생존성 향상을 위한 네트워크 관리 시스템의 구조
(Network Management System Architecture for Enhanced Network Survivability)

이 중 수 † 조 평 동 †† 김 남 훈 † 이 영 희 †††
(Joongsoo Lee) (Pyung-Dong Cho) (Namhoon Kim) (Young Hee Lee)

요 약 네트워크 관리 시스템은 네트워크를 모니터링하고 감독하여 잘 유지하고 활용성을 높이는 것을 목적으로 하는 시스템이다. 네트워크 관리 시스템 자체의 생존성이 보장되지 않으면, 네트워크를 안정한 상태로 유지하는 것은 매우 어렵다. 본 논문에서는 네트워크 관리 시스템 자체의 생존성을 높일 수 있도록 매니저들 간의 세션을 형성하고 SNA(Script of Neighbor's Agent)를 통해 고장이 발생한 매니저의 관리 작업을 세션 내의 다른 매니저들이 대행해 주도록 하여, 네트워크가 항상 관리될 수 있도록 하는 방법을 제안하였다. 제안한 방법은 하나의 관리 스테이션에 집중되던 부하를 하위의 매니저들에게 분산시키고, SNMP 메시지를 전송하기 위해 백본을 통과하는 트래픽을 줄일 수 있다. 제안한 방법의 우수성을 보이기 위해 노드의 오류에 관한 생존성 함수를 도출하였고, 그에 따른 분석 결과를 도시하였다.

Abstract Network management system is a system that aims at monitoring and configuring network elements in order to maintain the network reliable and to utilize highly. To keep the network in reliable state the survivability of network management system is indispensable. This paper proposes robust network management system architecture that supports the networks to always be in managed state. When one of managers in the session fails, the other managers in that session take a portion of the failed manager's job using SNA (Script of Neighbor's Agents). This architecture is very effective to increase survivability of the manager. Further more the load of central management station and the backbone traffic that is used in transferring SNMP message can be decreased in this architecture. To verify the survivability characteristics of the proposed method, we introduce the survivability function of manager faults and describe the analysis results.

1. 서 론

컴퓨터와 컴퓨터 네트워크가 실생활에서의 비중이 점점 커짐에 따라 네트워크에 발생한 고장이나 문제가 장시간 지속되는 경우 사회와 기업에 경제적으로 심각한 영향을 미치게 된다. 이러한 이유 때문에 많은 노력을 들여 네트워크 인프라의 운용 및 유지 보수와 관리에 많은 노력을 기울여 왔다. 이러한 네트워크와 관련된 유지, 보수, 환경설정 등의 일련의 활동을 네트워크 관리라 한다.

IAB(Internet Architecture Board)에서는 네트워크 관리 시스템(Network Management System)을 지원하기 위해 여러 가지 표준을 정하게 되었는데, SNMP (Simple Network Management Protocol)과 RMON (Remote Monitoring), MIB(Management Information Base) 등은 이런 표준의 대표적인 예라 할 수 있다[1][2][3][4]. 네트워크 관리를 위한 지속적인 노력의 결과로 네트워크 관리 시스템은 많은 발전을 거듭해 왔으나, 아직도 많은 연구가 이루어지고 있는 분야이다.

네트워크 관리 시스템의 다양한 기능 가운데 고장 관리(Fault Management)는 아주 중요한 부분을 차지한다. 네트워크의 활용성을 높일 수 있도록, 고장이 발생한 에이전트(Agent)를 빨리 고립시키고, 이 영향이 네트워크에 미치지 않도록 하는 것이 바로 고장 관리의

† 학생회원 : 한국정보통신대학원대학교 공학부
jslee@icu.ac.kr
nhkim@icu.ac.kr
†† 비 회 원 : 한국전자통신연구원 표준연구센터 연구원
pdcho@pec.etri.re.kr
††† 정 회 원 : 한국정보통신대학원대학교 공학부 교수
yhlee@icu.ac.kr
논문접수 : 1999년 11월 29일
심사완료 : 2000년 6월 26일

가장 큰 목적이다. 이 때문에 고장을 빨리 감지할 수 있는 알고리즘과 다른 네트워크 구성 요소에게 영향을 주지 않도록 고립시키는 방법 등에 대한 연구가 주류를 이루고 있다.

고장 관리에 관한 연구는 네트워크의 활용성과 안정성의 측면에 있어서 네트워킹에 직접적으로 관련이 있는 요소들에 대한 문제를 해결하는 방법을 제시하고 있다. 그러나, 네트워크 관리 시스템에서 네트워크 구성요소에 대한 고장만을 다루어서는 부족한 점이 있다. 네트워크 관리 시스템 자체에도 고장이 발생할 수 있기 때문이다. 네트워크 관리 시스템에 고장이 발생하면, 네트워크를 구성하는 에이전트로부터 데이타를 모니터링할 수 없으며, 경고(Alarm) 메시지 등의 중요한 정보의 손실은 관리의 신뢰성을 떨어뜨리는 요인이 된다. 이런 이유로 네트워크 관리 시스템의 오류에 대한 견실한 구조는 중점적으로 연구해야 할 중요한 주제인 것이다.

본 논문에서는 네트워크 관리 시스템 자체에 발생할 수 있는 고장이나 에러에 대하여 정의하고, 이런 문제들을 해결할 수 있도록 구조적인 접근 방법을 제시하고자 한다. 본 논문에서 제안하고자 하는 방법은 매니저들 간의 세션을 형성하고 SNA(Script of Neighbor's Agent)를 통해 세션 내의 매니저들에 의해 고장이 발생한 매니저의 관리를 대행해 주도록 하여, 네트워크가 "항상 관리되고 있는" 상태로 유지될 수 있도록, 즉, 네트워크에 발생할 수 있는 오류나 네트워크의 물리적인 정지, 혹은 고장이 발생하더라도 네트워크 관리 시스템은 영향을 받지 않도록 하여, 항상 네트워크를 관찰하고 관리할 수 있도록 하였다.

본 논문에서 제안한 방법은 다음과 같은 특징을 갖고 있다. 첫째, 계층적인 구조를 가짐으로써 하나의 관리 스테이션에 집중되던 부하를 하위의 매니저들에게 분산시킬 수 있으며, 둘째, SNMP 메시지를 전송하기 위해 백본을 통과하는 트래픽을 줄일 수 있다. 셋째, 네트워크를 관찰해야 할 의무가 있는 매니저의 주변에 발생할 수 있는 오류에 능동적으로 대처할 수 있는 구조를 가지고 있다.

본 논문의 구성은 다음과 같다. 2장에서는 기존 네트워크 관리 시스템의 연구 동향과 관련 연구들을 기술하고, 3장에서는 제안하는 생존성 향상을 위한 관리 시스템의 구조에 대해 자세하게 기술할 것이다. 4장에서는 제안한 방법에 대한 생존성 함수를 유도하여 성능을 분석하고, 5장에서 결론과 네트워크 관리 시스템의 향후 연구 방향을 제시한다.

2. 네트워크 관리 시스템의 동향과 관련연구

IAB(Internet Architecture Board)에서는 네트워크 관리 시스템을 지원하기 위한 지속적인 노력을 기울여 왔는데, 이 결과로 SNMP가 데이타 네트워크의 관리를 위한 프로토콜의 표준으로 정의되면서 네트워크 관리 시스템에 많은 발전을 가져왔다. 그러나, SNMP에서의 보안 문제와 중앙 집중적인 관리 구조에 의한 문제점 때문에, SNMPv2가 새로운 표준으로 정의되었다. SNMPv2가 표준으로 정의됨에 따라, 계층적인 네트워크 관리가 가능해 졌으며, 네트워크 관리 시스템의 구조에도 많은 변화가 생겼다. 그러나, SNMPv2에서도 여전히 보안문제는 완전히 해결되지 못했으며, 이에 따라 현재 IAB에서는 SNMPv3에 관련된 표준을 제정하게 되었다. SNMPv3는 1998년에 그 표준이 정해졌는데, 각 기능별로 모듈화하여 각 부분별로 새로운 표준이나 기능을 포함할 수 있도록 구성되었으며, 보안 기능이 강화되었다. 현재 SNMPv3 워킹 그룹에서는 RFC 2570-RFC 2575를[8][9][10][11][12][13] 완성하였으며, 하나의 인터넷 드래프트를 만들어두고 있다

SNMP 혹은, SNMPv2를 이용하여 계층적인 구조를 갖는 네트워크 관리 시스템을 구현하고자 하는 노력은 다양하게 진행되었다. [14]에서는 SubManager를 사용하여 계층적인 구조의 시스템을 설계하고, 간단한 프로토타입을 제시하였다. [5]에서는 스프레드 시트 패러다임을 소개하며, 이를 통하여 에이전트와 매니저 간의 추상적인 인터페이스를 구축하고, 동적인 정책 할당을 가능하게 하였다. [6]에서는 SNMP를 확장하여 계층적인 관리 구조를 가능케 하는 방법을 사용하고 있다.

네트워크 관리 시스템 가운데 고장 관리(Fault Management)에 관한 연구는 매우 중요한 부분이며, 이에 관련된 연구도 많이 진행되고 있다. 고장 관리 시스템은 네트워킹 요소들(예를 들어, 라우터, 물리적 전송선)에 발생한 문제를 빨리 해결하고자 하는 것을 목적으로 하고 있다[19]. 그러나, 고장 관리 시스템의 대부분이 관리하는 사람이 직접 연관되어 있다는 문제가 있었다. 이를 해결하기 위해 자동화된 고장 관리 시스템을 만들고자 하는 노력이 진행되었는데, [16]에서는 사람의 직접적인 작용이 없더라도 Hierarchical Domain-oriented Reasoning Mechanism을 사용하여 네트워크에 발생한 문제가 전파되는 것을 자동화된 방법으로 방지하고자 하였다. 그 외에도 특별한 네트워크 환경을 대상으로 한 고장의 고립화(isolation) 방법과 이에 필요한 제반 기술에 관한 연구가[17] 많이 이루어지고 있다.

이러한 고장 관리의 흐름을 살펴보면 통신이나 네트워킹에 필요한 요소들에 관한 관리를 목적으로 하고 있다. 그러나, 본 논문에서는 네트워크 관리 시스템 자체에도 오류가 발생할 수 있음에 착안하여 이를 해결하고자 새로운 구조의 네트워크 관리 시스템을 제안하고자 한다.

3. 생존성을 높이기 위한 관리 시스템의 구조

네트워크 관리 시스템은 매니저가 에이전트와의 메시지 교환을 통해 네트워크에 연결된 장치들의 상태에 관한 정보를 입수하여 이를 관리 스테이션에 전송함으로써 관리자(Administrator)에게 알려주는 시스템이다. 에이전트는 매니저로부터 받은 SNMP 메시지에 대해 응답을 할 수 있는 장치를 의미한다. 관리자는 관리 스테이션에 알려지는 정보를 참고하여 네트워크에 적용되는 정책을 결정하거나 수정하는 사람을 의미하며, 네트워크에 연결된 장치(에이전트)들의 설정을 수정하게 된다. 관리자가 얻게 되는 정보는 매니저가 수집하게 되므로, 네트워크를 항상 모니터링하기 위해서는 매니저가 안정된 상태를 유지하는 것이 중요하다.

네트워크 관리 시스템 전체에 영향을 미칠 수 있는 고장은 매니저의 주변에서 일어나게 된다. 네트워크에 발생하는 일반적인 오류의 경우 매니저가 이를 관리 스테이션에 알려줄 수 있는 방법이 있기 때문에 관리자가 쉽게 고장 상황을 파악하고 이에 대처할 수 있는 방법을 강구하게 되지만, 매니저가 정지하거나 매니저와 연결된 링크 등이 끊어지는 경우에는 오류 상황을 파악하기도 힘들 뿐만 아니라, 매니저에 일어난 고장을 파악하고 수정하는 동안 이 매니저에 의해 관리되고 있던 에이전트에 대한 정보는 관리 스테이션(즉, 관리자)에 전달되지 않는다. 네트워크에서 발생하는 오류가 단순히 하나의 영향을 미치는 오류가 아니라 통합적인 오류인 경우 이를 해결할 수 있는 구조를 갖고 있어야 한다.

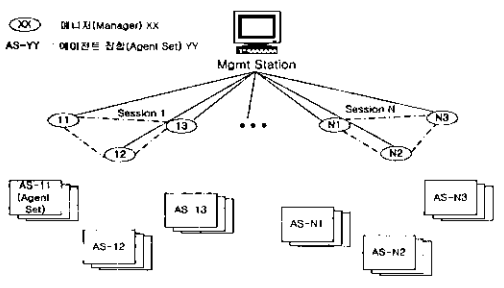


그림 1 오류 상황에 능동적으로 대처할 수 있는 관리 구조

그림 1은 매니저에 발생한 오류를 해결하기 위해 필요한 네트워크 관리 시스템의 구조를 보여주고 있다. 관리 시스템을 몇 개의 영역으로 나누어 각 영역 내의 매니저들 간에 세션을 형성하게 하여 각 매니저들은 세션 내의 매니저들과 연계하여 망을 관리하게 된다. 하나의 매니저가 관리하는 장치들을 에이전트 집합(Agent Set)으로 정의하였다. 각 매니저는 자신이 관리하는 에이전트들의 리스트를 가지고 있으며, 세션 내의 다른 매니저들이 관리하는 에이전트의 리스트도 저장해 두고 있다.

세션은 인터넷을 활용한 응용 서비스의 영역에서는 널리 알려진 개념이다. 이 개념을 도입하여 네트워크 관리 시스템을 오류에 대하여 유연하고 안정적인 형태로 만들 수 있다. 셋 이상의 매니저를 하나의 세션으로 묶어서 분류하고, 세션을 형성한 매니저들은 네트워크를 관리하는 작업을 수행함에 있어서 주기적으로 서로의 상태를 파악하며, 관리를 맡은 영역을 모니터링하여 그 결과를 관리 스테이션에 전송하게 된다.

본 논문에서는 모든 에이전트들에 대한 기본적인 설정들은 관리 스테이션에 저장되어 있으며, SNA는 각 매니저마다 다르게 설정이 되어 있으나 이 모든 설정이 관리 스테이션에 저장되어 있다고 가정하고 있다. 이는 매니저들이 다른 매니저들의 작업을 대신하게 되는 경우 에이전트들의 설정에 관해서는 따로 생각할 필요가 없음을 의미하며, 관리스테이션에서는 어느 매니저가 어떤 작업을 하고 있는지 자동적으로 파악할 수 있음을 뜻한다.

3.1 세션의 생성과 유지 및 매니저의 오류 판단 기법

본 구조에서 모든 매니저들은 각자 하나의 세션에 소속되어 있으며, 이를 통하여 상호간의 생존 여부를 확인한다. 세션의 생성 및 유지는 다음과 같이 이루어진다. 이 모든 메시지와 작업들은 응용 프로그램 수준에서 이루어지게 되므로 SNMP나 더 낮은 레벨의 전송 프로토콜과는 상관이 없다.

1. 각 매니저는 정의된 작업들을 시작할 때 SNA (Script of Neighbor's Agent)를 확인하여 세션에 관한 정보와 작업의 범위에 대한 정보를 얻게 된다.

2. 각 매니저는 같은 세션에 소속된 매니저들에게 SSN_INIT 메시지를 보내고, 다른 매니저로부터 이 메시지를 받으면, 그 매니저에 대한 로그(Log) 정보의 기록을 시작한다.

3. 각 매니저는 같은 세션에 소속된 매니저들에게 SSN_HELLO 메시지를 보냄으로써, 자신의 생존 여부

를 이웃의 매니저에게 알리게 되며, 다른 매니저로부터 이 메시지를 받으면 세션이 잘 유지되고 있는 상태로 간주한다.

4. SSN_HELLO 메시지를 받지 못한 경우, 메시지를 보내지 않은 매니저의 아이디(MGR_ID)를 MGR_FAULT 메시지와 함께 관리 스테이션에 보냄으로써 문제가 발생했음을 알린다.

각 매니저들은 위의 순서에 따라 같은 세션에 속하는 매니저를 파악하고, 자신이 처리해야 할 작업의 범위를 알게 된다. 매니저가 세션 내의 다른 매니저에 발생한 오류를 인지하고 이를 관리 스테이션에 전송하면, 최종적인 오류에 대한 결정은 관리 스테이션에서 이루어지게 된다. 관리 스테이션에서 오류임을 판단하는 과정은 다음과 같다.

1. 오류가 발생했음을 알리는 MGR_FAULT 메시지가 어떤 매니저로부터 도착하면, 관리스테이션은 이 매니저의 세션에 대한 정보를 검색하고, 이를 저장해둔다.

2. 1의 과정에서 MGR_FAULT 메시지를 전송한 매니저와 같은 세션에 속하는 모든 매니저들에게서 같은 메시지를 받게 되면 관리스테이션은 메시지와 함께 받은 MGR_ID를 검색한다.

3. MGR_ID를 통해 오류가 발생했으리라 추측되는 매니저가 최근에 보낸 SNMP 메시지들을 검색을 한다. 최근의 Report기록이 없으면 일단 이 매니저를 오류가 발생했다고 판단한다.

4. 오류가 발생했다고 판단하게 되면 관리 스테이션은 오류가 발생한 매니저와 같은 세션에 속하는 매니저들에게 JOB_DLG_CMD(Job Delegation Command), 즉 작업 이전 명령을 내림으로써 세션 내에서 이 오류를 처리할 수 있게 한다.

위와 같은 과정은 매니저의 오류에 관한 좀 더 정확한 판단을 위해 이루어지게 된다. 세션 내의 매니저들에 의해서만 오류를 판단하는 것은 충분하지 않기 때문이다. 예를 들어, 일시적으로 어떤 매니저가 처리해야 할 작업이 많아서 SSN_HELLO 메시지를 보낼 수 없는 상황일 수도 있다. 이런 경우 매니저들의 메시지에만 의존해서 문제의 발생을 판단하는 것은 정확한 판단을 내리기 힘들다. 또한 최근에 오류가 발생했으리라 추측되는 매니저로부터 도착한 Report를 검사하는 이유는 각 매니저들은 주기적으로 Report를 보내도록 설정이 되어 있으나 이 작업을 제대로 수행하지 못했음을 의미하는 것이며, 이 경우 이 매니저는 더 이상 매니저로서의 작업을 수행할 수 없다고 판단하게 된다.

세션을 형성하는 매니저의 수는 셋 이상으로 하는 것이 좋다. 세션을 구성하는 매니저의 수를 두 개만 두는 경우에는 두 매니저 간에 발생할 수 있는 연결 상태의 오류를 잘못 판단하여 매니저의 정지로 판단을 내릴 가능성이 있기 때문이다. 너무 많은 매니저에 의해 세션이 형성되는 경우에는 오류 상황에 대한 설정 작업이 복잡하고, 오류에 대한 결론을 내리는 데 어려움이 있기 때문에, 신속한 대처가 힘들고 관리 영역을 분할하는 데 어려움이 있다.

3.2 오류 처리 및 세션 복구 기법

몇 개의 매니저들이 세션을 형성하여 관리하게 되면 생존 가능한 관리 시스템의 구조적인 면을 살펴볼 때, 몇 가지 장점이 있다. 첫째, 세션 내의 매니저들의 상태를 쉽게 파악할 수 있다. 둘째, 오류가 발생한 매니저가 관리하던 영역을 다른 매니저들이 관리하게 하여 오류 상황을 쉽게 해결할 수 있다. 셋째, 매니저에 발생한 고장이 해결된 이후 쉽게 원래의 네트워크 관리 구조로 복원될 수 있다. 넷째, 관리 스테이션에서 관리자가 얻을 수 있는 네트워크의 정보는 오류상황을 판단할 때까지의 시간을 제외하고는 별 차이가 없다.

그림 2는 고장이 발생한 매니저를 관리 작업에서 제외시키고, 관리 영역을 세션 내의 다른 매니저들이 나누는 과정을 보여주고 있다. 매니저 N1과 N2는 매니저 N3에 문제가 발생했음을 알아낸 후, 관리 스테이션에

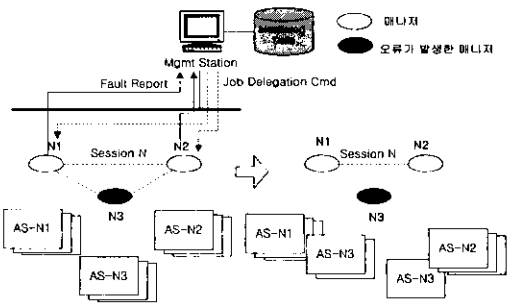


그림 2 오류가 발생한 매니저를 제외시키는 방법

이 상황을 알려주게 된다. 관리 스테이션으로부터 오류 상황임을 알리는 메시지가 도착하면 N1과 N2는 N3의 작업을 가져온다. N1과 N2는 모두 N3가 관리하던 영역의 리스트를 갖고 있으며, N3의 고장 시 수행할 작업에 대한 시나리오를 파악하고 있기 때문에 쉽게 고장 상황을 해결할 수 있다.

N1과 N2가 가지고 있어야 할 N3의 관리 대상이 되는 리스트는 관리자에 의해 설정된다. 이웃 매니저의 에이전트를 기술하는 스크립트 SNA(Script of

Neighbor's Agent)에 의해 이 리스트는 표현될 수 있다. SNA는 매니저의 오류의 경우 관리해야 할 리스트들을 간단히 기술하는 것이다. 이 스크립트를 사용하여 각 매니저가 오류상황이 발생했을 때 처리해야 할 일들을 모두 기술해 줄 수 있으므로, 능동적인 해결이 가능하며, 관리자의 간단한 설정으로 시스템의 문제를 해결할 수 있다.

3.3 SNA(Script of Neighbor's Agent)

SNA는 각 매니저의 역할과 이웃 매니저가 오류가 발생했을 때 수행되어야 할 과정에 대해 기술한 스크립트(Script)이다. 이는 네트워크 관리 시스템에서 사용하기 위해 특별히 설계할 수도 있으나, UNIX에서 사용되고 있는 스크립트(예를 들어, Tcl)를 사용한다 해도 별 무리가 없으리라 보인다. 본 논문에서는 이에 대한 구체적인 형태와 설계 기준 등에 관해서는 언급하지 않을 것이다.

SNA는 3.1절에서 기술한 바와 같이 매니저가 작업을 시작할 때 참조하도록 하여 자신의 역할과 세션에 관한 정보 등을 읽어올 수 있도록 하고 있다. 그림 3에 SNA의 간단한 예를 들어보았다.

위의 예는 3.2에서 설명한 그림 2의 세션 구조에서 매니저 N2에 설정할 수 있는 SNA를 기록한 것이다.

```
MGR_ID N2
Session SSN_ID N={N1,N3}
DefaultAgentSet {..Agent List..}
AgentSet=DefaultAgentSet
if(Failure equals N1)
     AgentSet=DefaultAgentSet+{..Agent
List..}
endif
if(Failure equals N3)
     AgentSet=DefaultAgentSet+{..Agent
List..}
endif
```

그림 3 SNA의 실례

DefaultAgentSet은 N2가 관리해야 하는 에이전트 집합을 기술하고 있으며, 에이전트 집합은 각각의 에이전트를 의미하는 AGENT_ID가 그 집합의 원소가 될 수 있다.

3.4 기능적 측면에서의 매니저의 구조

그림 4에 오류에 대응할 수 있는 매니저의 구조를 표현하였다. 이것은 SNMPv3에서 제시하고 있는 매니저의 구조[4]를 수용하여 새로운 모듈을 추가하고 기능을 확장한 구조이다.

패킷 처리 모듈(Packet Processing Module)은 SNMPv3의 Dispatcher와 같은 역할을 수행한다. 패킷을 전송하거나, 들어온 패킷을 메시지 처리 모듈(*Message Processing Module*)에 전달하고, 명령 처리 모듈(Cmd Processing Module)에서 요구하는 패킷을 처리한다. 또한, 오류를 알리는 메시지를 명령 처리 모듈에 전달하여 해당하는 작업이 이루어지도록 도와주게 된다. 패킷 처리 모듈은 매니저를 구성하는 여러 모듈간을 연결하는 중요한 일을 담당한다.

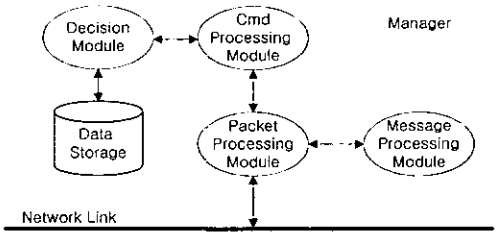


그림 4 매니저의 기능적 구조

메시지 처리 모듈(Message Processing Module)은 SNMP 메시지를 처리하는 모듈이다. SNMP 버전에 따라 서로 다른 구조를 가지게 되므로, 메시지 처리 모듈에서는 SNMP 메시지의 버전을 확인하고, 이에 따라 메시지를 처리하고 요구되는 동작을 수행하게 된다. 보안에 관련된 작업이 필요하다면, 메시지 처리 모듈에서 이런 작업을 수행하게 된다.

명령 처리 모듈(Cmd Processing Module)은 SNMP 명령을 만들어내고, SNMP 요구에 대한 응답을 받아들이는 것이 주된 기능이다. 명령 처리 모듈은 생존성 측면에서도 아주 중요한데, 세션간에 주고받는 메시지를 결정 모듈(Decision Module)로부터 받아들여 패킷 처리 모듈에 전달하게 된다.

결정 모듈(Decision Module)과 데이타 영역(Data Storage)은 매니저의 오류를 처리하고, 생존성을 보장하는데 아주 중요한 역할을 수행한다. 데이타 영역에는 현재 관리하고 있는 에이전트의 리스트를 갖고 있으며, 세션 내의 매니저들에 대한 정보를 담고 있다. 또한, SNA(Script of Neighbor's Agents)를 저장해 두었다가 결정 모듈의 요구에 따라 SNA를 넘겨주어 네트워크 관리 시스템의 오류를 고치는 데 도움을 준다. 데이타 영역에는 매니저를 관리하고 유지하는데 필요한 모든 데이타가 저장되어 있다.

결정 모듈은 주기적으로 다른 매니저들에게 메시지를 보내어 같은 세션 내의 매니저들에 이상이 없는지 확인하게 된다. 또, 데이타 영역을 관리하고 새로운 정보를 기록하게 하거나, 필요한 정보를 요청하는 등의 기능을 수행한다. 특히 결정 모듈은 관리 스테이션과 통신을 하는 메커니즘을 갖고 있으므로, SNA나 관리 정책 등의 변화를 설정하는 기능을 갖고 있다. 이를 통해 네트워크의 정책을 바꾸거나, 환경 설정 등의 작업이 관리 스테이션에서 이루어질 수 있다.

4. 결과 및 성능 분석

4.1 생존성에 관한 분석

네트워크 자체의 생존성에 관한 함수를 표현하는 방법과 그 접근 방법에 관한 연구가 진행되어 있는데[15], 이에 기반하여 생존성 함수를 본 시스템에 적용해 볼 수 있다. 본 연구에서는 매니저와 관련된 생존성의 연구에 그 관심이 있고, 생존성의 여부는 관리 스테이션에 관리 대상이 되는 에이전트의 정보가 전달될 수 있는가의 여부로 판단할 수 있다.

그림 5는 세션 내의 노드에 Fault가 발생했을 때 생존 가능한 집합과 그렇지 않은 집합을 표현하고 있다. 집합 S는 생존 가능한 집합을 의미하며, S^c는 생존 불가능한 집합을 의미한다. 세션 내의 매니저의 수를 M, 에이전트의 수를 A, 각 매니저의 관리를 받는 에

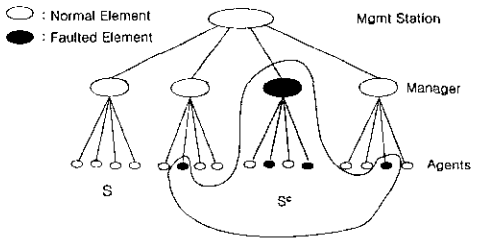


그림 5 세션 내에 Fault나 오류의 발생시 생존 가능성에 따른 집합 관계

이전트의 수를 A_k라 하자. A_k와 A사이에는 다음과 같은 관계가 성립한다.

$$A=\sum_{k=1}^{M} A_k \tag{1}$$

생존성에 관한 함수를 도출하고 분석을 수행하는 과정은 분석적이고 수학적인 과정을 따르며, 여러 가지의 표현 방법과 확률 변수들을 사용하였다. 이에 대해서는 새로운 표현이 나올 때마다 설명을 하고자 하였으며, 편의를 위하여 표 1에 본 논문에 사용된 표현과 이에 대한 설명을 정리해 보았다.

표 1 본 논문에 사용된 표현

Notation	Description
A	세션 내의 모든 에이전트의 수
A_k	k 번째 매니저가 관리하는 에이전트의 수
M	세션 내의 매니저의 수
S	생존성을 의미하는 Random Varaible
n	세션 내에 발생하는 Fault의 수
n_M	매니저에 발생하는 Fault의 수
n_A	에이전트에 발생하는 Fault의 수
N_s	생존성 S를 만족할 때, n개의 Fault가 발생할 수 있는 경우의 수
T	매니저와 관계없이 독자적으로 Fault가 일어난 에이전트의 수

4.1.1 오류 처리 메커니즘이 사용되지 않는 경우

세션 내에 n개의 Fault가 발생한다고 가정하면, 발생할 수 있는 모든 경우의 수는 ${}_{(M+A)}C_n$이 된다. 세션 내의 매니저나 에이전트에 발생할 수 있는 Fault의 확률 P_e와 생존성의 확률은 다음과 같다.

$$P_e=\frac{1}{{}_{(M+A)}C_n} \tag{2}$$

$$P[S=s\,|\,n]=\sum_{e:S_e=s} P_e=N_s\frac{1}{{}_{(M+A)}C_n} \tag{3}$$

N_s는 생존성 S를 만족할 때 n개의 Fault가 발생할 수 있는 모든 경우의 수를 의미한다. 각 매니저가 관리하는 에이전트의 수 A_k는 모두 같다고 가정하자. N_s를 얻기 위해 고려해야 할 것은 매니저에 발생하는 오류의 수이다. 매니저에 발생하는 오류의 수를 n_M이라 하고, 에이전트에 발생하는 오류의 수를 n_A라 하면, $n=n_M+n_A$의 관계가 성립한다. 매니저에 오류가 발생하는 경우, 그 매니저의 관리를 받던 모든 에이전트는 생존성이 없는 상태가 되기 때문에 오류가 일어나지 않았더라도 오류가 일어난 상태와 동일하다. 따라서 N_s는 다음과 같이 표현할 수 있다.

$$N_s=\binom{M}{n_M}\times\binom{A-n_M A_k}{n-n_M A_k} \tag{4}$$

이 식을 (3)에 대입하여, n과 n_M에 대한 확률로 표현할 수 있는데, 정리하면 다음과 같다.

$$P[S=s|n]=\frac{{}_{M}C_{n_M}\times{}_{(A-n_MA_t)}C_{(n-n_MA_t)}}{{}_{(M+A)}C_{n}} \quad (5)$$

과 같은 결과를 얻을 수 있으며, 이 식은 S에 대한 함수가 아니라, n과 n_M에 대한 함수이다. 이것은 본 논문에서 기술한 모델이 연속적인 모델이 아니라, 이산적인 모델로 표현되고 있기 때문이다.

If $n_M < M$, $P[S=s|n]=\frac{{}_{M}C_{n_M}\times{}_{(A-n_MA_t)}C_{(n-n_MA_t)}}{{}_{(M+A)}C_{n}}$ (6)

If $n_M = M$, $P[S=s|n]=0$ (7)

오류 처리 메커니즘을 사용하지 않은 계층적 구조에서의 생존성은 (6), (7)과 같이 정리될 수 있으며, 매니저에 오류가 발생함에 따라 시스템 자체의 생존성에 많은 영향을 미침을 알 수 있다.

4.1.2 매니저의 오류 처리 메커니즘을 사용하는 경우

비슷한 과정을 거쳐 이 경우의 생존성을 생각해볼 수 있다.

위에서 구한 N_s의 값이 n개의 오류가 발생했을 때, 생존성 S를 만족하는 모든 경우의 수를 구한 것임을 상기하자. 매니저에 발생하는 오류 n_M이 M보다 작은 경우에는 에이전트의 생존성은 매니저의 생존성에 영향을 받지 않으므로,

$$N_s = {}_{(M+A-n_M)}C_{n-n_M} \quad (8)$$

If $n_M < M$, $P[S=s|n]=N_s\frac{1}{{}_{(M+A)}C_{n}}=\frac{{}_{(M+A-n_M)}C_{n-n_M}}{{}_{(M+A)}C_{n}}$ (9)

If $n_M = M$, $P[S=s|n]=0$ (10)

과 같이 정리할 수 있다. 즉, 세션 내의 모든 매니저에 오류가 발생하는 경우를 제외하면, 매니저의 오류를 처리하는 메커니즘을 갖고 있는 네트워크 관리 시스템은 생존성을 유지할 수 있음을 알 수 있다.

표 2 네트워크 관리 시스템의 생존성

관리 시스템	생존성 ($n_M < M$ 인 경우)	$n_M = M$ 인 경우
오류 처리 메커니즘이 있는 경우	$\frac{{}_{(M+A-n_M)}C_{n-n_M}}{{}_{(M+A)}C_{n}}$	0
오류 처리 메커니즘이 없는 경우	$\frac{{}_{M}C_{n_M}\times{}_{(A-n_MA_t)}C_{(n-n_MA_t)}}{{}_{(M+A)}C_{n}}$	0

네트워크 관리 시스템에서 오류 처리 메커니즘이 있는 경우와 없는 경우에 대한 앞의 분석 결과를 표 2에 정리하여 보았다.

4.2 결과 분석

4.1절에서의 분석결과를 토대로 수치적인 작업을 통해 그림 6와 그림 7의 결과를 얻을 수 있었다. 그림 6는 같은 그래프를 세로 축을 로그 스케일로 취하여 그려진 것이며, 그림 7은 로그 스케일을 취하지 않은 경우이다. 이 두 그래프는 모두 오류 상황을 처리할 수 있는 방법을 사용하면, 생존성이 크게 감소되지 않음을 보여주고 있으며, 그렇지 않은 경우 생존성은 오류가 발생한 매니저의 수가 증가함에 따라 빨리 감소함을 보이고 있다.

그림에서 기존의 방법은 네트워크 관리 시스템 자체의 생존성을 고려하지 않은 경우의 생존성 감소를 의미하며, 오류 처리 메커니즘은 본 논문에서 기술한 구조를 가진 네트워크 관리 시스템의 경우를 나타낸다.

이 분석은 세션 내의 매니저의 수를 20, 발생할 수 있는 오류의 수를 1800으로 두고, 오류가 발생할 수 있는 매니저의 수가 증가할 때 시스템 자체의 생존성을

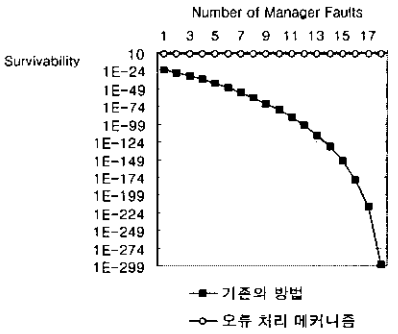


그림 6 생존성 비교(Log스케일)

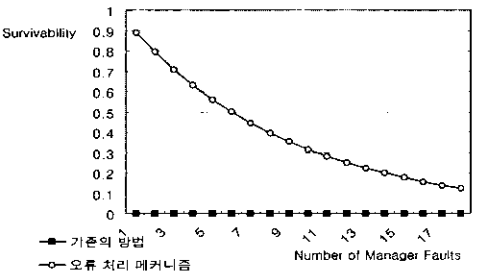


그림 7 생존성 비교

검증해본 결과이다.

네트워크 관리 시스템의 생존성을 높일 수 있는 메커니즘을 사용하면, 매니저에 오류가 발생하더라도 시스템 자체의 생존성에 미치는 영향을 최소화하여, 약간의 생존성 저하만을 가져오게 된다. 일반적인 네트워크 관리 시스템의 생존성이 매니저 하나에만 오류가 발생하더라도 거의 0에 가까운 값을 보여주는 것에 비해 매우 큰 값의 생존성을 유지하고 있다.

더 많은 수의 매니저들을 세션으로 연결할수록 오류 발생 시 더 큰 생존성을 유지할 수 있지만, 각 오류 상황에 대해 다른 설정을 요구하게 되므로, 네트워크 시스템의 설정을 위해 많은 노력이 필요하다. 그러므로, 세션으로 연결되어야 할 매니저의 수를 적절하게 결정하는 것이 필요하다. 일반적인 네트워크의 경우, 매니저에 고장이 발생할 확률이 매우 적음을 가정한다면, 많은 매니저를 세션으로 연결할 필요는 없다.

5. 결론 및 향후 연구 방향

본 논문은 계층적인 구조를 갖는 네트워크 관리 시스템에서 생존성을 높일 수 있는 관리 시스템의 구조와 오류가 발생했을 때 처리할 수 있는 메커니즘에 대해 제안하였다. 매니저들 간에 세션을 형성하여, 매니저의 정지 혹은 오류를 파악할 수 있으며, 오류가 발생하면 다른 매니저들이 오류가 발생한 매니저가 관리하던 에이전트에 대한 모니터링을 대신하게 되므로, 관리 시스템 자체의 생존성을 높일 수 있었다. 또한 본 논문에서 제안하는 네트워크 관리 시스템의 구조는 네트워크 관리 시스템의 매니저와 관리 스테이션 등의 소프트웨어 모듈만을 수정하여 네트워크를 직접적으로 구성하는 에이전트에는 아무런 영향을 주지 않음으로써 기존의 네트워크에 적용할 수 있는 장점이 있다.

본 논문에서는 시스템의 생존성에 관한 함수를 도출하여, 이를 통하여 제안한 시스템이 생존성 측면에서 적합한 선택임을 파악할 수 있었다. 오류를 처리할 수 있는 방법을 사용하지 않으면, 매니저에 발생한 오류가 시스템 전체에 많은 영향을 미치며, 오류가 발생하는 매니저의 수가 증가하면서 큰 생존성 저하가 일어나게 된다. 이에 비해, 본 논문에서 제안한 방법을 사용하면, 매니저의 수가 감소하더라도 다른 매니저에서 오류가 발생한 매니저들의 역할을 대신할 수 있기 때문에, 생존성이 감소하기는 하나, 어느 정도까지는 유지가 가능한 상태를 지속할 수 있다. 그러나 매니저 간의 정보를 공유해야 하는 문제와 오류가 발생한 매니저의 멤버들을 어떤 기준으로 남아 있는 매니저에게 분배할 것인가에 관한 문제 등을 추가적으로 고려하여야 한다.

네트워크 관리 시스템은 네트워크를 관리하고 모니터링하는 중요한 시스템이므로, 생존성을 높이는 것은 매우 중요한 문제이며, 많은 연구가 이루어져야 할 분야이다. 네트워크 관리 시스템은 인터페이스와 기능이 중요하기 때문에 이에 비해 생존성 측면에서의 연구가 부족한 면이 없지 않았다. 본 논문에서 제시한 방법은 구조적인 측면에서의 접근 방법이므로, 네트워크 관리 시스템의 설계, 구현 시에 적용할 수 있으리라 보여진다.

향후 본 논문에서 제안하고 있는 구조를 액티브 네트워크(Active Network)[20][21]을 기반으로 한 네트워크 관리 시스템에 적용, 발전시켜 구현을 할 계획이며, 현재는 액티브 네트워크 시스템 자체의 구현을 진행하고 있다.

참 고 문 헌

[1] M. Rose and K. McCloghrie, "Structure and Identification of Management Information for TCP/IP-based Internets," RFC1155, May 1990

[2] K. McCloghrie and M. Rose, "Management Information Base for Network Management of TCP/IP-based internets," RFC1156, May 1990

[3] J. Case, M. Fedor, M. Schoffstall, and J. Davin, "A Simple Network Management Protocol," RFC1157, May 1990.

[4] J. Case, K. McCloghrie, M. Rose, and S. Waldbusser, "Structure of Management Information for Version 2 of the Simple Network Management Protocol (SNMPv2)," RFC1902, January 1996

[5] A. S. Sethi, P. Kalyanasundaram, C. M. Sherwin, and D. Zhu, "A Hierarchical Management Framework for Battlefield Network," MILCOM 97 Proceedings, 1997.

[6] Y. Guo, T. Akatsuka, and Y. Hiranaka, "Hierarchical Network Management Based on Extended SNMP," T.IEE Japan, Vol.199-C, No.3, 1999.

[7] A. Balakrishnan, T. L. Magnanti, and Prakash Mirchandani, "Designing Hierarchical Survivable Networks," Operations Research, Vol. 46, No. 1, Jan.-Feb., 1998.

[8] Case, J., Mundy, R., Partain, D., and Stewart, B., "Introduction to Version 3 of the Internet-standard Network Management Framework," RFC 2570, April 1999

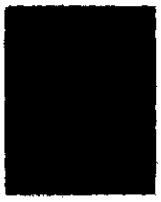
[9] Harrington, D., Presuhn, R., and B. Wijnen, "An Architecture for describing SNMP Management

Frameworks," RFC 2271, January 1998.
[10] Case, J., Harrington, D., Presuhn, R., and B. Wijnen, "Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)," RFC 2272, January 1998.
[11] Levi, D., Meyer, P., and B. Stewart, "SNMPv3 Applications," RFC 2273, January 1998.
[12] Blumenthal, U., and B. Wijnen, "The User-Based Security Model for Version 3 of the Simple Network Management Protocol (SNMPv3)," RFC 2274, January 1998.
[13] Wijnen, B., Presuhn, R., and K. McCloghrie, "View-based Access Control Model for the Simple Network Management Protocol (SNMP)," RFC 2275, January 1998.
[14] M. R. Siegl and G. Trausmuth, "Hierarchical network management: a concept and its prototype in SNMPv2," Computer Networks and ISDN Systems 28, pages 441-452, 1996
[15] S. C. View and K. W. Lu, "A Framework for Network Survivability Characterization," Proc. of the IEEE International Conference on Communications ICC'92 - Volume 1, 1992
[16] C. S. Chao, D. L. Yang, and A. C. Liu, "An Automated Fault Diagnosis System Using Hierarchical Reasoning and Alarm Correlation," IEEE Workshop on Internet Applications, Page(s): 120 127, 1999.
[17] D. Schuth, W. Nejdl, and R. Hager, "Fault management of infrastructure networks," IEEE 44th Vehicular Technology Conference, vol.1, 1994, pp 391 -395
[18] G.G. Berthet and N. Fischer, "A unified theory of fault diagnosis and distributed fault management in communication networks," IEEE AFRICON 4th Vol. 2, 1996, pp 776-781.
[19] G. Kar, "A methodology for fault management in heterogeneous networks," Second International Conference on Private Switching Systems and Networks, 1992, pp 144-148.
[20] David L. Tennenhouse et al, "A Survey of Active Network Research," IEEE communications magazine, Jan 1997. pp 80-86.
[21] D. Tennenhouse and D. Wetherall, "Toward an Active Network Architecture," Comp. Commun. Rev. vol 26, no 2, Apr. 1996.

이 중 수
1999년 2월 경북대학교 전자공학과 공학사. 1999년 2월 ~ 현재 한국 정보통신대학원 대학교(ICU) 석사과정. 1999년 3월 ~ 2000년 3월 한국통신 제 2 연구소 연구기획실 위촉연구원

조 평 동
1980년 연세대학교 전자공학과 졸업(공학사). 1995년 충남대학원 컴퓨터과학과 졸업(이학석사). 1980년 ~ 1997년 ISDN, 지능망, 통신처리시스템 개발. 1998년 ~ 현재 한국전자통신연구원 표준연구센터 책임연구원. 관심분야는 전기통신 기술기준, 지능망, Network architecture

김 남 훈
정보과학회논문지 : 정보통신
제 27 권 제 2 호 참조



이 영 희
1976년 2월 서울대학교 공과대학 공업교육학과 공학사. 1980년 2월 서울대학교 공과대학 공업교육학과 공학석사. 1981년 9월 ~ 1984년 6월 불란서 UTC(Universite de Technologie de Compiegne), 전산학 박사. 1984년 8월 ~ 1997년 12월 한국전자통신연구원, 정보통신표준연구센터 센터장. 1986년 7월 ~ 1987년 11월 IBM T. J. Watson 연구소 초빙과학자. 1998년 1월 ~ 현재 한국 정보통신 대학원 대학교(ICU) 교수.