferable to capture and store all the packets on the link without any loss in order to provide accurate network statistics and data sets for later fault analysis. It is not an absolute requirement for the ICN fault diagnosis system; however, the fault accuracy will be heavily affected by any loss of information.
- **Flow-based analysis:** As in Section 2.3, the aggregation of packet information into flows is required for efficient analysis and data storage.
- **Minimizing management overhead in the ICN:** The management traffic between the monitoring entities should not interfere with or overwhelm the ICN links. For example, it is not preferable to exchange a

large volume of traffic, such as collected packet traces, between the server and probe.

- **Combination of on-line/off-line analysis:** Our policy-based method relies on the DPI technique which may cause high system overhead. Instead of processing everything in real-time, we consider analyzing certain diagnosis categories with the collected and stored traces only.
- **Adaptive configuration:** The network administrators should be able to modify/insert/remove any configurable information, such as threshold and policy, without system stoppage. It should be an easily extensible architecture for additional policies and fault-related metrics
- **Support for various applications:** It should be flexible enough to provide data to various applications in diverse forms. When a new application, other than fault diagnosis, needs to use the monitoring system, it should be able to easily support that application without changing the structure of the system.

5.2 System Design

We implemented the diagnosis analysis system using C language, libpcap library, and xml libraries in a Linux environment. We designed the packet capture system with multi-threaded architecture to handle multiple NICs in a single capture system. Table X presents the required software packages and implementation environment.

Table X. Implementation environment

Categories	Details
OS	Red Hat Enterprise Edition 4/5, Cent OS 5
Environment	Kernel 2.6.x, gcc-3.4.x

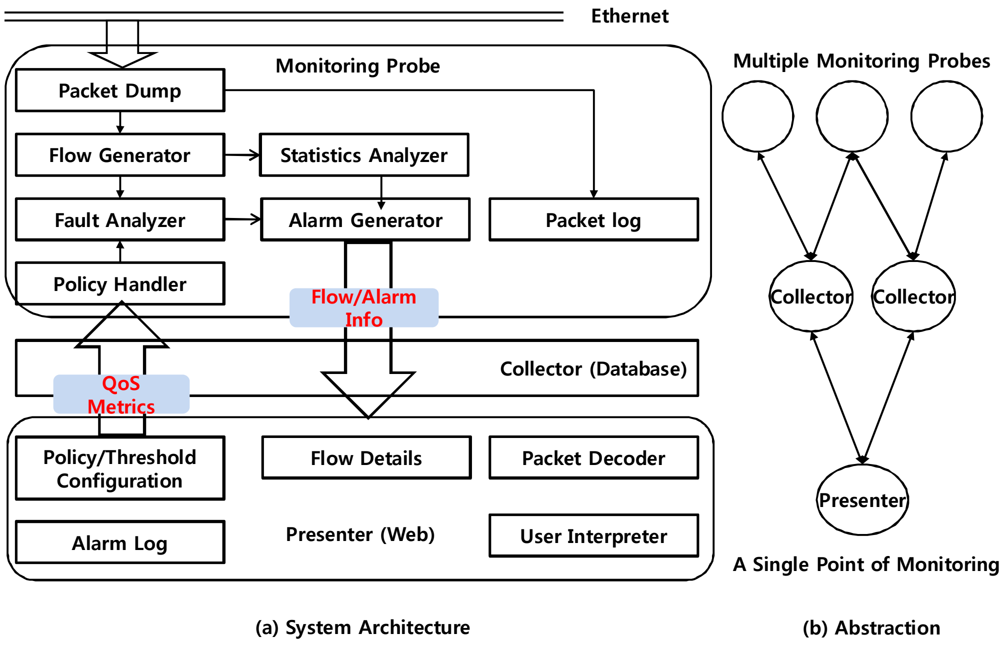
Software packages	httpd-2.2.2, php-5.2.5, mysql-5.0.27, jpgraph-1.13, mrtg-2.15.0, net-snmp-5.1.2, net-snmp-utils-5.1.2, Wireshark

5.2.1 RTFM/IPFIX Architecture

The Real-time Traffic Measurement (RTFM) architecture [73] was proposed by the IETF working group and it is a logical monitoring structure for different types of networks. It is composed of four entities that communicate by means of a suitable protocol which is not explicitly specified in the definition: Meter, Meter Reader, Manager, and Analysis Application.

- The meter is the component which has the task of accounting packets according to attributes such as their source and destination addresses. For traffic measurement, the meter follows a set of rules which specify the attributes of the traffic flows to be observed; a packet is counted if all its attribute values match.
- The manager is an application that configures and controls the activity of one or more meters and meter readers; it works according to the requirements of the applications that make use of the accounting data.
- The meter reader reliably collects and transfers the registered data between the other three applications.
- The analysis application is an application that handles accounting data for such purposes as the determination or management of network performances.

NetraMet [74] is an open-source implementation of the RTFM architecture for network traffic flow measurement. In the reference implementation of RTFM architecture, NeTraMet is the meter and NeMaC is both the manager and meter reader; these two elements exchange information through the SNMP protocol. A Rule Set, created by the user, contains the specifications of the tests on packet attributes, in particular the IP address and TCP/UDP port number of the source and destination. However, the initial version of NetraMet collects raw packets from network links without consideration of link speed. Many traffic monitoring systems influenced by the RTFM architecture need to be made compatible with the high-speed network links. In addition, the architecture for monitoring network traffic, such as IPFIX [75], recently has reached the RFC status as in 2009.



(a) System Architecture (b) Abstraction

Figure 13. ICN diagnosis system architecture

5.2.2 Architecture & Implementation

Figure 13 (a) illustrates the design of an ICN diagnosis system considering the distributed monitoring environment. Multipoint monitoring of ICN traffic is enabled with the deployment of one or more monitoring probe(s), a collector (database), and a Web-based presenter. Packet traces are forwarded to the probe via port mirroring in router/switch or passive signal tapping for fault analysis. The proposed system supports network diagnosis capability and remote accessibility. As illustrated in Figure 13 (b), its distributed architecture allows us to design a flexible monitoring system where multiple links can be monitored with a single point of data representation.

The monitoring probe is attached to the target link and is controlled remotely from the Web server running in the presenter. The Packet Collector captures packets from the interface and passes them on to the Flow Generator and Packet Log. The Fault Analyzer records the list of IP network oriented metrics mentioned in Section 4.3 and runs the validation check against the predefined thresholds by the network administrators. Note that the methods for choosing and evaluating the appropriate threshold values are presented in the upcoming section. If any of the metrics violate the defined alarm conditions, the alarm generator sends an email or SMS message. The metrics should be monitored for each flow. Flow here 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.

Binary packet log files are stored for post in-depth analysis if necessary. In fact, the I/O overhead for storing full packet traces is acceptable in a process control network environment because of the very low bandwidth, as described in Section 3.3. At last, the network administrator can attach the explanation to the alarm log. It is more of an intuitive opinion regarding the cause analysis. This information is later referenced to determine a mapping relationship between network metrics and previously known fault cases.

Flow and alarm information between the probe and presenter bridges over the database. Flow here 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. The proposed system is adapted from the RTFM architecture and concentrated on on-line analysis which mainly focused on detecting network performance QoS violation and remote accessibility. With the extension of policy handler and its offline analysis capability, the current system architecture is now capable of detecting application-layer faults which are strongly correlated to failures.

The Fault Analyzer comprises of online and offline processes in accordance to the corresponding fault categories in the OSI model (see Figure 14). The online process records the list of up to the transport-layer fault categories in Table VIII and runs a validation check against the predefined thresholds. If any of the metrics violates the defined alarm conditions, the alarm generator notifies the personnel via email or SMS message. A 10-20 second delay to allow for Web presentation is necessary to enable stable detection system performance. However, the application-layer fault analysis requires DPI in our policy-based approach. The online process for DPI is restricted due to system overheads involving overwhelming payload comparisons and system memory copy operations. Thus, we combine the offline process technique with the online system by using the event trigger approach. If a fixed amount of packet trace is collected, then it will notify the packet handler and proceed with signature matching in the application-layer fault detection module.

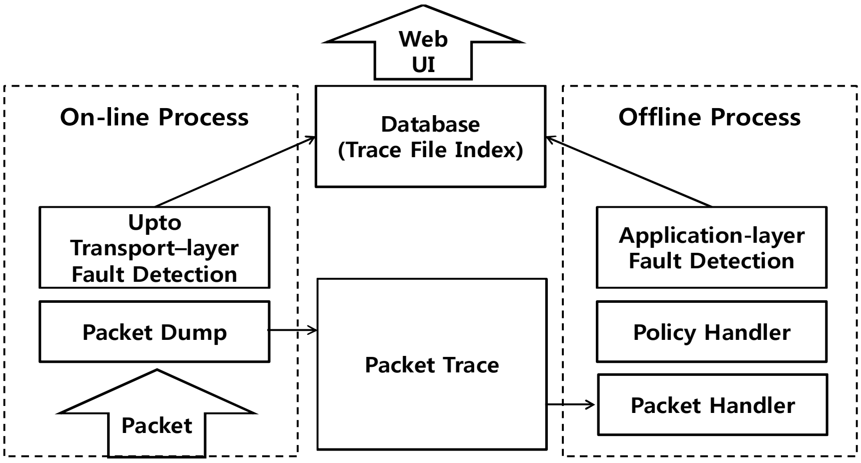
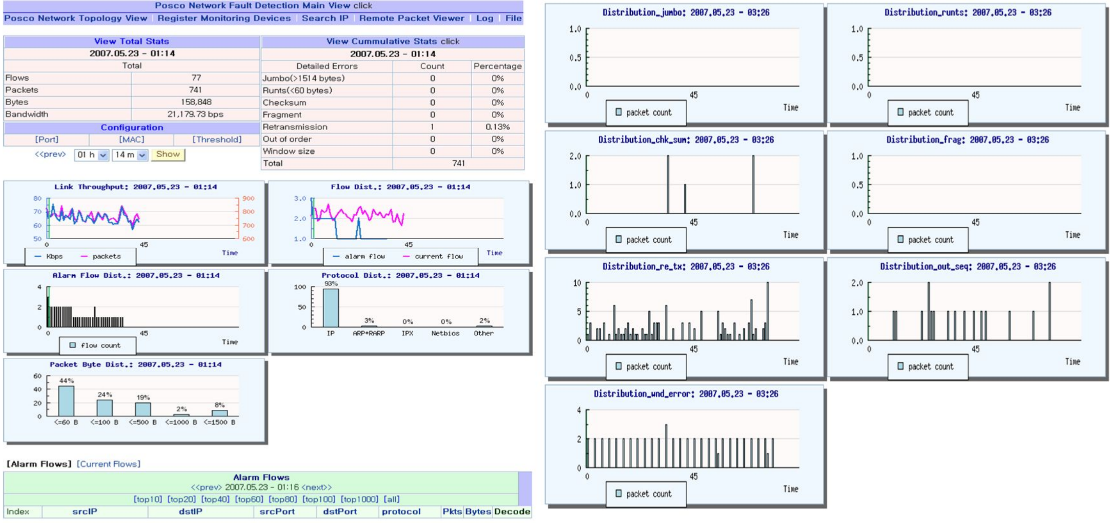


Figure 14. Detailed interaction within fault analyzer; an overview of combined on-line/off-line fault analysis methodology

(a) Main user interface

(b) Alarm occurrence graphs



Posco Network Fault Detection Main View click

Posco Network Topology View | Register Monitoring Devices | Search IP | Remote Packet Viewer | Log | File

View Total Stats	
2007.05.23 - 01:14	
Total	
Flows	77
Packets	741
Bytes	158,848
Bandwidth	21,179.73 bps

Configuration		
[Port]	[MAC]	[Threshold]

<<prev> 01 h 14 m Show

View Cummulative Stats click		
2007.05.23 - 01:14		
Detailed Errors	Count	Percentage
Jumbo(>1514 bytes)	0	0%
Runts(<60 bytes)	0	0%
Checksum	0	0%
Fragment	0	0%
Retransmission	1	0.13%
Out of order	0	0%
Window size	0	0%
Total	741	

[Alarm Flows] [Current Flows]

Alarm Flows
<<prev> 2007.05.23 - 01:16 <next>>
[top10] [top20] [top40] [top60] [top80] [top100] [top1000] [all]

Index	srcIP	dstIP	srcPort	dstPort	protocol	Pkts	Bytes	Decode

Link Throughput: 2007.05.23 - 01:14
80
70
60
50
900
800
700
600
0
45
Time
Kbps
packets
Flow Dist.: 2007.05.23 - 01:14
3.0
2.0
1.0
alarm flow
current flow
Alarm Flow Dist.: 2007.05.23 - 01:14
4
2
0
flow count
Protocol Dist.: 2007.05.23 - 01:14
100
50
93%
3%
0%
0%
2%
IP
ARP+RARP
IPX
Netbios
Other
Packet Byte Dist.: 2007.05.23 - 01:14
60
40
20
44%
24%
19%
2%
8%
<=60 B
<=100 B
<=500 B
<=1000 B
<=1500 B
Distribution_jumbo: 2007.05.23 - 03:26
Distribution_runts: 2007.05.23 - 03:26
Distribution_chk_sum: 2007.05.23 - 03:26
Distribution_frag: 2007.05.23 - 03:26
Distribution_re_tx: 2007.05.23 - 03:26
Distribution_out_seq: 2007.05.23 - 03:26
Distribution_wnd_error: 2007.05.23 - 03:26
1.0
0.5
0.0
2.0
10
5
packet count

Figure 15. Web-based user interface

For now, all the components are implemented in a single machine. Our implemented system satisfies the following functional requirements: traffic summary statistics analysis, packet decoding display, and alarm statistics analysis. Granting remote accessibility via the Web browser facilitates administrator awareness of the network status. Figure 15 (a) illustrates the user interface of the proposed system. The main user interface displays the bandwidth, alarm occurring flows, alarm occurring types, protocol distribution, etc. which reflect the overall status of the monitoring control networks. This information is updated every 20 seconds with a delay of about 40 seconds from the current time. Figure 15 (a) depicts flow records demonstrating the cumulative sum of traffic summary from flow commencement to its termination. The administrators are able to access each flow record even after its termination due to an active flow time-out of 30 seconds. Figure 15 (b) shows the alarm generation trail over the previous 30 minutes period. A single graph represents the occurrence of each alarm type. Note that these alarms are not always related to the actual failure of network; instead, they are early symptoms of possible network failures. Table XI illustrates the functionality comparison between the proposed system and Sniffer.

Table XI. Comparison of the proposed system and Sniffer

	Proposed System	Sniffer
Accessibility	Remote (Web-based)	On-site only
Applicability	Multiple link monitoring	A single link monitoring
Hardware flexibility	A single or multiple hardware platform for distributed architecture	A single hardware platform
Alarm definition	A new category of alarms can be added	Fixed
On-line packet logging	Lossless packet capture	Heavy overhead, occasional packet drops
On-line analysis	Yes	Yes
Packet decoding capability	Yes	Yes

There is hardly a case where a single metric infers the cause of failure. They occur in a chain of reactions; for example, if the error frames or the out-of-order packets arrive, there is a high chance of seeing retransmission trials soon after. Consequently, the whole sequence of transmissions might have been triggered from the cable cut or software bugs. The correlation coefficients are calculated for every combination of the proposed network metrics to determine their closeness in each fault case. If a certain group of metrics with strong association is repeatedly observed for a reasonable number of times, we can speculate the cause (from the list of fault cases) of a similar traffic pattern in future events. Finally, the user interpretation component represents the level of confidence by the network administers in reasoning the fault cause. The network administrator can intuitively rate each alarm for its accuracy.

Each fault case is represented by the group of metrics and their aggregated values from the previous analysis results. This set of data is referred to the knowledge base for cause analysis. Since the alarm condition is also based on the values of the proposed IP metrics, we cross-match a stream of alarms with the available set of failure cases that have previously observed in the network. The result from the cross-match can be reviewed by the network administrator to reduce any false positives in assigning the possible cause. Finally, the identified alarms are fed back to the knowledge base for the case updates to be consistent with the traffic nature.

We have designed the cause analysis procedure for offline use because we needed a sufficient number of network failure reports. The alarms relying on the measurement results cannot suggest appropriate countermeasures to fix the problem, but instead identify the troubled links or devices (e.g., IP address) that are suffering from the communication failures. Fast and accurate cause identification is more important than just alarm generation in the real-world network management and brings us one step closer to achieving fault-handling automation.

6 Evaluation: Prediction and Adaptive Decision

In this chapter, we describe the fault prediction technique using the statistical distribution model. We also present appropriate machine learning approaches to develop the system which is equipped with adaptive fault decision mechanisms. With the real-world deployment experience, the evaluation of these methods is discussed. Finally, the autonomic aspects of the proposed system and methods are presented.

6.1 Statistical Model for Alarm Patterns

Alarm patterns are projected against the real alarm occurrence ratios and fitted to the following distribution model.

6.1.1 Two-parameter Weibull Distribution

We present the alarm pattern and prediction analysis results gathered during the real-world deployments. Our fault detection and diagnosis system was deployed at two process control networks to monitor and detect faults at POSCO since June 2007. Our system also collects and logs traffic traces for the close examination of packets in the event of fault detection. A complete isolation of the PLC network from Internet or any other enterprise networks was guaranteed by assigning private IP addresses and physically detached networks to the PLC segments. We analyzed a week-long trace at one of the edge segments, and a typical working-hour trace at the plant backbone traffic traces – Segment A and B in Figure 2, respectively. Segment A refers to the top of a local PLC network and Segment B is the collection point where the multiple instances of Segment A level traffic are directed. The actual name of each process segment is undisclosed due to security reasons.

The system has reported a few categories of alarms since its deployment. Specifically, we have observed that the following three alarm categories are the most prevalent: window size error, out-of-sequence packet counts, and retransmission packet counts. We use the two-parameter Weibull distribution [42] to determine appropriate threshold values and to analyze the alarm cycle of our system over the continuous monitoring period. It is a probability measure of each alarm type over time based on the empirical data set of the alarm log. The two-parameter Weibull distribution is given by:

$$f(t) = \frac{\beta}{\eta}\left(\frac{t}{\eta}\right)^{\beta-1} e^{-\left(\frac{t}{\eta}\right)^{\beta}} \textit{ where } f(t) > 0,\ t>0,\ \beta>0,\ \eta>0 \tag{7}$$

t = time, η = scale parameter, β = shape parameter (slope)

Alarm ratios can be manipulated by the user specified threshold values; thus, it is important for the proposed system to determine appropriate threshold values while preserving the alarm credibility. Table XII illustrates the Weibull distribution fit values for each alarm case and its choice of threshold values. The fit to the empirical CDF distribution is given by a Weibull CDF distribution and reliability whose equations are:

$$F(t) = 1 - e^{-\left(\frac{t}{\eta}\right)^{\beta}} \tag{8}$$

$$R(t) = 1 - F(t) \tag{9}$$

Table XII. Parameters to fit for each alarm probability

Alarm Category	Threshold value	Segment A		Segment B	
		β	η	β	η
Window size error	1	1.4107	6627.8192	2.1505	372.6533
Out of sequence	10	1.5938	6941.4623	2.9220	313.0396
	50	1.6234	6957.8985	2.9220	313.0396

	100	1.6244	6959.1457	2.9220	313.0396
Retransmission	100	1.1852	6228.7397	2.9836	319.2062
	500	3.1364	276.5284	255.3608	264.4731
	1000	3.1567	277.3797	255.3608	264.4731

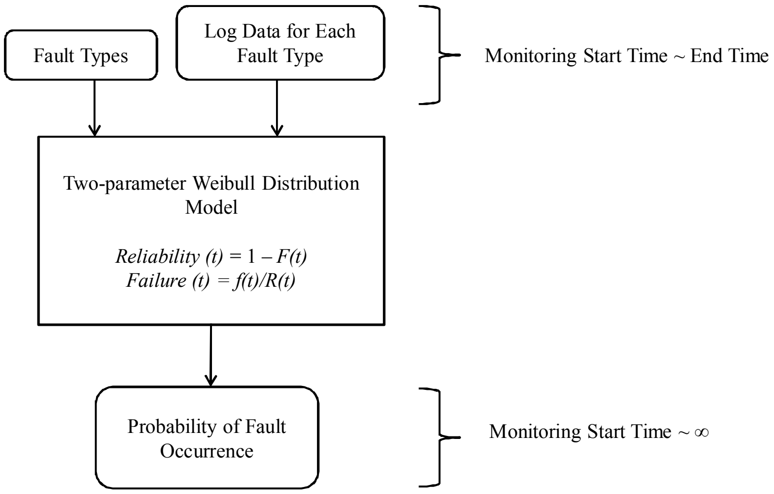

Fault Types
Log Data for Each Fault Type
Monitoring Start Time ~ End Time
Two-parameter Weibull Distribution Model
Reliability (t) = 1 – *F(t)*
Failure (t) = *f(t)/R(t)*
Probability of Fault Occurrence
Monitoring Start Time ~ ∞

Figure 16. Abstract view of the fault prediction model

6.1.2 Validation with Alarm Log

We focus on the Segment A traces for the alarm analysis of the proposed system. A longer trace is preferable to handle the process control network specifics in general. Figure 17 illustrates the probability measure of the window size error alarms. We are able to forecast the expected probability of window size error and provide a unique cycle of alarm. The time granularity of measurement is one minute. The threshold value for window size alarm report is equal to 1 because a single occurrence of such an abnormality has a high potential for causing communication failures.

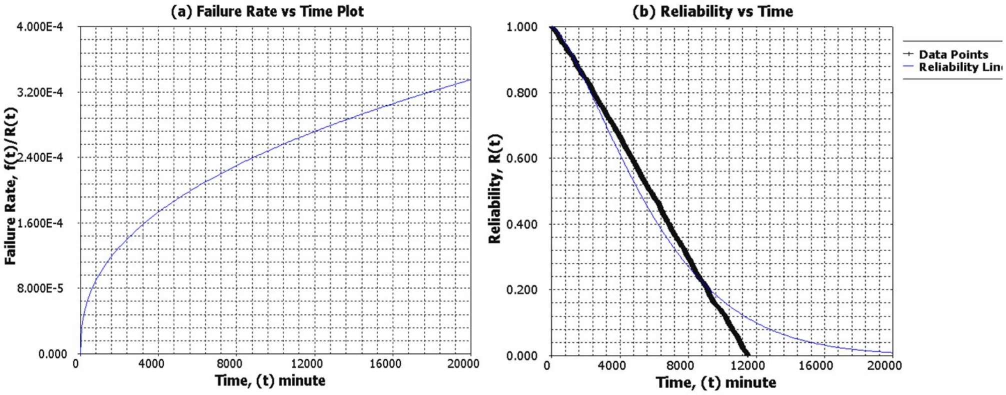


Figure 17. Segment A, window size error: Failure (alarm) probability and reliability fit to the empirical complementary CDF

Figure 18 (a) and (b) illustrate the out-of-sequence alarm probability model for threshold 10 and 50, respectively. Since the probability model for threshold 100 shows an almost perfect fit to (b), we have omitted the corresponding graph. Table III shows that these two sets of parameter values are very close to each other (1.6234→1.6244, 6957.8985→6959.1457). The curve in Figure 18 (a) is slightly steeper than (b), meaning a bit higher alarm ratio. However, the difference here is minimal. We can conclude that the alarm ratio remains steady after the threshold value of 50 is reached.

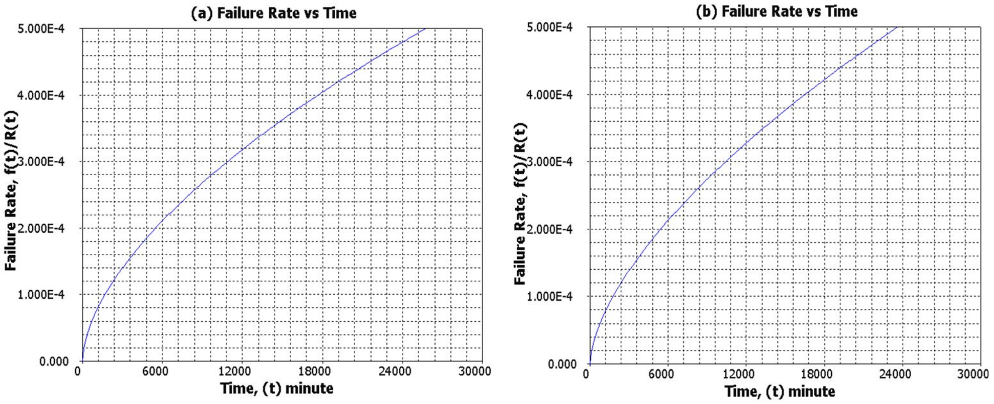


Figure 18. Segment A, out-of-sequence: Failure (alarm) probability for threshold 10 - (a) and 50 - (b)

Figure 19 (a) and (b) illustrate the retransmission alarm probability models for threshold 100 and 500, respectively. The last threshold value of 1000 also shows almost perfect fit to (b). In the case of (b), a significant reduction of alarm generation is observed compared to (a).

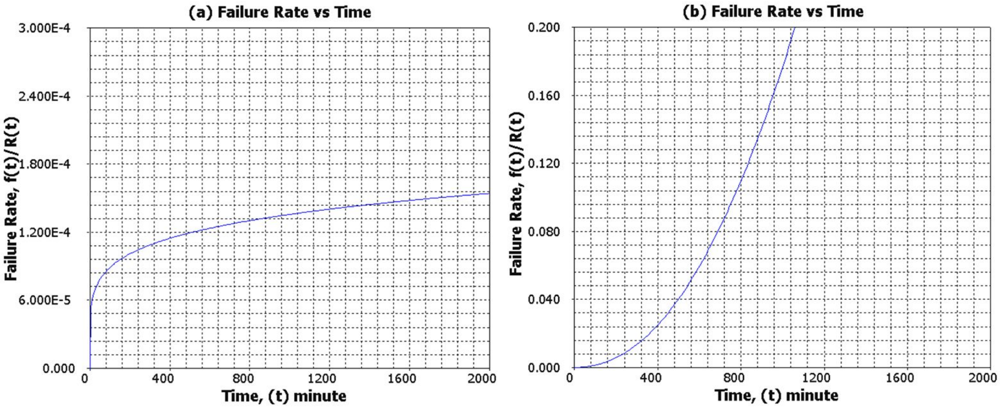


Figure 19. Segment A, retransmission: Failure (alarm) probability for threshold 100 – (a) and 500 – (b)

Based on the analysis results shown above, we have derived a failure probability model which can be bound to the threshold values of 50 and 500, out-of-sequence and retransmission, respectively. These values are set to generate the steady alarm ratios for Segment A. When applying these threshold values for Segment A, the networks will likely experience the alarm occurrence trails in Table XIII, which illustrates the probability measure of alarm reports. The first two categories show that the probability of occurrence of window error and out of sequence alarms increases as the monitoring period advances. Retransmission occurs continuously with 100% probability regardless of time because it was continuously observed throughout the monitoring period.

We have determined appropriate threshold values for alarm generation in this particular instance. In a similar fashion, we can validate the effectiveness of the determined threshold values for different networks and build a trajectory model of alarm generations.

Table XIII. Probability of alarm report

Segment A	Window error	Out of sequence	Retransmission
Minutes	Probability of Alarm (%)	Probability of Alarm (%)	Probability of Alarm (%)
1440 (1 Day)	10.95	7.45	100
2880 (2 Day)	26.54	21.24	100
4320 (3 Day)	42.11	36.95	100
5760 (4 Day)	55.97	52.09	100
7200 (5 Day)	67.49	65.25	100
8640 (6 Day)	76.62	75.85	100
10080 (7 Day)	83.58	83.88	100

6.2 Adaptive Decision using Machine Learning

There are some solid works of using data mining techniques to identify fault in IP data networks [43][44][45]. To the best of our knowledge, there has been no effort to evaluate fault detection using data mining techniques for ICN. We adapt some of these previous techniques to fulfill the ICN requirements.

6.2.1 ML Techniques

Many tests have been undertaken on the various classification and clustering techniques used for fault detection in IP networks, such as Support Vector Machines (SVMs) [46], which separate data into multiple classes (two in the basic case) through the use of a hyper-plane. Mukkamala et al. [45] used a more conventional SVM approach for fault detection. They used five SVMs, one of which was used to identify normal traffic, and the others to identify each of the four types of malicious activity in the KDD Cup [47] dataset. Many researchers have favored the use of SVMs. However, SVMs typically require a training time that is quadratic to the number of training examples, and are thereby inefficient for skewed data sets.

Decision trees are another very popular classification technique. Frank [44] 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 [45] and fuzzy logic [43]. One of the recent works using the clustering technique compares various outlier detection algorithms to identify the faults in the IP network [45]. Most of this research used the DARPA data set [48] of the KDD cup data set for testing their technique. However, although there has been some research done in this area, the traffic characteristics of control networks remain different. There is a definite need to analyze the faults in control networks, and to tune the data mining techniques to fit the control network security systems.

We add the data mining engine which can calculate the correlation between network error symptoms and actual faults in ICN to improve our existing system.

6.2.2 Approach: Decision Tree

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.

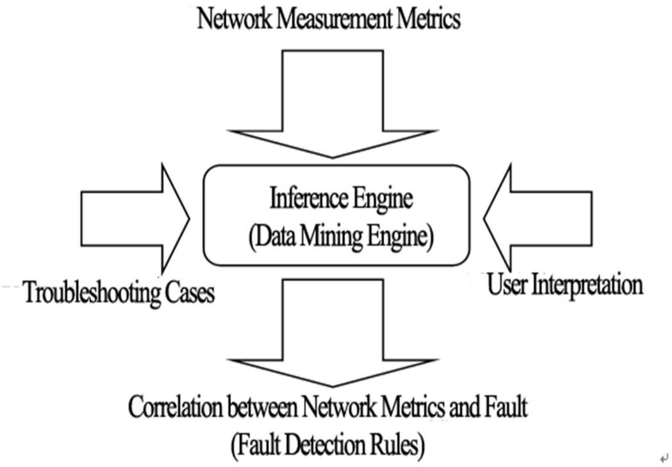


Figure 20. Conceptual view of data mining engine

Data mining technology can be considered as an inference engine for control network fault detection. Figure 20 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.

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 for unbalanced classification problems (i.e., they have been shown to be effective if the abnormal class has few training examples) [49].

Figure 21 illustrates an example of a fault decision tree. In this example, the source IP address and source port number label the nodes, while the fault and normal label the leaves. The labeled arrows are the edges.

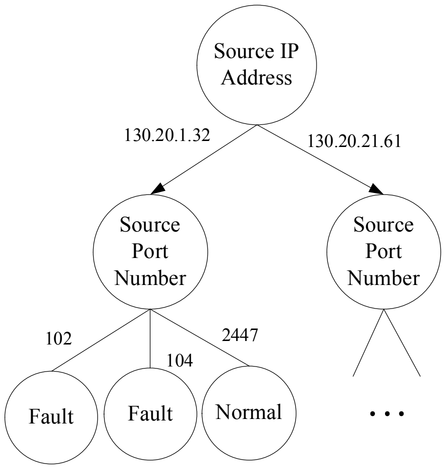


Figure 21. Example of fault decision tree

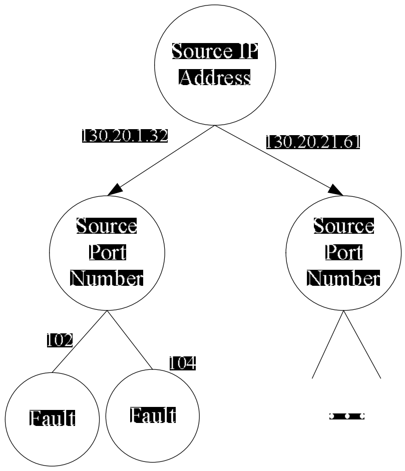


Figure 22. Example of pruned decision tree

We employed the C4.5 algorithm [50] 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 22 shows the post-pruning result of the decision tree portrayed in Figure 21. All connections are assumed to be normal if they are not classified as network faults.

Table XIV. Feature list

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

6.2.3 Validation with Ground Truth

We collected the packet trace from one of the least-congested and most frequent ICN faults in POSCO. The collected packet trace covered a 5-day period with a total traffic volume was about 75 Gbytes. 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. This was undertaken because individual packets cannot describe the communication pattern between network devices. After aggregating the traffic data into the flow, the features were extracted from the flow. Table XIV 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 to the statistical information of the connection logs and are 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 XIV to describe training or testing examples. Table XV 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.

To measure the accuracy of generated decision trees in regards to the network fault classification results, we used three measures that are widely used in data mining to evaluate the accuracy: precision, recall, and F1 metric [51]. 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 is a measure of how many identified or classified objects were correct and it is the ratio of TP to the sum of TP and FP.

Recall is a measure of 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:

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$F1 = \frac{2 \cdot recall \cdot precision}{recall + precision}$$

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

First, we used the standard C4.5 algorithm to create decision trees which classify connections based on the features listed in Table XIV. In this experiment decision trees were pruned during their construction to generalize information.

All three evaluation metrics have exhibited very high accuracy (over 99%) because the communication in control networks is much simpler than that within the 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. Each connection is described by a pair of source IP and destination IP addresses. 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 XV. 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 XVI. 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 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 the number of nodes. As such, pruning affected the accuracy of the classification results.

Multi-stage classification: In the other two experiments, a 2-stage decision tree classification process was used in order to use IP addresses as important features. We constructed decision trees with only IP addresses in the first stage of classification. These decision trees classified the fault based on IP addresses so that we could primarily detect IP related faults. The 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 quantity of FNs and FPs. However, this methodology was also unable to 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. Some fault cases appeared only once in the entire data set and were not properly trained in the context of cross-validation. To detect these faults, 22 decision tree independent rules were needed. Although using extra rules can cause over fitting, it is desirable in control network fault detection due to the characteristics for 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 sufficiently trained.

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.

6.3 Integration to Fault Diagnosis System

This section provides the design of a fault detection and diagnosis system for ICN using classification rules generated by the data mining engine which uses decision tree. Figure 23 illustrates high-level system architecture of the process control network traffic monitoring system.

The Fault Analyzer records the list of IP network oriented metrics listed in Table VIII 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 the user's interpretation about actual network faults. The network administrator can attach the explanation to the alarm log via the User Interpretation interface and provide an intuitive opinion about cause analysis. This information is used to generate a decision tree in order to determine mapping relationships between the network metrics and previously known fault cases. Binary packet log files are stored for further offline analysis if necessary.

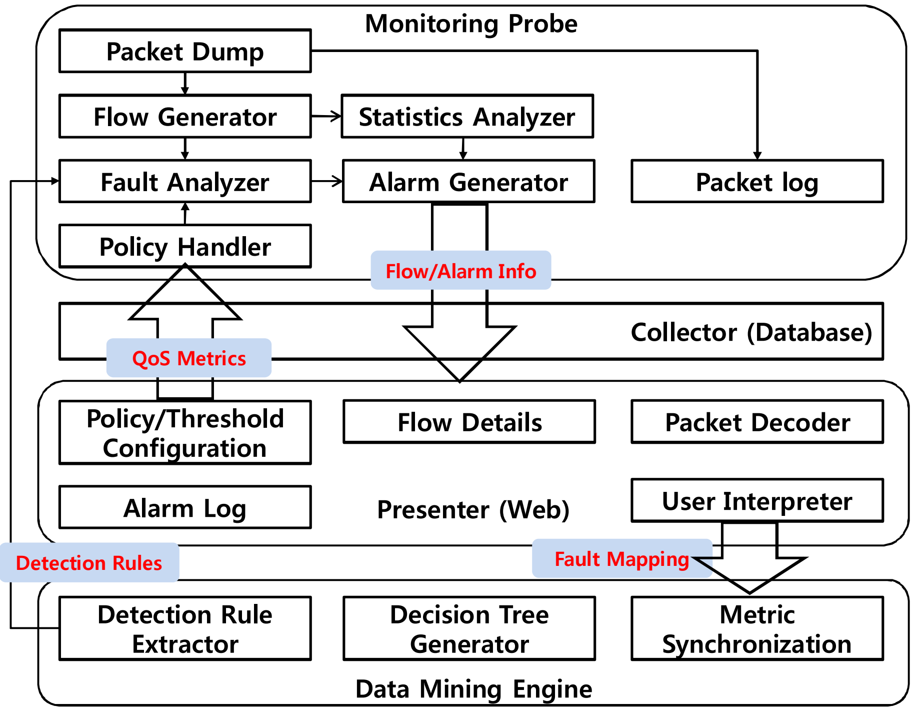


Figure 23. ICN diagnosis system architecture with ML decision engine

Although there have been numerous studies on detecting network anomalies by investigating traffic from the Internet data plane, there has been little research focus on the monitoring and analyzing ICN. The fault detection system presented in this section provides a new systematic approach to detecting control network faults. 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.

6.4 Damage Cost Function

The performance of an ICN is directly related to the net profit of the industrial process that uses it (e.g., Profit loss = Production loss ratio * Trouble hours). Any loss due to ICN faults and their corresponding failures can require damage cost analysis. The damage cost function can be modeled against the fault occurrence ratios and its future prediction model defined in this paper. Unlike the Internet, ICNs allow us to estimate the economical consequences in case of network or system malfunctioning. This is not easy for other data networks (e.g., Internet) because there is too much variance in fault occurrence, and no clear cost estimation for such complex environment exists. The expected damage cost function [78] for ICN is as follows:

$$\begin{aligned} \mathrm{E}[C_d] &= \\ &= \int_0^1 \Big[\int_0^\infty C_d(x) f_{X|L}(x|l)\,dx\Big](P|\mathrm{F}, t_l) f_L(l)\,dl \\ &\approx C_0(x'_{li}) \cdot \int_0^L \frac{1}{L} e^{-t} dt \end{aligned} \tag{10}$$

l is the loss impact ratio of a fault category. It implies the severity ratio (0-1) of faults and how much production loss can be expected, where 1 is a complete process failure (e.g., Cost loss is 100%). $(\mathrm{P}|\mathrm{F}, \mathrm{t_l})$ represents the present impact factor of future costs at the fault occurrence time. We can derive C_0 as the maximum damage cost in which L is 1, or ICN stoppage. In (7), the future cost is uniformly distributed, where $(\mathrm{P}|\mathrm{F}, \mathrm{t_l}) = \int_0^\mathrm{L} \frac{1}{\mathrm{L}} \mathrm{e}^{-\mathrm{t}} \mathrm{dt}$.

Since the fault occurrence can be modeled in discrete interval and by Weibull distribution (Section 6.1), we could replace the uniform part with our prediction model:

$$\mathrm{E}[\mathrm{C_d}] = \sum_0^1 \mathrm{C_d}(\mathrm{x}) \sum_{\mathrm{n}=1}^{\infty} \Big[\int_0^\mathrm{L} \frac{\mathrm{n}}{\mathrm{L}} \mathrm{e}^{-\mathrm{t}} \mathrm{dt}\Big]\, \mathrm{f}(\mathrm{t_l}; \mathrm{n}, \eta) \tag{11}$$

$$= \sum_0^1 C_d(x) \sum_{n=1}^{\infty} \Big[\int_0^L \frac{n}{L} e^{-t} dt\Big] \frac{n}{\eta} \left(\frac{t_l}{\eta}\right)^{n-1} \mathrm{e}^{-\left(\frac{t_l}{\eta}\right)^{\mathrm{n}}} \tag{12}$$

Thus, the expected damage cost function is formulated using the loss impact ratio of each fault and its prediction model.

6.5 Autonomic Aspect

The concept of autonomic computing by IBM has penetrated many research areas. It is best described with self-* capability handling any system configuration, control, monitoring, and etc. We illustrate how to apply FOCALE architecture [76], a novel autonomic network architecture, to our ICN fault diagnosis system. The FOCALE architecture provides the framework for developing future fault diagnosis systems

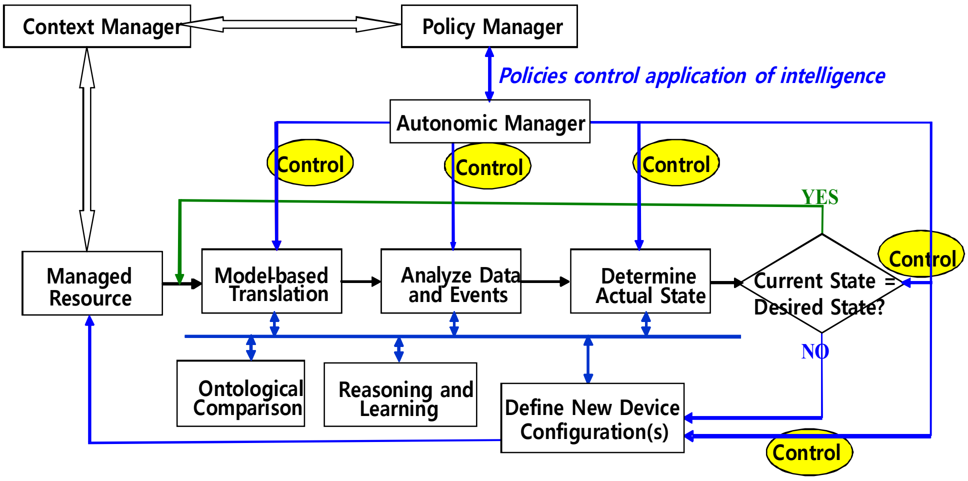


Figure 24. FOCALE architecture

Figure 24 presents the FOCALE architecture. Self-governing infers that the diagnosis systems decide its action plans, such as automatic updates of threshold values, fault decision engine update, and control of monitoring probes. We plot the tasks for ICN diagnosis requirements to each FOCALE component:

- Managed resource: This refers to PC, PLC, machineries and the diagnosis system itself which consists of multiple monitoring and analysis probes.

- Model-based translation & State decision: Two-state model transition of fault and non-fault events (on/off model) is proposed. The proposed QoS metrics determines whether the ICN system is in a good state.
- Analyze data and events: It refers to the choice of fault detection algorithms. This work presented a threshold- based method as well as policy-based fault detection approach.
- Ontological comparison: It requires defining relations between the measurement metrics and failure cases. Ontology representation of the metrics differs from decision tree where there is no definite path to the fault case. The ontology information will be cross-referenced while determining the fault state.
- Reasoning and learning: It infers to the learning database for fault case decisions by the ML engine approach and the collected fault information will be later cross-referenced with the fault prediction results in temporal aspect.
- Define new device configuration (action): This implies counteraction in case of a fault event. It may be a manual instruction to engineers or simply an alarm notification to network operators. Possible counteractions in ICN, troubleshooting cases, may be device isolation, replacement, and observation.
- Autonomic manager: It delivers user-defined policy and handles periodic updates of its rules. The state is bridged to the state decision component via system notification.
- Policy manager: It handles user input for threshold and policy configuration via Web.
- Context manager: It defines the threshold categories, policy attributes, and temporal aspect of the diagnosis results (on-line or off-line analysis).

7 Conclusion

This section summarizes the overall contents of the thesis and lists a set of contributions. Suggested areas for future work are also discussed.

7.1 Summary

Mission-critical ICN supports secure and reliable communications of devices in a process controlling or manufacturing environment. ICN has been heavily dependent on the proprietary protocols from various vendor-specific technologies. Many of them are being migrated to IP networks in order to consolidate various different types of networks into a single common network. This is undertaken to simplify the network operation, administration and maintenance and to reduce operational expenses and capital expenditures. Despite the current wide deployment of ICN, most operators have very little knowledge on how to manage their ICN reliably and securely. This is mainly due to the operators' unfamiliarity with the various faults that occur on ICN and defensive network maintenance strategy. The current process of detecting and diagnosing faults is mostly manual and thus the operators usually only detect problems after noticeable malfunctioning has occurred.

Communication failures reflect the status of machineries and their operation correctness, and can be fatal in any stage of ICN. Since all the machineries operate within a tight boundary for control delay and order, there is a possibility that a single communication failure to a device may delay or even force a shutdown of an entire plant. The question arises whether these network performance metrics are sufficient to run valuable diagnostics [5] of ICN components and their communications. Any abnormality decision with respect to typical IP traffic behavior does not necessarily coincide with ICN fault cases. A precise and specific diagnostic technique for ICN is required to remove the uncertainty of detecting a lead to trouble cases.

This thesis presents that the absence of advanced fault diagnosis techniques leads to the development of new methodologies which are suitable for ICN. We describe unique traffic characteristics and categorize the faults of ICN. We also propose novel fault diagnosis, prediction, and adaptive decision methodologies and verify them with the real-world ICN data from POSCO. Finally, this thesis proposes fault diagnosis system architecture for ICN. Our experience in developing the fault diagnosis system provides a firm guideline for understanding the fault management mechanisms in a large ICN. Overall, this thesis presents the complete cycle for handling faults in IP networks within the scope of ICN: Monitoring, analysis, and prediction.

The followings are summary of this thesis.

- This thesis provided an overview of the existing fault diagnosis techniques and previous traffic analysis results in ICN.
- The categorization of real-world ICN faults and their related phenomenon are described.
- This thesis presented the traffic characteristics and monitoring features for faults using the real-world ICN traffic traces and fault cases.
- This thesis designed fault diagnosis system architecture for ICN which targets for remote, scalable, and flexible fault detection and analysis. The detail design and implementation of fault diagnosis system for ICN is also described.
- This thesis proposed new fault diagnosis and prediction methodologies.

- This thesis validated the proposed methodologies by implementing a prototype system and applying it to a real-world ICN environment, POSCO.
- Our fault diagnosis 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.

7.2 Contributions

The area of monitoring and analyzing ICN networks has received little attention from the research community thus far. ICN networks are much more vulnerable in case of network outage because the consequences can be very costly. An active diagnosis of potential communication failures is crucial to maintaining a fault-tolerant network operating environment. The fault diagnosis system presented in this thesis provides a new systematic approach to detecting ICN specific faults.

The followings are the expected findings of the thesis.

- The limitations of current fault diagnosis techniques are described. The absence of sophisticated, but desired, fault diagnosis techniques is also highlighted.
- ICN traffic characteristics and models are presented by analyzing the real-world traffic traces.
- The categorization of real-world ICN faults and their related phenomenon are described.
- The detail design and implementation of fault diagnosis system for ICN is described.

- Future directions for fault diagnosis system are addressed by proposing the prediction and adaptive decision methodologies.

The followings are key contributions of the thesis.

- The problems of existing primitive diagnosis techniques for ICN are addressed with the real-world case study. The absence of the advanced fault diagnosis techniques leads to the development of new methodologies which are suitable for ICN.
- Traffic characteristics of ICN, which are distinguishable from the conventional IP networks, are summarized and utilized to identify the network features for fault diagnosis.
- By sharing the design and implementation experiences of fault diagnosis system architecture for ICN, a firm guideline for developing fault management mechanism in a large ICN is presented.
- Novel fault diagnosis, prediction, and adaptive decision methodologies are described and validated with the real-world ICN data.
- A complete monitoring, analysis, and prediction cycle of analyzing fault in IP networks is described. This thesis shows that ICN is an excellent test-bed for demonstrating the accuracy and efficiency of the developed diagnosis methodologies.

7.3 Future Work

There is a growing need for accurate fault diagnosis of IP-based ICN. Based on the real-world ICN deployment experience, this thesis investigated the ICN fault categories in each OSI reference model layer. We propose diagnosis techniques to cover all layers, including the network QoS faults and application-layer faults. A distributed design of ICN fault diagnosis and monitoring system is also provided. The contributions in this thesis cover more than the proposal of

ICN fault detection; in addition, we share our experiences of failure prediction, adaptive fault decision engine using ML, and self-governing detection system architecture. We evaluated each with real network statistics and communication behavior.

We consider enhancing the diagnosis system performance and reliability to cope with multi-gigabit environments in ICN. Although it is not very common to use gigabit optics other than the backbone yet, we are planning to adapt the specialized network interface cards with NPUs for ICN diagnosis purpose. Such a procedure should also support the programmable interface for exporting our fault diagnosis implementations.

Instead of predicting fault occurrences, we would like to establish the prediction model for actual network failures (rather than fault) by matching the generated alarms with real network outage cases. A large collection of failure cases will be crucial in order to build an accurate model. The SVM technique will be applicable if the sufficient enough cases are observed. We also plan to generate fault detection rules using data sets collected from other locations in POSCO.

The current diagnosis system passively reacts to failure events. We consider adapting the user experience for problem solving into the system. We rely on establishing ontology information for ICN fault and failure. With a clear relation ontology map, we will be able to define a specific ‘control’ method in order to solve troublesome cases.

Based on the traffic characteristic analysis, we will investigate the ICN network traffic growth model which can be used for network planning and anomaly detection. Furthermore, the comparison study between ICN and the ordinary IP networks will also be helpful for identifying and formulating the systematical differences of ICNs.

References

[1] Fieldbus Foundation, FF-581-1.3, “FOUNDATION Specification: System Architecture,” 2003.

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

[3] MODBUS.ORG, “Modbus Application Protocol V1.0,” 2002.

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

[5] A. Bellini, F. Filippetti, C. Tassoni, and G. Capolino, “Advances in Diagnostic Techniques for Induction Machines,” IEEE Transactions on Industrial Electronics, Vol. 55, No. 12, Dec. 2008, pp. 4109-4108.

[6] G. Schickhuber and O. McCarthy, “Distributed Fieldbus and Control Network Systems,” Computing & Control Engineering Journal, Feb. 1997, pp. 21-32.

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

[8] 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), 2001, pp. 641-647.

[9] Stefan Soucek and Thilo Sauter, “Quality of Service Concerns in IP-Based Control Systems,” IEEE Transactions on Industrial Electronics, Vol. 51, No. 6, 2004, pp. 1249-1258.

[10] Harvey Wohlwend, SEMATECH DOC ID #: 01084153D-ENG, “e-Diagnostics Guidebooks: Revision 2.1,” July 12, 2005. http://ismi.sematech.org/emanufacturing/ediagguide.htm/.

[11] J. Ploennigs, M. Guertler, M. Neugebauer, and K. Kabitzsch, "Automated Measurement-based Device Traffic Modeling in Control Networks," IEEE Conference on Industrial Informatics, June 23-27, 2007, pp. 27-32.

[12] J. Jasperneite and P. Neumann, "Measurement, Analysis, and Modeling real-time Source Data Traffic in Factory Communication Systems," WFCS, Porto, Portugal, Sept. 6-8, 2000, pp. 327-333.

[13] Y. Won, M. Choi, M. Kim, H. Noh, J. Lee, H. Hwang, and J. Hong, "Measurement Analysis of IP-based Process Control Networks," Asia Pacific Network Management Symposium (APNOMS), Sapporo, Japan, Oct. 10-12, 2007.

[14] B. Andersson, N. Pereira, W. Elemenreich, E. Tovar, and F. Pacheco, "A Scalable and Efficient Approach for Obtaining Measurements in CAN-Based Control Systems," IEEE Transactions on Industrial Informatics, Vol. 4, No. 2, May 2008.

[15] J. Feld, "PROFINET - Scalable Factory Communication For All Applications," IEEE Workshop on Factory Communication Systems, Sept. 22-24, 2004, pp. 33-38.

[16] S. Soucek, T. Sauter, and T. Rauscher, "Scheme to Determine QoS Requirements for Control Network Data over IP," 27th Annual Conference of the IEEE Industrial Electronics Society, 2001, pp. 153-158.

[17] J. Ploennigs, M. Neugebauer, and K. Kabitzsch, "Fault Analysis of Control Networks Designs," IEEE Conference on Emerging Technologies and Factory Automation, Vol. 2, Sept. 19-22, 2005, pp. 477-484.

[18] B. Xi, Y. Fang, M. Chen, and J. Liu, "Use of Ethernet for Industrial Control Networks," IEEE Conference on Industrial Electronics and Applications, May 24-26, 2006, pp 1-4.

[19] J. Moyne and D. Tilbury, "The Emergence of Industrial Control Networks for Manufacturing Control, Diagnostics, and Safety Data," IEEE Proceedings, Vol. 95, No. 1, Jan. 2007, pp. 29-47.

[20] F. Lian, J. Moyne, and D. Tilbury, "Performance Evalution of Control Networks: Ethernet, ControlNet, and DeviceNet," IEEE Control Systems Magazine, Feb. 2001, pp. 66-83.

[21] H. Zimmermann, “OSI Reference Model – The ISO Model of Architecture for Open Systems Interconnection,” IEEE Transactions on Communications, Vol. 28, No. 4, April 1980, pp. 425-432.

[22] M. Antolovic, K. Acton, N. Kalappa, S. Mantri, J. Parrott, J. Luntz, J. Moyne, and D. Tilbury, “PLC Communication using PROFINET: Experimental Results and Analysis,” IEEE Conference on Emerging Technologies and Factory Automation, Sept. 2006, pp. 1-4.

[23] X. Hao and L. Wu, “Performance Evaluation of Industrial Ethernet and Its Modeling,” International Conference on Information Acquisition, June 21-25, 2004, pp. 527-531.

[24] POSCO, http://www.posco.com/.

[25] Arndt Lüder and Kai Lorentz, “IAONA Handbook Industrial Ethernet, Industrial Automation Open Networking Alliance,” ISBN 3-00-016934-2, 2005.

[26] Libpcap, http://www.tcpdump.org/.

[27] Tcptrace, http://jarok.cs.ohiou.edu/software/tcptrace/.

[28] T. Kunz, T. Barry, X. Zhou, J.P. Black, and H.M. Mahoney, “WAP Traffic: Description and Comparison to WWW Traffic,” ACM Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems, Aug. 2000.

[29] Myung-Sup Kim, Young J. Won, and James W. Hong, “Characteristic Analysis of Internet Traffic from the Perspective of Flows,” Elsevier Computer Communications, Volume 29, Issue 10, June 19, 2006, pp. 1639-1652.

[30] E. Strangas, S. Aviyente, and S. Zaidi, “Time-Frequency Analysis for Efficient Diagnosis and Failure Prognosis for Interior Permanent-Magnet AC Motors,” IEEE Transactions on Industrial Electronics, Vol. 55, No. 12, Dec. 2008, pp. 4191-4199.

[31] H. Kim and B. Karp, “Autograph: Toward automated, distributed worm signature detection,” 13th Usenix Security Symposium, San Diego, CA, USA, Aug. 2004.

[32] S. Singh, C. Estan, G. Varghese, and S. Savage, “Automated Worm Fingerprinting,” 6th USENIX Symposium on Operating Systems Design and Implementation, San Francisco, CA, USA, Dec. 6-8, 2004.

[33] S. Singh, C. Estan, G. Varghese, and S. Savage, “The EarlyBird System for Real-time Detection of Unknown Worms”, UCSD Tech Report CS2003-0761, Aug. 2003.

[34] W. Scheirer and M. Chuah, “Comparison of Three Sliding-Window Based Worm Signature Generation Schemes,” Technical Report LU-CSE-05-025, Lehigh University, 2005.

[35] Sven Ehlert and Sandrine Petgang, “Analysis and Signature of Skype VoIP Session Traffic,” Franunhofer FOKUS Technical Report NGNI-SKYPE-06b, Berlin, Germany, July 2006.

[36] Laurent Bernaille and Renata Teixeira, “Early Recognition of Encrypted Applications,” Passive and Active Monitoring Conference, Louvain-la-neuve, Belgium, April 5-6, 2007, pp. 165-175.

[37] NetScout Sniffer, http://www.netscout.com/.

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

[39] Snort, http://www.snort.org/.

[40] Bro Intrusion Detection System, http://www.bro-ids.org/.

[41] Karp-Rabin Algorithm, http://www-igm.univ-mlv.fr/~lecroq/string/node5.html/.

[42] Weibull Distribution. http://www.weibull.com/LifeDataWeb/weibull_probability_density_function.htm/.

[43] J. E. Dickerson and J. A. Dickerson, “Fuzzy network profiling for intrusion detection,” 19th International Conference of the North American Fuzzy Information Processing Society, Atlanta, GA, USA, July 2000, pp. 301–306.

[44] J. Frank, “Artificial intelligence and intrusion detection: Current and future directions,” 17th National Computer Security Conference, Baltimore, MD, USA, 1994.

[45] S. Mukkamala and A. H. Sung, “Identifying Significant Features for Network Forensic Analysis Using Artificial Intelligent Techniques,” International Journal of Digital Evidence, Vol. 1, Issue 4, 2003, pp. 1–17.

[46] N. Cristianini and J. Shawe-Taylor, “An Introduction to Support Vector Machines,” Cambridge University Press, 2000.

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

[48] 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,” DARPA Information Survivability Conference and Exposition (DISCEX), Vol. 2, Los Alamitos, CA, USA, 2000, pp. 12-26.

[49] Francois Denis, “Pac Learning from Positive Statistical Queries,” ALT 1998.

[50] J. Quinlan, “C4.5: Programs for Machine Learning,” Morgan Kaufmann Publishers, 1993.

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

[52] Luca Deri, Industry Report, “IP Monitoring: Trends and State of the Art,” 2007.

[53] S. Waldbusser, “Remote Network Monitoring Management Information Base Version 2 using SMIv2,” IETF RFC 2021, Jan. 1997.

[54] Ahang Shiyong, Wu Chengrong, and Guw Wei, “Network Monitoring In Broadband Network”, Web Information Systems Engineering, in Proceedings of the Second International Conference, Vol. 2, Dec. 3-6, 2001, pp. 171 - 177.

[55] Napatech, http://www.napatech.com/.

[56] Inveatech, http://www.invea-tech.com/.

[57] Endace, http://www.endace.com/.

[58] Force 10, http://www.force10networks.com/products/pseries.asp/.

[59] Intel IXP family, http://www.intel.com/design/network/products/npfamily/

[60] Cavium Networks, http://www.cse.wustl.edu/ANCS/2006/OCTEON%20presentation%20for%20ANCS%202006.pdf/.

[61] Tilera, http://www.ele.uri.edu/barc2006/talks/skp/skp-1-talk-barc2006-AgarwalA.pdf/.

[62] IETF Working Group PSAMP (Packet Sampling), http://www.ietf.org/html.charters/psamp-charter.html/.

[63] C. Estan and G. Varghese, "New Directions in Traffic Measurement and Accounting," ACM SIGCOMM Internet Measurement Workshop, San Francisco, CA, USA, Nov. 2001, pp. 75-80.

[64] Cisco Systems, NetFlow, http://www.cisco.com/web/go/netflow/.

[65] P. Phaal, S. Panchen, and N. McKee, "InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks," IETF RFC 3176, Sept. 2001.

[66] Daniel W. McRobb, "cflowd Design," CAIDA, Sept. 1998.

[67] N. Stanton, "Modelling Human Alarm Initiated Activities: Implications for Alarm System Design," IEE Colloquium on Man-Machine Interfaces for Instrumentation, Oct. 23, 1995, pp. 8/1-8/4.

[68] Stenfan William, Undisclosed publication, Lulea University of Technology, 2008.

[69] ITU. X.733, "Information Technology - Open Systems Interconnection - Systems Management: Alarm Reporting Function," 1992.

[70] 3GPP, Technical Report, "3GPP TS 32.111-2: Alarm Integration Reference Point (IRP)," 2007.

[71] F. Caruso, D. Milham, S. Orobec, and T. Technologies, "Emerging Industry Standard for Managing Next Generation Transport Networks: TMF MTOSI," IEEE/IFIP Network Operations and Management Symposium, Vancouver, BC, Canada, April 3-7, 2006, pp. 1-15..

[72] TeleManagement Forum, OSS/J, Aug. 14, 2008, http://www.tmforum.org/ossj/.

[73] N. Brownlee, C. Mills and G. Ruth, “Traffic Flow Measurement: Architecture,” IETF RFC 2722, Oct. 1999.

[74] N. Brownlee, “Traffic Flow Measurement: Experiences with NeTraMet,” IETF RFC2123,” Mar. 1997.

[75] G. Sadasvian, N. Brownlee, and B. Claise, and J. Quittek, “Architecture for IP Flow Information Export,” IETF RFC 5470, Mar. 2009.

[76] B. Jennings, Sven van der Meer, S. Balasubramaniam, D. Botvich, M. O Foghlu, W. Donnelly, and J. Strassner, “Towards Autonomic Management of Communications Networks,” IEEE Communications Magazine, Vol. 45, Nov. 10, Oct. 2007, pp. 112-121.

[77] W. Zhang, M. S. Branicky, and S. M. Phillips, “Stability of Networked Control System,” IEEE Control Systems Magazine, Feb. 2001, pp.84-99.

[78] S. Park, “Code Calibration for Cost Effective Seismic Designload,” KORUS, Vol. 2, June 26, 2001, pp. 48-52.

[79] ITU Recommendation X.733, “Information Technology – Open Systems Interconnection - System Management: Alarm Reporting Function,” 1992.

[80] Stanford University, Network Monitoring Tools, http://www.slac.stanford.edu/xorg/nmtf/nmtf-tools.html/.

Biography

Education

2006-2009 : Ph.D., Dept. of Computer Science and Engineering,
POSTECH, Korea
2004-2006 : MS, Dept. of Computer Science and Engineering, POSTECH, Korea
1999-2003 : B. Mathematics, Honors Computer Science, Faculty of Mathematics,
University of Waterloo, ON, Canada

Publications: International Journal/Magazine Papers

1. Young J. Won, Mi-Jung Choi, James W. Hong, Chan-Kyu Hwang, and Jae-Hyoung Yoo, “Measurement of Download and Play and Streaming IPTV Traffic”, IEEE Communications Magazine, Vol. 46, No. 10, October, 2008, pp. 154-161. **(SCI)**

2. 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. **(SCI)**

3. Myung-Sup Kim, Young J. Won, and James W. Hong, “Characteristic Analysis of Internet Traffic from the Perspective of Flows”, Elsevier Computer Communications, Vol. 29, Issue 10, June 19 2006, pp. 1639-1652. **(SCIE)**

4. Myung-Sup Kim, Young J. Won, and James Won-Ki Hong, “Application-Level Traffic Monitoring and Analysis on IP Networks”, ETRI Journal, Vol.27, No.1, Feb. 2005, pp.22-42. **(SCI)**

Publications: International Conference/Workshop Papers

5. Jae Yoon Chung, Byungchul Park, Young J. Won, John Strassner, and James W. Hong, “An Effective Similarity Metric for Application Traffic Classification”, 12th IEEE/IFIP Network

Operations and Management Symposium (NOMS), Osaka, Japan, Apr. 19-23, 2010. (Accepted to appear)

6. Sung-Su Kim, Young J. Won, John Strassner, and James W. Hong, "Manageability of the Internet: Management with New Functionality", 12th IEEE/IFIP Network Operations and Management Symposium (NOMS), Osaka, Japan, Apr. 19-23, 2010. (Accepted to appear)

7. Jae Yoon Chung, Byungchul Park, Young J. Won, John Strassner, and James W. Hong, "Traffic Classification Based on Flow Similarity", 8th IEEE International Workshop on IP Operations & Management (IPOM), Venice, Italy, Oct. 29-30, 2009.

8. Seong-Cheol Hong, Jin Kim, Byungchul Park, Young J. Won, and James W. Hong, "Traffic Growth Analysis over Three Years in Enterprise Networks", 15th Asia-Pacific Conference on Communications, Shanghai, China, Oct. 8-10, 2009.

9. Suman Pandey, Young J. Won, Hong-Taek, Ju, and James. W. Hong, "Dimensioning of IPTV VoD Service in Heterogeneous Broadband Access Networks", 12th Asia-Pacific Network Operations and Management Symposium (APNOMS), Jeju Island, Korea, Sept. 23-25, 2009.

10. Sung-Su Kim, Young J. Won, Mi-Jung Choi, James W. Hong, and John Strassner, "Towards Management Requirements of the Future Internet", IEEE/IFIP Workshop on Management of the Future Internet (conjunction with IM 2009), New York, USA, June 5, 2009, pp. 1-6.

11. Byungchul Park, Young J. Won, Hwanjo Yu, James W. Hong, Hong-Sun Noh, and Jang Jin Lee, "Fault Detection in IP-based Process Control Networks using Data Mining", 11th IEEE/IFIP Integrated Network Management (IM), New York, USA, June 1-5, 2009, pp. 211-217.

12. Byung-Chul Park, Young J. Won, Mi-Jung Choi, Myung-Sup Kim, and James W. Hong, "Empirical Analysis of Application-level Traffic classification using Supervised Machine Learning", 11th Asia-Pacific Network Operations and Management Symposium (APNOMS), LNCS 5297, Beijing, China, Oct. 22-24, 2008, pp. 474-477.

13. Young J. Won, Byung-Chul Park, Mi-Jung Choi, James W. Hong, Hee-Won Lee, Chan-Kyu Hwang, Jae-Hyoung Yoo, "End-User IPTV Traffic Measurement of Residential Broadband

Access Networks", 6th IEEE International Workshop on End-to-End Monitoring Techniques and Services, Salvador, Brazil, April 7, 2008, pp. 95-100.

14. Byung-Chul Park, Young J. Won, Myung-Sup Kim, and James Won-Ki Hong, "Towards Automated Application Signature Generation for Traffic Identification", 11th IEEE/IFIP Network Operations and Management Symposium (NOMS), Salvador, Brazil, April 7-11, 2008, pp. 160-167.

15. Young J. Won, Mi-Jung Choi, Myung-Sup Kim, Hong-Sun Noh, Jun Hyub Lee, Hwa Won Hwang, and James W. Hong, "Measurement Analysis of IP-Based Process Control Networks", 10th Asia-Pacific Network Operations and Management Symposium (APNOMS), LNCS 4773, Sapporo, Hokkaido, Japan, Oct. 10-12, 2007, pp. 385-394.

16. Young J. Won, Mi-Jung Choi, Jang Jin Lee, Jun Hyub Lee, Hwa Won Hwang, and James W. Hong, "Detecting Network Faults on Industrial Process Control IP Networks", 7th IEEE International Workshop on IP Operations & Management (IPOM), LNCS 4786, San Jose, CA, USA, Oct. 31-Nov. 2, 2007, pp. 184-187.

17. Young J. Won, B.C. Park, S.C. Hong, K.B. Jung, H.T. Ju, James W. Hong, "Measurement Analysis of Mobile Data Networks", 8th Passive and Active Measurement Conference (PAM), LNCS 4427, Louvain-la-neuve, Belgium, April 5-6, 2007, pp. 223-227.

18. Young J. Won, Byung-Chul Park, Hong-Taek Ju, Myung-Sup Kim, and James W. Hong, "A Hybrid Approach for Accurate Application Traffic Identification", 4th IEEE International Workshop on End-to-End Monitoring Techniques and Services, Vancouver, Canada, April 3, 2006, pp. 1-8.

19. Seung-Hwa Chung, Young J. Won, Deepali Agrawal, Seong-Cheol Hong, and James Won-Ki Hong, "Detection and Analysis of Packet Loss on Underutilized Enterprise Network Links", 3rd IEEE International Workshop on End-to-End Monitoring Techniques and Services, Nice, France, May 15, 2005, pp. 164-176.

20. Myung-Sup Kim, Young J. Won, Hyung-Jo Lee, James W. Hong, and Raouf Boutaba, "Flow-based Characteristic Analysis of Internet Application Traffic", 2nd IEEE International Workshop on End-to-End Monitoring Techniques and Services, San Diego, California, USA,

October 3, 2004, pp. 62-67.

Domestic Journal/Conference Papers

3 journal & 8 conference papers (in Korean), available upon request

Patents

1. Registration date: 2007-04-16, Registration #: 10-0710047, “Apparatus for Traffic Identification on Internet Protocol Network Environment”
2. Registration #: 10-2008-0135020, “Method and System for Detecting Error in Process Control Network”, FILED in Dec. 2008.
3. Registration #: 10-2008-0133944, “Method and Apparatus for Predicting Error in Process Control Network”, FILED in Dec. 2008.
4. Registration #: 10-2009-0046093, “Signature Generation Apparatus for Network Behavior of Applications, Collection Server, Detection System for Network Behavior, and Signature Generation Method for Network Behavior”, FILED in May 2009.

Research/Project Experience

- Title: “Self-organizing Network Expert System” – Samsung Electronics (2009-)

This research project will analyze how current wireless networks are managed, and develop a new approach that uses an expert system and production rules to self-configure and -organize wireless network resources (Mobile WiMax and LTE).

- Title: “IT Convergence for Ubiquitous Autonomic Systems” – Ministry of Education, Science, and Technology, Korea (2009-)

This is part of the research work proposed under our World Class University grant from the Korean government. My work involves investigating how autonomic mechanisms can be applied to manage new ubiquitous computing systems that use bio-informatics, nano-technologies, and networking technologies for building ubiquitous computing applications (called ubiquitous health and ubiquitous environment applications in Korea).

- Title: “Fault Detection, Diagnosis, Prediction for IP-based Industrial Control Networks” – Ph.D. Thesis (2009)

This thesis proposes novel fault diagnosis, prediction, and adaptive decision methodologies, and verifies them with real-world ICN data from POSCO.

- Title: “Collect, Analyze, and Share for Future Internet (CASFI) – Manageability of Future Internet” – Ministry of Knowledge Economy, Korea (2008-)

CASFI is a 5-year-long government funded project that focuses on high-precision network performance measurement and analysis. This project is developing new performance measurement and analysis mechanisms for the current Internet; this will be used to gain insight to develop better manageability approaches for the future Internet. I have focused on investigating different manageability challenges for the Future Internet. For example, I have developed an automated signature generation system and similarity-based traffic classification algorithms. I also have given a talk at the CAIDA-WIDE-CASFI annual joint workshop since 2008.

- Title: “Fault Detection & Prediction for Industrial Control Networks” – POSCO (2008-2009)

This work was a follow-on project from the above fault monitoring project of POSCO. I extended the capabilities of their existing fault diagnosis system to include fault prediction and adaptive decision.

- Title: “Strategic Planning Towards All-IP Based Enterprise Networks” – POSCO (2008)

This project was funded by POSCO. I performed a comprehensive strategic analysis that explained how POSCO could migrate from proprietary protocols and devices to an all-IP based enterprise networks in the near future. It involved the simulation and impact analysis using OPNET for different scenarios, including their business and manufacturing networks.

- Title: “IPTV Traffic Measurement and Dimensioning Methodology” – POSTECH Internal

Project (2008)

This project focused on IPTV measurement and proposed a more efficient delivery infrastructure for IPTV using xDSL and FTTH. A comparison study between unicast and multicast delivery of VoD was conducted. It also included an optimal dimensioning methodology analysis for IPTV network deployment.

- Title: “Triple Play Services Traffic Impact Analysis on Broadband Access Networks – Korea Telecom (2007)

Korea Telecom provides full scale triple- and quadruple-play services over IP networks. This project focused on traffic impact analysis on broadband access networks (e.g., xDSL, FTTH) that used large-scale triple play services. It involved traffic measurement and analysis of each of the services provided, along with analyzing how the user utilized each service. I was responsible with providing analysis results and projection models for these services.

- Title: “Remote Fault Monitoring System for Industrial Control Networks” – POSCO (2006-2007)

This project was funded by POSCO, the second largest iron and steel manufacturer in the world. POSCO operates many different types of specialized IP networks for their manufacturing process and various business-specific uses. I was responsible for developing an advanced fault monitoring system for a set of their Industrial Control Networks (ICN); this system was modular and independent of any one specific ICN. I also developed a set of related analysis techniques for handling different types of ICN-specific alarms and faults.

- Title: “Performance Analysis of Underutilized Enterprise Networks” – Purdue University (2006)

This work was a part of a joint research effort with Purdue University, USA. It focused on understanding why the performance of underutilized network resources still degraded. This involved both active and passive monitoring and analysis.

- Title: “Traffic Analysis and Application-specific Billing Systems for Commercial Mobile Data Networks” – nTelia (2006)

This was a confidential research project for a large Korean vendor. The project goal developed sophisticated billing schemes for Internet users over 2G and 2.5G cellular phone networks. It involved real-time packet inspection for improved classification and billing by using advanced traffic characteristic analysis in mobile data networks.

- Title: “A Hybrid Approach for Accurate Application Traffic Identification” – Master Thesis (2006)

This thesis proposed a hybrid approach of signature matching and flow relationship methods to more accurately identity application traffic. This approach defined a priority-based signature matching scheme that used a limited number of packet samples to replace the existing conventional signature matching mechanism. The result of this thesis showed how to combine the above two approaches to improve the accuracy of signature matching while using the flow relationship mapping to uncover concealed application traffic.

- Title: “MPLS Metric Tree” – Bell University Network Lab at the University of Waterloo & Bell Canada (2005)

This was a part of a larger research project with the University of Waterloo and Bell Canada. I investigated a set of low-level MPLS network performance metrics and translated them to higher layer business metrics.

- Title: “Application-level Traffic Classification Methodology for High-Speed Networks” – Korea Telecom (2005)

Korea Telecom is the number one ISP in the country; its R&D department operates the network monitoring system for its subscribers. An important problem was analyzing user behavior of P2P application usage. This type of traffic was increasing at a significant rate; therefore, this project focused on analyzing this type of traffic for a variety of different scenarios. I participated in developing and implementing the Flow Relationship Mapping (FRM) algorithm in KT’s high-speed network monitoring system.

- Title: “Traffic Monitoring and Control System for High-speed Networks” – Korea Industrial Technology Foundation (2003-2005)

This was a two-year government-funded project focused on the design and development of traffic monitoring and control for a high-speed backbone network. This led to developing a Next Generation Traffic Monitoring System (NG-MON) for performing host- and application-analysis of IP network traffic metrics as well as low-level security analysis of networks and systems. The developed system has been tested on various networks, including the Korean Internet eXchange point (KIX) and other private enterprise networks.