에이전트들 간의 협력을 통한 RBR 기반의 네트워크 구성 장애 관리 알고리즘

조 광 종† · 안 성 진†† · 정 진 욱†††

요 약

본 논문에서는 시스템의 네트워크 구성 장애를 관리하기 위한 관리 모델과 에이전트들 간의 협력을 통한 장애의 진단 및 복구 알고리즘을 제시하고 있다. 관리 모델에는 장애의 검출, 진단, 복구의 세 단계로 이루어지며 각각은 RBR(Rule-Based Reasoning)에 기반으로 하여 규칙기반 지식 데이터베이스에 있는 규칙을 이용하여 네트워크의 구성 장애를 진단하고 복구한다. 또한 관리 도메인 상의 네트워크에 분포하고 있는 여러 에이전트들 간의 협력을 통하여 시스템 단독으로는 해결할 수 없는 복잡한 문제를 해결하거나 네트워크의 상황까지 고려하여 진단하고 복구함으로써 효율적인 시스템의 네트워크 구성 관리 알고리즘을 제시하고 있다.

RBR Based Network Configuration Fault Management Algorithms using Agent Collaboration

Kwang Jong Cho† · Seong Jin Ahn†† · Jin Wook Chung†††

ABSTRACT

This paper proposes fault diagnosis and correction algorithms using agent collaboration, and a management model for managing network configuration faults. This management model is composed of three processes-fault detection, fault diagnosis and fault correction. Each process, based on RBR, operates on using rules which are consisted in Rule-based Knowledge Database. Proposed algorithm solves the complex fault problem that a system could not work out by itself, using agent collaboration. And the algorithm does efficiently diagnose and correct network configuration faults in abnormal network states.

키워드 : 시스템의 네트워크 구성 장애(Network configuration fault of system), 에이전트(agent), RBR, 장애 검출(Fault detection), 장애 진단(Fault diagnosis), 장애 복구(Fault correction), 지식기반 관리(Knowledge-based management), 에이전트 간의 협력(agent collaboration)

1. 서 론

최근 발달된 컴퓨터의 통신 기술과 데이터 통신 기술을 바탕으로 인터넷은 폭발적으로 성장하게 되었다. 또한 인터넷을 구성하고 있는 여러 가지 네트워크 장비와 장비에 연결된 시스템들은 점차 복잡해지고 방대해져 가고 있다. 이러한 상황 속에 시스템 혹은 네트워크에 장애가 발생하였을 때 장애를 관리하기는 무척 어려운 일이다. 현재는 이러한 장애를 해당 분야의 전문가가 직접 관리하거나 전문가의 견해가 축적된 전문가 시스템(Export System)을 통해 관리하고 있다. 그러나 이러한 방법으로는 발생한 장애를 관리하기에는 많은 한계가 있다. 왜냐하면 전문가가 시스템 혹은 네트워크의 모든 상황을 알기란 쉽지 않으며 이러한 전문가의 지식이 들어있는 전문가 시스템 역시 마찬가지이기 때문이다[4, 6].

특히 시스템의 네트워크 구성 장애는 장애 발생 원인을 명확히 판단하기 어렵다. 시스템의 네트워크 구성 관련 장애는 그 장애 발생 원인이 다양하지만 실제 나타나는 장애 증상이 동일한 경우가 많기 때문이다. 예를 들어 시스템에서 웹(WWW) 서비스가 되지 않는다고 하여 IP 혹은 서브넷 마스크(Subnet Mask)와 같은 시스템의 네트워크 구성 정보의 장애라고 단정하기는 어렵다. 이러한 장애는 네트워크 자체의 다운, 시스템의 네트워크 인터페이스 장치의 고장 혹은 네트워크를 구성하고 있는 중간 라우터의 다운 등과 같은 다양한 장애의 원인이 존재할 수 있기 때문이다[3].

이러한 모호한 네트워크 구성 장애를 효과적으로 관리하기 위해서 RBR(Rule-Based Reasoning)을 기반으로 하여 네트워크의 장애 검출, 진단 그리고 복구가 하는 연구가 있다[5]. RBR기반의 네트워크 장애 관리는 네트워크의 장애를 어느 정도 명확하게 진단할 수 있을 뿐만 아니라 복잡한 네트워크 환경에 쉽게 적응할 수 있다는 장점이 있다[8-10]. RBR기반의 네트워크 장애 관리가 적용된 시스템으로는 LODES 시스템이 있다. 그러나 이 시스템은 LAN 상의 장애 검출 및 위치 확인을 위한 규칙 기반 전문가 시스

† 준 회 원 : 성균관대학교 대학원 전기전자및컴퓨터공학부
†† 종신회원 : 성균관대학교 컴퓨터교육과 교수
††† 종신회원 : 성균관대학교 전기전자및컴퓨터공학부 교수
논문접수 : 2001년 12월 11일, 심사완료 : 2002년 6월 25일

템으로 장애 발생에 대한 진단이 가능하지만 시스템 스스로 이러한 장애를 제어하고 복구하지는 못한다[1].

또한 무엇보다도 시스템의 네트워크 구성 장애를 시스템 자체에서 판단하는 것은 많은 한계가 있다. 즉 외부 혹은 내부 네트워크의 상황을 해당 시스템 혼자서 판단하기는 어렵다. 따라서 이러한 장애의 원인을 명확하게 규명하기 위해서는 네트워크의 구성 장애를 관리하기 위한 또 다른 에이전트와의 통신을 통한 협력이 필요하다.

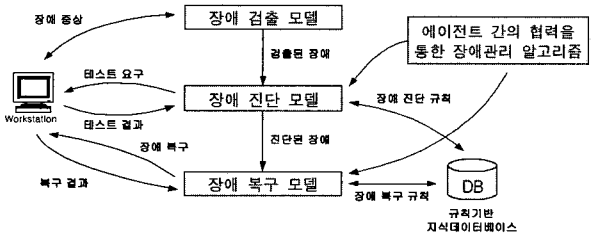
따라서 본 논문에서는 네트워크의 구성 장애를 보다 효과적으로 관리하기 위해 장애의 검출, 진단, 복구의 세 과정으로 나누어 각 부분을 RBR을 기반으로 하여 스스로 진단하고 자동 복구하는 네트워크 구성 장애 관리 방법론을 제시하고 있다 또한 에이전트들 간의 유기적인 협력을 통하여 시스템 자체적으로 파악할 수 없는 내부 혹은 외부네트워크의 상황을 정확히 파악함으로써 정확한 장애의 진단뿐만 아니라 효율적인 장애 복구를 위한 방법을 제공하고 있다.

그리고 본 논문에서는 시스템의 네트워크 구성 장애를 검출하기 위해 기존의 관리 시스템에서 사용하는 수동적인 접근 방법, 즉 SNMP(Simple Network Management Protocol)와 같은 관리 프로토콜을 사용하여 주기적으로 네트워크의 상태를 모니터링하여 시스템 혹은 네트워크의 이상 유무를 판단하는 방법을 사용하지 않고 PING과 같은 관리 도구를 사용하여 능동적으로 네트워크의 장애를 탐지하는 기법을 사용하였다. 능동적인 장애 관리 기법은 수동적인 접근 방법이 가지고 있는 시스템 자원의 낭비를 막고 시스템 스스로 자신의 시스템을 관리할 수 있다는 장점이 있다. 그리고 실제 네트워크 환경에서 장애 사례를 기반으로 하여 실험을 통해 본 논문에서 제시하는 에이전트들 간의 협력을 통한 RBR 기반의 네트워크 구성 장애 관리 방법을 검증하였다[7].

2. 네트워크 구성 장애 관리 모델

2.1 장애 관리 모델

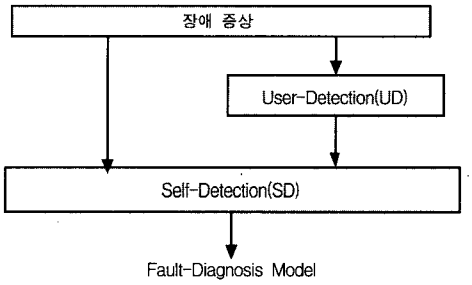
시스템의 네트워크 구성 장애 관리는 크게 장애 검출, 장애 진단, 장애 복구의 3과정으로 이루어진다. 장애 검출 단계에서는 시스템에서 발생한 네트워크 구성 장애를 검출하는 단계로 여러 가지 장애 감지 툴을 이용하여 장애를 검출하는 단계이다. 장애 진단 단계는 검출된 장애 증상을 바탕으로 진단 규칙(Rule)을 적용하여 시스템의 어느 위치에, 어떤 시스템 객체에 장애가 발생하였는지를 판단하는 단계이다. 장애 진단 규칙은 규칙기반 지식 데이터베이스에 저장되어 있으며, 장애 진단 시에 이러한 규칙을 이용한다. 장애의 진단 시에는 에이전트들 간의 협력을 통한 진단이 이루어진다. 이러한 에이전트의 협력을 통한 진단 과정은 기존의 규칙기반 지식 데이터베이스에 들어 있는 진단 규칙과 결합하여 더욱 효율적인 장애 진단을 가능하게 한다. 또한 진단된 결과가 올바른지 시스템에 결과를 테스트 해봄으로써 오판으로 인한 잘못된 장애 진단을 장애 복구 과정이 진행되기 전에 미연에 방지한다. 마지막으로 장애 복구 단계에서는 진단된 장애를 바탕으로 규칙기반 지식 데이터베이스를 검색하여 해당 복구 규칙을 찾는다. 그리고 검색한 복구 규칙을 이용하여 장애를 복구한다. 복구 시에도 역시 에이전트간의 협력을 통한 복구가 이루어지며 장애 복구 후 올바른 장애 복구 과정이 이루어졌는지 확인한다. 그리고 기존의 규칙으로 새로운 사실을 이끌어 내기 위해 후향 추론 알고리즘을 사용한다[2].



(그림 1) 장애 관리 모델

2.2 장애 검출 모델(Fault Detection Model)

장애 검출 모델은 시스템의 네트워크 구성 장애를 검출하는 모델이다. 여기에서 네트워크의 구성 장애는 물리적인 장애(Physical fault)와 논리적 장애(logical fault)로 구분한다. 물리적인 장애는 하드웨어 장애를 말하는 것으로 시스템의 네트워크 인터페이스 카드(Network Interface Card ; NIC)에 케이블이 제대로 꽂혀 있지 않은 경우, NIC 자체의 고장, 네트워크 장비와 시스템을 연결하는 네트워크 회선(line)이 끊어진 경우, 네트워크 자체의 다운 등이 있다. 그리고 논리적인 장애는 소프트웨어 장애를 말하는 것으로 네트워크 인터페이스 카드 드라이버가 제대로 설치되어 있지 않은 경우, NIC를 active 시키지 않은 경우, 네트워크 구성 설정이 잘못된 경우, 라우팅 경로 혹은 데몬 프로세스의 장애 등이 있다.



(그림 2) 장애 검출 모델

장애 검출은 (그림 2)과 같이 시스템 자체에 의한 장애 감지(Self-Detection ; SD) 과정과 시스템 사용자에 의한 장애 검출(User-Detection ; UD) 과정으로 이루어져 있다. SD 과정은 주기적으로 디폴트 게이트웨이(Default Gateway) 혹은 특정 IP로 PING Test를 수행함으로써 시스템의 네트워크 구성에 관한 장애가 발생하였는지를 검출하는 과정이다. 또한 SD 과정에서 수행하는 이러한 PING 테스트는 그 결과

를 분석하여 발생한 장애가 TCP/IP의 IP계층 이하의 장애인지 혹은 응용 계층의 장애인지를 구분하여 준다. 이렇게 하여 검출된 장애는 다음 단계인 장애 진단 모듈로 전달하게 된다. UD 과정은 시스템 사용자가 WEB, TELNET, FTP 등과 같은 응용 서비스를 이용할 때 발생하는 장애를 검출하는 과정이다. 검출된 장애 사실은 장애 진단 모듈로 곧바로 전달되는 것이 아니라 SD 과정을 거침으로써 발생한 장애를 계층으로 구분한 후 장애 진단 모듈로 진행하게 된다.

2.3 장애 진단 모델(Fault Diagnosis Model)

장애 진단 모델은 (그림 3)과 같이 크게 두 부분으로 구분된다. 하나는 에이전트들 간의 협력을 통한 진단으로 네트워크 구성 장애가 발생한 시스템에 위치한 에이전트는 동일 네트워크 혹은 다른 네트워크에 존재하는 다른 에이전트와 두 단계 질의와 응답의 동작으로 장애를 진단한다. 에이전트들 간의 협력을 통한 진단은 본 논문의 3장에서 제시하고 있다. 다른 하나는 항목별 장애 진단으로 에이전트들 간의 협력을 통해서도 해결되지 못한 장애를 진단하는 것이다. 항목별 장애 진단으로는 다음과 같은 것들이 있다.

- 기본 연결 구성 장애 진단(Default Connectivity Configuration Fault Diagnosis, DCCFD)

시스템의 기본적인 연결 설정 상태를 진단하는 모듈로 NIC의 상태, 케이블의 상태, IP주소, 서브넷 마스크, 브로드캐스트(Broadcast) 주소 등의 설정이 올바른지 진단하는 모듈로 [2]의 기본 연결 설정 장애 진단 규칙을 적용하였다.

- 네트워크 환경 장애 진단(Network Environment Fault Diagnosis, NEFD)

시스템에서 검출된 장애가 시스템 자체의 네트워크 구성 장애가 아닌 시스템이 위치한 내부 네트워크 혹은 관리 도메인 상의 다른 네트워크의 장애를 진단하는 모듈이다. 즉 네트워크의 장애인지 시스템의 장애인지를 판단하는 것이다. 그리고 네트워크 환경에 대한 진단된 장애는 시스템 자체적으로는 해결이 불가능하므로 관리자에게 네트워크 환경에 장애가 발생하였다는 사실을 통보하는 것으로 한다. 네트워크 환경 장애 진단 후 다시 한번 PING 테스트를 통해 그 결과를 기준으로 다음 단계로 진행하게 된다.

- 라우팅 구성 장애 진단(Routing Configuration Fault Diagnosis, RCFD)

시스템이 라우팅 기능을 하고 있을 때, 정적 라우팅 혹은 동적 라우팅 시 나타나는 장애를 진단하는 모듈이다. 주로 디폴트 라우트, 라우팅 테이블, 라우팅 데몬 프로세스에 대한 장애를 진단하는 모듈로 [2]의 라우팅 구성 장애 진단 규칙을 적용하였다.

- 네임 서비스 구성 장애 진단(Name Service Configuration Fault Diagnosis, NSCFD)

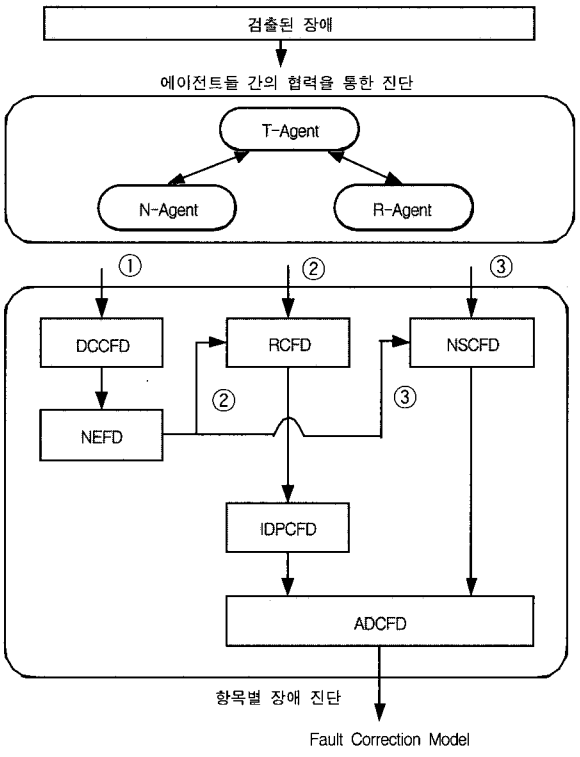
네임 서비스에 대한 구성 장애를 진단하는 모듈로 시스템이 DNS로 동작하고 있을 때와 단지 DNS를 이용하는 Resolver로 동작하고 있을 때의 구성 장애를 진단한다. [2]의 네임 서비스 구성 장애 진단규칙을 적용하였다.

- 인터넷 데몬 프로세스 구성 장애 진단(Internet Demon Process Configuration Fault Diagnosis, IDPCFD)

인터넷 데몬 프로세스 즉 inetd 프로세스에 대한 장애를 진단하는 모듈로 [2]의 인터넷 데몬 프로세스 구성 장애 진단 규칙을 적용하였다.

- 어플리케이션 데몬 구성 장애 진단(Application Demon Configuration Fault Diagnosis, ADCFD)

TCP/IP관련 어플리케이션 서비스를 제공하는 각 데몬 프로세스에 대한 장애를 진단하는 모듈이다. 관련 대몬들은 텔넷 데몬(telnetd), 파일 전송 데몬(ftpd), HTTP 데몬(httpd), 메일 데몬(smtpd, imapd, pop3d) 등이다[12].



(그림 3) 장애 진단 모델

그리고 에이전트들 간의 협력을 통한 진단 후 PING 테스트를 통하여 <표 1>과 같은 결과를 바탕으로 항목별 장애 진단의 각 모듈로 진행하게 된다[11].

<표 1> PING 테스트 결과와 그 표기

PING 테스트 결과	표기
"No answer", "connection timed out", "cannot connect"	①
"Network unreachable"	②
"Unknown host"	③

2.4 장애 복구 모델(Fault Correction Model)

장애 복구 모델 역시 장애 진단 모델과 같이 두 부분으로 구성되어 있다. (그림 4)에서 보듯이 에이전트들 간의 협력을 통한 복구와 항목별 장애 진단 단계에 대응하는 항목별 장애 복구가 그것이다. 에이전트들 간의 협력을 통한 복구과정은 3장에서 제시하고 있으며 항목별 장애 복구는 다음과 같다.

- 기본 연결 구성 장애 복구(Default Connectivity Configuration Fault Correction, DCCFC)

기본 연결 구성 장애 진단모듈에서 진단된 결과에 대응하는 복구 규칙을 적용하여 장애를 복구하는 모듈이다. [2]의 기본 연결 설정 장애 복구 규칙을 적용하였다.

- 라우팅 구성 장애 복구(Routing Configuration Fault Correction, RCFC)

라우팅 구성 장애 진단에 대응하는 복구 모델이다. [2]의 라우팅 구성 장애 복구 규칙을 적용하였다.

- 네임 서비스 구성 장애 복구(Name Service Configuration Fault Correction, NSCFC)

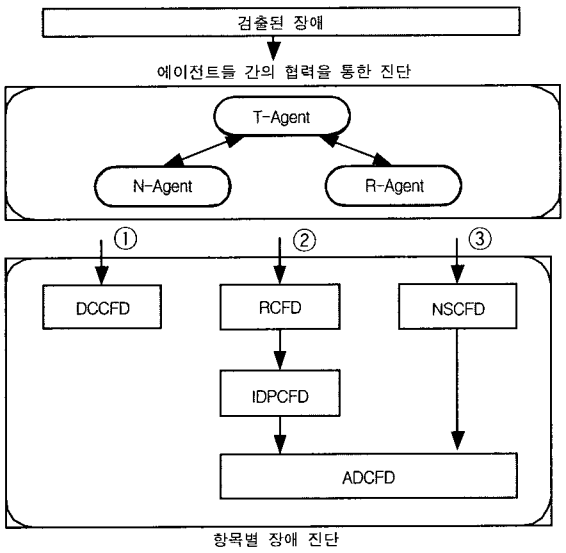
네임 서비스 구성 장애 진단에 대응하는 복구 모델이다. [2]의 네임 서비스 구성 장애 진단규칙을 적용하였다.

- 인터넷 데몬 프로세스 구성 장애 복구(Internet Demon Process Configuration Fault Correction, IDPCFC)

인터넷 데몬 프로세스 구성 장애에 대한 복구 모델로 [2]의 인터넷 데몬 프로세스 구성 장애 진단 규칙을 적용하였다.

- 어플리케이션 데몬 구성 장애 복구(Application Demon Configuration Fault Correction, ADCFC)

각 어플리케이션 데몬에 대한 복구 과정이다.



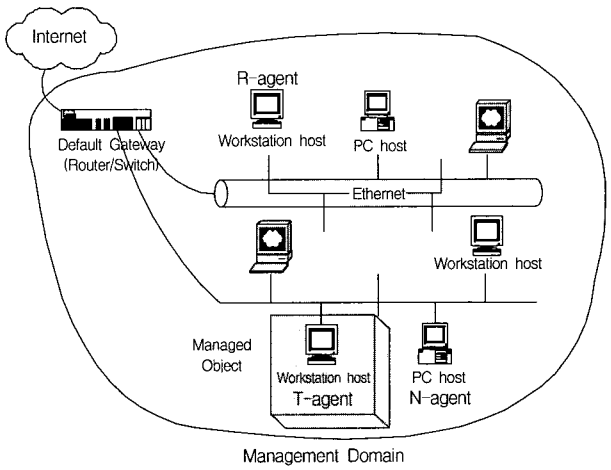
(그림 4) 장애 복구 모델

3. 에이전트 간의 협력

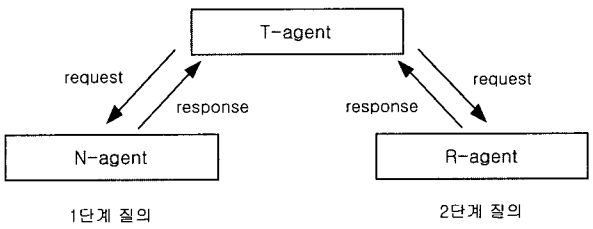
3.1 에이전트의 구성

네트워크 구성 장애를 효과적으로 관리하기 위해세 가지의 에이전트를 고안하였다. 네트워크에서의 위치에 따라 3종류의 에이전트로 구분된다. 장애를 관리하고자 하는 시스템 자체에 설치되는 T-Agent(This-Agent), T-Agent와 같은 내부 네트워크 상에 존재하는 N-Agent(Neighbor-Agent) 그리고 T-Agent와 다른 네트워크에 있는 R-Agent(Remote-Agent)가 있다. 에이전트들은 모두 같은 관리 도메인(Management Domain) 상에 존재하며 에이전트 상호 간은 브로드캐스트 혹은 UDP를 이용하여 서로 통신한다. (그림 5)는 관리 도메인 상에 관리하고자 하는 관리 객체(Managed Object)에 설치된 T-Agent를 비롯한 네트워크 구성 장애 관리를 위한 에이전트들을 네트워크 상에 보여주고 있다.

그리고 관리 도메인에 설치된 각 에이전트들은 통신을 통하여 상호 협력함으로써 더욱 효과적으로 네트워크 장애를 진단하고 복구한다. 협력시 T-Agent는 질의와 응답의 동작으로 N-Agent와 R-Agent로부터 네트워크 진단 및 복구를 위한 정보를 얻는다. 단순하게는 N-Agent와 R-Agent의 네트워크 설정에 관한 정보를 얻거나, 나아가서는 네트워크에 연결된 다른 시스템(주로 디폴트 게이트웨이)으로의 PING 테스터를 요구하며, PING 테스트의 결과 정보를 에이전트로부터 얻는다.



(그림 5) 에이전트의 구성



(그림 6) 에이전트들간의 관계

여기서 (그림 6)에서와 같이 T-Agent가 N-Agent에게 질의하는 단계를 1단계 질의라 하며 T-Agent가 R-Agent에게 질의하는 단계를 2단계 질의라고 정의한다. 보통 네트워크 구성 장애의 진단 및 복구과정은 1단계 질의와 2단계 질의 조합으로 이루어진다.

3.2 에이전트의 상태

각 에이전트들은 디폴트 게이트웨이로의 PING 테스트 후 그 결과에 따라 다른 상태로 천이하게 된다. <표 2>는 PING 테스트 결과에 따른 에이전트의 상태와 천이를 위한 이벤트를 표시하고 있다.

T-Agent, N-Agent 그리고 R-Agent의 상태를 표시하고 있다.

〈표 2〉 에이전트의 상태 정의

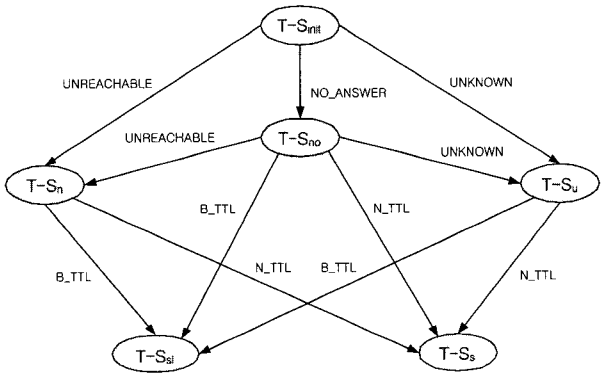
에이전트의 상태	상태 설명(PING 테스트 결과)	이벤트 표기
$S_{init,}$	에이전트의 초기 상태	-
S_s	TTL 값이 정상적일 경우	N_TTL
S_{sl}	TTL 값이 아주 큰 경우	B_TTL
S_{no}	"No answer", "connection timed out", "cannot connect"	NO_ANSWER
S_u	"Unknown host"	UNKNOWN
S_n	"Network unreachable"	UNREACHABLE

(TTL = Time To Live)

또한 <표 3>은 이벤트에 따른 각 에이전트의 상태를 표기하고 있다. 각 에이전트의 상태 표기에서 T로 시작하는 것은 T-Agent의 상태를 말하는 것이며, N과 R로 시작하는 것은 각각 N-Agent와 R-Agent의 상태를 말하는 것이다.

〈표 3〉 각 에이전트의 상태 표기

이 벤 트	각 에이전트의 상태 표기
초기상태(init)	$T\text{-}S_{init,}$, $N\text{-}S_{init,}$, $R\text{-}S_{init,}$
N_TTL	$T\text{-}S_{s,}$, $N\text{-}S_{s,}$, $R\text{-}S_{s,}$
B_TTL	$T\text{-}S_{sl}$, $N\text{-}S_{sl}$, $R\text{-}S_{sl}$
NO_ANSWER	$T\text{-}S_{no}$, $N\text{-}S_{no}$, $R\text{-}S_{no}$
UNKNOWN	$T\text{-}S_u$, $N\text{-}S_u$, $R\text{-}S_u$
UNREACHABLE	$T\text{-}S_n$, $N\text{-}S_n$, $R\text{-}S_n$



(그림 7) T-agent의 상태 천이도

그리고 (그림 7)은 T-Agent의 상태 천이를 도식화한 것이다. 상태의 천이 과정 역시 PING 테스트 결과에 의해 결정된다. 초기 상태는 모든 상태가 가능하며 최종 상태로는 $T\text{-}S_{sl}$, $T\text{-}S_s$상태가 가능하다.

3.3 에이전트들 간의 협력 알고리즘

시스템의 네트워크 구성 장애가 검출되면 T-Agent는 N-Agent 혹은 R-Agent와 서로 통신을 하여 협력함으로써 장애 관리에 필요한 유용한 정보를 추출해 내거나 추출한 장애 관리 정보를 바탕으로 발생한 장애를 진단 및 복구한다. 본 논문에서는 장애의 진단 및 복구를 에이전트들 간의 협력 알고리즘을 (그림 8)~(그림 11)과 같이 제시하고 있다. 본 논문의 알고리즘에서 관리 객체에 설치된 T-Agent, N-Agent, R-Agent 각각은 IP가 설정된 상태임을 전제 조건으로 한다. 여기서 T-Agent가 N-Agent와 R-Agent에게 PING 테스트 요청을 하였을 때 네트워크 혹은 시스템의 다운으로 인해 T-Agent가 아무런 응답을 받지 못할 때의 T-Agent의 상태를 $T\text{-}S_{nr}$상태로 정의한다.

먼저 (그림 8)은 에이전트들 간의 초기 협력 알고리즘으로 T-agent가 초기에 구동될 때 동작하는 알고리즘으로 N-agent로부터 네트워크의 구성정보를 얻거나, PING 테스트를 하여 그 결과에 따라 각 세부모듈로 분기하게 한다.

Handler_main

```
If ( Network Configuration fault detected ) then
{
    T-Agent request N-Agent's NCI ;
    If ( N-Agent's response non-exist ) then
        T-Agent's interior network is down State ;
    Else {
        T-Agent get N-Agent's NCI ;
        If ( N-Agent's Collision rate is high ) then
        {
            T-Agent request Other N-Agent's Collision rate ;
            If ( Other N-Agent's Collision rate is high ) then
                Inform network manager that this network is saturation
                state ;
        }
    }

    T-Agent execute D-PING Test ;
    If ( T-Agent's State == T-Sno ) then
        Handler_T-Sno ;
    Else if ( T-Agent's State == T-Su ) then
        Handler_T-Su ;
        Else if ( T-Agent's State == T-Ssl ) then
        Handler_T-Ssl ;

    T-Agent execute I-PING Test ;
    If ( T-Agent's State == T-Sn ) then
        Inform network manager that DG's routing Error ;

}
```

NCI : Network Configuration Information(Subnet Mask, Broadcast Address, Default Gateway(DG), DNS Address, etc)
Collision rate = collision/opkts
D-PING Test : PING Test to Default gateway
I-PING Test : PING Test to specific system connected Internet

(그림 8) 에이전트들 간의 초기 협력 알고리즘

다음으로 (그림 9)은 "No answer" 관리를 위한 협력 알고리즘으로 에이전트간의 초기 협력 알고리즘에서 PING 테스트 결과 "No answer" 결과가 나왔을 때 동작하는 알고리즘으로 N-agent로 PING 테스트를 요구하여 그 결과를 바탕으로 시스템의 네트워크 구성 장애를 관리한다. 또한 이 알고리즘에서는 위에서 설명한 1 단계 질의뿐만 아니라 2 단계 질의를 사용함으로써 에이전트들간의 유기적인 협력을 이용하여 장애를 관리하고 있다.

Handler_T - S_{no}

```
T-Agent requests N-Agent D-PING Test ;
If ( N-Agent's State == N - Ss ) then
{
   If ( T-Agent's DGA != N - Agent's DGA ) then
      T-Agent's DGA = N-Agent's DGA ;
   T-Agent executes D-PING Test ;
   If ( T-Agent's State == T - Ssl or T - Ss ) then
      Fault was corrected ;
}
Else if ( N-Agent's State = N - Sno ) then
{
   T-Agent requests R-Agent D-PING Test ;
   If ( R-Agent's State == R - Ss, R - Ssl ) then
      T-Agent's DGA = R - Agent's DGA ;
      T-Agent execute D-PING Test ;
      If ( T-Agent's State == T - Ss, T - Ssl ) then
         Fault was corrected ;
      Else ( T-Agent's State == T - Sno )
         Interfaces from DG to T-Agent interior network have
            fault ;
   If ( R-Agent's State == R - Sno ) then
      T-Agent request R-Agent T-PING Test ;
      If ( R-Agent's State == R - Sno ) then
         T-Agent or R-Agent interior network was down ;
      Else if ( R-Agent's State == R - Ss, R - Ssl )
         DG Error ;
}
Else if ( N-Agent's State = N - Snr )
   T-Agent interior network was down ;
```

DGA : Default Gateway Address
T-PING Test : PING Test to T-Agent

(그림 9) "No answer" 관리를 위한 협력 알고리즘

Handler_T-S_u

```
T-Agent requests N-Agent D-PING Test ;
If ( N-Agent's State == N-Ss ) then
{
   If ( T-Agent's DNSA != N-Agent's DNSA ) then
      T-Agent's DNSA = N-Agent's DNSA ;
   T-Agent executes D-PING Test ;
   If ( T-Agent's State == T-Ssl or T-Ss ) then
      Fault was corrected ;
}
Else if ( N-Agent's State = N-Su ) then
{
   T-Agent requests R-Agent D-PING Test ;
   If ( R-Agent's State == R-Ss, R-Ssl ) then
      T-Agent's DNSA = R-Agent's DNSA ;
      T-Agent execute D-PING Test ;
      If ( T-Agent's State == T-Ss, T-SsL ) then
         Fault was corrected ;
   Else if ( R-Agent's State == R-Su ) then
      DNS Error ;
}
```

DNSA : DNS Address

(그림 10) "Unknown host" 관리를 위한 협력 알고리즘

(그림 10)은 "Unknown host" 관리를 위한 협력 알고리즘으로 에이전트간의 초기 협력 알고리즘에서 PING 테스트 결과 "Unknown host" 결과가 나왔을 때 동작하는 알고리즘으로 N-agent로 PING 테스트를 요구하여 그 결과를 바탕으로 시스템의 네트워크 구성 장애를 관리한다.

마지막으로 (그림 11)은 "Big TTL" 관리를 위한 협력 알고리즘으로 에이전트간의 초기 협력 알고리즘에서 PING 테스트 결과 TTL 값이 클 경우 N-agent로 PING 테스트를 요구하여 그 결과를 바탕으로 장애를 관리한다.

Handler_T-S_{sl}

```
T-Agent requests N-Agent D-PING Test ;
If ( N-Agent's State == N-Ssl ) then
{
  Inform network manager that this network is saturation state ;
  This network was needed to divide into sub-networks ;
}
```

(그림 11) "Big TTL" 관리를 위한 협력 알고리즘

4 실험 및 고찰

4.1 실험 환경

본 논문에서 제시하고 있는 시스템의 네트워크 구성 장애 관리를 위한 에이전트간의 협력을 통한 장애 관리 알고리즘의 타당성을 평가하기 위해 성균관대학교의 네트워크를 관리 도메인으로 하여 실험을 하였다. 디폴트 게이트웨이와 T-Agent, N-Agent, R-Agent가 설치될 각 시스템들에 관한 설명은 <표 4>과 같다. 여기에서 시스템 1, 시스템 2, 시스템 3은 같은 내부 네트워크(LAN)에 연결된 상태이며 시스템 4는 관리 도메인상의 다른 네트워크에 연결되어 있는 시스템이다.

<표 4> 실험 대상

시 스 템	IP 주소	모 델	OS
디폴트 게이트웨이	203.252.53.1	Xylan omni Switch	ROM
시스템 1(T-Agent)	203.253.53.41	Ultra SPARC 60	Solaris 2.7
시스템 2(N-Agent)	203.252.53.45	PC PIII933	wowlinux 7.1
시스템 3(N-Agent)	203.252.53.42	Ultra SPARC 1	Solaris 2.6
시스템 4(R-Agent)	203.252.45.161	Ultra SPARC 2	Solaris 2.6

4.2 장애 시나리오에 따른 실험 결과

아래의 네 가지 시나리오는 에이전트간의 협력을 통한 장애 관리를 테스트하기 위한 시나리오이다.

(1) 시나리오 1 : 시스템 1의 네트워크 구성 정보 중 디폴트 게이트웨이 정보를 삭제하였다. 그 후 시스템 1에서 에이전트들 간의 협력을 통한 장애 관리 알고리즘을 적용하였다. 이때 시스템 2, 시스템 3, 시스템 4는 정상적으로 동작하고 있는 상태이다. 아래의 <표 5>는 시나리오 1에 관한 협력을 통한 장애 관리 알고리즘을 적용했을 때의 각 에이전트의 상태와 중간 과정을 나타내고 있다.

〈표 5〉 시나리오 1의 장애 진단 및 복구 결과

단 계	T-Agent의 상태	N-Agent의 상태	R-Agent의 상태	수행한 PING TEST(or Broadcast)	수행 모듈	결과
1	-	-	-	Broadcast	Main_ Handler	T-Agent get N-Agent's NCI ;
2	$T\text{-}S_{no}$	-	-	D-PING	Main_ Handler	-
3	$T\text{-}S_{no}$	$N\text{-}S_n$	-	D-PING	Handler _$T\text{-}S_{no}$	T-Agent's DNSA = N-Agent's DNSA
4	$T\text{-}S_s$	$N\text{-}S_n$	-	D-PING	Handler _$T\text{-}S_{no}$	Fault was corrected ;

(2) 시나리오 2 : 시스템 1, 시스템 2, 시스템 3이 연결된 LAN 허브의 전원을 꺼 놓음으로써 내부 네트워크를 다운시켰다. 그리고 T-Agent에서 에이전트의 협력을 통한 장애 관리 알고리즘을 적용하였다.

〈표 6〉 시나리오 2의 장애 진단 및 복구 결과

단 계	T-Agent의 상태	N-Agent의 상태	R-Agent의 상태	수행한 PING TEST(or Broadcast)	수행 모듈	결과
1	-	-	-	Broadcast	Main_ Handler	T-Agent's interior network is down State ;

(3) 시나리오 3 : 동일 LAN 네트워크에 존재하는 시스템 1, 시스템 2, 시스템 3의 DNS 주소 설정을 모두 삭제하였다. R-Agent가 설치되어 있는 시스템 4는 정상적으로 동작하고 있는 상태이다.

〈표 7〉 시나리오 3의 장애 진단 및 복구 결과

단 계	T-Agent의 상태	N-Agent의 상태	R-Agent의 상태	수행한 PING TEST(or Broadcast)	수행 모듈	결 과
1	-	-	-	Broadcast	Main_ Handler	T-Agent get N-Agent's NCI ;
2	$T\text{-}S_u$	-	-	D-PING	Main_ Handler	-
3	$T\text{-}S_u$	$N\text{-}S_u$	-	D-PING	Handler _$T\text{-}S_u$	-
4	$T\text{-}S_u$	$N\text{-}S_u$	$R\text{-}S_s$	D-PING	Handler _$T\text{-}S_u$	-
5	$T\text{-}S_a$	$N\text{-}S_u$	$R\text{-}S_s$	D-PING	Handler _$T\text{-}S_u$	Fault was corrected ;

(4) 시나리오 4 : 시스템 1, 시스템 2, 시스템 3이 연결된 내부 네트워크에 UDP 트래픽 발생기를 이용하여 네트워크 트래픽을 계속 발생시켰다.

〈표 8〉 시나리오 4의 장애 진단 및 복구 결과

단 계	T-Agent의 상태	N-Agent의 상태	R-Agent의 상태	수행한 PING TEST(or Broadcast)	수행 모듈	결 과
1	-	-	-	Broadcast	Main_ Handler	T-Agent get N-Agent's NCI ;
2	-	-	-	-	Main_ Handler	Inform network manager that this network is saturation state ;

4.3 시나리오별의 장애 진단 및 복구 결과 정리

본 논문의 실험에서는 에이전트간의 협력을 통한 네트워크의 구성 장애 관리를 실험 환경에서 테스트하였다. 네 가지의 시나리오들 중 시나리오 1과 시나리오 3은 시스템 스스로 진단 및 복구가 정상적으로 완료되었고 시나리오 2와 시나리오 4는 시스템 스스로 진단은 정상적으로 완료되었으나, 그 진단 내용이 시스템이 아닌 시스템이 속해 있는 네트워크의 현재 상태와 연관된 것이므로 시스템 혹은 네트워크에게 진단 내용을 통보하고 있다.

〈표 9〉 시나리오별의 장애 진단 및 복구 결과

시나리오	장애 내용	장애 관리 결과
시나리오 1	시스템의 디폴트 게이트웨이 정보 삭제	장애 진단 및 복구 완료
시나리오 2	내부 LAN 네트워크의 다운	관리자에게 진단 내용 통보
시나리오 3	동일 LAN 네트워크에 있는 전체 시스템의 DNS 정보 삭제	장애 진단 및 복구 완료
시나리오 4	내부 LAN 네트워크의 과부하	관리자에게 진단 내용 통보

관리자는 이러한 통보받은 진단 정보를 이용해 장애를 확인하고 복구할 수 있다. 시나리오 2와 같이 내부 네트워크가 다운된 상태는 네트워크 케이블, 허브 혹은 스위치와 같은 네트워크 장비 자체를 테스트 해봄으로써 장애를 확인하여, 이를 복구할 수 있다. 그리고 내부 네트워크의 과부하로 진단된 시나리오 4는 내부 네트워크를 작은 몇 개의 세그먼트로 분리하거나 내부 네트워크에 연결되어 있는 시스템의 수를 조정하여 발생한 장애를 복구할 수 있다. 또한 시나리오 4의 경우 내부 네트워크에 연결된 시스템의 특정 응용 프로세스가 발생시키는 트래픽의 양을 점검해봄으로써 이를 해결할 수도 있다.

5. 결 론

본 논문에서는 시스템의 네트워크 구성 장애를 에이전트들 간의 협력을 통하여 진단, 복구하는 알고리즘을 제시하였다. 에이전트들은 네트워크에서의 위치에 따라 T-Agent, N-Agent, R-Agent로 구분되며 질의와 응답의 형태로 협

력하는 과정을 통하여 시스템에서 발생하는 네트워크 구성 장애를 진단하고 복구한다. 또한 RBR 기반의 네트워크 구성 장애에 대한 관리 모델을 제시하였다. 이러한 관리 모델은 장애의 검출, 진단, 복구의 세 가지 세부 모델로 구분되며 진단과 복구 시에는 위에서 언급한 에이전트들 간의 협력을 통한 진단 및 복구 과정과 복합적으로 이루어지고 있다. 그리고 에이전트들 간의 협력을 통한 장애 관리 과정에 의해 발생한 장애는 네트워크 관리자에 의해서가 아닌 자동적으로 진단 및 복구가 이루어진다.

본 논문에서 제시하고 있는 협력을 통한 시스템의 네트워크 구성 장애 관리 알고리즘은 시스템의 네트워크 구성 장애를 기존의 시스템 자체에서 그 원인을 규명할 수 없는 문제들을 주위 시스템과의 협력은 물론, 네트워크 환경이나 상황과 결부시켜 발생한 장애를 진단하고 복구하는 알고리즘을 제시하고 있다는 점에서 큰 의미를 부여할 수 있다. 또한 RBR 기반의 항목별 장애의 진단 및 복구 과정은 규칙기반의 지식 데이터베이스를 이용하여 추론 기법을 적용하여 장애를 관리함으로써 네트워크의 환경이 변화하거나 네트워크에 다른 새로운 프로토콜 혹은 기술이 접목되었을 때 능동적으로 대처할 수 있다.

그리고 기존의 시스템 혹은 네트워크 관리 툴들이 지향하는 수동적인 관리 방법에서 벗어나 시스템 스스로 시스템의 상황이나 네트워크의 상황을 PING과 같은 관리 도구를 사용하여 관리함으로써 스스로 진단 및 복구를 할 수 있을 뿐 아니라 RMON, SNMP 등의 부가적인 시스템 혹은 프로토콜을 필요로 하지 않는 능동적인 관리 방법을 제시하고 있다.

참 고 문 헌

[1] 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.

[2] Tae In Hwang, Seongjin Ahn, Jin Wook Chung, A Study on the Rules and Algorithm for the Diagnosis and Recovery of Routing Configuration, SCI 2000, World Multiconference on Systemics, Cybernetics and Informatics, Vol.4, pp.137-141, 2000.

[3] Roy Sterritt, Discovering Rules for Fault Management, Engineering of Computer Based System, Proceedings. Eighth Annual IEEE International Conference an Workshop, pp. 190-196, 2001.

[4] Wesley W. Chu, System Management Research via Behavior Characterization, Proceedings of the IEEE First International Workshop on, pp.1-6, 1993.

[5] Denise W.Gurer, Irfan Khan, Richard Ogier, Renee Keffer, An Artificial Intelligence Approach to Network Fault Management, SRI International, Menlo Park, California, USA.

[6] Kohei OHTA, Takumi MORI, Nei KATO, Hideaki SONE, Glenn MANSFIELD, Yoshiaki NEMOTO, Divide and Conguer Technique for Network Fault Management, Proceedings of ISINM97, 1997.

[7] Ewerton L. Madruga, Lianc M. R. Tarouco, Fault Management Tools for a Cooperative and Decentralized Network Operations Environment, IEEE Journal on selected areas in communications, Vol.12, No.6, 1994.

[8] 조규억, 안성진, 정진욱, RBR을 이용한 지연 장애 검출 및 진단알고리즘에 관한 연구, 정보처리학회논문지, 제7권 제8호, pp.2620-2630, 2000.

[9] 조강홍, 안성진, 정진욱, LAN 상의 장애 검출 및 위치 확인을 위한 규칙 기반 장애 진단 에이전트 시스템, 정보처리학회논문지, 제7권 제7호, pp.2169-2178, 2000.

[10] 윤정미, 안성진, 정진욱, INBANCA기법을 이용한 웹 서버 장애 진단 및 복구기법, 정보처리학회논문지, 제7권 제8호, pp.2497-2504, 2000.

[11] Craig Hunt, TCP/IP Network Administration, O'Relly & Associates, 1998.

[12] Stevens, TCP/IP Illustrated, Vol.1, pp.66-96, 1998.

조 광 종

e-mail : kjcho@songgang.skku.ac.kr

2000년 성균관대학교 전기전자 및 컴퓨터공학부 졸업 (학사)

2000년~현재 성균관대학교 전기전자및컴퓨터공학부 대학원 석사과정

관심분야 : 네트워크 관리, 시스템 장애 관리, 인공지능, GRID

안 성 진

e-mail : sjahn@comedu.skku.ac.kr

1988년 성균관대학교 정보공학과 졸업 (학사)

1990년 성균관대학교 대학원 정보공학과 졸업(석사)

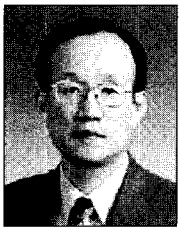
1990년~1995년 한국전자통신연구원 연구전산망 개발실 연구원

1996년 정보통신 기술사 자격 취득

1998년 성균관대학교 대학원 정보공학과 졸업(박사)

1999년~현재 성균관대학교 컴퓨터교육과 조교수

관심분야 : 네트워크 관리, 트래픽 분석, Unix 네트워킹



정 진 욱

e-mail : jwchung@songgang.skku.ac.kr

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 전기전자및컴퓨터공학부 교수

관심분야 : 컴퓨터 네트워크, 네트워크 관리, 네트워크 보안