규칙 기반 라우팅 구성 장애 관리 시스템의 설계 및 구현

정회원 황 태 인*, 안 성 진**, 정 진 욱*

Design and Implementation of Rule-based Routing Configuration Fault Management System

Tae In Hwang*, Seongjin Ahn**, Jin Wook Chung* *Regular Members*

요 약

이 논문에서는 시스템의 라우팅 구성 장애를 진단하고 복구하기 위한 규칙과 알고리즘을 제시하고, 이를 이용하여 피관리 시스템에서 발생한 라우팅 구성 장애를 자동으로 관리할 수 있는 시스템을 JAVA를 이용하여 설계하고 구현하였다. 라우팅 구성 장애 관리를 위하여 네트워크 구성 관리 생성 규칙, 라우팅 구성 장애 진단 생성 규칙, 라우팅 구성 장애 복구 생성 규칙을 제안하였으며 후향 추론 알고리즘을 기반으로 이런 생성 규칙간의 상호 연동을 위하여 메타 규칙을 적용하여 시스템을 구현하였다. 구현한 시스템을 시나리오에 기반하여 규칙, Blackboard, 목표의 변화 과정을 보여줌으로써 실험 결과를 제시하였다. 이 논문에서는 이질적이고 급변하는 네트워크 환경에 능동적으로 대처할 수 있는 시스템 구현을 통해서 시스템의 라우팅 구성 장애 관리에 드는 비용과 시간을 줄이고, 시스템의 네트워크 구성 장애 관리를 위한 방법론을 제시하고자 한다.

ABSTRACT

In this paper, we have defined the rules and the algorithm for diagnosis and recovery of routing configuration fault on a system. By using them, we have implemented the Java-based system that can manage routing configuration fault automatically. To manage routing configuration fault, the production rule for network configuration management, the production rule for routing configuration fault diagnosis, and the production rule for routing configuration fault recovery have been proposed. Rule-based routing configuration fault management system has been implemented on the basis of backward chaining algorithm and applied for meta rules for the purpose of interconnecting the production rules. We have derived the experimental result from transition process of the rules, the Blackboard, the goals based on scenarios. Through the implementation of dynamically applicable system in heterogeneous and rapidly changing network environments, we have proposed the methodology for network configuration fault management. Also, we expect that network configuration manager can reduce time and cost wasted for routing configuration fault management.

Ⅰ. 서 론

인터넷의 폭발적인 성장과 함께 네트워크 장비들 뿐만 아니라 이런 장비들에 연결되어 있는 각기 다른 운영체제가 적재된 시스템들 역시 계속해서 증가하고 있는 추세이다. 하지만 기하 급수적으로 증가하고 있는 이런 피관리 시스템들의 수에 비해 이들을 관리하기 위한 전문 지식을 갖춘 관리자는 터무니없이 부족한 상태이다. 이로 인해 시스템의 네

* 성균관대학교 전기전자 및 컴퓨터공학부(tihwang@songgang.skku.ac.kr)
** 성균관대학교 컴퓨터 교육과(sjahn@comedu.skku.ac.kr)
논문번호 : 00088-0306, 접수일자 : 2000년 3월 6일

트워크 구성 장애를 자동으로 관리해줄 수 있는 시스템에 관한 연구가 필요하게 되었다[1-3].

현재까지 개발되어져 있는 시스템들은 대부분 일반적인 프로그래밍 기법에 의해서 네트워크 구성 장애를 자동으로 진단하고 복구하는 시스템이다. 하지만, 현재와 같이 이질적이고 계속해서 변화해 가는 네트워크 환경에서 정적으로 프로그램된 시스템들은 적응력이 떨어지며 변화해 가는 네트워크 환경에 대처하기 힘들다. 그렇기 때문에 이런 시스템은 바람직하지 않으며 규칙이나 사례 기반 시스템의 설계 및 구현이 필요하다[4]. 규칙이나 사례를 기반으로 프로그램된 시스템을 피관리 시스템에 도입한다면 변화된 정보에 대한 규칙 및 사례들만을 수정하여 이질적이고 급변하는 네트워크 환경에 쉽게 대처할 수 있다.

네트워크 장애 관리를 위한 시스템으로 LODES와 CRITTER가 구현되었다[5-6]. LODES는 LAN 상의 장애 검출 및 위치 확인을 위한 규칙 기반 전문가 시스템이며, CRITTER는 네트워크를 사례에 기반하여 관리하기 위한 시스템이다. 또한, ENCORE (Ensemble of Cooperative Reflector Agents) 와 DUMBO(Descobrindo solUcoes Manipulando uma Base de Ocorrencias) 시스템에 관한 연구가 최근에 수행되어지고 있다[7-8]. 하지만 이런 구현된 시스템과 연구가 진행중인 시스템은 네트워크 상의 장애를 검출하고 위치 확인은 가능하지만 장애가 발생한 시스템을 직접 제어하여 잘못된 네트워크 구성 정보를 찾아내고 복구하지는 못한다. 즉, 장애가 발생한 네트워크를 진단하여 장애 발생 위치를 추적하여 진단 및 자문은 일부 가능하지만 시스템 내에서 발생한 장애를 관리자 개입 없이 스스로 시스템의 네트워크 구성 장애를 복구하기 위한 연구는 이루어지지 않았다[9-11].

시스템의 TCP/IP 네트워크 구성 관리와 관련하여 시스템에서 발생할 수 있는 네트워크 장애들 중에서 라우팅 구성 장애를 진단하고 복구하기 위하여 후향 추론 알고리즘에 기반한 생성 규칙을 제시하고 장애 진단 및 복구를 위한 규칙 기반 시스템을 구현하였다. 이질적이고 급변하는 네트워크 환경에 쉽게 대처할 수 있는 시스템 개발을 통해서 대규모의 네트워크를 관리하는 관리자가 시스템에서 발생한 네트워크 구성 장애를 관리하기 위해 드는 시간과 비용을 줄이고, 시스템에서 발생할 수 있는 다른 네트워크 구성 장애 문제 해결을 위한 방법론을 제시하고자 한다.

Ⅱ. 라우팅 구성 장애 진단 및 복구 생성 규칙

1. 관련 연구

CRITTER는 TTR(Trouble Ticketing System)과 CBR(Case-Based Reasoning)기법을 이용하여 네트워크 장애를 복구하기 위한 지식을 표현함으로써 예상치 못한 문제에 대한 해를 제공하고자 하는 취지에서 연구가 수행되었다. 장애 복구 전문 지식이 규칙이 아니라 사례에 기반하여 표현되었다. 유사한 경우를 검색하는 메커니즘을 통해 예기치 못한 문제에 대한 해결책을 찾는다. 그와 유사한 연구로 DUMBO(Descobrindo solUcoes Manipulando uma Base de Ocorrencias) 시스템이 현재 연구가 진행중에 있다. 이 시스템 역시 TTR을 이용하여 장애를 Trouble Ticket으로 나타내며 Ticket은 설명부와 해결부로 나누어진다. 설명부에서는 장애가 발생할 당시의 시스템의 상태 및 증상이 기입되어 있으며 해결부에는 실제로 관리자로부터 조언을 받기 위한 부분이다.

위의 두 시스템은 TTR과 CBR 기법을 이용하여 네트워크 구성 장애를 복구하기 위한 지식을 표현하였다는 공통점을 가지고 있지만 몇 가지 차이점이 있다. CRITTER 시스템은 Ticket에 들어 있는 설명부 필드와 해결부 필드가 여러 개의 필드로 복잡하게 나누어져 있으며 이 필드에 들어가는 값들이 기호화되어 있다. 반면, DUMBO 시스템은 Ticket에 들어 있는 설명부와 해결부가 아주 간단한 구조로 되어 있다. 특히, 해결부는 거의 구조화 되어있지 않으며 문제에 대한 해결은 텍스트 형태로 입력된다. CRITTER는 추론 단계(input, retrieve, adapt, propose, process) 중 Propose단계에서 사용자의 개입을 통한 사용자의 조언을 얻을 수 있지만 DUMBO 시스템은 다양한 추론 단계에서 사용자의 개입을 허용한다. 이를 통해 능숙한 사용자가 문제를 처리하는데 있어 더 많은 도움을 준다.

LODES 시스템은 RBR(Rule-Based Reasoning) 기법을 이용하여 TCP/IP 관련 문제 해결을 위한 지식을 표현하며 복잡하고 이질적인 네트워크 환경에서 일어나는 문제들을 해결하기 위한 취지에서 연구가 수행되었다. 이 시스템의 중요한 특징은 TCP/IP와 관련한 지식을 가지고 있다는 것이다. 물리적으로 분산되어져 있는 각각의 네트워크는 LODES 시스템이 존재하며 문제의 본질에 따라 독립적 및 협동관계에 의해 문제를 진단한다. 그리고

문제의 증상을 감지하기 위한 패킷 감지기를 가지고 있다. 패킷 수집뿐만 아니라 장애가 발생한 호스트의 응답을 확인하기 위해 패킷을 장애가 발생한 호스트로 보내어 장애를 진단한다. 그와 유사한 연구로 ENCORE(Ensemble of Cooperative Reflector Agents) 시스템이 현재 연구가 진행 중에 있다. LODES 시스템이 각각의 LAN 세그먼트에 상주하면서 장애를 모니터링하고 진단하는 시스템인데 반해 ENCORE 시스템은 AS(Autonomous System)에 상주하면서 각각의 AS들 사이의 협동에 의해 발생할 수 있는 장애를 진단하는 시스템이다.

CRITTER와 DUMBO 시스템은 CBR기법을 이용한 시스템으로써 장애를 감지할 때마다 관리자의 지식을 Ticket형식으로 표현하여 해결책을 제시하며 이를 데이터 베이스에 저장한다. 이 방식은 예기치 못한 문제가 발생할 수 있는 도메인을 대상으로 했을 경우 관리자의 조언을 통해 문제 해결에 도움을 줄 수 있다. 하지만, 도메인이 한정되어 있는 시스템의 장애를 진단하고 복구하기 위한 시스템으로는 복잡하며 관리자에게 번거로운 작업을 요구하는 시스템이다. 장애가 감지된 시점의 시스템의 상태 정보를 통해서 네트워크 전문가가 이에 대한 조언을 제시한다는 것 역시 상당히 힘든 일이다. 위와 같은 문제로 인해 이 논문에서는 RBR을 이용한 시스템을 구현하였다.

LODES 와 ENCORE 시스템은 RBR기법을 이용한 시스템으로써 관리자의 지식을 미리 규칙으로 정의해 두며 이를 통해 관리 도메인에 대한 장애를 진단하는 시스템이다. 이들 시스템은 네트워크에 연결되어 있는 각각의 네트워크화된 장비에 내재되어 있는 시스템이 아니라 적게는 LAN 세그먼트내에 하나 또는 AS에 하나의 시스템이 존재하면서 독립적으로 또는 서로간의 협동에 의해 문제를 해결하는 방식이다. 하지만 이 방식은 실제로 각각의 네트워크화된 장비에서 발생하는 네트워크 구성 장애를 해결해 줄 수는 없으며 관리자는 문제가 발생하였을 경우 이 시스템에 직접 가서 문제를 해결해야 한다. 이 논문에서 제시한 시스템은 각각의 네트워크화된 장비에 내재되어 시스템에서 발생하는 네트워크 구성 장애를 모니터링함으로써 시스템에서 발생하는 문제를 시스템 내에서 스스로 장애를 진단하고 복구함으로써 관리자가 장애가 발생한 장소로 직접 가서 시스템을 제어해야 하는 번거로움을 줄일 수 있는 시스템이라는 점에서 위의 시스템들과 다르다.

2. 진단 및 복구 생성 규칙 표현

생성 규칙은 전문지식을 나타내는 가장 빠른 방법이며 이 방법은 전문가 시스템에서 가장 많이 사용되고 있다. 사실을 결합시킬 경우 결론에 도달하기 위하여 규칙을 사용하며 이 결론은 또한 새로운 사실이 된다. 이것의 장점은 이해하기 쉽고 단편적인 지식 표현에 적합하며 새로운 지식을 추가하기가 용이하다는 것이다. 또한 인간의 행동을 쉽게 모델화할 수 있다. 단점으로는 비효율적이고 불투명하며 시스템 구조를 고려하지 않은 상태에서 지식을 표현하므로 많은 양의 지식을 표현할 수 없다는 것이다.

이 논문에서는 후향 추론을 기반으로 한다. 추론 방법의 선택은 전문가 시스템을 응용하는 분야의 특성에 따라서 선택이 이루어지게 된다. 전향 추론은 설계 및 계획형 문제에 주로 이용되며 후향 추론 방식은 진단 및 분류 등의 문제에 주로 이용된다. 라우팅 장애를 진단하고 복구하기 위한 추론 방법으로는 후향 추론이 적합하다. 후향 추론 규칙 기반 시스템 설계를 위해서는 다음과 같은 단계가 필요하다. 첫 번째, 문제를 정의해야 한다. 두 번째, 목표를 정의해야 한다. 세 번째, 목표를 달성하기 위한 규칙을 설계해야 한다. 네 번째, 사용자를 위한 인터페이스를 설계해야 한다. 이 장에서는 문제 및 목표를 정의하고 생성규칙을 설계한다.
제안한 규칙은 추후 확장을 고려하여 문제를 네트워크 구성 장애로 정의하고 장애 유형은 다음과 같이 네 가지로 정의하였다.

표 1. 네트워크 구성 장애 유형

장애 유형	설명
라우팅 구성 장애	디폴트 라우터 설정 오류 및 동적 라우팅 실패로 인하여 발생하는 외부 네트워크와의 통신 장애를 말한다.
도메인 네임 서비스 구성 장애	도메인 네임 서버 설정 오류로 인하여 도메인 네임을 IP 주소로 변환하지 못하기 때문에 발생하는 장애를 말한다.
인터넷 데몬 프로세스 구성 장애	데몬 프로세스 구동 실패로 인하여 응용 프로그램이 올바르게 동작하지 않기 때문에 발생하는 장애를 말한다.
IP 중복 장애	서로 다른 시스템이 같은 IP 를 사용하고 있을 경우 발생할 수 있는 장애를 말한다.

규칙을 표현하기 위해 후향 추론 방식을 지원하는 LEVEL 5 쉘에서 제공하는 DISPLAY, CHAIN 명령을 이용하였다. DISPLAY 명령을 사용하여 중간에 중요한 정보들을 출력하였다. 또한 각각 분리되어 있는 전문가 모듈 사이에 제어를 넘겨주고 받

기 위해 CHAIN 명령을 사용한 메타 규칙을 적용하였다. 각 모듈 사이의 통신은 Blackboard를 이용하였으며, Blackboard는 각 모듈에 의해 결정된 문제에 대한 정보를 담고 있으며 이 정보는 문제 해결을 위해 다른 모듈에서 사용되어진다[12]. 장애 원인으로 판명된 사실이 장애 복구 모듈에서 복구가 성공적으로 이루어졌다면 다른 전문가 모듈에 영향을 주지 않기 위해서 DEL 명령을 사용하여 그 사실을 Blackboard에서 삭제하였다.

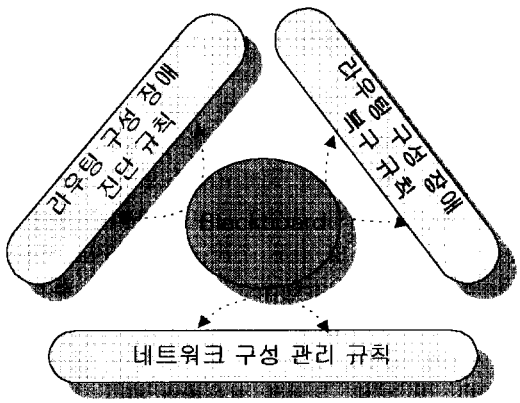

그림 1. 생성 규칙과 Blackboard 간의 관계

[그림 1]과 같이 본 논문에서는 생성규칙을 3개의 모듈 즉, 네트워크 구성 관리 생성 규칙 모듈(M1), 라우팅 구성 장애 진단 생성 규칙 모듈(M21), 라우팅 구성 장애 복구 생성 규칙 모듈(M22)로 나누었으며 각각의 생성 규칙을 표기하기 위해 다음과 같이 정의하였다.

[정의 1] 각 모듈이 추구하는 목표의 집합을 G라고 정의하며 다음과 같다. $G = \{ g11, g12, g211, g212, g221, g222 \}$

[정의 2] 현존 사실과 추론에 의해 새로 생성된 사실들의 집합을 F라고 정의하며 다음과 같다. $F = \{ f1, f2, f3, f4, f5, f6, f7, f8, f9, f10, f11 \}$

[정의 3] 추론 중에 어떤 사실을 얻기 위해 필요한 질의함수의 집합을 Q라고 정의하며 다음과 같다. $Q = \{q1, q2, q3, q4, q5, q6, q7, q8, q9, q10, q11, q12, q13, q14, q15, q16, q17, q18 \}$

[정의 4] 질의함수 수행 후에 발생할 수 있는 결과 값, 즉, 사건의 집합을 E 라고 정의하며 다음과 같다. *E = { unix, win, linux, up, down, exist, notexist, noresponse, response , success, fail, match, notmatch }*

M1

```
GOAL  g11, g12

RULE 1 : IF  q1 = UNIX  THEN  g11  AND  f1
RULE 2 : IF  q2 = fail  AND  f1
         THEN  g12  AND  f2  AND  CHAIN(M21)

END
```

g11 : 시스템 유형 판별 g12 : 장애 유형 판별 f1 : 시스템 유형 = 유닉스 f2 : 장애 유형 = 라우팅 구성 장애	q1 : 시스템 운영체제 검사 q2 : 1홉까지의 경로 추적 결과 검사

그림 2. 네트워크 구성 관리 생성 규칙

[그림 2]의 네트워크 구성 관리 생성 규칙의 목표는 시스템의 운영체제를 판별하고 발생한 장애가 네트워크 구성 장애 유형 중 어떤 장애 유형에 속하는지를 판별하는 것이다.

M21

```
GOAL  g211, g212

RULE 3 : IF  q3 = down  THEN  g211  AND  f3  AND  DISPLAY (f3)
RULE 4 : IF  q4 = exist  THEN  g211  AND  f4  AND  DISPLAY(f4)
RULE 5 : IF  q5 = notmatch  THEN  g211  AND  f5  AND  DISPLAY(f5)
RULE 6 : IF  q6 = notmatch  THEN  g211  AND  f6  AND  DISPLAY(f6)
RULE 7 : IF  q5 = match  THEN  g211  AND  ¬ f5
RULE 8 : IF  q6 = match  THEN  g211  AND  ¬f6
RULE 9 : IF  q7 = noresponse  AND  q8 = exist AND  ¬f5  AND  ¬ f6
         THEN  g21  AND  f7  AND  DISPLAY(f7)
RULE 10 : IF  q8 = notexist  THEN  g211  AND  f8 AND  DISPLAY(f8)
RULE 11 : IF  q9 = exist  OR  q10 = exist  THEN
          f8  AND  f9  DISPLAY(f9)
RULE 12 : IF  q9 = notexist  AND  q10 = notexist  THEN  f8  AND  f10
          DISPLAY(f10)
RULE 13 : IF  CHAIN(M22)  THEN  g212

END
```

g211 : 라우팅 장애 유형 판별 g212 : 장애 복구 모듈로 전이 f3 : 장애 원인 = 인터페이스 나운 f4 : 장애 원인 = 케이블 연결 상태 불량 f5 : 장애 원인 = 네트워크 주소 변경 f6 : 장애 원인 = 서브넷 마스크 변경 f7 : 장애 원인 = 디폴트 게이트웨이 다운 또는 디폴트 게이트웨이 주소 변경 f8 : 장애 원인 = 디폴트 엔트리 부재 f9 : 장애 원인 = 디폴트 엔트리 삭제 f10 : 장애 원인 = 라우팅 데몬 프로세스 다운	
q3 : 인터페이스 상태 검사 q4 : 로그정보에 "disconnected cable" 엔트리 존재 유무 검사 q5 : 백업된 네트워크 주소와 현재 네트워크 주소 일치 여부 검사 q6 : 백업된 서브넷 마스크 주소와 현재 서브넷 마스크 주소 일치 여부 검사 q7 : 디폴트 게이트웨이 주소로의 ping 수행 결과 검사 q8 : 라우팅 테이블에 "default" 또는 "0.0.0.0" 엔트리 존재 유무 검사 q9 : DefaultRouter 파일 존재 유무 검사 q10 : 라우팅 데몬 프로세스 존재 유무 검사	

그림 3. 라우팅 구성 장애 진단 생성 규칙

[그림 3]의 라우팅 구성 장애 진단 생성 규칙의 목표는 시스템내의 라우팅 구성 장애를 유발한 원인이 무엇인지를 식별하는 것이며 식별된 원인을 라우팅 장애 복구 생성 규칙 모듈에 전달하는 것이다.

M22

```
GOAL  g221, g222

RULE14 : IF  f3  AND  q11 = success  THEN g221  AND  DEL(f3)
RULE 15 : IF  f4  AND  q12 = success  THEN g221  AND  DEL(f4)
RULE 16 : IF  f5  AND  q13 = success  THEN g221  AND  DEL(f5)
RULE 17 : IF  f6  AND  q14 = success  THEN g221  AND  DEL(f6)
RULE 18 : IF  f7  AND  q15 = success  THEN g221  AND  DEL(f7)  AND
          DEL(-f5)  AND  DEL(-f6)
RULE 19 : IF  f9  AND  q16 = success  THEN g221  AND  DEL(f9)
RULE 20 : IF  f10  AND  q17 = success  THEN g221  AND  DEL(f10)
RULE 21 : IF  q18 = exist  THEN g222  AND  f11  AND  DISPLAY(f11)
RULE 22 : IF  q18   notexist  THEN  g222  AND  DEL(f2)
          AND DISPLAY(라우팅 장애 복구 완료)

END
```

g221 : 장애 복구
g222 : 장애 복구 결과 출력
f3 : 장애 원인 = 인터페이스 다운
f4 : 장애 원인 = 케이블 연결 상태 불량
f5 : 장애 원인 = 네트워크 주소 변경
f6 : 장애 원인 = 서브넷 마스크 변경
f7 : 장애 원인 = 디폴트 게이트웨이 다운 또는 디폴트 게이트웨이 주소 변경
f9 : 장애 원인 = 디폴트 엔트리 삭제
f10 : 장애 원인 = 라우팅 데몬 프로세스 다운
f11 : 라우팅 장애 복구 실패

q11 : 다운된 인터페이스를 UP 상태로 변경
q12 : 케이블 연결 점검 후 인접호스트에 대해 ping 실행
q13 : 인접 호스트로부터 네트워크 주소를 얻어온 후 그 주소로 대체
q14 : 인접 호스트로부터 서브넷 마스크 주소를 얻어온 후 그 주소로 대체
q15 : 인접 호스트로부터 디폴트 게이트웨이 주소를 얻어온 후 그 주소로 대체
q16 : 디폴트게이트웨이 주소를 라우팅 테이블에 추가
q17 : 동적 라우팅 데몬 프로세스를 실행시킴
q18 : f1, f2 이외의 사실 존재 유무 검사

그림 4. 라우팅 구성 장애 복구 생성 규칙

[그림 4]의 라우팅 구성 장애 복구 생성 규칙의 목표는 장애 원인에 대해서 어떠한 식으로 복구할 것인지에 대한 방법을 결정하며 실제로 복구를 수행하는 것이다.

3. 라우팅 구성 장애 진단 및 복구를 위한 후향 추론 알고리즘

위에 설명된 생성 규칙을 이용하여 새로운 사실을 유추해내기 위해서는 추론 알고리즘이 필요하다. 이 논문에서는 후향 추론 알고리즘을 기반으로 여러 다른 규칙들이 추후에 서로 연결 가능할 수 있도록 하기 위하여 메타 규칙을 적용하여 라우팅 구성 장애를 진단하고 복구하기 위한 추론 알고리즘을 다음과 같이 제시한다.

초기화 과정에서 GoalTable, Blackboard, RuleTable을 생성한다. GoalTable에 첫번째로 실행되는 생성규칙 모듈의 G(Goals)를 추가한다. 목표 항목(GoalEntry)과 RuleTable의 결론부(RuleTable.CC)가 일치하는 규칙이 있는지 찾는다. 만약 여러 개가 있다면 첫번째 규칙을 적용(trigger)한다. 적용된 규칙은 TD상태로 바뀐다. 만약 적용된 규칙의 조건부

```
Initialization :
    Table Blackboard
    Table RuleTable
    Stack GoalTable
    GoalTable.Push(M.G)

while(!GoalTable.Empty())
{
   GoalEntry = GoalTable.Pop();
   while((ID = Scan(RuleTable.CC,GoalEntry)) != NotFound)
   {
      if(TD.PC == TU)
      {
          Blackboard.Add(CC.F);
          if(TD.CC == CHAIN(M))
          {
            M = RuleTable.Transition();
            GoalTable.Push(M.G);
          }
          TD.Fire();
      }
      else if(TD.PC == FA) TD.Discard();
      else if(TD.PC == CHAIN(M))
      {
          RuleTable.Transition(M);
          GoalTable.Push(M.G);
      }
   }
}
```

G = Goals	Scan(CC, GoalEntry) : CC 와 Goal 이 일치하는 active rule 스캔
F = Facts	Add() :테이블에 엔트리 추가
M = Modules	Discard() : rule을 discard 시킴
A = active rule	Transition() : 다른 모듈로 전이
D = discarded rule	Pop() : GoalTable 스택에서 있는 Goal 을 Pop 함
TD = triggered rule	Push() : GoalTable 스택에 Goal 을 Push 함
CC = conclusion clause	Empty() : GoalTable 스택이 비어 있는지 검사
PC = premise clause	Fire() : rule을 fire 시킴
FD = fired rule	
FR = free clause	
FA = false clause	
TU = true clause	

그림 5. 라우팅 구성 장애 관리를 위한 후향 추론 알고리즘

(TD.PC)가 참이라면 Blackboard에 결론부의 사실들(CC.F)을 추가하고 적용된 규칙은 FD(fired rule) 상태로 바뀐다. 만약 적용된 규칙의 결론부에 CHAIN 명령이 있다면 다른 RuleTable로 전이하며 GoalTable에 바뀐 생성규칙 모듈의 G를 추가한다. 만약 적용된 규칙의 조건부가 거짓이라면 적용된 규칙은 폐기(Discard)된다. 만약 적용된 규칙의 조건부에 CHAIN 명령이 있다면 결론부와 마찬가지로 다른 RuleTable로 전이하고 새로운 목표가 GoalTable에 추가된다. 위와 같은 과정이 적용된 A(active rule)가 없을 때 까지 반복된다. 적용된 A가 없다면 새로운 목표가 팝(pop)되고 또 다시 위와 같은 과정이 반복된다. 만약 GoalTable에 G가 모두 팝되고 없다면 프로세스는 종료한다.

Ⅲ. 규칙 기반 라우팅 구성 장애 관리시스템 설계 및 구현

1. 구조

라우팅 구성 장애 관리 시스템(RCFMS)은 관리

자와 대화하기 위한 UI 모듈, 생성 규칙 모듈들을 관리하고 추론하기 위한 IE 모듈, GoalTable, RuleTable, Blackboard를 통합 관리하기 위한 ITM 모듈, 각 생성 규칙 모듈이 필요로 하는 질의 함수를 실행시키고 결과값을 반환하는 QM 모듈, 복구 과정에서 마스터 호스트로부터 네트워크 구성 정보를 얻어오기 위해 필요한 연결 관리를 담당하는 CM 모듈로 구성되어 있으며 각각의 모듈과 세부 모듈들에 대한 자세한 설명은 아래와 같다.

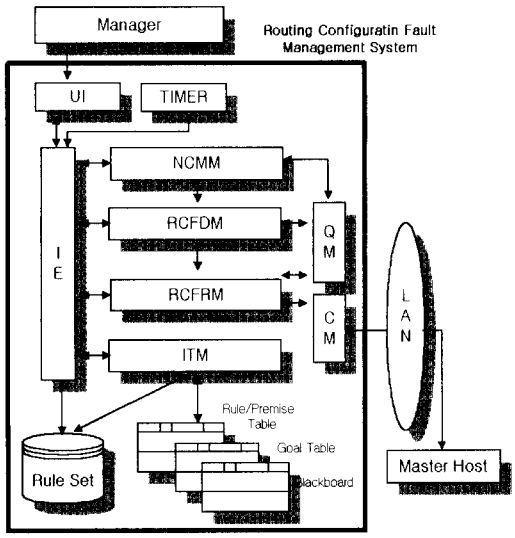

그림 6. 라우팅 구성 장애 관리 시스템 구조도

1) UI(User Interface) : 관리자로부터 사건을 받아들이기 위한 인터페이스이다. 관리자는 UI를 통해서 네트워크 구성 장애에 대해 자문을 요청한다.
2) TIMER : 일정시간 간격으로 대기 상태에 있는 IE 모듈에게 시스템의 라우팅 구성 장애를 진단해줄 것을 요청하기 위한 메시지를 보낸다.
3) IE(Inference Engine) : 관리자의 요청에 의해서 또는 자체 타이머의 동작으로 추론 엔진이 구동하게 된다. 2장에서 설명한 라우팅 구성 장애 관리를 위한 후향 추론 알고리즘에 따라 규칙에 기반하여 추론하는 모듈이다. NCMM, RCFDM, RCFRM 모듈의 실행에 필요한 RuleTable, GoalTable, Blackboard의 정보를 ITM 모듈로부터 얻고 또한 갱신된 정보를 ITM 모듈에게 전달한다.
4) NCMM(Network Configuration Management Module) : 시스템의 TCP/IP 네트워크 구성을 전반적으로 관리하며 2장에서 설명한 네트워크 구성 관리 생성 규칙에 따라 동작한다. 시스템의 운영체제 유형과 장애 유형을 식별하여 만약 라우팅 구성 장애라면 RCFDM 모듈을 호출한다.
5) RCFDM(Routing Configuration Fault Diagnosis Module) : 라우팅 구성 장애를 진단하는 모듈로서 장애가 발생하게 된 원인이 무엇인지를 2장에서 설명한 라우팅 구성 장애 진단 생성 규칙에 따라 식별한다. 식별 후 장애 복구를 위해 RCFRM 모듈을 호출한다.
6) RCFRM(Routing Configuration Fault Recovery Module) : 라우팅 구성 장애를 복구하기 위한 모듈이다. RCFDM 모듈에서 식별한 장애 원인에 기반하여 2장에서 설명한 라우팅 구성 장애 복구 생성 규칙에 따라 복구를 수행한다.
7) ITM(Integrated Table Manager) : GoalTable, RuleTable, Blackboard의 엔트리를 갱신, 추가 및 삭제하는 역할을 수행한다. IE 모듈로부터 갱신된 정보를 얻어온 후 각 테이블을 갱신하고 IE 모듈이 필요로 하는 정보를 제공한다.
8) QM(Query Manager) : NCMM, RCFDM, RCFRM 모듈에 의해 호출되며 각 모듈에서 필요로 하는 질의 함수를 동작시킨 후 결과값을 각 모듈로 전달한다.
9) CM(Communication Manager) : RCFMS와 마스터 호스트 사이의 연결 관리를 위해 호출되는 모듈로서 복구를 위해 RCFRM 모듈에 의해 호출된다.

2. 상태 천이 및 동작

라우팅 구성 장애 관리 시스템은 초기 상태, 대기 상태, 추론 상태, 질의 함수 실행 상태, 결과 출력 상태의 5가지 상태를 가진다. 초기 상태는 서버를 데몬으로서 가동시키고 RuleTable, GoalTable, Blackboard를 초기화 시킨다. 이 상태에 초기화가 정상적으로 완료되면 대기 상태로 가서 타이머 종료, 관리자 입력과 같은 사건이 발생할 때까지 기다린다. 사건이 발생하면 추론 상태가 되며 추론 알고리즘에 따라 추론을 수행한다. 추론 과정에서 생성 규칙의 조건부의 참, 거짓을 식별하기 위해 질의 함수 실행 상태로 간다. 실행 후 결과 값과 함께 추론 상태로 되돌아간다. 추론 과정 중에 밝혀진 사실들

을 관리자에게 통보하기 위해 결과 출력 상태로 간다. 결과 출력이 끝나면 다시 추론 상태로 되돌아온다. GoalTable에 목표 항목이 없다면 추론 과정을 종료하고 대기 상태로 간다. 라우팅 구성 장애 관리 시스템의 상태 천이와 동작은 [표 2]와 [그림 7]에 자세히 나타나있다.

표 2. 상태 천이 표

현재 상태	사건	동작	다음 상태
초기 상태	초기화 완료	RuleTable, GoalTable, Blackboard 생성 및 초기화 수행 후 기존 네트워크 구성 정보를 백업	대기 상태
대기 상태	타이머 종료 관리자 입력	타이머 종료, 관리자 입력과 같은 사건이 발생할 때까지 기다림	추론 상태
추론 상태	질의	추론 과정에서 생성 규칙 조건부의 참, 거짓을 판단하기 위해 질의 함수를 구동시킴	질의 함수 실행 상태
	중간결과보고	추론 과정에 얻어진 사실들을 통보하기 위해 출력 함수 동작시킴	결과 출력 상태
	추론 완료	라우팅 구성 장애를 진단하고 복구하기 위한 추론 과정이 모두 완료	대기 상태
질의 함수 실행 상태	응답	질의 함수 실행 후 결과 값을 통보	추론 상태
결과 출력 상태	출력 완료	추론 과정에서 사실을 출력한 후 추론 상태로 천이	추론 상태

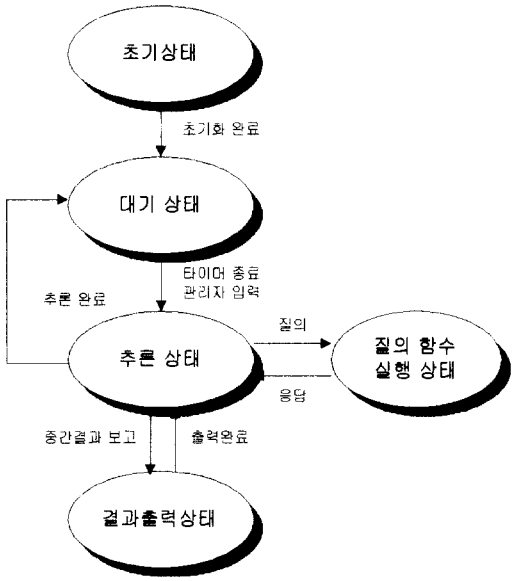

그림 7. 상태 천이도

3. 클래스 계층도

라우팅 구성 장애 관리 시스템 구현에 사용된 클래스는 [그림 8]에 나타나 있다. RCFMS(Routing Configuration Fault Management System) 클래스는 라우팅 구성 장애 관리 시스템의 주 클래스 역할을 하며 초기화 실행뿐 아니라 세 개의 쓰레드 클래스(Timer, UserInterface, ITM)를 호출한다. Timer 클래스는 일정 주기로 IE 쓰레드 클래스를 생성한다. UserInterface 클래스는 관리자로부터의 요청이 왔을 경우 IE 쓰레드 클래스를 호출한다. ITM클래스는 세 개의 관리자 클래스(Blackboard Manager, GoalTableManger, RuleTableManger)를 호출한다. 각각의 클래스는 Blackboard, GoalTable, RuleTable의 갱신, 수정, 삭제 기능을 수행한다. IE 클래스는 추론 엔진으로 ITM 클래스와 상호 연동하여 추론을 수행한다. IE 클래스는 NCMM 클래스를 호출하여 시스템 유형 및 장애 유형을 식별한 후 RCFDM 클래스를 호출한다. RCFDM 클래스를 통해 장애 원인을 식별한 후 IE 클래스는 RCFRM 클래스를 호출하여 장애 복구를 수행한다. NCMM, RCFDM, RCFRM 클래스는 Query Manager, Display 클래스를 통하여 질의에 대한 결과를 얻고 밝혀진 사실에 대한 출력을 수행한다. QueryManager 클래스는 질의에 대한 결과를 얻기 위해 필요한 파일 처리를 위해 FileIO 클래스를 호출하며, 질의 함수 수행을 위해 RunProcess 클래스를 호출한다. RCFRM 클래스는 장애 복구 과정 중에 LAN 세그먼트의 마스터 호스트로부터 네트워크 구성 정보를 얻기 위해 필요한 연결 설정 관리를 수행하는 Communication Manager를 호출한다.

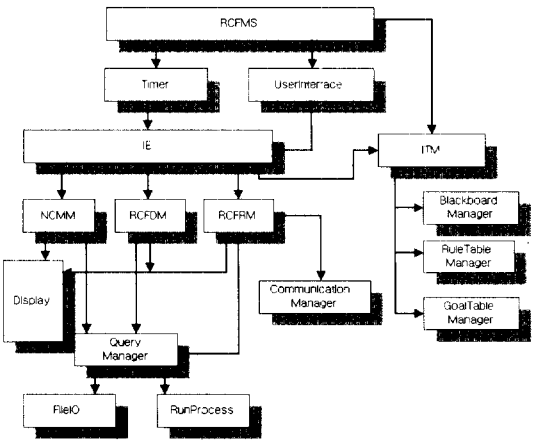

그림 8. 클래스 계층도

Ⅳ. 실험 및 고찰

1. 실험 환경

성균관대학교의 203.252.53.0 네트워크에 연결되어 있는 시스템을 실험 대상으로 하였다.

203.252.53.0 네트워크에는 약 160 여 개의 호스트가 연결되어 있으며 100Mbps Ethernet LAN이다. RCFMS가 탑재되어 있는 피관리 시스템, 203.252.53.0 네트워크 관련 구성 정보를 가지고 있으며 각 호스트의 요구에 응답하는 마스터 호스트, 디폴트 게이트웨이에 대한 자세한 설명은 [표 3]에 나타나 있다.

표 3. 실험 대상 시스템

	IP 주소	시스템	운영체제
디폴트 게이트웨이	203.252.53.1	Xylan omni switch/router	ROM
피관리 시스템	203.252.53.42	Sun Ultra SPARC 1	SunOS release 5.5.1
마스터 호스트	204.253.52.41	Sun SPARCStation-20	SunOS release 5.6

2. 장애 진단 및 복구 과정

1) 시나리오 1 : 인터페이스가 다운된 경우

① 실험 환경 구성 : UP 상태에 있는 이더넷 인터페이스(le0)를 ifconfig le0 down 명령을 사용하여 DOWN 상태로 전환하였다. 전환 후 RCFMS를 구동시키고 장애 진단 및 복구 요청을 하였다.

② 장애 진단 및 복구 과정

표 4. 시나리오 1의 장애 진단 및 복구 과정

단계	규칙 번호	규칙 상태 천이	조건절 번호	조건절 상태	Black board	모듈	Goal Table
0	-	-	-	-	-	M1	g11, g12
1	1	A->TD ->FD	(1)-1	TU	f1	M1	g12
2	3	A->TD ->FD	(2)-1 (2)-2	TU TU	f1, f2	M1 ->M21	g211, g212
3	3	A->TD ->FD	(3)-1	TU	f1, f2, f3	M21	g212
4	13	-	(13)-1	CHAIN	f1, f2, f3	M21 ->M22	g221, g222
5	14	A->TD ->FD	(14)-1 (14)-2	TU TU	f1, f2	M22	g222
6	22	A->TD ->FD	(22)-2	TU	f1	M22	-

2) 시나리오 2 : 케이블 연결상태 불량인 경우

① 실험 환경 구성 : 시스템의 LAN 카드에 연결되어 있는 케이블을 뽑은 후 RCFMS를 구동시키고 장애 진단 및 복구 요청을 하였다.

② 장애 진단 및 복구 과정

표 5. 시나리오 2의 장애 진단 및 복구 과정

단계	규칙 번호	규칙 상태 천이	조건절 번호	조건절 상태	Black board	모듈	Goal Table
0	-	-	-	-	-	M1	g11, g12
1	1	A->TD ->FD	(1)-1	TU	f1	M1	g12
2	2	A->TD ->FD	(2)-1 (2)-2	TU TU	f1, f2	M1 ->M21	g211, g212
3	4	A->TD ->FD	(4)-1	TU	f1, f2, f4	M21	g212
4	13	-	(13)-1	CHAIN	f1, f2, f4	M21 ->M22	g221, g222
5	15	A->TD ->FD	(15)-1 (15)-2	TU TU	f1, f2	M22	g222
6	22	A->TD ->FD	(22)-2	TU	f1	M22	-

3) 시나리오 3 : 디폴트 엔트리가 삭제된 경우

① 실험 환경 구성 : 피관리 시스템의 라우팅 테이블의 송신지 주소가 default또는 0.0.0.0 인 엔트리를 route delete default 203.252.53.1 1 명령을 사용하여 삭제한 후 사용자 인터페이스를 통해 네트워크 구성 장애를 해결해 줄 것을 요구하였다.

② 장애 진단 및 복구 과정

표 6. 시나리오 3의 장애 진단 및 복구 과정

단계	규칙 번호	규칙 상태 천이	조건절 번호	조건절 상태	Black board	모듈	Goal Table
0	-	-	-	-	-	M1	g11, g12
1	1	A->TD ->FD	(1)-1	TU	f1	M1	g12
2	2	A->TD ->FD	(2)-1 (2)-2	TU TU	f1, f2	M1 ->M21	g211, g212
3	10	A->TD ->FD	(10)-1	TU	f1, f2, f8	M21	g212
4	11	A->TD ->FD	(11)-1 (11)-2	TU FA	f1,f2, f8,f9	M21	g212
5	13	-	(13)-1	CHAIN	f1, f2, f8, f9	M21 ->M22	g221, g222
6	19	A->TD ->FD	(19)-1 (19)-2	TU TU	f1, f2	M22	g222
7	22	A->TD ->FD	(22)-2	TU	f1	M22	-

4) 시나리오 4 : 동적 라우팅 상태에서 라우팅 데몬 프로세스 다운

① 테스트 방법 : 정적 라우팅 상태에 있는 시스템을 동적 라우팅 상태로 전환하기 위해 defaultrouter파일을 삭제한 후 동적 라우팅 데몬 프로세스를 구동하기 위한 in.routed와 in.rdisc 파일을 다른 위치에 백업한 후 제거시켰다. 시스템을 재가동 후 RCFMS에게 장애 진단 및 복구를 요청하였다.

② 장애 진단 및 복구 과정

5) 시나리오 5 : 디폴트 게이트웨이가 다운되었거나 주소가 변경된 경우

① 테스트 방법 : 마스터 호스트에서 디폴트 게

표 7. 시나리오 4의 장애 진단 및 복구 과정

단계	규칙 번호	규칙 상태 천이	조건절 번호	조건절 상태	Black board	모듈	Goal Table
0	-	-	-	-	-	M1	g11, g12
1	1	A->TD ->FD	(1)-1	TU	f1	M1	g12
2	2	A->TD ->FD	(2)-1 (2)-2	TU TU	f1, f2	M1 ->M21	g211, g212
3	10	A->TD ->FD	(10)-1	TU	f1, f2, f8	M21	g212
4	12	A->TD ->FD	(12)-1 (12)-2	TU TU	f1, f2 f8, f10	M21	g212
5	13	-	(13)-1	CHAIN	f1, f2, f3	M21 ->M22	g221, g222
6	14	A->TD ->FD	(14)-1 (14)-2	TU TU	f1, f2	M22	g222
7	22	A->TD ->FD	(22)-2	TU	f1	M22	-

이트웨이 IP 주소 요구에 응답할 수 있는 데몬 프로세스를 구동시킨 후 피관리 시스템의 defaultrouter 파일에 있는 IP 주소(203.252.53.1)를 203.252.53.253로 변경한 다음 시스템을 재가동 시켰다. 장애 진단 및 복구를 위해 RDFMS를 구동 시켰다.

② 장애 진단 및 복구 절차

표 8. 시나리오 5의 장애 진단 및 복구 과정

단계	규칙 번호	규칙 상태 천이	조건절 번호	조건절 상태	Black board	모듈	Goal Table
0	-	-	-	-	-	M1	g11, g12
1	1	A->TD ->FD	(1)-1	TU	f1	M1	g12
2	2	A->TD ->FD	(2)-1 (2)-2	TU TU	f1, f2	M1 ->M21	g211, g212
3	7	A->TD ->FD	(7)-1	TU	f1, f2, ¬f5	M21	g212
4	8	A->TD ->FD	(8)-1	TU	f1, f2, ¬f5, ¬f6	M21	g212
5	9	A->TD ->FD	(9)-1 (9)-2 (9)-3 (9)-4	TU TU TU TU	f1, f2, ¬f5, ¬f6 f7	M21	g212
6	13	-	(13)-1	CHAIN	f1, f2, ¬f5, ¬f6 f7	M21 ->M22	g221, g222
7	18	A->TD ->FD	(18)-1 (18)-2	TU TU	f1, f2	M22	g222
8	22	A->TD ->FD	(22)-2	TU	f1	M22	-

3. 실험 결과

위의 시나리오에 따라 [표 9]에서는 Blackboard에 기록되었던 사실들을 바탕으로 실험 결과를 제시하였다. 장애 진단 및 복구 과정 중간에 Blackboard에 기록된 사실들은 장애 원인들을 포함하고 있으며 마지막 단계에 기록된 사실 중에 장애 원인에 대한 사실이 없다면 복구가 정상적으로 완료 되었다는 것을 알 수 있다.

표 9. Blackboard상태에 기반한 실험 결과표

	장애 원인 관련 사실들	마지막 단계	결과
시나리오 1	인터페이스가 다운(f3)	f1	진단 및 복구 완료
시나리오 2	케이블 연결 상태 불량(f4)	f1	진단 및 복구 완료
시나리오 3	디폴트 엔트리 삭제(f9)	f1	진단 및 복구 완료
시나리오 4	라우팅 데몬 프로세스 다운(f10)	f1	진단 및 복구 완료
시나리오 5	디폴트 게이트웨이 다운 또는 주소 변경(f7)	f1	진단 및 복구 완료

V. 결 론

이 논문에서는 시스템에서 발생할 수 있는 라우팅 구성 장애 관리의 중요성을 인지하고, 이질적인 네트워크 환경에 대처할 수 있는 규칙 기반의 라우팅 구성 장애 관리 시스템(RCFMS)을 JAVA를 사용하여 구현하였다. 후향 추론 알고리즘을 기반으로 여러 다른 생성 규칙과의 상호 연동을 위하여 메타 규칙을 적용하였으며 라우팅 구성 장애 관리를 위한 생성 규칙을 세 개의 모듈로 나누었다. 네트워크 구성 관리 생성 규칙 모듈은 시스템의 운영체제 유형과 여러 네트워크 구성 장애 유형을 분류하기 위한 규칙들을 포함하였다. 라우팅 구성 장애 진단 생성 규칙 모듈은 장애가 발생하게 된 원인을 발견하기 위한 규칙들을 포함하였다. 라우팅 구성 장애 복구 생성 규칙 모듈은 장애 원인에 기반하여 복구를 위한 규칙들을 포함하였다.

라우팅 구성 장애 관리 시스템(RCFMS)은 관리자와 대화하기 위한 UI 모듈, 제안한 생성 규칙 모듈을 관리하고 추론하기 위한 IE 모듈, GoalTable, RuleTable, Blackboard를 통합 관리하기 위한 ITM 모듈, 각 생성 규칙 모듈이 필요로 하는 질의 함수를 실행시키고 결과값을 반환하는 QM 모듈, 복구 과정에서 마스터 호스트로부터 네트워크 구성 정보를 얻어오기 위해 필요한 연결관리를 담당하는 CM 모듈로 구성하였다. 다섯 가지의 시나리오에 기반한 규칙, Blackboard, 모듈, Goal의 변화 과정을 보임으로써 실험 결과를 제시하였다.

이 논문에서 제시한 규칙과 알고리즘을 이용한 시스템 개발이 갖는 의미는 크게 세 가지로 나누어 볼 수 있다. 첫 번째로, 시스템에서 발생한 라우팅 구성 장애를 규칙에 기반하여 자동으로 진단하고 복구함으로써 대규모의 네트워크에 존재하는 피관리 시스템에서 발생한 라우팅 구성 장애를 관리자가 직접 관리하는데 드는 시간과 비용을 줄일 수 있다는 것이다. 두 번째로, 다른 운영체제 기반의 장비에 라우팅 구성 장애 관리 시스템을 적용시 프로그램의 절차를 모두 변경할 필요 없이 미리 정의되어

있는 UNIX 기반의 규칙에 다른 운영체제의 규칙, 즉 Windows, Linux에 적용 가능한 규칙을 추가하기만 하면 쉽게 적응 가능하다는 것이다. 세 번째로, 규칙과 후향 추론 알고리즘을 통해 이질적이고 급변하는 네트워크 환경에 쉽게 대처할 수 있는 시스템 개발을 위한 방법론을 제시했다는 것이다.

참 고 문 헌

[1] R. J. Clack, R. Hutchins, S. W. Register, The Role of Human Operators in Configuring, Managing, and Troubleshooting Interconnected Computer Networks, Systems, Man and Cybernetics, Intelligent Systems for the 21s Century, IEEE International Conference, Vol.5, pp.4201-4206, 1995

[2] Show-Way Yeh, Chunan-lin Wu, Hong-Dah Sheng, Expert System Based Automatic Network Fault Management System, Computer Software and Applications Conference, COMPSAC 89, Proceedings of the 13th Annual International, pp.767-774, 1989

[3] 김학준, 망 장애관리 전문가시스템 연구, 숭실대학교 석사학위논문 , 1991

[4] L. Lewis, A Review of Rule-Based Appoaches to Network Management, Technical Note ctron-lml-92-02, 1992

[5] L. Lewis, A Case-Based Reasoning Approach to the Management of Faults in Communications Networks, INFOCOM '93, Proceedings of the 12th Annual Joint Conference of the IEEE Computer and Communications Societies, Vol.3, pp.114-120, 1993

[6] Toshiharu Sugawara, A Cooperative LAN Diagnostic and Observation Expert System, Computers and Communications, Conference Proceeding of the 9th Annual International Phoenix Conference, pp.667-674, 1990

[7] Akashi O, Sugawara T, Murakami K, Takahashi N, Multiagent-based Cooperative Inter-AS Diagnosis in ENCORE, Network Operations and Management Symposium 2000, NOMS 2000, IEEE/IFIP 2000, pp. 521-534, 2000

[8] Melchiors C, Tarouco L.M.R, Trouble Network Faults Using Past Experience, Network Operations and Management Symposium 2000, NOMS 2000, IEEE/IFIP 2000, pp.549-562, 2000

[9] 김순철, 최영수, 정진욱, LAN 세그먼트 관리를 위한 PC 기반의 RMON 에이전트 및 관리자 시스템에 관한 연구, 정보처리학회 추계학술발표논문집, 제6권, 제2호, pp.186-191, 1999

[10] 유승근, 최영수, 정진욱, 네트워크 및 시스템 관리를 위한 웹 기반 통합 관리 시스템의 설계 및 구현, 정보처리학회 추계학술발표논문집, 제6권, 제2호, pp.173-178, 1999

[11] 조강홍, 안성진, 정진욱, 박형우 RMON MIB을 이용한 LAN 성능 및 장애 파라미터 추출에 관한 연구, 정보처리학회 추계학술발표논문집, 제4권, 제2호, pp.943-948, 1997

[12] B.Hayes-Roth, A Blackboard architecture for control, Artificial Intelligence, Vol.26, pp.255-321, 1985

황 태 인(Tae In Hwang) 학생회원

1999년 2월 : 성균관대학교 정보공학과 졸업

1999년 3월~현재 : 성균관대학교 전기전자 및 컴퓨터공학부 석사과정

<주관심 분야> 데이터 통신, 네트워크 관리, 인공지능

안 성 진(Seongjin Ahn) 정회원

1988년 : 성균관대학교 정보공학과 졸업 (학사)

1990년 : 성균관대학교 대학원 정보공학과 졸업 (석사)

1990년~1995년 : 한국전자통신연구원 연구 전산망 개발실 연구원

1996년 : 정보통신 기술사 자격 취득

1998년 : 성균관대학교 대학원 정보공학과 졸업 (박사)

1999년~현재 : 성균관대학교 컴퓨터교육과 전임강사

<주관심 분야> 네트워크 관리, 트래픽 분석, Unix 네트워킹

정 진 욱(Jin Wook Chung) 정회원

1974년 : 성균관대학교 전기공학과 학사
1979년 : 성균관대학교 대학원 전자공학과 석사
1991년 : 서울대학교 대학원 계산통계학과 박사
1982년 ~ 1985년 : 한국과학기술 연구소 실장
1981년 ~ 1982년 : Racal Milgo Co. 객원연구원
1985년 ~ 현재 : 성균관대학교 전기전자 및 컴퓨터공학부 교수

<주관심 분야> 컴퓨터 네트워크, 네트워크 관리, 네트워크 보안