논문 01-26-6A-16 한국통신학회논문지 '01-6 Vol.26 No.6A

SNMP 트래픽 최적화를 위한 폴링 방식에 관한 연구

정회원 김 민 우*, 박 승 균*, 오 영 환*

A Study on the Polling Mechanism for Optimizing the SNMP Traffics

Min-Woo Kim*, Seung-gyun Park*, Young-hwan Oh* ***Regular Members***

요 약

인터넷 보급의 확대와 서비스의 다양화에 따라 사용자 트래픽의 증가와 서비스에 대한 질적인 요구가 증가함으로써 이에 대응하는 인터넷 관리 또한 그 추세를 망 구성요소에 대한 관리뿐만 아니라 서비스나 어플리케이션 측면의 관리로 그 방향을 전환하고 있다. 이러한 추세는 망 관리 시스템의 관리기능 강화를 가져오게 되고 그에 따라 망 관리 트래픽의 증가를 가져오게 된다. 이러한 망 관리 트래픽의 증가는 오히려 사용자 트래픽의 서비스 저하를 가져오기도 한다.

본 논문에서는 관리 어플리케이션들이 발생시키는 관리정보 갱신요구 중에서 중복되는 트래픽을 통합하기 위한 폴링 계층을 해결책으로써 제안한다. 제안한 폴링 계층은 Controller와 Record Table 그리고 Poller 세가지로써 구성하였다. 그리고 트래픽의 통합에 대한 효율을 높이기 위해서 시간원점이라는 개념을 사용하여 관리정보 갱신요구의 발생이 주기적으로 어긋나는 것을 최대한 억제하도록 하였다. 또한 관리정보의 실시간적인 획득을 위하여 기존 연구에서 트래픽 통합을 위해 제시하였던 대기시간을 줄이기 위한 방안을 제시하였다.

ABSTRACT

As the Internet is more popular and has more various services, it increases user traffics and demands on quality of service. Hence, focus in the Internet management is shifted toward service- and application-targeted management from network component-level management. It leads to increase management traffics, which is caused by intensive management functions of a NMS(Network Management System). If management traffics increase, it could decrease service performance of user traffics.

In this thesis, we propose a polling layer as solution for integration of overlapped traffics among whole traffics by management applications. The proposed polling layer is composed of three components which are Controller, Record Table, and Poller. And, for the purpose of promoting efficiency of traffic integration, we restrain demand of management information refreshment from being made crisscross by using notion of time origin. In addition, for the real-time gathering management informations, we propose to reduce waiting-time proposed by existing research for optimizing management traffics.

Ⅰ. 서 론

인터넷의 보급이 확대되고 사용자계층이 다양해지면서 인터넷을 통한 서비스 또한 다양해지게 되었다. 이전의 텍스트 위주의 문자정보에서 벗어나 영상회의(video conference)나 인터넷 폰(internet phone)등과 같은 멀티 미디어 환경에 적합한 서비스들도 등장하게 된다. 이제는 더 이상 망에 이상이 생겼을 때 아이콘의 색깔 변화로 나타내어지던 NMS (Network Management System)를 사용하던 시대는

* 광운 대학교 통신망 연구실(minuo519@explore.kwangwoon.ac.kr)
논문번호 : 00495-1230, 접수일자 : 2000년 12월 30일

지나갔다. 대량의 트래픽 전송과 서비스의 품질을 만족시키기 위해서 이전의 망 구성요소에 대한 관리에서 벗어나 점차적으로 서비스나 어플리케이션 측면의 관리쪽으로 이동하는 추세이다[1][2]. 이에 따라서 관리 대상이 되는 MIB가 증가되고 NMS의 기능성-더욱 더 강화된 모니터링 기능-에 대한 요구가 늘어나게 되었다. 이 두가지 사실은 관리 대상 망에 현저한 트래픽의 증가를 가져오게 되었으며 망 관리 트래픽이 사용자 트래픽에 주는 영향은 오히려 서비스의 질적인 저하를 가져오게 되는 것이다[3]. 이러한 망 관리를 위한 NMS의 기능성에 대한 요구를 충족시키고 망 관리 트래픽의 최적화를 위해 등장한 것이 폴링 계층의 삽입과 캐싱의 개념의 도입이라 할 수 있다[4][5]. 폴링 계층을 삽입함으로써 망 관리 어플리케이션에서 중복적으로 발생하는 관리정보 갱신 요구들을 감소시키고자 하고 캐싱 개념에서는 일정한 시간 간격을 두고 SNMP 에이전트들로부터 수집한 관리정보들을 특정한 임시저장소(cache)에 저장한 후 여러 관리 어플리케이션들이 공동으로 사용할 수 있도록 하는 것이다. 그러나 이러한 방식들은 관리 정보를 어플리케이션이 원하는 그 시간에 가져올 수 없다는 단점을 갖고 있다.

이러한 단점을 극복하기 위해서 본 논문에서는 폴링 계층을 사용하여 망 관리 트래픽을 감소시키면서 대기시간을 없앨 수 있는 방안을 제시한다. 본 논문에서의 폴링 계층은 관리정보의 수집을 위해서 SNMP를 망 관리 프로토콜로 사용한다.

본 논문은 2장에서 SNMP의 관리정보 수집방법에 대해서 살펴보고 3장에서 제안한 폴링 계층의 구조와 동작에, 그리고 4장에서 제안한 폴링 계층의 성능에 대해서 보여주고 이전에 제안 되었던 방식과 비교 분석한다. 마지막으로 5장에서 이 논문을 결론 짓고자 한다.

Ⅱ. SNMP를 이용한 망 관리

2.1 SNMP (Simple Network Management Protocol)

SNMP 표준은 망 구성 요소에 관한 관리정보가 논리적으로 원격지 사용자들에 의해 조사되거나 변경될 수 있게 지원하는 간단한 표준 프로토콜을 정의 한다. SNMP는 TCP/IP를 기반으로 하는 망들을 관리하기 위해 간단하게 운영 가능한 구조의 시스템을 제공한다. SNMP의 기본 원리는 감시 데이터가 너무 빈번히 왕래하여 다른 데이터의 소통을 방해하지 않도록 망 자체의 기능에 대한 부담을 최소화 하는 것이다.

2.1.1 SNMP의 구성

SNMP의 구조적인 모델은 관리 시스템(manager)과 피관리 시스템들로 이루어진다. 관리 시스템들은 피관리 시스템들을 감시하고 통제하는 관리 어플리케이션들을 수행한다. 피관리 시스템으로는 호스트, 게이트웨이, 단말기, 서버등과 같은 장비이며, 이 장비에는 망 관리 스테이션들로부터 요청된 망 관리 기능들을 수행하는데 책임이 있는 관리 수행자(agent)가 있다. SNMP에서 관리 시스템과 피관리 시스템 사이의 관리정보 교환의 구조는 그림1과 같다.

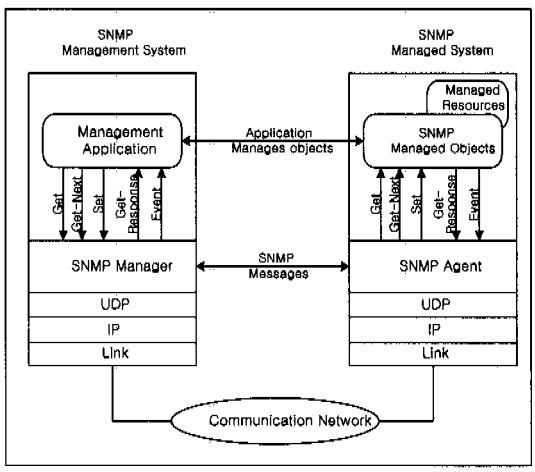


그림 1. SNMP의 구조

2.1.2 SNMP 메시지

관리 시스템과 피관리 시스템이 주고 받는 정보의 대부분은 피관리 자원에 관련된 정보이다. 정보에는 관리자가 관리자원의 정보를 요구하면 피관리 시스템이 그에 따른 정보를 검색하여 응답하거나, 관리자가 적절한 조치 사항과 함께 정보를 넘겨주어 피관리 시스템이 피관리 자원에 대한 상응된 조치 및 그에 따른 정보 반영 작업을 하게 하는 것, 피관리 시스템이 먼저 관리자에게 정보 교환 요구를 시작하는 것 등이 있다. 구체적으로 SNMP 메시지에는 다음과 같은 다섯가지가 있다.

- GetRequest : 피관리 시스템이 가지고 있는 관리 정보를 관리 시스템이 수집한다.
- GetNextRequest
- SetRequest : 관리 시스템의 지시로 피관리 시

스템이 가지고 있는 관리정보의 값을 설정한다.

- GetResponse : 관리 시스템의 명령에 대해 피관리 시스템이 응답하는데 사용한다.
- Trap : 기기의 장애등 이상상태가 피관리 시스템에 발생했을 때에 피관리 시스템이 그 이벤트 정보를 관리 시스템에 통지한다.

2.1.3 SNMP의 관리정보 획득방법

TCP/IP 기반의 망 관리 표준인 SNMP에서 관리정보를 교환하는 방식은 사건보고 방식과 폴링 방식이 있다.

사건보고 방식의 경우에는 관리 시스템이 별도의 관리정보 요구를 하지 않으며, 사건 보고가 없는 피관리 시스템은 기본적으로 사건이 일어나지 않은 것으로 판단한다.

폴링 방식의 경우에는 관리 시스템이 여러 피관리 시스템들에게 관리정보를 요구하기 위하여 명령을 보내고 응답 패킷을 수신하는 것이다. 관리 시스템과 피관리 시스템들 사이의 관계는 일대일의 논리적인 연결 관계이고, 피관리 시스템들 서로 간에는 아무런 연결관계가 없다. 이와 같은 폴링 방법의 장점은 피관리 시스템들이 응답을 하지 않는 경우에는 이상이 발생하였음을 판단할 수 있어 관리 행위에 신뢰성이 있다는 것이다.

그러나 위의 폴링 방법은 관리 시스템에서 피관리 시스템들로 폴링하는 주기가 짧아지거나 관리 시스템 내의 관리 어플리케이션의 수가 증가하게 되면 이에 비례하여 관리정보 패킷의 수가 증가하게 되어 중간 노드나 관리 시스템이 속한 망이 혼잡해질 가능성이 있다. 또한 이러한 관리정보의 폭주로 인한 망 혼잡 가능성은 관리대상 MIB가 증가하고 관리 시스템의 기능성이 높아짐에 따라서 증가한다.

2.2 기존 연구의 고찰

SNMP의 표준 모델을 사용하였을 때, 관리 대상 MIB가 증가하고 관리 시스템 내에서 관리 어플리케이션의 수가 증가할 때, 중복된 관리정보 패킷을 발생하게 되는데 이러한 중복된 관리정보 패킷은 망의 효율을 떨어뜨린다. 이러한 단점을 보완하기 위해서 폴링 계층을 사용하는 폴링 방식이 등장하게 되었다.

관리 어플리케이션과 SNMP 에이전트 사이에 폴링 계층을 삽입함으로써 여러 관리 어플리케이션으로부터 관리정보 갱신 요구를 받아들여서 공통된 폴링 패킷을 생성하는게 그 목적이다. 기존 연구에서는 이 부분에 이미지(image)라는 요소(component)를 사용하여 관리 어플리케이션내의 객체 속성(Object Attribute)의 특징을 정의하고 그 이미지에 의해서 여러 관리 어플리케이션을 통합하고자 한다. 이때, 5가지의 파라미터를 사용하여 특징을 정의하는데 이때에 대부분의 파라미터가 시간에 관한 것이고 또한 그 파라미터들로 시스템 내에서의 대기시간을 정의하고 있다. 그 대기시간에 의해서 더 많은 수의 관리정보 갱신 요구를 통합할 수 있는 것이다. 그러나 여러 개의 파라미터를 사용하면서 각각의 파라미터에 대응하는 시스템 자원들을 많이 필요로 하게 되었다. 그리고 관리정보 갱신 요구를 통합하기 위해서 각각의 요구들이 시스템 내에서 대기하는 시간을 갖게 됨으로서 관리정보의 실시간성이 떨어지는 결과를 초래하게 되었다.

Ⅲ. 제안한 폴링 방식

제안한 폴링 방식은 기본적으로 SNMP의 정보교환 모델을 기초로 하고 있다. 본 논문이 제안하는 폴링방식이 사용되기 위한 망 관리 시스템 모델은 다음과 같다.

3.1 망 관리 시스템 모델

본 논문에서 사용하는 망관리 시스템 모델은 그림 2에서 보여지는 것처럼 계층(layer) 개념을 사용한다. 본 모델은 관리 어플리케이션 계층과 폴링 계층으로 구성되어있다.

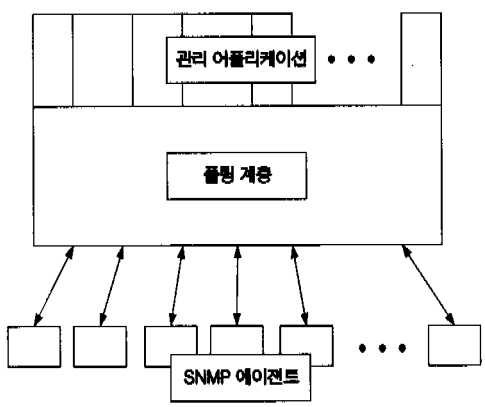


그림 2. 망 관리 시스템 모델

관리 어플리케이션 계층에서의 각각의 관리 어플리케이션들은 관리되는 망의 기능적인 요구에 의해서 추가되거나 감소될 수 있다. 그러므로 관리 어플

리케이션들은 새로운 관리어플리케이션의 시작과 이미 동작하고 있는 어플리케이션의 종료가 다른 어플리케이션들에 영향을 끼치지 않도록 각각 독립적으로 동작하게 된다. 또한 각각의 관리 어플리케이션은 그 동작 특성에 따라 독립적인 관리정보 갱신 주기를 갖는다.

폴링 계층의 목적은 다른 동작특성을 갖는 상위 관리 어플리케이션들의 관리정보 갱신요구를 적은 수의 SNMP 메시지로 통합하는 것이다. 이때에 관리 어플리케이션에서는 폴링 계층을 SNMP 에이전트로서 인식하게 되고, SNMP 에이전트에서는 폴링 계층을 관리 어플리케이션으로써 인식한다. 여기서 폴링 계층은 관리정보 갱신요구와 그 응답을 SNMP 에이전트와 관리 어플리케이션에 전달할 책임이 있다.

3.2 폴링 계층의 구조

본 논문에서 제안한 폴링 계층의 구성은 다음과 같다.

- Controller : 관리 어플리케이션으로부터의 관리정보 갱신요구를 수집하고 수집된 요구를 Record Table에 넘겨준다. 그리고 동시에 Poller에 SNMP 패킷을 만들도록 요구한다. 또한 SNMP 에이전트들로부터의 응답을 관리 어플리케이션에 분배하는 기능을 한다.
- Record Table : 관리정보 갱신요구를 각 관리 어플리케이션과 목적 에이전트 그리고 그 변수명에 따라 기록하고 그 값을 응답이 되돌아올 때까지 유지한다.
- Poller : Controller의 요구에 따라서 Record Table을 감시하고 그 값을 이용하여 SNMP 패킷을 작성하고 해당 에이전트에 전송한다. 또한 전송후 에이전트의 응답을 감시하는 역할을 한다.

폴링 계층의 구조는 그림 3과 같으며 관리정보 갱신 요구의 흐름은 실선으로 그리고 응답의 흐름은 점선으로 표시하였다.

관리정보 통합을 위해 사용되는 폴링 계층에서 관리정보 갱신요구의 송수신 절차를 살펴보면 다음과 같다.

관리정보 갱신요구의 송신을 위한 절차

단계 0.
- Controller가 관리 어플리케이션들의 관리정보 갱신요구를 감시

단계 1
- Controller에 도착한 관리정보 갱신요구를 관리 어플리케이션의 주소와 에이전트의 주소 그리고 대상이 되는 관리정보 변수 별로 분류, Record Table에 기록한다.

단계 2
- Controller가 Poller로 SNMP 패킷 작성을 요구한다.

단계 3
- Poller에서 SNMP 패킷을 작성하고 각각의 SNMP 에이전트로 송신

관리정보 갱신응답의 수신을 위한 절차

단계 0.
- Poller는 응답여부를 감시

단계 1
- Poller에 수신된 응답을 Controller에 전달

단계 2
- Controller는 응답이 어떤 관리 어플리케이션의 객체속성으로 가는지를 Record Table의 기록에서 확인 후 전달

단계 3
- 전달한 응답에 대한 기록을 Record Table에서 삭제

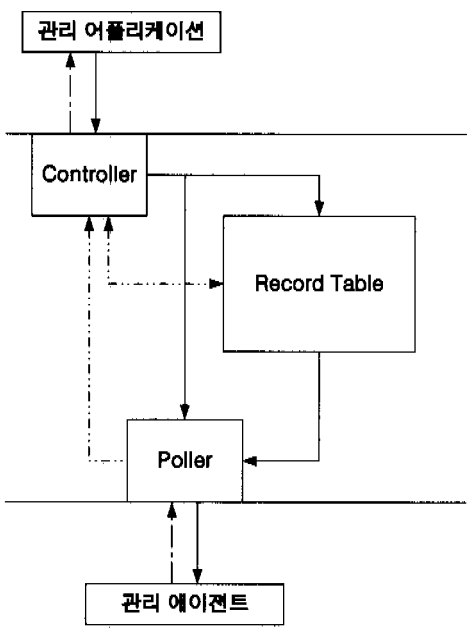
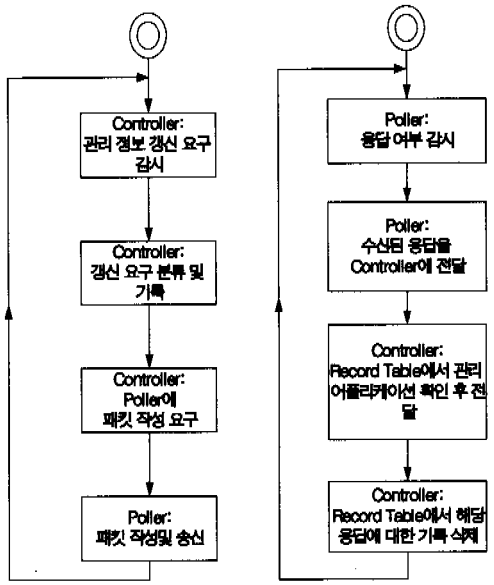


그림 3. 제안한 폴링 계층의 구조



(a) 송신시 (b) 수신시

그림 4. 관리정보 갱신을 위한 송수신 절차

그림 4의 수행 절차는 폴링 계층 운용시에 관리정보의 갱신 요구를 SNMP 패킷으로 작성하여 송신하는 과정과 그 응답을 수신하는 절차를 보여준다. 그림 4(a)에서 두 번째 단계와 네 번째 단계 즉, 관리정보 갱신요구를 분류해서 기록하고 패킷작성을 하는 단계에서 본 논문에서 제안한 폴링 알고리즘을 적용한다. 그림 4(b)는 제안한 폴링 알고리즘에 의해 통합된 관리정보 갱신요구의 응답을 관리 어플리케이션에 분배하는 과정을 나타낸 것으로 통합된 관리정보 응답을 Controller가 Record Table에서 해당하는 관리 어플리케이션의 객체 속성을 확인한 후 전달하는 과정을 나타낸 것이다.

3.3 폴링 알고리즘

제안한 알고리즘은 각각의 관리 어플리케이션으로부터 발생되는 관리정보 갱신요구를 통합하기 위한 알고리즘이다. 각각의 관리 어플리케이션은 서로 독립적으로 동작을 하며 그 관리정보 갱신주기 또한 독립적이다. 이때에 서로 다른 관리 어플리케이션들의 폴링의 동기를 맞춰주기 위해서 시간원점(t_0)라는 개념을 둔다. 여기서 시간원점은 각각의 관리 어플리케이션들이 동작을 시작하려 할 때, 그 시작시간에 대한 참조점이 되는 것이다. 예를 들어 서로 독립적으로 동작하는 두 개의 관리 어플리케이션이 있고 두 관리 어플리케이션의 관리정보 갱신 주기가 같다고 가정을 했을 때, 만약 시작시간에 대한 참조점이 없다면 두 관리 어플리케이션의 관리정보 갱신 요구는 통합되어질 수 없을 것이다. 그림4는 시간원점에 대한 개념을 설명하고 있다. 그림 4에서는 관리정보 갱신주기가 Δt_1인 관리 어플리케이션(A_1)과 관리정보 갱신주기 Δt_2인 관리 어플리케이션(A_2)간에 시간원점을 사용할 때와 하지 않을 때를 그림으로 설명하고 있다. 그림 5(b)에서 살펴보면 A_1이 관리정보 갱신을 요구하고 그 후에 약간의 시간간격으로 A_2가 관리정보 갱신을 요구한다. 그러한 동작을 그림 5(a)와 비교해서 보면 전체 관리정보 갱신요구의 수가 많음을 알 수 있다.

시간원점을 통한 관리 어플리케이션들간의 관리정보 갱신요구의 발생 시점을 최대한 통합하고 관리 어플리케이션의 객체 속성이 대응하는 에이전트 내의 변수(variable)이름에 따라서 같은 변수로 대응하는 객체속성을 통합하게 된다. 그 이후에 같은 에이전트로 가는 폴링 메시지들을 하나의 SNMP 패킷으로 통합하게 된다. 변수의 이름과 SNMP 에이전트에 의한 통합은 Controller가 Record Table에 관리정보 갱신요구를 기록하면서 이루어지게 된다.

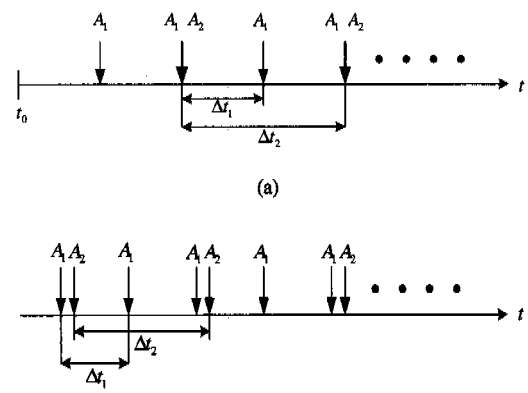


그림 5. 관리정보 갱신주기 (a) 시간원점을 사용했을 때 (b) 시간원점을 사용하지 않았을 때

관리정보 갱신요구를 기록할 때에 SNMP 에이전트의 이름, 변수명, 관리 어플리케이션의 객체속성의 이름 순으로 기록하게 된다. 이 때 객체 속성에는 변수에 해당하는 모든 객체 속성의 이름을 기록하게 된다. 그림 6은 Record Table의 작성 예이다. 이러한 기록이 끝나면 Controller는 Poller에 SNMP 패킷 작성을 요구하게 되는데 이때에 SNMP 패킷

의 최대크기는 484옥텟임을 감안하면 최대로 포함될 수 있는 변수의 개수는 약 20개로 한정된다.

SNMP 에이전트 1	변수 1	객체 속성
	변수 2	객체 속성
	변수 3	객체 속성
	•	
	•	
	•	
	변수 N	
SNMP 에이전트 2	변수 a	
	변수 b	
	•	
	•	
	변수 x	
•		
•		
•		
•		
•		
SNMP 에이전트 이름	변수명	해당 객체 속성명

그림 6. Record Table 작성의 예

만약 하나의 SNMP 에이전트에 동시에 보내어지는 관리정보 갱신요구 대상이 되어지는 변수의 수가 20개가 넘을 경우는 또하나의 SNMP 패킷을 추가적으로 요구하게 된다.

Ⅳ. 성능 분석 및 고찰

본 논문에서 제안한 폴링 방식을 평가하기 위해서 다음과 같은 성능분석 환경을 두었다.

- 관리 어플리케이션의 수나 SNMP 에이전트의 수는 랜덤하게 결정한다.
- 관리 어플리케이션은 랜덤한 수의 객체 속성을 갖는다.
- SNMP 에이전트는 랜덤한 수의 관리정보 변수를 갖는다.
- 관리 어플리케이션의 객체 속성은 SNMP 에이전트의 변수에 랜덤하게 대응된다.
- 각각의 관리 어플리케이션의 관리정보 갱신주기는 랜덤하게 결정하나 10초 단위로 한다.
- 망 관리 시스템에서 에이전트까지의 전송지연은 무시한다.
- 통합을 위하여 시스템 내에 대기하는 시간 외의 시스템의 처리시간은 고려하지 않는다.

이와같은 조건하에서 실험을 하고 결과를 관리정보 갱신 요구가 낮을 때와 높을 때 두가지로 나누어서 그 차이를 산출하고 기존 연구와의 성능차이를 알아본다. 또한 시스템 내에서의 대기시간을 분석해 본다.

4.1 관리정보 트래픽의 분석

4.1.1 낮은 관리정보 갱신요구하에서의 분석

그림 7과 그림 8은 관리 어플리케이션의 수가 적을 때 발생하는 결과이다. 평균 86개의 관리정보 갱신요구가 발생하였고 제안한 폴링 방식을 사용하였을 때에 평균 74개의 패킷이 발생하고 기존의 폴링 계층을 사용한 방식에서는 평균 70개의 패킷이 발생한다. 본 논문에서 제안한 폴링방식을 사용했을 때는 약 14%정도의 패킷 감소율을 보였으며 기존 방식을 사용하였을 때는 약 19%의 패킷 감소율을 보였다. 기존의 방식이 약 5%의 성능우위를 보인다.

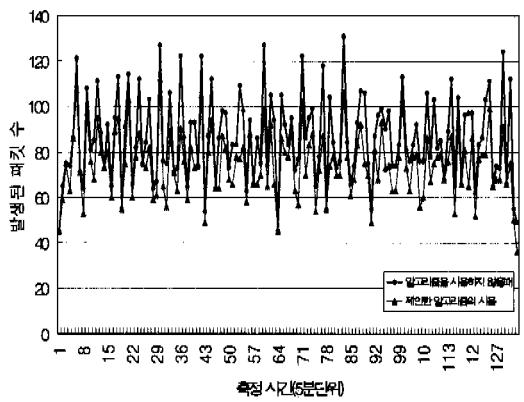


그림 7. 낮은 관리정보 갱신요구일 때 트래픽 분석

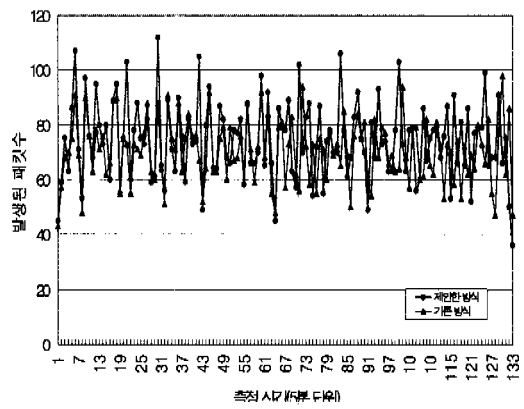


그림 8. 낮은 관리정보 갱신요구일 때 기존방식과의 비교

4.1.2 높은 관리정보 갱신요구하에서의 성능분석

그림 9과 10는 관리 어플리케이션의 수가 많을 때 발생된 결과이다. 평균 288개의 관리정보 갱신요구가 발생하였으며 제안한 방식에서 평균 124개의 패킷이 발생 그리고 기존의 폴링계층을 사용하는 방식에서는 평균 111개의 패킷이 발생하였다. 본 논문에서 제안한 방식을 사용했을 때는 약 57%의 패킷 감소량을 보였고 기존방식을 사용했을 때는 약 62%의 패킷 감소량을 보였다. 높은 관리정보 갱신요구하에서도 역시 기존 방식이 약 5%정도의 우위를 보인다.

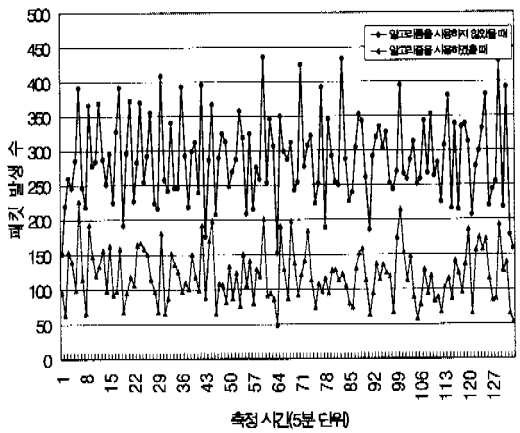


그림 9. 높은 관리정보 갱신요구일 때 트래픽 분석

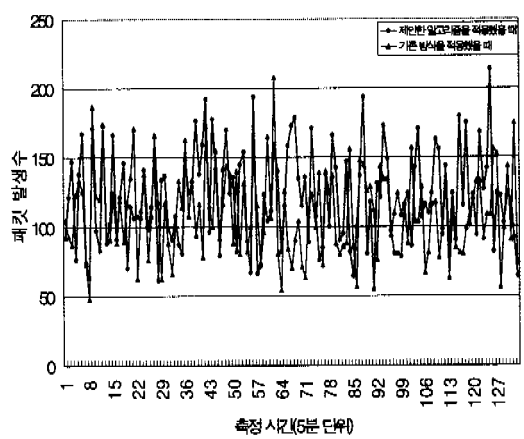


그림 10. 높은 관리정보 갱신요구일 때 기존방식과의 비교

4.2 시스템 내에서의 대기 시간 비교

그림 11은 관리정보 갱신요구를 통합하기 위해서 각각의 관리정보 갱신요구가 시스템 내에서 대기하는 시간이 관리 어플리케이션의 수에 따라서 평균적으로 어떻게 변해가는지를 나타낸 그래프이다. 기존 방식에서는 관리 어플리케이션이 증가함에 따라서 대기시간이 감소하는 것이 보이지만 약 40초에서 80초의 대기시간을 보이는 반면 본 논문에서 제안한 방식에서는 관리 어플리케이션의 수와 상관없이 대기시간이 없음을 알 수 있다.

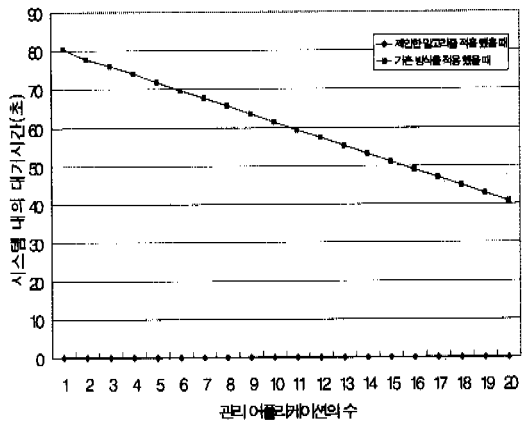


그림 11. 기존방식과의 대기시간 비교

4.3 고찰

이상의 결과에서 살펴보면 기존방식과 제안한 방식 모두 관리 어플리케이션이 증가함에 따라서 발생하는 관리 대상이 되는 망에 대한 관리 트래픽의 부담을 줄일 수 있었다. 그리고 패킷 감소율로만 보자면 기존의 방식이 관리정보 갱신 요구의 정도에 상관없이 약 5%의 우위를 보인다. 그러나 관리정보 갱신 요구의 통합을 위해서 시스템 내에서 대기하는 시간들 평균적으로 따져 보면 기존 방식에서는 약 40~80초의 시간동안 다른 요구가 들어오는지 기다려야 하므로 관리정보의 실시간성이 떨어진다고 할 수 있다. 관리정보의 실시간성은 서비스나 어플리케이션 측면의 관리에서 중요한 역할을 한다. 예를 들면 시스템에 대한 성능관리를 위해서 MIB에서 필요한 정보를 추출해 낼 때, 정확한 통계관리를 위해서 관리 어플리케이션이 요구하는 시점에서 정보를 가져와야 할 필요성이 있다. 따라서 망의 사용율이 높아지고 서비스가 다양해짐에 따라서 관리정보 패킷 발생율을 낮추는 것과 관리정보의 실시간성 모두 무시할 수 없는 상황이 되었다. 따라서 어떠한 측면을 중시하여 망관리를 할 것인가에 대한

선택은 망을 사용하는 사용자의 성향에 따라서 망 관리자가 효율적인 망관리를 할 수 있는 방법을 선택하여야 할 것이다.

V. 결 론

본 논문에서는 인터넷 망 관리에서 증가하는 망 관리 트래픽을 억제하면서 관리정보의 실시간성을 유지할 수 있는 방안으로써 새로운 형태의 폴링 계층을 사용하는 방식을 제안하였다. 제안한 폴링 방식은 각각의 관리 어플리케이션에서 발생하는 중복된 관리정보 갱신요구를 효과적으로 통합하기 위해서 시간 원점의 개념을 사용하여 관리정보 갱신요구의 발생이 서로 어긋남으로써 추가적인 SNMP 트래픽이 발생되는 것을 억제하고 Record Table을 이용함으로써 같은 시점에 발생하는 관리정보 갱신요구들을 통합하였다. 또한 관리정보의 실시간적인 획득을 위하여 기존방식에서 사용되었던 시스템 내의 관리정보 갱신요구의 대기시간을 없앰으로써 관리정보의 실시간성을 높였다.

성능분석을 통하여 비교한 결과를 보면 트래픽 억제 측면에서는 낮은 관리정보 갱신요구하에서보다는 높은 갱신요구하에서 더 많은 트래픽 통합이 이루어지는 것을 볼 수 있었다. 그럼으로써 NMS의 기능강화에 따른 관리 어플리케이션이 증가되더라도 관리 대상이 되는 망에 끼치는 관리 트래픽의 영향을 줄일 수 있음을 알 수 있었다. 그리고 기존의 방식이 제안한 방식보다 평균 5%정도의 추가적인 패킷감소를 가져왔지만 대기시간이 길어서 관리정보의 실시간적인 측면에서는 제안한 방식이 효과적임을 볼 수 있었다. 또한 관리 어플리케이션의 수가 증가하여 관리정보 갱신요구가 크게 증가하더라도 관리 트래픽의 증가는 그 정도가 미미하여 제안한 폴링 방식을 사용할 때 관리 어플리케이션의 수가 관리 대상 망에 끼치는 영향을 줄일 수 있었다.

참 고 문 헌

[1] P. G. S. Flofissi, Y. Yemini and D. Florissi, "QoSockets: a New Extension to the Sockets API for End-to-End Application QoS Management", Proceedings of the Sixth IFIP/IEEE International Symposium on Integrated Network Management, pp. 655-668, 1999.

[2] P. Parnes, K. Synnes and D. Schefstrom, "Real-time Control and Management of Distributed Applications using IP-multicast", Proceedings of the Sixth IFIP/IEEE International Symposium on Integrated Network Management, pp. 901-914, 1999.

[3] C. Amley, "Network Management's Impact on Managed Networks", Proceedings of the 19th Conference on Local Computer Networks, pp. 404-410, 1994.

[4] M. Chekhrouhou and J. Labetoulle, "An Efficient Polling Layer for SNMP", Proceedings of the 2000 IEEE/IFIP Network Operations and Management Symposium, pp. 477-490, 2000.

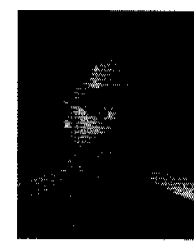
[5] F. Stamatelopoulos and B. Maglaris, "A Caching Model for Efficient Distributed Network and System Management", Proceedings of the 3rd IEEE International Symposium on Computers and Communications, pp. 226-230, 1998.

[6] W. Stallings, SNMP, SNMPv2, SNMPv3, and RMON 1 and 2, Addison-Wesley, 1999.

[7] M. A. Miller, Managing Internetworks with SNMP, M&T Books, 1999.

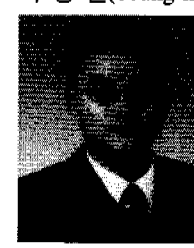
[8] 대한민국 전산망 표준, "단순 망 관리 규약 (SNMP) 표준", 체신부, 1993. 12.

김 민 우(Min-woo Kim) 정회원



1999년 2월 : 광운대학교 전자통신공학과 졸업
2001년 2월 : 광운대학교 전자통신공학과 석사
<주관심 분야> SNMP, TMN 체계의 통신망 운용 관리

박 승 균(Seung-kyun Park) 정회원



1993년 2월 : 광운대학교 전자통신공학과 졸업
1995년 2월 : 광운대학교 전자통신공학과 석사
1999년 2월~ 현재 : 광운대학교 전자통신공학과 박사과정

<주관심 분야> SNMP, TMN 체계의 통신망 운용 관리

오 영 환(Young-hwan Oh) 정회원
·한국 통신학회 논문지 제 26권 제 4A호 참조