韓國컴퓨터情報學會 論文誌
第15卷 第2號, 2010. 2.

2010-15-2-3-2

네트워크 관리 프로토콜 SNMP의 성능 향상

나 호 진*, 조 경 산**

Improving the Performance of Network Management Protocol SNMP

Ho-Jin Na*, Kyungsan Cho**

요 약

규모가 커지고 라우터, 스위치, 서버 등과 같은 구성 요소가 다양해지고 있는 네트워크를 효율적으로 지원할 수 있는 SNMP(Simple Network Management Protocol)가 네트워크 관리의 표준 프로토콜로 가장 널리 사용되고 있다. 하지만, SNMP는 네트워크 과부하, 처리 지연 그리고 테이블 검색의 비효율성과 같은 성능적 문제점이 있다. 본 연구에서는 SNMP의 성능 개선을 위해 SNMP-GetBulk 응답메시지 내의 중복되는 OID 정보를 줄이는 기법과 NMS 캐쉬와 새로운 SNMP-GetUpdate 요청이 결합된 기법의 두 가지 제안을 제시한다. 분석을 통해 첫 번째 제안 기법은 네트워크 과부하를 감소시키고, 두 번째 기법은 SNMP 요청의 처리 지연과 테이블 검색 문제를 개선함을 보였고, 이로 인해 네트워크 관리의 확장성도 개선할 수 있다.

Abstract

SNMP(Simple Network Management Protocol) is most commonly used as a standard protocol for effective network management by supporting the increasing size of the network and the variety of network elements such as router, switch, server and so on. However, SNMP has performance drawbacks of network overhead, processing latency, and the inefficiency in data retrieval. In this paper, we propose two schemes to improve the performance of SNMP; 1) the first scheme to reduce the amount of redundant OID information within a SNMP-GetBulk response message, 2) the second scheme of newly proposed SNMP-GetUpdate message combined with the cache in MNS. Through the analysis with real experiments, we show that our first scheme reduces the network overhead and the second scheme improves the processing latency and the retrieval of SNMP MIB tables. And, therefore the scalability of network management can be improved.

▸ Keyword : SNMP(Simple Network Management Protocol), 네트워크 관리(Network Management), 성능 개선 (Performance Improvement), 네트워크 과부하(Network overhead), 지연(Latency)

• 제1저자 : 나호진 교신저자 : 조경산
• 투고일 : 2010. 01. 19, 심사일 : 2010. 01. 27, 게재확정일 : 2010. 02. 22.
*단국대학교 대학원 박사과정 **단국대학교 컴퓨터학부 교수
※이 연구는 단국대학교 2009학년도 대학연구비의 지원으로 연구되었음.

I. 서 론

다양한 네트워크 장비들이 출연하고 네트워크의 규모가 거대해 짐에 따라 효율적인 네트워크 관리 기법들이 제안되었고, 그 중에서 SNMP가 표준 프로토콜로 가장 널리 사용되고 있다[1][2][3]. SNMP 네트워크 관리 구조는 그림1과 같이 관리될 장비인 NE(Network Element)들을 감시하고 관리하는 NMS(Network Management System), 자원 정보를 제공하기 위한 SA(SNMP Agent), NMS와 SA 사이에 관리 대상 객체를 정의하는 MIB(Management Information Base)로 구성되며, NMS와 NE의 통신을 위한 네트워크 관리 프로토콜(Network Management Protocol)인 SNMP을 이용하여 네트워크를 관리한다[4].

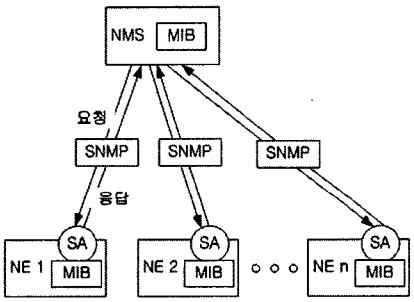

그림 1. SNMP의 네트워크 관리 구조
Fig. 1. Network management architecture of SNMP

SNMP는 NMS가 NE에게 정보를 요청하고 NE가 NMS에게 응답하는 방식으로 동작하여, NMS가 NE로부터 정보를 수집하거나 NE에게 정보를 설정하기 위해 사용되는 프로토콜이다.

하지만, SNMP는 비효율적인 인코딩과 중복되는 정보로 인한 네트워크 과부하, SNMP 메시지 교환에 의해서 발생하는 처리 지연, 그리고 NMS가 얼마나 많은 데이터를 검색해야 할지 모르는 상황에서 최대 메시지 크기로 SA에 요청함에 따라 필요하지 않은 데이터도 응답메시지에 포함되는 테이블 검색의 비효율성과 같은 취약점이 존재한다.

이러한 취약점으로 인해 SNMP에 의한 네트워크 관리의 확장성에 한계를 가지므로, 이를 극복하기 위해 SNMP의 개선이 필요하게 되었고 많은 관련 연구가 제시되었다.

본 연구에서는 기존 연구들을 분석하고 활용하여 SNMP의 성능을 향상 시킬 두 가지 기법을 제안한다. 첫째 제안은 네트워크 과부하 문제를 개선하기 위해 SNMP-GetBulk 응답메시지에서 중복 표현되는 OID 정보를 줄이는 기법이다. 두 번째 제안은 메시지의 처리 지연과 테이블 검색 문제를 개선하기 위해 NMS 캐쉬와 결합된 SNMP-GetUpdate 기법이다.

첫째 제안 기법은 기존 SNMP 방법에 비해 메시지의 크기를 줄여 네트워크 과부하를 감소시키고, 두 번째 기법은 NMS 캐쉬와 효율적인 테이블 검색을 통해 기존 SNMP에 비해 메시지의 크기를 줄이고, 메시지의 전송 횟수도 감소시키는 것으로 분석되었다.

SNMP의 성능 향상을 위한 제안과 검증을 제시하는 본 논문은 다음과 같이 구성된다. 2장에서는 SNMP의 특성과 관련 연구들을 분석하고, 3장에서는 SNMP의 성능 모델과 두 가지 성능 개선 기법을 제안한다. 4장에서는 3장에서 제안한 기법들의 성능 평가를 제시하고, 5장에서는 결론 및 향후 연구 방향에 대해 기술한다.

II. SNMP의 특성 및 관련 연구 분석

본 장에서는 SNMP의 특성을 논의하고, SNMP와 관련된 기존 연구를 분석한다.

1. SNMP의 특성

표준화된 네트워크 관리 프로토콜인 SNMP는 기능과 특성이 개선되면서 첫 버전인 SNMPv1에서 SNMPv2와 SNMPv3로 확장되었다[5]. SNMPv1은 요청과 응답으로 동작하며 NMS가 SA의 객체 값을 검색하는 SNMP-Get과 SNMP-Get Next, NMS가 SA의 객체 값을 설정하는 SNMP-Set 그리고 SA의 이벤트를 NMS에게 알리기 위한 SNMP-Trap을 제공한다. 테이블 형태로 SA에 존재하는 많은 객체의 값을 읽으려면, 많은 SNMP-Get(GetNext)을 반복해야 하는 문제점을 극복하기 위해 SNMPv2에는 SNMP-GetBulk 기능이 추가 되었다. 또한, SNMP의 보안 문제를 개선하기 위해 SNMPv3가 도입되었다.

Header						Data			
Version	Community	PDU Type	Request ID	Error Status	Error Index	OID-Value 1	OID-Value 2	···	OID-Value n

(1) Get-request, Get-next-request, Set-request, Get-response, SNMPv2-trap, Inform-Request PDU

Header						Data			
Version	Community	PDU Type	Request ID	Non-repeaters	Max-repetitions	OID-Value 1	OID-Value 2	···	OID-Value n

(2) GetBulk-Request PDU

그림 2. SNMP의 PDU 형식
Fig. 2. SNMP PDU format

NMS와 SA는 그림2와 같은 형식의 메시지를 UDP로 교환한다. PDU의 Header 영역은 크기가 고정되고, Data 영역은 OID(Object Identifier)와 값(Value)의 쌍으로 구성되며 객체의 수에 따라 크기가 유동적이다. OID는 NE의 특성을 기술한 계층적이고 구조화된 데이터베이스인 MIB에 정의한다. NMS가 MIB에 정의된 OID의 값을 요청하면 NE는 OID와 값의 쌍으로 응답한다.

SNMP-Get, SNMP-GetNext는 NMS의 요청에 대해 SA가 하나의 객체로 응답하는 반면, SNMP-GetBulk는 SA가 다량의 객체를 응답하므로 효율적이고 빠르게 큰 테이블의 값들을 검색할 수 있다. 하지만 SNMP-GetBulk에는 여전히 처리 지연, 네트워크 과부하 그리고 테이블 검색 문제가 존재한다.

2. 관련연구

SNMP는 다음과 같은 성능적 취약점을 가진다. 첫째, SNMP에서 정의된 코딩 기법과 중복 표현되는 정보 때문에 SNMP 메시지의 크기가 증가하여 네트워크 과부하 문제가 발생한다. 특히, 중복 표현되는 OID의 정보에 의한 네트워크 과부하가 대표적이다. 둘째, SNMP 처리를 위해 NMS와 SA 사이의 메시지 교환에 의해 처리 지연 문제가 발생한다. 셋째, 메시지 교환회수를 줄이기 위한 SNMP-GetBulk 메시지는 데이터의 일관성과 데이터 오버슛(data overshoot)으로 인한 테이블 검색 문제가 발생한다[6].

2.1 네트워크 과부하를 개선하기 위한 기존 연구

네트워크 과부하 문제는 비효율적인 인코딩(Encoding), 중복되는 정보 그리고 불필요한 객체의 검색 문제에 의해 발생한다.

SNMP에서 사용하는 인코딩 기법은 BER이며, BER은 네트워크 과부하 측면에서 비효율적이다[7]. 이런 문제를 개선하기 위해, PER(Packed Encoding Rules), LER(Lightweight Encoding Rules), DER(Distinguished Encoding Rules), 그리고 CER(Canonical Encoding Rules) 기법이 제안되었다[8]. 그러나 PER은 BER에 비해 인코딩 시간이 증가한다는 단점이 있고, LER은 인코딩 길이가 증가한다는 단점이 있고, DER은 일정한 형태(Definite form)에서만 사용할 수 있다는 단점이 있다. 위의 4가지 대체 기법들은 구현을 위해 기존 구조에 많은 수정을 필요로 하므로 새로운 인코딩 방법으로 정착되지 못하고 있다.

MIB을 구성하는 객체의 식별자인 OID는 계층적 정보로 표현되어, 중복 표현이 많이 발생하므로, 이 문제점을 개선하기 위한 여러 기법들이 연구되었다. 이전 OID와 중복되지 않는 부분만 OID로 표시하는 ODC(OID Delta Compression)의 기법이 제안되었지만 실제 압축률은 높지 않다[9]. OID 사이의 일 대 다 매핑을 통하여, 열에 대한 이름을 한번만 표현하고 객체 값을 연속적으로 표시하는 OID Suppression 기법이 제안되었지만 표현이 모호해지는 단점이 있다[10]. 기업에 부여되는 Enterprise OID 정보를 줄여서 정보의 양을 줄이는 Enterprise OID 최적화기법이 제안되었지만, EMS(Enterprise Management System)에서만 사용 가능하고 NMS에서는 사용할 수 없다는 단점이 있다[11].

2.2 처리 지연 문제를 개선하기 위한 기존 연구

SNMP에서 발생하는 처리 지연 문제는 네트워크를 통해 전송되는 메시지 교환에 의해 발생한다.

여러 쓰레드를 사용하여 동시에 MIB 테이블의 서로 다른 객체를 검색하여 지연 문제를 개선한 Pipeline 탐색이 제안되었지만, NMS와 SA 사이의 통신 복잡성이 증가하고 전송되는 요청과 응답메시지 수는 줄어들지 않는다는 단점이 있다[6]. 또한, SNMP-GetNext와 SNMP-GetBulk는 이전 응답에 의존해서 요청을 하기 때문에, 여러 쓰레드를 동시에 적용할 수 없다.

NMS와 SA 사이에 메시지 전송 횟수를 줄이기 위하여 NMS에 캐쉬를 사용하는 기법이 제안되었지만, SNMP-Get과 SNMP-GetNext 메시지 교환할 때만 캐쉬에 접근이 가능하다는 단점이 있다[12].

2.3 테이블 검색 문제를 개선하기 위한 기존 연구

SNMP-GetBulk에서 발생 가능한 테이블 검색 문제는 데이터의 일관성과 데이터 오버슛에 의해 발생한다. 큰 테이블의 데이터를 두 메시지로 나눠서 전송하는데, 두 번째 전송 전에 테이블의 값이 변경된다면 두 메시지 사이에 일관성이 유지되지 못하는 문제가 발생한다. 또한, NMS는 SNMP-Get Bulk를 통해 얼마나 많은 데이터를 검색해야 할지 모르기 때문에, 객체의 최대 반환 개수를 지정하고 요청한다. 이 때문에 NMS는 필요하지 않는 데이터를 전송 받는 데이터 오버슛 문제가 발생한다. 이와 같은 문제점을 개선하기 위한 다음과 같은 기법들이 연구되었다.

SNMP-Get, SNMP-GetNext 그리고 SNMP-GetBulk는 행을 기준으로 객체를 검색하는데, 이를 보완하기 위해 열을 기준으로 객체를 검색하는 GetRow, GetNextRow, GetBulkRow 기법이 제안되었다[13]. NMS가 필요한 데이터의 끝을 표시하여 불필요한 객체가 검색되지 않도록 하는 GetBulkBumper 기법, MIB에 정의된 하나의 테이블 만을 검색하는 GetSubtree 기법, 역 방향으로 OID를 증가 하면서 객체를 찾는 GetPrev 기법, 관리자가 이전에 폴링한 객체 중 변경된 객체 정보만 탐색하여 전송하는 GetModify 기법이 제안되었다[14][15][16]. 그

리고 조건에 만족하는 객체만을 검색하기 위하여 필터 기법을 이용한 GetObject, GetRows 그리고 getRangeTable 기법이 제안 되었다[17][18]. 테이블 검색 문제를 개선하기 위한 이들 기법은 특정한 환경에서만 성능 향상을 보이며, 상황에 따라 성능이 저하될 수 있는 단점이 있다.

III. 성능 모델과 성능 개선 기법

1. SNMP 성능 모델

SNMP에서는 그림3과 같이 NMS에서 요청메시지를 생성하여 SA에게 전송한다. SA에서는 요청메시지를 분석하고 응답메시지를 생성 한 후, NMS에게 전송한다.

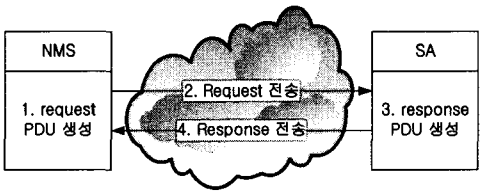

그림 3. SNMP 처리 과정
Fig. 3. Sequence of SNMP procedures

SNMP 요청에서 응답까지의 SNMP 처리 시간(TProcess)은 1) NMS에서 SNMP 요청메시지 생성 시간(TReqGen)과 2) 네트워크를 통한 SNMP 요청메시지 전송 시간(TReqTrans), 3) SA에서 SNMP 응답메시지 생성 시간(TRespGen) 그리고 4) 네트워크를 통한 SNMP 응답메시지 전송 시간 (TRespTrans)으로 구성되며, 식(1)과 같이 표현할 수 있다.

$$T_{Process} = T_{ReqGen} + T_{ReqTrans} + T_{RespGen} + T_{RespTrans} \quad \cdots (1)$$

TRespGen은 전송되는 메시지를 해석하고 파싱하는 메시지 디코딩시간(TMsgDecoding), OID 등록 처리기를 확인하여 해당 모듈을 호출하는 디스페치시간(TDispatch), 요청한 객체의 값을 생성하는 SNMP 값 생성시간(TValueGen), 그리고 SNMP 응답메시지를 생성하는 인코딩시간(TMsgEncoding)의 합으로 표현할 수 있다. 요청된 객체가 하나 일 때, SNMP 응답 생성 시간은 식(2)와 같이 표현 할 수 있다.

$$T_{RespGen} = T_{MsgDecoding} + T_{Dispatch} + T_{ValueGen} + T_{MsgEncoding} \quad \cdots (2)$$

식(2)에 $T_{MsgDecoding}$, $T_{Dispatch}$, 그리고 $T_{MsgEncoding}$의 시간은 응답 객체가 증가해도 한번만 처리하지만 $T_{ValueGen}$은 응답 객체 수만큼 객체의 값을 생성해야 하므로 객체 수에 비례해서 시간이 소요되므로 식(2)는 다음과 같이 다시 표현할 수 있다.

$$T_{RespGen} = T_{MsgDecoding} + T_{Dispatch} + \sum_{i=1}^{n} T_{ValueGen} + T_{MsgEncoding} \quad (3)$$
$$(n : SNMP\text{응답 객체의 수})$$

NMS에서 필요한 모든 정보를 검색하는 총 처리 시간(T_{Total})은 식(1)과 같이 단일 메시지 교환을 하는 경우도 있지만, 많은 경우에는 여러 개의 요청과 응답을 통하여 다중 메시지 교환을 하며 이 경우는 식(4)로 표현된다.

$$T_{Total} = \sum_{i=1}^{k} T_{Process} \ (k: \text{메시지 교환(요청/응답)쌍의 수}) \quad \cdots (4)$$

식(4)에서 메시지 교환을 최소화 하면 처리 시간이 감소하여 SNMP 성능을 향상 시킬 수 있다.

SNMP 처리 시간에 영향을 미치는 주요 요소로는 식(3)의 응답 객체의 수에 따른 메시지의 크기와 식(4)의 메시지 교환 쌍의 수이다.

2. 제안기법

2.1 중복되는 OID 정보를 줄이는 제안

SNMP-GetBulk 응답메시지의 데이터 영역은 OID와 값의 쌍이 연속적으로 표현되는데, OID에는 중복되는 정보가 많이 포함되는 문제가 있다. 본 제안에서는 함께 전송되는 여러 OID 내의 중복되는 정보를 하나로 줄여 응답메시지에 더 많은 OID와 값의 쌍을 전송하여 SNMP의 성능을 개선하고자 한다.

비트	8	7	6	5	4	3	2	1
의미	class		p/c	Tag number				

a) Identifier octets

비트	8	7	6	5	4	3	2	1
의미	form	Contents octets length(Max=128)						

b) Length octets

비트	8	7	6	5	4	3	2	1
의미	L/S	subidentifier						

C) Contents octets

그림 4. SNMP 메시지의 인코딩
Fig. 4. Encoding of SNMP message

그림2에서 보인 SNMP PDU내에 전송되는 OID와 값의 쌍

은 그림4와 같이 Identifier octets, Length octets 그리고 Contests octets로 구성된다.

기존 Length octets의 구조에서는 Contents octets length가 7비트로 subidentifier를 128개까지 표현할 수 있다. 본 제안에서는 그림5와 같이 Length octets의 구조에서 7번째 비트를 중복되는 OID 정보를 표현하기 위한 OID Flag 비트로 사용하고, Contents octets length를 6비트(최대 64개 까지의 subidentifier를 표현)로 수정한다.

비트	8	7	6	5	4	3	2	1
의미	form	OID Flag	Contents octets length(Max=128)					

그림 5. 수정된 "Length octets" 필드
Fig. 5. Modified "Length octets" field

SNMP 응답 메시지에서 테이블의 첫 번째 객체에 대해서는 7번째 비트(OID Flag)를 1로 설정하고 테이블을 의미하는 OID, 리프 노드(leaf node)를 의미하는 OID 그리고 인스턴스를 의미하는 OID를 포함한 완전한 OID 정보가 Contents octets의 구조에 표현된다. 그리고 테이블의 두 번째 객체부터 테이블의 마지막 객체까지는 그림5의 7번째 비트를 0으로 설정하고, 동일 테이블에서는 테이블 OID가 중복 정보이므로 이를 제외하고 리프 노드 OID와 인스턴스 OID만을 표현한다.

따라서, 제안 기법과 같이 중복되는 테이블 OID 정보를 생략하면, SNMP-GetBulk의 응답메시지에 더 많은 객체의 값을 포함시킬 수 있어서 네트워크 과부하를 줄일 수 있다.

2.2 NMS 캐쉬와 SNMP-GetUpdate를 결합한 제안

NMS가 필요한 최신 정보를 가지고 있다면, SA와 SNMP 메시지 교환이 필요 없게 된다. 이러한 목적으로 NMS에 캐쉬 기법을 적용할 수 있으며, 또한 캐쉬에 저장되지 않는 객체만을 SA에 요청하는 기법을 제안한다.

SA에 저장된 객체 정보는 네트워크 구성 요소와 같은 쉽게 변하지 않는 정적 정보, 네트워크 상태와 이벤트에 의해서 변하는 동적 정보, 그리고 지속적으로 값이 변하는 통계 정보로 분류 할 수 있다[4]. 본 연구에서는 [4]에서 제시된 정보의 분류를 [15]에서 연구된 NMS 캐쉬에 적용한다. 즉, 본 연구에서는 정적 정보와 동적 정보만을 NMS 캐쉬에 저장하도록 제안하여 캐쉬의 효율성을 높이고 NMS와 SA사이의 과도한 메시지 교환을 줄이고자 한다. 이를 위해서는 다음과 같은 세 가지 기능을 추가하도록 제안한다.

첫 번째, MIB에 객체가 정적, 동적 또는 통계 정보 인지를 표시하기 위해 MIB의 OBJECT-TYPE MACRO에 Static, Dynamic 그리고 Statistic의 값을 갖는 "InfoType"을 그림6과 같이 제안한다.

```
OBJECT-TYPE MACRO ::=
 BEGIN
  TYPE NOTATION ::=
   InfoType

 ====== 중간 생략 ======

   InfoType ::=
          "Static"
          | "Dynamic"
          | "Statistic"
 END
```

그림 6. OBJECT-TYPE MACRO
Fig. 6. OBJECT-TYPE MACRO

두 번째, SA에서 MIB을 구현할 때 정적 또는 동적 객체의 정보가 변경되면 NMS에게 변경 내용을 전송하여 NMS에서 SA의 최신 상태 정보를 유지하도록 한다.

세 번째, NMS에서 SA의 최신 정보를 저장하고 관리하기 위해 표1과 같은 테이블을 NMS에서 관리한다.

표 1. NMS에서 관리하는 필드
Table 1. Fields managed in NMS

필드	기능
IMAC	SA 식별
BTime	SA의 시스템이 초기화된 시간
OID	SA의 객체를 식별하기 위한 식별자
Value	객체의 값
LUTime	마지막으로 업데이트한 시간
IType	Static, Dynamic, Statistic 정보 구분

또한, 본 연구에서는 SNMP-GetBulk를 개선하여 NMS 캐쉬에 저장되지 않는 통계 정보의 요청만을 SA에서 검색하여 NMS로 전송하는 SNMP-GetUpdate 메시지 기법을 제안한다. 이때, [14]에서 제안된 객체의 시작 OID와 마지막 OID를 SNMP-GetUpdate 요청에 포함하도록 설정한다.

즉, NMS 캐쉬와 SNMP-GetUpdate를 결합한 두 번째 제안에서는 SA에 저장된 객체 정보 중에서 정적 및 동적 정보는 NMS 캐쉬를 통해 관리하여 처리 지연을 줄이고, 통계 정보는 SNMP-GetUpdate 요청을 통해 테이블 검색 문제점을 개선하도록 한다.

IV. 검증 및 평가

SNMP 성능 평가와 3장에서 제안한 기법의 성능 검증을 위한 시스템 환경은 NMS와 SA가 동일하며, 하드웨어는 IBM System x3650 M2를 사용하였고, 운영체제는 CentOS 5.4 x86_64를, 소프트웨어는 Net-SNMP5.4.2.1을, MIB RFC1213-MIB과 IF-MIB을 사용하였다.

1. SNMP의 성능 평가

본 절에서는 SNMP 처리시간을 3.1절의 성능 모델을 기반으로 분석한다.

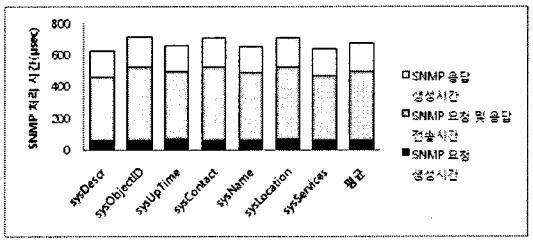

그림 7. SNMP-Get 처리 시간
Fig. 7. Processing time of SNMP-Get

그림7은 SNMP에서 참조가 가장 많은 RFC1213-MIB의 "system" 객체들을 SNMP-Get를 통해 얻는 처리 시간을 식(1)의 요소들로 구성하여 분석한 그래프이다. 요청 및 응답 전송 시간이 가장 큰 비중을 차지하므로, 네트워크를 통한 전송 회수가 SNMP 통신에 큰 영향을 준다.

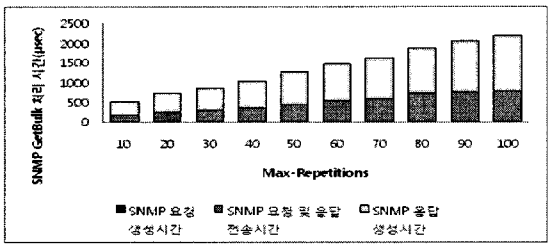

그림 8. SNMP-GetBulk 처리 시간
Fig. 8. Processing time of SNMP-GetBulk

그림8은 Max-repetitions을 10에서 100까지, 10만큼씩 증가시키며, SNMP-GetBulk의 처리 시간을 분석한 그래프이다. Max-Repetitions가 증가하면 SNMP 요청 및 응답 전송시간과 SNMP 응답 생성시간이 함께 증가하므로 불필요한 객체는 검색하지 않는 SNMP-GetBulk의 효율적인 검색이 필요하다.

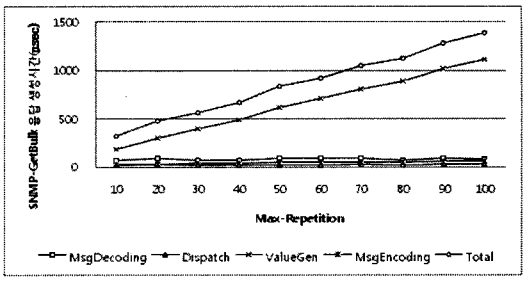

그림 9. SNMP-GetBulk 응답 생성 시간
Fig. 9. Response generation time of SNMP-GetBulk

그림9는 그림8에서 SNMP 응답 생성 시간인 TRespGen의 수행 시간을 식(2)의 시간 성분으로 표현했다. SNMP 값 생성 시간은 Max-Repetitions의 증가에 비례하여 증가된다. 따라서 불필요한 객체를 검색하지 않고 식(3)의 응답 객체 수 n을 최소화 하면 응답 생성 시간이 감소하여 SNMP 통신의 성능을 향상 시킬 수 있다.

2. 제안 기법의 성능 평가

본 절에서는 본 연구에서 제안한 두 기법에 의한 SNMP 성능 개선도를 분석한다.

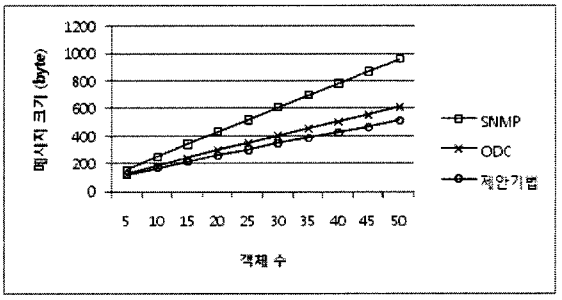

그림 10. 첫 번째 제안에 대한 메시지 크기

Fig. 10. Message size for the first proposal

첫 번째 제안 기법의 평가를 위해 SNMP-GetBulk 요청에 대한 응답 메시지 크기를 분석하였다. 그림10은 SNMP-GetBulk 요청의 Max-repetitions을 5부터 50까지 5씩 증가 하면서 실제 전송되는 SNMP-GetBulk 응답메시지의 크기를 비교한 그래프이다. 그림에서 SNMP는 기존 SNMP의 경우이고, ODC는 OID Delta Compression 기법을 사용한 경우이다. Max-repetitions

가 50일 때 SNMP는 958byte이고, ODC는 615byte이며, 제안 기법은 517byte로 메시지 크기가 작아진다.

따라서, 앞에서 분석된 바와 같이 첫 번째 제안 기법에 의해 전송되는 OID의 정보 크기가 감소하므로 응답메시지 크기가 작아져서 메시지의 세그먼트를 최소화 하고 네트워크 과부하를 줄여 SNMP 통신의 성능을 향상 시킬 수 있다.

두 번째 제안을 평가하기 위해 IF-MIB의 ifTable 정보의 요청에 대한 처리 성능을 분석하였다. IF-MIB의 ifTable 테이블은 총 22개의 객체로 구성되는데, 그 중에서 ifIndex, ifDescr, ifType, ifMtu의 4개 객체는 정적 정보로 분류되고 ifSpeed, ifPhysAddress, ifAdminStatus, ifOperStatus, ifLastChange의 5개 객체는 동적 정보로 분류되어 이들은 MNS 캐쉬에 저장된다. 나머지 13개의 객체는 통계 정보로 SNMP-GetUpdate 메시지에 의해 요청된다. 모든 정보 요청이 동일한 횟수로 요청된다면, 9개의 동적 정보 객체가 모두 캐쉬에 저장될 때 전체 요청에 대한 NMS 캐쉬의 히트율은 41%가된다.

두 번째 제안 기법을 적용할 때의 전송된 메시지의 크기, 메시지 전송 회수, 그리고 SNMP 총 처리 시간은 기존 기법에 비해 그림11-그림13과 같이 개선된다. 그림에서 GetBulk는 Max-repetitions을 100(Net-SNMP Agent는 최대 100까지만 허용)으로 설정하고 ifTable의 첫 번째 객체부터 100개의 객체를 SNMP-GetBulk을 통해 요청한 경우이다. GetSubtree는 [14]에서 제안된 GetSubtree 기법으로 ifTable의 마지막 객체까지만 요청하는 경우이다. 제안 기법을 나타내는 GetUpdate는 ifTable의 갱신된 통계 정보를 SNMP-GetUpdate를 통해 요청하는 경우이다. 이때, MNS 캐쉬의 초기화 과정은 포함하지 않았고 NMS 캐쉬의 히트율은 41%(정적/동적 정보가 캐쉬에 저장되었다고 가정) 인 경우에 대해 분석하였다.

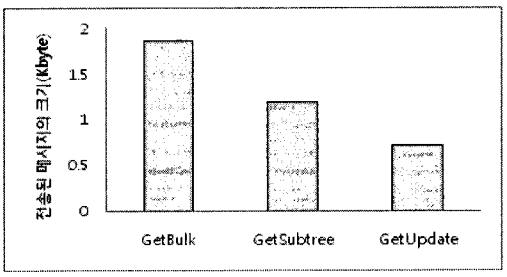

그림 11. 두 번째 제안에 대한 메시지의 크기
Fig. 11. Message size for the second proposal

그림11은 식(3)에서 제시된 응답 객체 메시지들의 전체 크기를 비교한 그래프이다. GetBulk는 1.86Kbyte이고, GetSubtree는 1.18Kbyte이며, GetUpdate는 0.71Kbyte로 메시지 크기가 감소된다.

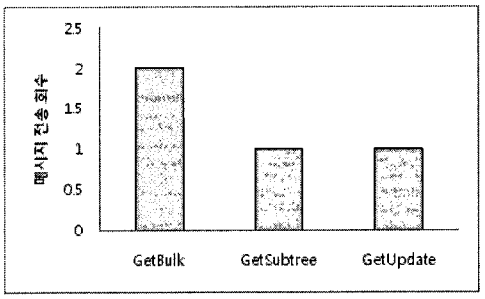

그림 12. 메시지 전송 회수
Fig. 12. Number of message exchanged

그림12는 식(4)의 메시지 교환 회수를 비교한 그래프이다. GetBulk는 하나의 메시지가 MTU를 초과하기 때문에 메시지 전송 회수가 2회이고, GetUpdate는 GetSubtree와 동일하게 1회로 메시지 전송회수가 감소함을 보인다.

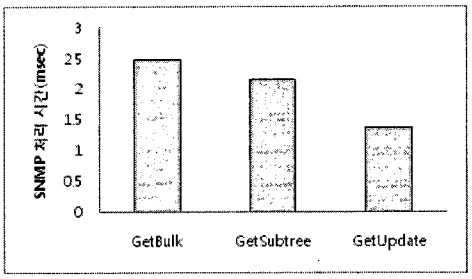

그림 13. SNMP 총 처리 시간 비교
Fig. 13. SNMP total processing time

그림13은 식(4)의 SNMP 총 처리 시간을 비교한 그래프이다. GetBulk는 2.48msec이고, GetSubtree는 2.15msec이며, GetUpdate는 1.37msec로 SNMP 처리 시간이 감소한다.

두 번째 제안 기법의 또 다른 검증을 위해 더 많은 정보를 전송하는 ifXTable에 대한 요청 처리를 분석하였다. 통계 객체 12개와 정적/동적 객체 7개를 포함하고 각 객체에 인스턴스가 24개가 존재하는 ifXTable에 기존의 SNMP-GetBulk를 적용한 결과로 메시지 크기는 5.29Kbyte, 전송 회수는 6회, 총 처리 시간은 4.653msec이다. 반면에 제안한 SNMP-GetUpdate를 적용한 결과는 메시지 크기는 3.17Kbyte, 전송 회수는 3회, 총 처리 시간은 2.967msec로 제안 기법이 그림11-그림13에서 보인 결과와 유사한 비율로 성능의 개선을 보였다.

따라서, 두 번째 제안 기법은 정적/동적 정보는 NMS 캐쉬를 통해 관리하고 통계 정보는 SNMP-GetUpdate 요청을 통

해 객체를 검색하여 테이블 검색 문제가 해결되고 총 처리 시간이 감소하는 SNMP의 성능 향상을 보인다.

V. 결 론

SNMP는 네트워크 관리를 위해 가장 많이 사용되는 프로토콜이지만 네트워크 과부하, 처리지연, 그리고 테이블 검색의 비효율성과 같은 문제점을 가지고 있다.

본 연구에서는 SNMP의 문제점을 개선하기 위해 다음과 같은 두 가지 기법을 제안하였다. 첫째, 네트워크 과부하 문제를 개선하기 위해 SNMP-GetBulk의 응답메시지 내에 중복해서 나타나는 OID정보를 한번만 표현하는 기법을 제안하였다. 둘째, 처리지연과 테이블 검색 문제를 개선하기 위해 NMS 캐쉬와 결합된 SNMP-GetUpdate 메시지 기법을 제안하였다.

첫 번째 제안 기법은 동일 테이블 상의 다수의 객체를 요청할 경우에 효율적인데, 제시된 분석에서는 기존 SNMP에 비해 53%의 크기로 메시지가 감소함을 보였고 따라서 네트워크 과부하 문제의 개선안이 될 수 있다. 두 번째 제안 기법은 NMS가 관리할 정보가 정적, 동적, 통계 정보들로 혼합되어 있으면서 정적 및 동적 정보의 비율이 높고 변경이 적으며 또한 주기적으로 통계정보를 수집하는 환경에 적합하다. 정적 및 동적 객체가 NMS 캐쉬에 저장된 경우의 분석에서 기존 SNMP의 GetBulk에 비해 제안된 GetUpdate가 메시지 크기, 메시지 전송 회수 및 처리 시간이 현저히 개선되어 처리 지연과 테이블 검색의 문제를 개선할 수 있음을 보였다.

본 연구의 제안에 의해 SNMP의 문제점을 개선하면, NMS는 더 많은 NE를 관리할 수 있고, 관리할 정보의 양을 증가시킬 수 있으므로 네트워크 관리의 확장성이 증대된다. 또한 확장된 네트워크에서도 더 적은 수의 NMS로 전체 네트워크를 관리할 수 있으므로 네트워크 관리 비용을 줄일 수 있다.

실제 본 연구의 분석에서는 캐쉬의 초기화 및 관리 비용을 고려하지 않았고 또한 저장 객체의 정적, 동적 및 통계 정보의 비율에 따라 성능 개선 정도가 달라진다. 따라서, 적용된 환경에 따른 분석이 향후 연구로 요구된다.

SNMP의 효율성을 더욱 개선하기 위해서는 SNMP 기반의 중앙 집중적인 네트워크 관리 방법의 단점을 개선한 이동 에이전트 기반의 분산 네트워크 관리 방법에 대한 연구가 필요하여 현재 진행 중이다. 아울러 SNMP 기반 네트워크 관리와 이동 에이전트 기반 네트워크를 융합한 네트워크 관리도 향후 연구 과제이다.

참고문헌

[1] 정경권, 박현식, 최우승, "RFID를 이용한 일상생활 모니터링 시스템 개발," 한국컴퓨터정보학회논문지, 제 14권, 제 7호, 49-56쪽, 2009년 7월.

[2] 박문화, "스마트 라우터를 사용한 효율적인 네트워크 관리 시스템," 한국컴퓨터정보학회논문지, 제 10권, 제 6호, 253-260쪽, 2005년 12월.

[3] 강경인, 박경배, "이동 애드 혹 네트워크에서의 트래픽 관리," 한국컴퓨터정보학회논문지, 제 14권, 제 9호, 29-35쪽, 2009년 9월.

[4] William Stalling, "*SNMP, SNMPu2, SNMPu3, RMON 1 and 2*," Addison Wesley, 1996.

[5] R. Presuhn, "Version 2 of the Protocol Operations for the Simple Networks Management Protocol(SNMP)," Internet Engineering Task Force(IETF), RFC 3416, 2002.

[6] R. Sprenkels, J.P. Martin-Flatin, "Bulk transfers of MIB data," The Simple Times, Vol. 7, No. 1, pp. 1-7, 1999.

[7] International Telecommunication Union, "Information technology - ASN.1 encoding rules : Specification of Basic Encoding Rules(BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules(DER)," ITU-T Recommendation X.690, 2002.

[8] International Telecommunication Union, "Information technology - ASN.1 encoding rules : Specification of Packed Encoding Rules(PER)," ITU-T Recommendation X.691, 2002.

[9] J. Schoenwaelder, "SNMP payload compression," draft-ietf-nmrg-snmpcompression-01.txt, 2001.

[10] S. McLeod, D. Partain, and M. White, "SNMP object identifier compression," draft-ietf-eosoidcompression-00.txt, 2001.

[11] C. Jagadish, S. T. Prakash and T. Gonsalves, "Network Management Traffic Optimization," Procs. of 14th National Conference on communications, 2008.

[12] O. Said, "A Novel Technique for SNMP Bandwidth Reduction: Simulation and Evaluation," International Journal of Computer Science and Network Security, Vol. 8, No. 2, pp. 208-214, 2008.

[13] L. Heintz "*SNMP row operations extensions,*" draft-ietf-eos-snmp-rowops-01.txt, 2001.

[14] M. Malowidzki, "GetBulk worth fixing," Simple Times, Vol. 10, No. 1, pp. 3-6, 2002.

[15] D. Breitgand, D. Raz, and Y. Shavitt, "SNMP GetPrev: an efficient way to browse large MIB tables," Procs. of Integrated Network Management 2001, pp. 437-452, 2001.

[16] Park SH, Park MS. "An efficient transmission for large MIB tables in polling-based SNMP," Procs. of the 10th International Conference on Telecommunications, Vol. 1, pp. 246-252, 2003.

[17] W. Hardaker, *"Object Oriented Protocol operations for the SNMP Protocol,"* draft-ietf-eos-oops.00.txt, 2003.

[18] C. Yen-Cheng, C. Io- Kuan, "SNMP GetRows - An Effective Scheme for Retrieving Management Information from MIB Tables," International Journal of Network Management, Vol. 17, pp. 51-67, 2007.

저 자 소 개

나 호 진

1998 : 단국대학교 응용물리학과(학사)
2001 : 단국대학교 전산통계학과 (이학석사)
2006년 3월~현재 : 단국대학교 컴퓨터과학(박사과정)
관심분야 : 네트워크 관리, 지능형 에이젠트, 네트워크 통신, 임베디드 컴퓨팅

조 경 산

1979 : 서울대학교 전자공학과(학사)
1981 : 한국과학원 전기전자공학과 (공학석사)
1988 : 텍사스 대학원(오스틴) 전기전산공학과(Ph.D.)
1988~1990 : 삼성전자 컴퓨터부문 책임연구원, 실장
1990~현재 : 단국대학교 컴퓨터학부 교수
관심분야 : 네트워크시스템 및 이동통신보안, 컴퓨터시스템