제18회 한국정보처리학회 추계학술발표대회 논문집 제9권 제2호 (2002.11)

ICMP를 이용한 실시간 장애 탐지 시스템의 구현

김현구, 민지영, 장범환, 정태명
성균관대학교 전기전자 및 컴퓨터 공학과
e-mail:{hkkim, jymin, bhchang}@rtlab.skku.ac.kr,
tmchung@ece.skku.ac.kr

Implementation of A Real-Time Fault Detection System using ICMP

Hyun-Ku Kim, Ji-Young Min, B. Chang, Tai-M Chung
School of Electrical & Computer Engineering,
SungKyunKwan University

요약

1990년대를 지나면서 컴퓨팅 환경은 급속히 변화 하였다. 과거 메인 프레임 중심의 환경에서 네트워크를 기반으로 클라이언트/서버 모델 중심의 분산 컴퓨팅 환경으로 전환되었다. 이러한 환경에서 복잡하고 유지하기 힘든 데이터 네트워크를 관리를 수행하고, 또한 유용한 정보를 줄 수 있는, 효과적인 네트워크 관리의 필요성이 대두되었다. 이러한 필요에 의해서 네트워크 관리의 개념이 등장하였다. 네트워크 관리에는, 기능적 관점에서, 장애 관리, 구성 관리, 계정관리, 성능 관리, 그리고 보안 관리로 나누어진다[1]. 그런데 네트워크가 발전함에 따라, 네트워크에서 발생할 수 있는 각종 장애를 신속히 발견하여 관리자에게 통보하고 적절한 처리를 할 수 있는 기능이 특히 요구되고 있다. 하지만 기존의 SNMP 트랩 기반의 장애 관리는 기능 면에서 부족한 점이 많다[2]. 예를 들어, 불규칙적인 트랩 메시지, 또는 SNMP 패킷의 소실 때문에 정확한 장애 발견에 문제가 생기게 된다. 본 논문에서는 SNMP 장애 관리 기능을 보완하기 위해 ICMP Echo/Reply 메시지를 이용한 장애 처리 시스템의 구조와 알고리즘에 대해 설명하고, 기존의 SNMP 기반의 NMS와 어떻게 연동될 수 있는 방안에 대해서 기술하도록 하겠다..

1. 서론

네트워크가 발전함에 따라, 네트워크에서 발생할 수 있는 각종 장애를 신속히 발견하고 관리자에게 통보하고 적절한 처리를 할 수 있는 기능이 특히 요구되고 있다. 장애 관리는 비정상적인 동작이나 관리 객체를 탐지하고, 그 장애가 더 커지지 않게 장애 노드를 고립시키고 복구하는 일련의 과정 및 방법의 집합이다[3]. 주요 기능으로는 에러나 장애의 로그, 장애에 대한 탐지와 추적, 장애 복구, 관리자에게 발생한 장애에 관한 보고 및 이미 내려진 정책에 따른 자동화된 처리 등이 있다. 장애 관리에서 제일 중요한 것 중의 하나는 신속하고 정확한 장애의 탐지이다. 물론 장애를 미리 예견해 장애가 발생하기 전에 조치를 취하는 것이, 최선의 방법이겠지만 아직 완벽히 장애를 예측 할 수 있는 시스템은 존재하지 않는다.

현재 대부분의 망 관리 시스템들은 SNMP 프로토콜을 지원하고 있다. 현재 출시되고 있는 대부분의 오브젝트들이 SNMP를 지원하기 때문에 그 영향력을 무시 할 수 없다[2]. 하지만 SNMP는 장애 관리 면에서 상당히 취약한 면을 가지고 있다. 트랩 메시지의 불규칙적인 생성과 패킷을 소실은 정확한 장애 감지를 어렵게 한다. 또한 SNMP 요청 메시지를 주기적으로 폴링하여 장애 여부를 감지하는 방법 역시 SNMP 트랩을 사용할 때의 문제점을 가질 뿐만 아니라 관리 네트워크가 커질 경우 장애를 탐지하는 시스템과 네트워크 많은 로드를 주게 되어 감지 속도 역시 느려질 수 밖에 없다[3]. 이러한 문제점을 보안하기 위해서 이미 많은 연구 기관이나 업체에서 ICMP 메시지를 장애 관리나 네트워크 토폴로지에 활용하고 있다. 본 논문에서는 SNMP 장애 관리 기능을 보완하기 위해 ICMP 메시지를 이용한 장애 처리 시스템의 구조와 알고리즘에 대해 기술하도록 하겠다.

2. 관련연구

2.1. 주요 네트워크 장애 감지 방법

◆ SNMP Trap Messages : 가장 간단하고 구현하기 쉬운 장애 감지 방법이다. 장애를 감지하기 위해 별도의 루틴 없이 SNMP 에이전트에서 발생하는 트랩 메세지를 바탕으로 장애 감지를 하게 된다[2].

◆ SNMP request Messages : SNMP 트랩 메세지를 보완하여 주기적으로 각 관리 오브젝트들에 SNMP 요청 메시지를 날려 응답 여부에 따라 장애 감지를 하게 된다[2]. 즉, 응답이 오면 정상 작동 중이고, 오지 않으면 객체에 이상이 생긴 것이다. SNMP 트랩 메시지보다는 정확하지만 네트워크에 걸리는 부하가 크다.

◆ Ping/traceroute(ICMP echo/replay) : SNMP 대신 ICMP echo/reply 메시지를 이용해 주기적으로 각 관리 오브젝트의 장애 여부를 체크하는 방법이다. ICMP 메시지는 애플리케이션 레이어가 아니라 네트워크 레이어이기 때문에 SNMP에 비해 상대적으로 적은 로드로 좋은 결과를 얻을 수 있어 한번에 더 많은 폴링을 할 수 있다[6][7].

◆ 그 외에 텔넷이나 기타 시스템 고유의 프로토콜을 사용하는 방법이 있다. 이 방법들 역시 대부분이 주기적으로 관리 오브젝트를 폴링해서 응답 여부에 따라 장애 감지를 하게 된다[6][7].

2,2. Libnet & Libpcap 라이브러리

Libnet는 로우레벨(네트워크 레이어 이하) 패킷을 다루기 위한 C 라이브러리이다. Libnet은 이더넷 프레임 및 IP 패킷을 직접 조작할 수 있는 환경을 제공하고, 패킷의 헤더 필드에 대한 완전한 제어를 할 수 있다. 또한 로우레벨 네트워크 프로그래밍에서 발생할 수 있는 많은 지루한 작업들을 자동화할 수 있는 함수들을 제공한다. 어떤 사람들은 libnet을 단지 기본적인 해킹툴로만 보지만, Libnet은 매우 강력한 네트워크 보안 프로그램을 만드는데 사용될 수 있다[4].

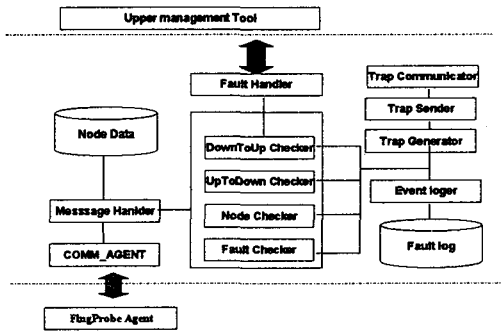
이더넷 카드의 경우 자신을 패킷만을 상위 레이어로 전달하지만 이더넷 카드에 감지되는 모든 패킷을 카피해 상위 레이어로 전달할 수 있는 기능을 제공 모드를 가지고 있다. Libpcap은 이러한 패킷 캡쳐를 위해 사용되는 라이브러리이다. 가장 좋은 사용 예는 TCPDUMP나 각종 스니퍼 프로그램들을 들 수 있다[5].

3. 시스템 설계 및 구조

우리의 시스템은 기존의 SNMP 기반의 망관리 시스템을 보완해서 사용할 수 있도록 확장성 있는 에이전트-마스터 구조를 사용하였다[3]. 즉, 기존의 중앙집중식 폴링이 방법 대신 에이전트를 이용해 네트워크의 관리 트래픽이 한 곳에 집중되지 않도록 폴링 포인트를 분산 배치해 네트워크의 관리 트래픽으로 인한 네트워크의 성능 저하를 줄일 수 있도록 하였다. 시스템은 기존의 트랩 모듈이나 SNMP 요청 폴링 부분을 대신하여 실제 장애 관련 처리나 통계 정보를 위한 로깅 기능을 수행하는 FaultMaster와 장애를 감지하는 에이전트인 FaultProber, 관리 노드 정보와 장애 로그를 위한 DB로 구성된다.

3.1 FaultMaster

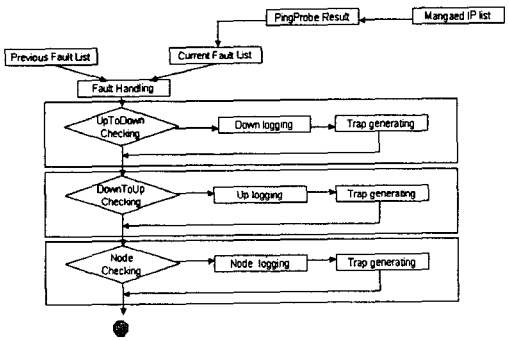
FaultMaster는 FaultProber에서 받은 장애 인터페이스 정보를 바탕으로 실제 장애 처리를 수행하는 부분이다. 탐지한 장애 정보에 대한 사용자 인터페이스로의 전송, 장애 로깅, 에이전트와의 장애 및 관리 정보 교환 기능을 수행한다. FaultMaster의 기능적 구조는 [그림1]과 같이, 사용자 인터페이스 쪽으로의 장애 정보 전송을 위한 트랩 모듈, 장애 처리 모듈, 에이전트와의 통신 모듈, DB 엑세스 모듈로 구성되어 있다.



[그림1] FaultMaster 시스템 구조

FaultMaster의 장애 처리 알고리즘에 대해 알아보면, 에이전트에서 받은 인터페이스 장애 리스트와 DB에 저장되어 이전 장애에 대한 인터페이스 장애 리스트를 비교해 장애 처리를 수행하게 된다. 장애 처리는 [그림2]에서와 같이 크게 3부분으로 나누어진다. 이전 폴링에서는 동작 중이었지만, 현재 폴링에서는 다운 된 경우, 이전 폴링에서 다운 상태였지

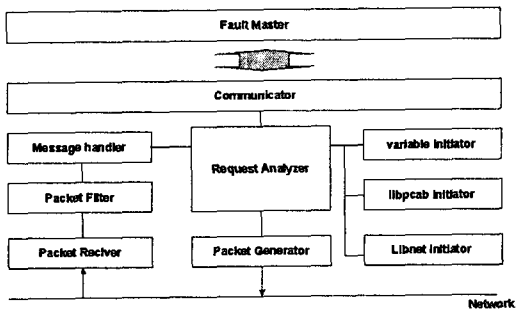
만, 현재 폴링에서는 동작 중인 경우, 마지막으로 시스템에 대한 동작여부를 판별하게 된다. 첫 번째 동작상태였다가 다운 상태로 된 인터페이스의 체크는 현재 장애 리스트의 값들 중, 이전 장애 인터페이스 리스트들 중 없는 값들을 찾는다. 여기에서 나온 값들이 동작 상태였다가 다운 상태로 된 인터페이스가 되고, 해당하는 로그를 DB에 기록한 후 사용자 인터페이스로 다운 트랩 메시지를 보내게 된다. 두 번째로 다운 상태였다가 동작 상태로 변한 인터페이스의 체크는 첫 번째 과정의 반대로, 이전 인터페이스 장애 리스트의 값 중, 현재 폴링에서의 장애 리스트 없는 값들을 검사한다. 이 값들이 다운 상태였다가 동작 상태로 된 인터페이스가 되고, 역시 해당 로그를 DB에 기록한 후 FaultMaster로 인터페이스 업에 해당하는 트랩 메시지를 보낸다. 세 번째 루틴은 시스템(관리 노드)의 장애 여부를 체크하는 과정이다. 인터페이스가 한 개인 시스템의 경우에는 앞에서 설명한 알고리즘을 그대로 사용하게 되고, 2개 이상인 경우에는 시스템의 인터페이스를 모두 체크해 시스템의 업다운 상태를 검사하게 된다. 즉, 하나 이상의 인터페이스가 업 상태이면 동작 중인 시스템이 되고, 해당 인터페이스가 모두 다운 상태이면, 시스템도 다운 인 것으로 판별한다.



[그림2] 장애 검사 알고리즘

3.2 FaultProber 에이전트

FaultProber 에이전트는 ICMP Echo/Reply 메시지를 이용하여 관리 노드의 인터페이스의 장애 여부를 조사하는 에이전트이다. FaultMaster로부터 받은 관리 대상 인터페이스 리스트에 대해 Libnet 라이브러리를 사용하여 애플리케이션 레벨에서 ICMP 메시지를 생성한다[4]. 대규모 네트워크를 지원하기 위해서 네트워크의 중요 지점들에 설치된다. FaultProber의 주요 모듈은 그림[3]과 같이, 통신포트, 및 버퍼 초기화, libnet 라이브러리 초기화, Libpcap 라이브러리 초기화를 하는 초기화 모듈들, ICMP 메시지를 생성하는 Packet generator, ICMP 메시지를 받은 Packet Reciver, 필요한 ICMP Replay 메시지만 선별하는 Packet Filter, FaultMaster로 보낼 리턴 값을 만드는 Message Handler 부분과 실제 마스터 부분과 통신을 수행하는 communicator로 구성된다.



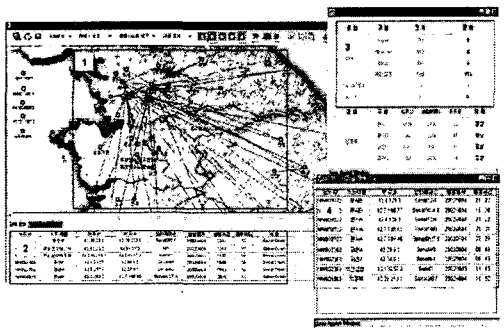
[그림3] FaultProber 에이전트 시스템 구조

4. 시스템 구현

우리의 시스템은 Raw 소켓을 사용함으로써 짧은 시간에 많은 패킷을 생성할 수 있어, 한번에 많은 인터페이스의 장애 여부를 조사할 수 있다. 실제로 Libnet을 사용하여 프로그램 작성 시 많은 패킷을 한번에 전송할 때 패킷 손실이 일어난다. 테스트에 의하면 초당 500개 이상의 패킷을 한번에 전송할 때 패킷 손실이 일어나며, 타 네트워크까지 전송을 염두에 둘 경우 초당 200개 이상의 패킷을 전송하지 않는 것이 안정적이라는 결과를 얻었다. 또한 초당 200개의 패킷이라 할지라도 매 초 200개씩 4000개의 패킷을 연속으로 전송해본 결과, 경유 네트워크의 트래픽 문제로 많은 패킷들이 손실되는 것을 발견할 수 있었다. 본 시스템은 각 인터페이스 별로 2개의 ICMP 메시지를 생성하여 전송하며, 응답이 오지 않는 주소를 체크해 다시 테스트를 반복하는 방식을 사용하였다. 1회 2개 패킷을 3회에 걸쳐서 보냈을 때도 응답이 없는 서버는 장애가 발생했다고 판정하였고, 신뢰할 수 있는 결과를 얻었다. 테스트 시 4회째부터의 결과는 3회째와 같은 결과를 나타내었다. 기존의 SNMP 폴링 방법에서는 1000~1500개 사이의 인터페이스를 폴링 하는데 10분 이상이 소요되었지만 우리의 시스템에서는 1분 주기로 폴링했을 경우

에도 SNMP를 사용했을 때 보다 더 좋은 결과를 얻었다.

[그림4]는 SNMP 기반의 망관리 시스템과 FaultMaster를 결합하여 장애 관리를 수행하는 예를 보여 준다. FaultMaster와 에이전트인 FaultProber는 설정에 따라 주기적으로 동작하며 관리 노드 및 인터페이스의 상태를 체크해 그 결과를 NMS로 보낸다. 네트워크 토폴로지와 연동해 관리자가 별도의 다른 동작 없이 관리 네트워크의 상태를 쉽게 모니터링 할 수 있고, 장애 정보 테이블에 장애 노드에 대한 상세한 정보를 표시하게 된다. 또한 토폴로지 상태 표시와 더불어 관리 네트워크를 카테고리별로 분류해 장애 상태에 대한 기본적인 통계를 보여 준다. 그 외에 장애 로그를 바탕으로 노드별 총 장애 시간, 시간대별 장애수, 관리 도메인별 장애수등 다양한 포맷으로 관리자에게 제공해준다.



[그림4] 기존의 NMS와의 연동

각 모듈을 컴포넌트화하여 필요한 부분만을 쉽게 다른 시스템에 적용할 수 있게 하였다. 마스터쪽 모듈은 자바를 사용하였고, FaultProber 모듈은 Libnet과 Libpcap 라이브러리를 사용하기 위해 C를 사용하여 구현하였고, 에이전트와 통신을 위해서 TCP를 사용하였다.

5. 결론 및 향후 계획

지금까지 우리가 구현한 ICMP 메시지 기반의 장애 탐지 시스템에 대해 살펴보았다. 우리의 시스템은 단순하고 확정성 있는 시스템이며, 또한 분산 망관리 환경에 적합하도록 설계되었다. 현재는 기본적인 장애 관리 기능인 장애 탐지와 장애에 대한 통계 정보 제공의 기능을 제공한다.

장애의 빠른 탐지도 중요하지만 그보다 더 중요한 것은 장애에 대한 신속한 복구이다. 현재의 SNMP나 ICMP 기반에서는 장애 탐지는 가능 하지만 장애 복구는 불가능하다. 자동화된 장애 탐지 툴은 많이 있지만, 그에 비해 자동화된 장애 복구 솔루션은 이제 연구 및 시작 단계이다[8]. 또한 장애가 일어나기 전에 미리 예측할 수 있다면 장애 때문에 발생하는 비용보다 훨씬 적인 비용으로 장애를 예방하여 언제나 최적의 네트워크 환경을 유지할 수 있을 것이다. 보통 장애가 일어나기 전에는 장애를 예측할 수 있는 정상적인 경우와는 다른 현상이 생기는 경우가 많다. 이러한 경우들을 체계적으로 분류하여 사용할 수 있는 장애 예측 알고리즘에 대한 연구도 필요할 것이다.

향후에는 자동화된 복구 기능과 장애 예측 모듈을 추가해 보다 어느 환경에서나 적합한 장애 관리 시스템으로 발전해 나갈 것이다.

참고문헌

[1] Aiko. Pras, "Network Management Architectures", CTIT

[2] David Perkins, Evan McGinnis, "Understanding SNMP MIBs", Prentice Hall PTR

[3] Heinz-Gerd Hegering, Sebastian Abeck. and Bernhard Neumair, "Integrated Management of Networked Systems: Concepts, Architectures, and Their Operational Application", Morgan Kaufmann Publishers

[4] http://libnet.sourceforge.net

[5] http://www.cet.nau.edu/~mc8/Socket/Tutorials

[6] Ramesh Govindan, Hongsuda Tangmunarunkit, "Heuristics for Internet Map Discovery", CA90292

[7] Nelson Tang, Binay Sugla, "Netmap: A Network Discovery Tool", Lucent Tech, 1998

[8] J.-F. Huard and A. A. Lazar, "Fault Isolation based on Decision-Theoretic Troubleshooting", CTR Tech. Rep. 442-96-08, Feb. 1996

[9] Merin Hughes, Michael Shoffener, and Derek Hamner, "JAVA Network Programming", 2nd Ed., Manning