논문 01-26-2A-4 한국통신학회논문지 '01-2 Vol.26 No.2A

세그먼트 클러스터링 방식을 이용한 이동 에이전트형 망 관리 시스템의 구현 및 성능 분석

정회원 노 정 민*, 윤 종 호*

Implementation and Performance Analysis of Mobile Agents with Segment Clustering Scheme for Network Management

Jeong Min Noh*, Chong Ho Yoon* *Regular Members*

요 약

현재 SNMP를 이용한 망 관리 시스템은 각 에이전트들에 대한 폴링 방식으로 운영되기 때문에, 대역폭 소비가 많은 단점이 있다. 최근에는 이를 극복하기 위해 이동 에이전트 방식의 망 관리 시스템이 연구되고 있지만 이에 대한 수식적 분석과, 실제 성능 검증은 미비한 상태다.

본 논문에서는, 이동 에이전트형 망 관리 시스템을 Java로 구현하고, 이것의 성능을 소모 대역폭 관점에서 SNMP의 성능과, 수식 및 실험적으로 비교 분석하였다. 그 결과, 단일 세그먼트 내에서는 로밍 방식을 사용하는 이동 에이전트의 성능이 데이터 누적으로 인해서 기존 방식에 비해 떨어지지만, 다중 세그먼트 환경에서는 세그먼트간 이동이 적은 이동 에이전트 방식의 성능이 기존 방식에 비해 우수함을 알 수 있었다.

그리고, 다중 세그먼트 환경에서 기존의 로밍 방식에 비해 대역폭 소비가 적은 세그먼트 클러스터링 방식을 제안했다. 이것은 세그먼트 단위로는 폴링 방식을 사용하고, 한 세그먼트 안에서는 로밍 방식을 사용한다. 성능 분석 결과 다중 세그먼트 환경에서는 세그먼트 클러스터링 방식의 성능이 가장 우수함을 알 수 있었다.

ABSTRACT

It has been noted that the polling scheme used for the conventional network management system(NMS) consumes excessive bandwidth. To overcome this problem, recent activities have been focused on the development of mobile agents. However, there are few numerical analysis and performance verification results.

In this paper, we compare the performance of the NMS based on mobile agent with the conventional one based on SNMP in terms of required bandwidth, and verify by its implementation with Java. From the numerical and experimental results, the performance of mobile agents is poorer than the conventional one in a single segment network due to the accumulation of collected data; while it performs better than the conventional one in a multi-segment network, because it can save a more bandwidth for migrating from one segment to another one.

To save further bandwidth in a multi-segment network, we also propose a new segment clustering scheme for mobile agents. Under this scheme, each host in a segment, namely a cluster, is managed by a mobile agent. The mobile agent is dispatched from the manager to a segment. After gathering objects in the segment, the mobile agent should be returned to the manager. Then, the mobile agent will be dispatched to another segment. From the numerical analysis and experiments, we find that the performance of the proposed segment clustering scheme shows better efficiencies than others.

* 한국항공대학교 항공통신정보공학과
논문번호 : 00461-1206, 접수일자 : 2000년 12월 6일

Ⅰ. 서 론

망 관리 시스템은 네트워크에 관련된 정보를 수집, 저장, 관리하며 적절한 처리를 하여 원하는 정보를 보여주는 시스템이다[1]. 최근 들어 인터넷의 사용이 급증하면서 망 관리 시스템이 관리하는 네트워크 개체도 크게 증가하고 있다. 따라서 이런 다양하고 복잡한 네트워크 개체를 효율적으로 관리하기 위한 연구가 꾸준히 진행중이다.

현재 대부분의 망 관리 시스템은 1990년 표준으로 제정된 SNMP 방식을 사용한다. SNMP를 이용한 망 관리 시스템은 관리 정보를 폴링으로 주고받는 중앙 집중형 방식이다. 따라서 구현은 쉽지만, 관리 개체가 늘어날수록 대역폭 소비가 많고 관리 프로세스의 병목 현상이 있는 단점이 있다.

최근에는 이러한 문제점을 극복하기 위해 이동 에이전트 방식의 망 관리 시스템이 연구되고 있다. 이동 에이전트는 폴링 방식과 같이 네트워크를 통해서 프로그램의 데이터를 전달하는 것이 아니라, 프로그램 코드 자체를 전달하는 방법이다. 그리고 이동 에이전트는 실행된 시스템에 종속되는 것이 아니라, 자기 자신을 네트워크 상의 다른 시스템으로 이동하며 필요한 일을 수행한다. 따라서, 기존의 중앙 집중형 망 관리 방식에 비해 네트워크 사용 대역폭을 크게 줄일 수 있으며, 관리 개체를 이동하면서 필요한 관리 동작을 수행하기 때문에, 중앙의 관리자가 해야하는 일을 줄여 주는 장점이 있다[2][3].

하지만, 아직까지 이동 에이전트형 망 관리 시스템에 관한 연구는 개념 정립과[4], 구현방안, 그리고 지역 네트워크에서 기존의 SNMP 폴링 방식과의 성능을 비교한 정도이며, 시스템 성능에 대한 수식적인 분석과, 실제로 이를 구현하여 검증한 결과는 찾아보기 어렵다.

본 논문에서는, 이동 에이전트를 이용한 망 관리 시스템의 성능을 수식적으로 분석하고 실제로 구현하였으며, 단일/다중 세그먼트 환경에서 기존의 SNMP 기반 망 관리 시스템과 비교하여 그 효율성을 검증하였다. 이를 위해서, 먼저 이동 에이전트형 망 관리 시스템의 대역폭 사용량을 수식적으로 분석하였다. 그리고 Java를 사용한 시스템을 구현하여, 실제 망 상황에서의 성능과 수치적 분석결과를 검증하였다.

본 서론에 이어 제 2장에서는 SNMP 망 관리 시스템과 이동 에이전트 망 관리 시스템에 대해 다루고, 제 3장에서는 이동 에이전트 망 관리 시스템 구현 방법에 대하여 다루며, 4장에서는 이에 대한 성능을 분석한다. 그리고 마지막으로 제 5장에서 결론을 맺는다.

Ⅱ. SNMP 및 이동 에이전트형 망 관리 시스템의 비교

2.1 SNMP 기반의 망 관리 시스템

중앙 집중형 망 관리 시스템은 전체 네트워크에서 관리하는 요소의 각 지점과 특정한 속성에 주소와 이름을 지정하고, 네트워크 요소들은 주기적으로 자신이 가진 정보를 중앙 제어 센터에 제공한다. 여기서 에이전트는 각각의 네트워크 자원들에 붙어서 각각의 자원들에 대한 정보를 수집하며, 매니저는 각각의 에이전트가 관리하는 정보를 수집해 전체 네트워크를 관리한다.

인터넷 관리체제에 대한 표준화는 IETF에 의해 이루어지고 있으며, 인터넷 관리체제의 표준 프로토콜은 Simple Network Management Protocol(SNMP)이다[4]. SNMP는 매니저와 에이전트로 구성되는 중앙 집중형 관리 구조이다. SNMP에 의해 관리되는 정보는 Management Information Base(MIB)에 저장되며, 매니저와 에이전트는 request와 response 메시지를 통해 정보를 주고받는다[5].

SNMP에서 에이전트의 내용을 매니저에게 전달할 때는 그림 1과 같이 폴링 방식을 사용한다. 폴링은 관리자의 요청에 의한 응답 방식으로서, 매니저가 에이전트에게 원하는 정보를 요구(request)하면 에이전트는 매니저가 원하는 정보를 찾아서 응답(response)해 준다. 이것은 SNMP에서 정보를 교환하는 주된 방법으로서 구현이 쉬운 장점이 있지만, 네트워크에 부하를 많이 주는 단점이 있다.

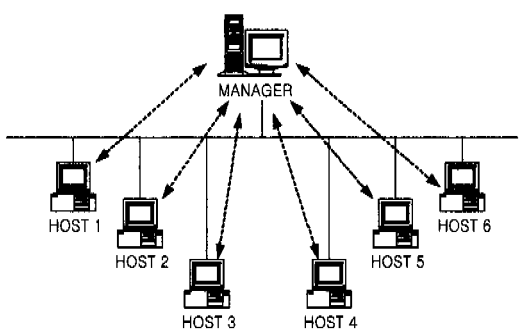


그림 1. SNMP 폴링 방식.

2.2 이동 에이전트를 이용한 망 관리 시스템

에이전트는 사용자나 컴퓨터를 대신해서 독자적으로 동기, 비동기적으로 프로세스를 수행하는 기능을 가진 시스템이다. Yemini와 Goldszmidt는 네트워크 관리를 중앙의 매니저가 전부 처리하는 것이 아니라 각각의 에이전트에 위임(Delegation)하여 중앙집중형 관리 방식의 단점을 줄이는 방법을 제안했다[6]. 이 개념에 이동 코드(Mobile Code) 개념을 추가하여 코드 이동, 프로세스 수행, 비동기적인 통신수행, 데이터 운반 등의 기능을 독자적인 권한을 가지고 수행하도록 한 것이 이동 에이전트이다. 이것은 학습, 추론, 계획의 기능을 가지고 주위의 실행환경에 적응하며, 어떤 방식으로 주어진 업무를 수행할 것인가에 대해 미리 계획하고, 이 계획에 의해 통신과 필요한 작업을 수행한다.

이동 에이전트에서 이동되는 코드에는 실행 가능한 코드와 메모리 상태, 실행에서 얻은 데이터 등을 포함하며, 이것은 이동된 시스템에서 다시 활성화되는 성질이 있다. 또한 이동 에이전트는 실행된 시스템에 종속되는 것이 아니라, 자기 자신을 네트워크 상의 다른 시스템으로 이동하며 필요한 일을 수행한다. 그리고 에이전트의 기능이 한번 수행으로 끝나는 것이 아니라 지속적으로 상태 및 절차를 변화시켜가며 업무를 수행하는 특성을 지닌다. 최근에는 객체 직렬화(Object Serializing) 기술을 이용하여 Mobile Agent(MA)의 상태, 수행한 데이터, 통신 정보등을 저장 및 실행하여 인스턴스화 되는 특성을 가진다.

이동 에이전트형 망 관리 시스템은 이러한 이동 에이전트의 기능을 사용하여 망 관리를 수행하는 시스템으로서, 기존 시스템의 네트워크 대역폭의 과다 사용과 관리 프로세스의 병목 현상을 줄이고, 지능적인 망 관리를 위해 제안되었다. Bieszczad는 이동 에이전트를 사용하여 망 관리를 수행하는 전체적인 구조를 제안했다[7]. 그리고 Galavas와 Greenwood는 이런 이동에이전트형 망 관리 시스템의 성능을 대역폭 사용량을 기준으로 분석했다[8]. 이동 에이전트는 폴링 방식과 같이 네트워크를 통해서 관리하는 호스트와 1:1로 통신하는 것이 아니라, 그림 2 와 같이 에이전트의 코드와 데이터가 로밍방식으로 호스트에게 전달된다. 따라서 기존의 중앙 집중형 망 관리 방식에 비해 네트워크 사용 대역폭을 크게 줄일 수 있으며, 관리 개체를 이동하면서 필요한 관리 동작을 수행하기 때문에, 중앙의 관리자가 해야하는 부하를 감소시키는 장점이 있다.

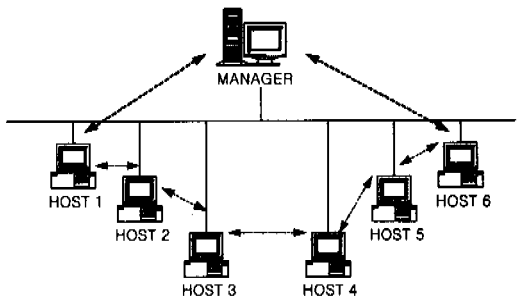


그림 2. 로밍형 이동 에이전트 방식.

하지만 이동 에이전트형 망 관리 시스템은 에이전트의 이동 메카니즘과 시스템의 코드가 복잡하다. 또한 Bieszczad와 Galavas가 제안한 방식은 로밍을 사용하기 때문에 이동 호스트가 많아질수록 데이터량이 증가하는 단점이 있으며, 이더넷과 같은 공유 매체를 사용하는 경우에는 더 심각해진다. 따라서 효율적인 이동 에이전트형 망 관리 시스템의 구축을 위해서는 경량 이동 에이전트의 개발과, 네트워크 특성에 맞는 이동 방식이 필요하다.

2.3 제안하는 세그먼트 클러스터링 방식의 망 관리 시스템

이동 에이전트는 네트워크로 연결된 호스트를 돌아다니면서 자료를 수집한다. 그런데 호스트가 다중 세그먼트에 분포되어 있는 경우에는, 이동 에이전트가 세그먼트를 통과할 때마다 같은 대역폭을 반복해서 소비한다. 이런 대역폭 중복 사용은 세그먼트 이동이 많아질수록 점점 심해진다. 그러나 망 관리 시스템이 관리하는 전체 네트워크를 세그먼트 단위의 클러스터로 묶고, 에이전트의 이동을 한 클러스터 내에서만 하도록 한다면, 세그먼트 통과로 인한 대역폭 낭비를 최소화 할 수 있다. 따라서, 본 논문에서는 클러스터 단위로는 폴링을 하고, 각 클러스터 내에서는 로밍 방식을 사용하는 세그먼트 클러스터링 방식을 제안한다. 세그먼트 클러스터링 방식의 구조는 그림 3 과 같다.

그림에서 매니저는 세그먼트 중의 한 곳에 소속되어 있고, 세 개의 세그먼트가 있으며, 각 세그먼트에는 2개의 호스트가 있다. 이동 에이전트를 관리하는 매니저는 이 세그먼트를 기준으로 클러스터링해서, 클러스터 별로 이동 에이전트의 경로를 고정시킨다. 즉 세그먼트1을 담당하는 에이전트는 호스트1, 호스트2 의 순서로 이동하여 데이터를 수집하

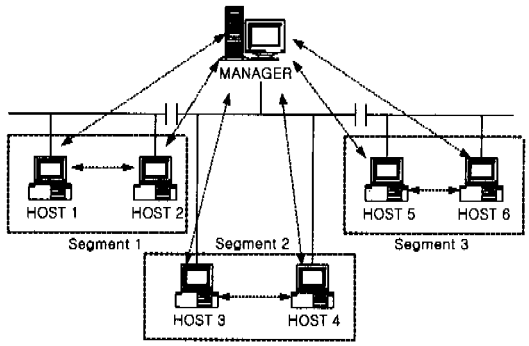


그림 3. 세그먼트 클러스터링 방식.

며, 나머지 세그먼트2와 세그먼트3도 이와 동일한 방법으로 데이터를 수집한다. 로밍 방식과 세그먼트 클러스터링 방식을 대역폭 소비 관점에서 비교한 것은 그림 4 와 같다.

그림 4 에서 볼 수 있듯이, 로밍 방식은 관리하는 호스트를 모두 이동하기 때문에, 호스트를 이동할 때마다 관리 데이터가 누적되고, 세그먼트를 통과할 때마다 대역폭을 중복해서 사용한다. 그러나 세그먼트 클러스터링 방식은 세그먼트 별로 로밍을 하기 때문에, 에이전트의 세그먼트 통과와, 데이터 누적으로 인한 대역폭 낭비를 막을 수 있는 장점이 있다.

이것은 소요되는 시간을 비교했을 때도 마찬가지이다. 전달 매체마다 Maximum Transfer Unit (MTU) 값은 고정되어 있기 때문에, 한 호스트에서 다른 호스트로 데이터를 전달할 때 걸리는 시간은 누적된 데이터의 배수만큼 증가한다. 따라서 세그먼트 클러스터링 방식을 사용하면 대역폭 낭비는 물론 소요 시간도 줄일 수 있다.

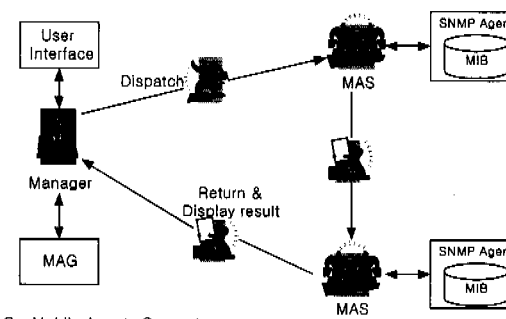
또한 실제로 전체 네트워크를 세그먼트 단위로 클러스터링 할 때는 매니저 프로그램에서 이를 논리적으로 구성하기 때문에, 클러스터링을 위해 물리적으로 네트워크를 재구성해야하는 오버헤드는 발생하지 않는다.

Ⅲ. 이동 에이전트형 망 관리 시스템의 구현

3.1 구현한 이동 에이전트 시스템 개요

본 논문에서 구현한 시스템은 기존의 SNMP구조를 활용하면서, 동시에 이동 에이전트 기능을 추가로 제공하도록, 다중지원 방식으로 구성되었다. 이것은 매니저가 파견한 이동에이전트가 호스트에 도착해서 관리 정보를 얻을 때, 로컬 SNMP 폴링을 사용하기 위함이다. 따라서 국지적으로 SNMP 폴링을 사용하면 기존의 SNMP 장비를 그대로 사용할 수 있으며, 폴링과정에서 트래픽을 발생시키지 않는 장점이 있다.

구현한 시스템은 크게 매니저와, 이동 에이전트, 이동 에이전트 서버로서, IBM의 Java 기반 공개 이동 에이전트 시스템인 aglets을 기본 모델로 사용해서 구현했다[9]. 이것의 전체적인 구조는 그림 5와 같다.



MAG : Mobile Agents Generator
MAS : Mobile Agents Server

그림 5. 구현한 이동 에이전트 시스템의 전체 구조.

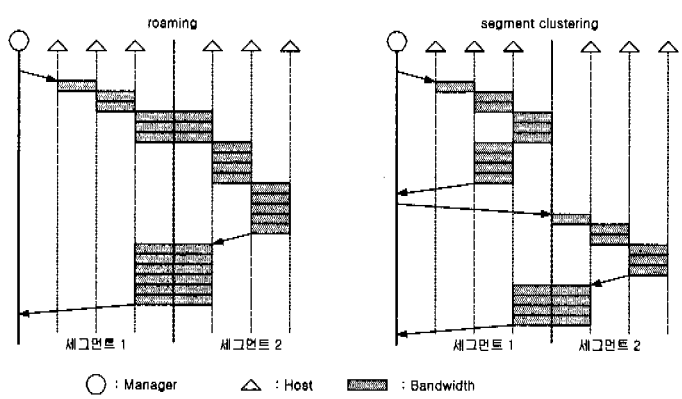


그림 4. 로밍 방식과 세그먼트 클러스터링 방식 비교.

먼저 매니저는 전체 네트워크를 총괄적으로 관리하는 개체이다. 이 안에는 이동 에이전트를 설정하고 결과를 볼 수 있는 사용자 인터페이스가 있다. 그리고 이동 에이전트를 발생시키는 Mobile Agent Generator(MAG)를 가지고 있어서, 이것을 통해 새로운 이동 에이전트를 만들 수 있고, Dispatcher를 이동 에이전트를 파견할 수 있다. 또한 돌아오는 이동 에이전트를 받아들여서, 그것이 수집한 데이터를 저장하고 관리하도록 하였다.

다음에 구현한 것은 이동 에이전트이다. 이동 에이전트는 순수한 이동 에이전트 코드와 MAS에서 얻는 데이터로 구성된다. MAS에서 얻는 데이터는 SNMP의 MIB 객체이기 때문에, 이동 에이전트의 데이터 구조도 MIB의 트리 구조와 같은 형태로 구현했고, 이것은 이동하는 호스트에 따라 가변적으로 변한다. 그리고 에이전트가 이동하는 경로는 사전에 매니저에서 입력되며, 에이전트는 그 경로에 따라서 움직인다.

마지막으로 구현한 것은 관리 호스트에 있는 Mobile Agent Server(MAS)이다. MAS는 매니저나 다른 호스트로부터 도착한 이동 에이전트를 받아들이고, 로컬의 SNMP 에이전트가 관리하는 데이터 중에서 이동 에이전트에 선택된 항목을 폴링을 통해 수집하고, 이것을 이동 에이전트에 적재한다. 작업이 끝나면 MAS는 이동 에이전트를 다른 호스트에 이동시킨다. 다른 호스트에서도 같은 작업을 반복하며, 결국 모든 호스트를 거친 후에는 다시 매니저로 돌아와 결과를 보고하도록 하였다.

3.2 매니저의 구조

호스트로 이동 에이전트를 파견하고, 다시 수집하는 매니저의 구조는 그림 6 와 같다.

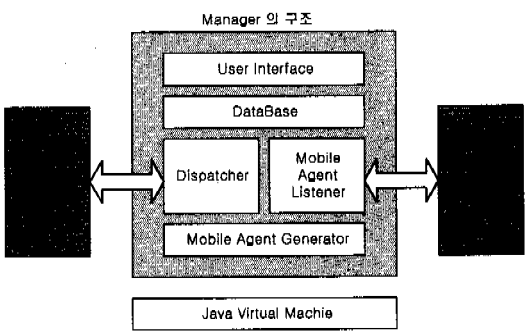


그림 6. 구현한 매니저의 구조

매니저는 이동 에이전트를 파견하고 다시 받아들여서, 네트워크 정보를 총괄적으로 관리 하는 개체이다. 매니저는 aglets을 기반으로 구현했으며, 플랫폼에 독립적인 시스템을 개발하기 위해서 Java를 사용했다. 따라서 이것은 Java 가상 머신 상에서 동작한다.

매니저의 최 하단 부는 MAG를 구현했다. 이것은 사용자의 지시에 따라서 이동 에이전트를 생성한다. MAG 상단에는 Dispatcher와 Mobile Agent Listener(MAL)을 구현했다. Dispatcher는 MAG에서 발생시킨 이동 에이전트를 파견하고, MAL은 호스트에서 자료 수집을 끝낸 이동 에이전트를 받아들인다. 이 상단에는 이동 에이전트의 자료를 저장하는 데이터 베이스와, 관리자의 지시를 받는 사용자 인터페이스가 있다. 매니저의 동작과정은 그림 7 과 같다.

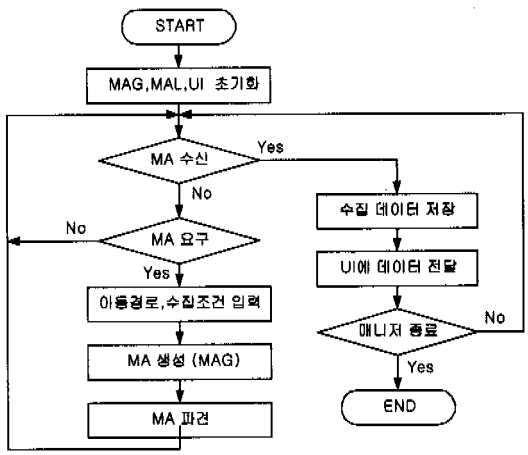


그림 7. 매니저의 동작 과정.

처음 매니저가 시작되면 MAG와, MAL, 그리고 User Interface(UI)를 초기화하고, 대기 상태로 들어가 관리자의 명령을 기다린다. 그 후 관리자로부터 이동 에이전트를 발생시키라는 명령을 UI를 통해서, 에이전트의 이동 경로, 자료 수집조건 등과 함께 입력받는다. 그 후 매니저는 이것을 MAG에 전달해서 새로운 이동 에이전트를 만들고, Dispatcher를 통해 호스트로 전달한다. 그리고 호스트로부터 이동 에이전트가 도착하면, 수집한 정보를 데이터베이스에 저장하고, UI를 통해서 관리자에게 보여준다.

3.3 MAS의 구조

호스트에서 이동 에이전트를 받아들이고 관리 동작을 수행하는 MAS의 구조는 그림 8 과 같다.

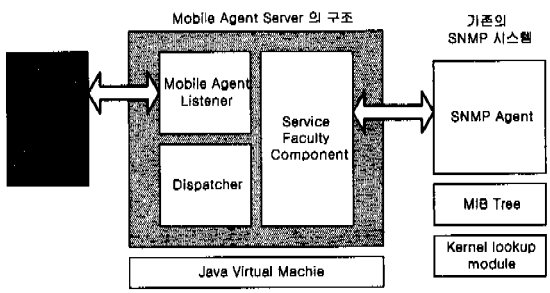


그림 8. 구현한 MAS의 구조

MAS는 이동 에이전트 관리 서버로서, 매니저와 마찬가지로 aglets을 기반으로 구현했으며, 매니저와 인터페이스 통일을 위해 Java를 사용했다. 따라서 MAS의 최 하단부도 Java 가상 머신으로 구성된다. 이것의 상위 계층에는 이동 에이전트를 받아들이는 MAL과, 이동 에이전트를 다른 호스트로 파견하는 Dispatcher로 구성된다. 그리고 그 옆에는 Service Faculty Component(SFC)를 구현했는데, 이것은 기존의 SNMP 에이전트와 이동 에이전트를 연결시키는 인터페이스로서, 로컬 폴링으로 얻은 SNMP 에이전트의 데이터를 이동 에이전트에 적재하는 역할을 담당한다. MAS의 전체적인 동작은 그림 9 와 같다.

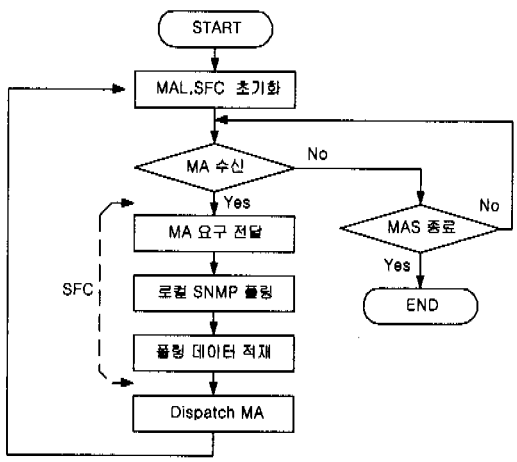


그림 9. MAS의 동작 구조.

처음 MAS가 시작되면, MAL과 SFC를 초기화시키고, 이동 에이전트를 수신하기 위한 대기 상태로 들어간다. 매니저나 다른 호스트로부터 이동 에이전트가 도착하면, MAS는 MAL을 통해서 그것을 받아들이고, SFC에게 이동 에이전트의 요구 사항을 전달한다. SFC는 이것을 받아, 폴링을 통해 로컬 SNMP 에이전트에 의뢰하고, 그 결과를 다시 이동 에이전트에 전달한다. 그리고 정보 수집이 끝나면, Dispatcher를 통해 다른 MAS로 이동 에이전트를 파견한다.

3.3 이동 에이전트의 구조

이동 에이전트는 매니저에서 사용자의 요구에 따라 생성되며, 그 동작은 그림 10 과 같이 생성(Create), 파견(Dispatch), 사멸(Dispose)의 과정을 거친다.

먼저 이동 에이전트는 생성 과정을 통해서 만들

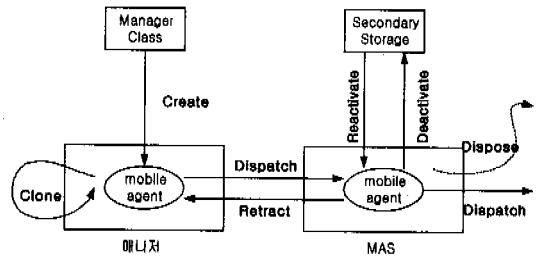


그림 10. 이동 에이전트 동작 과정.

어진다. 생성은 에이전트의 템플릿에서 매니저의 요구에 따라 그 특성에 맞게 인스턴스화 된다. 생성된 이동 에이전트는 관리 대상으로 파견되거나, 귀환(Retract)되며, 관리 호스트에서 잠시 비활성화(Deactive)되었다가 나중에 활성화(Activate)되기도 한다. 여기서 파견, 귀환, 이동(Fetch), 메시지 전달(Message) 등의 이동 에이전트 동작은 Agent Transfer Protocol(ATP)의 규약에 따라 이루어진다[10]. 이동 에이전트의 전달을 위한 매니저와 호스트 사이의 주소체계는 IP 주소를 사용했다.

이동 에이전트는 관리하는 호스트를 이동하면서 호스트의 자료를 수집한다. 따라서 수집한 자료를 가지고 이동하기 위해서는 적당한 자료구조가 필요하다. 본 논문에서는 기존의 SNMP 에이전트가 가지고 있는 자료를 수집하기 때문에, 이동 에이전트도 SNMP의 MIB의 구조에 따라 트리(tree)형으로 구성했다. 트리는 MIB의 색인인 OBJECT-IDENTIFIER에 따른 계층을 이룬다[11]. 본 논문에서 적용한 것은 MIB-II이며, 트리의 수는 에이전트가 경유하는 호스트의 수에 따라 가변적으로 변한다.

Ⅳ. 성능 분석

이동 에이전트형 망 관리 시스템의 대역폭 사용량과 새로 제안한 세그먼트 클러스터링 방식 망 관리 시스템의 대역폭 사용량을 수치적으로 분석하고, 실제 구현한 시스템을 통해 이를 검증한다.

4.1 실험 환경

본 논문에서 구현한 이동 에이전트형 망 관리 시스템은 기존의 여러 가지 콤포넌트를 사용했다. 앞서 밝힌 대로, 이동 에이전트는 IBM에서 개발한 aglets을 사용했다. 그리고 aglets을 동작시키는 전체적인 망 관리 시스템은 Java를 사용해 구현하였으며, TCP/IP 프로토콜 스택 중에서 TCP를 사용한다. 물론 호스트에서 SNMP 에이전트로의 로컬 폴링은

UDP 기반의 SNMP 메시지를 사용한다. 매니저와 호스트로 사용한 시스템은 펜티엄III 이다. 관리 항목은 RFC1213에 정의되어 있는 MIB-II를 사용했다. 그리고 관리 대상 호스트들은 세그먼트 당 3개를 지정했으며, 모두 7개의 세그먼트를 사용했다.

4.2 이동 에이전트형 망 관리 시스템의 대역폭 사용량 분석

현재 대부분의 네트워크는 효율적인 트래픽 사용을 위해, 집단의 특성에 따라 여러 개의 세그먼트로 나뉘어져 있다. 이때 패킷 전달에 따른 대역폭 사용량은 로컬에서 패킷이 전달될 때에 비해, 다른 세그먼트로 패킷을 전달할 때가, 패킷이 경유하는 세그먼트 수로 인해 더 크다. 따라서 이동 에이전트형 망 관리 시스템의 대역폭 사용량 분석을 위해서는, 매니저와 관리 호스트가 로컬 네트워크에 있는 단일 세그먼트 환경과, 매니저와 관리 호스트가 여러 세그먼트에 분산되어 있는 다중 세그먼트 환경을 고려해야 한다.

(1) 매니저와 관리 호스트가 같은 단일 세그먼트 내에 있는 경우

가장 기본적인 경우는 매니저와 관리 호스트가 같은 세그먼트 내에 존재하는 경우이다. 이 경우 평균 request/response 패킷의 바이트 단위 크기인 S_{req} 에 대하여, N_o 개의 호스트에서, k 개의 MIB 객체수를, 횟수 p 만큼 폴링 한다고 할 때, 송수신되는 $2*N_o$ 개의 패킷이 사용하는 SNMP 방식의 네트워크 대역폭 사용량 B_{snmp} 는 다음과 같이 구해진다.

$$B_{snmp} = 2 * S_{req} * N_o * k * p. \quad (1)$$

반면에 이동 에이전트형 망 관리 시스템의 네트워크 대역폭 사용량 B_{mobile} 은 다음과 같이 구해진다.

$$B_{mobile} = (N_o + 1) * S_c + \sum_{i=1}^{N_o} (S_s * i). \quad (2)$$

여기서, S_c 는 이동하는 에이전트 코드의 바이트 단위 크기이며, 이것은 호스트를 이동할 동안 항상 일정한 값을 가진다. 그리고 S_s 는 에이전트가 관리 호스트에서 얻는 객체의 정보의 크기이며, 호스트에 따라 변할 수 있다.

식(2) 의 첫 번째 항에서 에이전트의 경로는 N_o 의 호스트를 경유하고 돌아오기 때문에, 기존의 N_o 개의 호스트에 1이 더해진다. 그리고 두 번째 항에서 나타낸 객체 정보의 크기는 매니저에서 에이전트를 파견할 때는 발생하지 않으며, 첫 번째 호스트부터 마지막 목적지인 매니저로 돌아갈 동안, 호스트를 거칠 때마다 누적된다.

관리 호스트의 수를 고정시키고 호스트에서 수집하는 객체의 정보를 폴링 간격을 늘리며 증가시켰을 때, 기존의 SNMP 방식의 대역폭 사용량과 이동 에이전트형 망 관리 시스템의 대역폭 사용량을 비교한 것은 그림 11 과 같다.

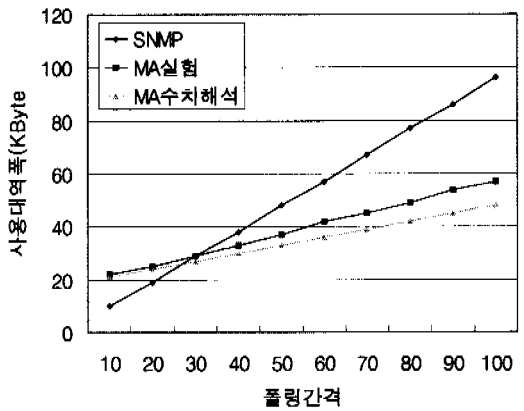


그림 11. 폴링 간격 증가에 따른 대역폭 사용량 비교.

이 실험을 통해 기존 SNMP 방식은 수집하는 객체의 정보가 증가할수록 폴링 간격도 비례해서 증가하기 때문에 사용 대역폭 B_{snmp} 는 선형적으로 증가하는 반면, 이동 에이전트 방식은 폴링 방식을 사용하지 않고, 이동 에이전트가 한번 이동했을 때 그곳의 데이터를 모두 수집한다. 따라서 소요대역폭 B_{mobile} 은 객체의 정보가 늘어나도 이동하는 호스트의 수와 호스트에서 수집한 전체 데이터의 크기에만 비례해서 증가하기 때문에, SNMP 방식에 비해서 증가폭이 작다. 그림 11은 식(2)의 수치해석으로 이동에이전트의 대역폭 사용량을 나타낸 'MA수치해석'과, 실제 구현한 이동에이전트를 사용하여 결과를 추출한 'MA실험' 등 두 가지 경우를 함께 보였다. 실제 실험에서 이동 에이전트 시스템은 TCP/IP 스택에서 TCP를 사용하기 때문에, SYN/ACK 패킷등이 추가되었다. 그 결과 실험 치의 대역폭 사용량이 이론 치보다 조금 더 증가했다. SNMP 방식은 UDP를 사용하기 때문에 SYN/ACK 패킷 추가에 의한 대역폭 증가는 없다.

따라서 단일한 호스트에서 수집하는 객체 수와, 폴링 간격이 증가할 경우에는, 이동 에이전트를 이용한 방식이 SNMP 방식에 비해 대역폭을 적게 사용함을 알 수 있다.

그림 12 은 관리 객체의 수는 고정시키고 관리 대상 호스트를 증가 시켰을 때, SNMP 방식과 이동 에이전트 방식의 대역폭 사용량을 비교한 것이다.

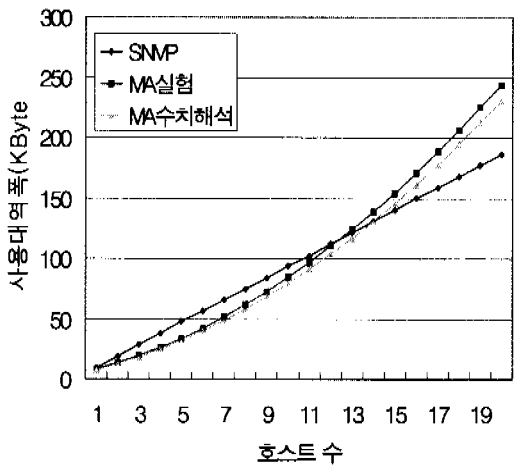


그림 12. 관리 대상 호스트 증가에 따른 대역폭 사용량 비교(호스트당 50개의 객체)

그림 12 에서 볼 수 있듯이, 호스트에서 수집하는 객체 수를 고정시키고 관리 호스트를 늘린 경우는, 이동 에이전트 방식이 가지고 다니는 정보의 수가 누적되기 때문에 SNMP 방식보다 대역폭을 많이 사용하는 것을 알 수 있다. 따라서 단일 세그먼트 내에서 이동 에이전트 방식의 대역폭 사용량이 SNMP 방식에 비해 항상 적은 것은 아니다. 식(3)에서 알 수 있듯이, 식(2)의 값이 식(1) 의 값보다 작은 경우, 즉 호스트를 경유하면서 누적된 정보 S_s의 크기가 폴링 간격 증가에 의한 정보의 증가량보다 작은 경우에 한해서 이동 에이전트 방식의 대역폭 사용량이 SNMP 방식에 비해서 작다.

$$(N_o+1)*S_c+\sum_{i=1}^{N_o}(S_s*i) < 2*S_{req}*N_o*k*p. \quad (3)$$

(2) 관리 호스트들은 단일 세그먼트에 모여 있고, 호스트와 관리자는 다른 세그먼트에 있는 경우

이것은 관리 호스트들은 단일 세그먼트에 모여있는 반면, 호스트와 관리자가 다른 세그먼트에 떨어져 있는 경우이다. 실험은 떨어져 있는 세그먼트 수를 증가하면서, 두 방식의 대역폭 사용량을 비교했다. 이 경우 SNMP 방식의 대역폭 사용량 B_{snmp}는 식(4)와 같다. 이 식은 기본적으로 로컬 폴링을 나타내는 식(1)에 세그먼트의 수인 N_s를 추가한 형태이다. 관리 호스트들이 전부 단일 세그먼트 내에 모여있기 때문에, 대역폭 사용량 B_{snmp}는 세그먼트 개수인 N_s 에 비례해서 증가함을 알 수 있다.

$$B_{snmp}=2*S_{req}*N_o*k*N_s*p. \quad (4)$$

그리고, 이 경우 이동 에이전트형 망 관리 시스템의 대역폭 사용량 B_{mobile}은 식(5) 와 같다.

$$B_{mobile}= 2(N_s-1)*S_c+(N_o+1)*S_c+\sum_{i=1}^{N_o}(S_s*i)+(N_s-1)\sum_{i=1}^{N_o}S_s \quad (5)$$

여기서 N_o 는 한 세그먼트에 있는 호스트의 수이며, $\sum_{i=1}^{N_o}S_s$는 N_o 번째 호스트에서 누적된 데이터의 크기이다. 식(5)의 첫 번째 항은 에이전트 코드가 호스트들이 있는 세그먼트에 도착하고, 다시 매니저로 돌아가면서 사용한 대역폭의 크기이다. 따라서 총 이동 세그먼트 수는 (N_s-1) 개이며, 매니저와 호스트를 왕복하기 때문에 2를 곱한다. 두 번째 항은 목적지 세그먼트 내에서 에이전트 코드가 이동하면서 사용한 대역폭의 크기이다. 따라서 총 이동 횟수는 (N_o+1) 이다. 세 번째 항은 목적지 세그먼트에서 누적된 호스트 객체 정보의 크기이다. 그리고 마지막 항은 목적지의 마지막 N_o 호스트에 누적된 객체 정보를 매니저가 있는 원래 세그먼트로 돌려보낼 때 사용한 대역폭의 크기이다. 이렇게 매니저와 호스트가 서로 다른 세그먼트에 있는 경우 B_{mobile}은 처음 매니저에서 호스트들이 있는 세그먼트로 에이전트를 파견할 때와, 정보 수집을 마치고 매니저로 돌아갈 때만 세그먼트를 통과하기 때문에, 폴링 간격이 늘어날 때마다 세그먼트가 증가되는 B_{snmp} 비해서 훨씬 작다.

관리 호스트들이 단일 세그먼트에 모여있고 매니저와 호스트들이 서로 다른 세그먼트에 있는 경우, 세그먼트 개수 증가에 따른 B_{mobile}와 B_{snmp}를 비교하면 그림 13 와 같다. 그림 13 에서 호스트의 수는 3개이고, 각 호스트에서 가져오는 객체 정보의 수는 50개이며, 매니저와 호스트 사이의 세그먼트 수만 하나씩 증가 시켰다.

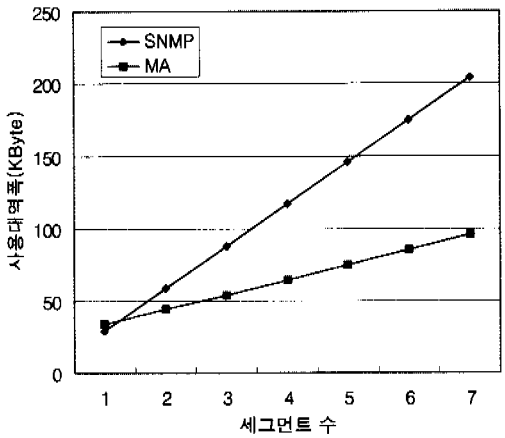


그림 13. 세그먼트 증가에 따른 대역폭 사용량 비교(3개의 호스트, 호스트당 50개의 객체).

그림에서 볼 수 있듯이 호스트 수를 고정시키고 세그먼트 수만 증가시키면 매니저와 호스트 사이의 이동 량이 많은 SNMP 방식이 이동 에이전트 방식에 비해 대역폭을 많이 소비함을 알 수 있다.

(3) 관리 호스트들이 여러 세그먼트에 분산되어 있는 경우

이것은 관리 호스트들이 여러 개의 세그먼트에 분산되어 있는 경우이다. 이 경우, 관리 호스트는 매니저와 같은 세그먼트에 있거나, 다른 여러 세그먼트에 분포한다. SNMP 방식의 대역폭 사용량 B_{snmp}은 식(6) 과 같다. 여기서 i 는 세그먼트 수를 의미하고, $N_{o(i)}$ 는 i 번째 세그먼트에 있는 호스트 수를 의미한다. 식(6)은 식(4)에서 나타낸 세그먼트 별 폴링 대역폭을 전부 합친 것이다.

$$B_{snmp} = \sum_{i=1}^{N_s} 2 * S_{req} * N_{o(i)} * k * i * p. \quad (6)$$

반면에, 이동 에이전트형 망 관리 시스템은 하나의 이동 에이전트가 각 세그먼트를 돌아다니면서 데이터를 수집해야 한다. 따라서 이동 에이전트형 망 관리 시스템의 대역폭 사용량 B_{mobile}은 다음과 같다.

$$B_{mobile} = 2*(N_s-1)*S_c + (N_o+1)*S_c + \sum_{i=1}^{N_o}(S_s * i) + (N_s-1)\sum_{i=1}^{N_o} S_s + \sum_{j=1}^{N_s-1}\left(\sum_{i=1}^{N_s-1} N_{o(i)}\right) * S_s. \quad (7)$$

식(7)의 첫 번째, 두 번째, 세 번째, 네 번째 항의 내용은 식(5)와 같다. 그리고 마지막 항은세그먼트를 이동할 때 중복되는 객체 정보를 더한 것인데, 이것은 한 세그먼트 내에서는 호스트를 이동할 때 객체 정보를 누적시키고, 각 세그먼트의 객체 정보량을 전부 합친 것이다.

SNMP 방식은 세그먼트 수에 비례해서 사용 대역폭이 증가하고, 전체 대역폭 사용량은 이것의 누적치 이기 때문에, 이동 에이전트형 망 관리 시스템이 SNMP 방식에 비해서 대역폭을 적게 소비함을 알 수 있다. 하지만 하나의 이동 에이전트가 모든 호스트를 돌아다니기 때문에, 관리 호스트가 늘어날수록 데이터 누적으로 인해 대역폭 소비량이 크게 증가한다. 그림 14 는 관리 호스트가 여러 세그먼트에 분산된 경우, 두 방식의 대역폭 사용량을 비교한 것이다. 그림 14 에서 호스트의 수는 세그먼트를 하나 증가시킬 때마다 세 개씩 증가시켰으며, 한 호스트에서 가져오는 객체는 50개이다.

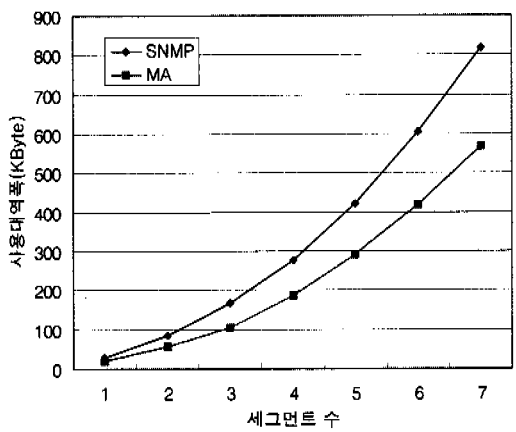


그림 14. 호스트 수와 세그먼트 증가에 따른 대역폭 사용량 비교(세그먼트 당 3개의 호스트, 호스트당 50개의 객체)

그림에서 볼 수 있듯이 세그먼트를 증가시킬수록 SNMP 방식이 이동 에이전트 방식에 비해서 대역폭을 많이 소비함을 알 수 있다.

4.3 제안한 세그먼트 클러스터링 방식의 성능 분석

앞서 살펴본 대로 다중 세그먼트에 관리 호스트들이 분산된 경우, 이동 에이전트가 모든 호스트를 돌아다니면 데이터 누적뿐만 아니라 세그먼트 이동으로 인한 대역폭 낭비가 많다. 따라서 본 논문에서는 이런 다중 세그먼트 환경에서 대역폭 사용량을 줄이기 위해서 세그먼트 클러스터링 방식을 제안했

다. 제안한 세그먼트 클러스터링 방식은 세그먼트 단위로는 폴링 방식을 사용하고, 각 클러스터 내에서는 이동 에이전트 방식을 사용한다. 클러스터링 방식에서 소모하는 대역폭은 다음과 같다.

$$B_{cluster} = \sum_{i=1}^{N_s}[(N_{\alpha(i)}+1)S_c + 2(i-1)S_c] + \sum_{i=1}^{N_s}[\sum_{j=1}^{N_{\alpha i}}(S_s * j) + (i-1)\sum^{N_{\alpha i}} S_s] \quad (8)$$

식(8)에서 대괄호로 묶은 첫 번째 항은 에이전트 코드가 전체 클러스터를 이동하면서 사용한 대역폭의 합이고, 두 번째 항은 호스트 객체 정보 크기의 합이다. 먼저 첫 번째 항을 보면, 이동 에이전트는 클러스터로 구분한 각 세그먼트 내에서 모든 호스트를 돌아다닌다. 따라서 i 번째 세그먼트의 호스트 개수에 1이 많은 $N_{\alpha(i)}+1$ 곳을 이동하고, 매니저에서 목적 세그먼트로 이동했다가 다시 매니저로 돌아오기 때문에 $2(i-1)$ 을 더해 주어야 한다. 그리고 에이전트가 각 클러스터에서 사용한 대역폭을 전부 더한다. 대괄호의 두 번째 항을 보면, 클러스터로 구분한 i 번째 세그먼트 내에서는 호스트를 돌면서 객체 정보를 얻어오기 때문에, j 개의 호스트를 누적한 $\sum_{j=1}^{N_{\alpha i}}(S_s * j)$ 만큼의 대역폭을 사용하고, 누적된 마지막 객체 정보의 값을 매니저로 다시 전송할 때 $(i-1)\sum^{N_{\alpha i}} S_s$ 만큼의 대역폭을 추가로 소비한다. 그리고 마지막으로 이동 에이전트가 각 클러스터에서 사용한 대역폭을 전부 더한다.

SNMP 방식과 단일 이동에이전트 방식, 그리고 세그먼트 클러스터링 방식 등의 세 가지를 실험한 결과, 그림 15 과 같이 제안한 세그먼트 클러스터링 방식의 대역폭 사용량이 가장 적음을 알 수 있다. 그림 15의 실험 환경은 그림 11과 같다.

V. 결 론

본 논문에서는 세그먼트 클러스터링 방식을 사용하는 이동 에이전트형 망 관리 시스템의 성능을 소모 대역폭 관점에서 수식적으로 분석하고, 실제로 구현해서 단일/다중 세그먼트 환경에서 기존의 SNMP 기반 망 관리 시스템과 비교하여 검증하였다. 이를 위해서 먼저 이동 에이전트형 망 관리 시스템의 대역폭 사용량을 수식적으로 분석하였다. 그리고 플랫폼에 독립적인 Java를 사용해서 이동에

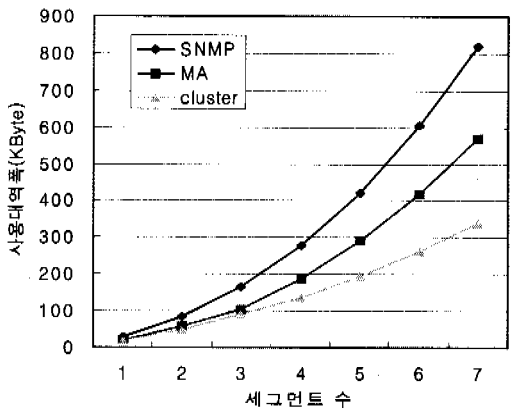


그림 15. 세그먼트 증가에 따른 대역폭 사용량 비교(세그먼트 당 3개의 호스트, 호스트당 50개의 객체)

이전트형 망 관리 시스템을 구현하였으며, 이것을 단일 세그먼트와 다중 세그먼트 환경의 망 관리 시스템에 적용하여 실제 대역폭 사용량을 검증하였다.

그 결과, 단일 세그먼트 내에서는 관리 호스트가 증가할수록 이동 에이전트 방식의 대역폭 사용량이 기존의 SNMP 방식에 비해 많아지지만, 다중 세그먼트 환경에서는 이동 에이전트 방식의 대역폭 사용량이 기존 방식에 비해 적음을 알 수 있었다. 그리고 관리 호스트들이 다중 세그먼트에 분포한 경우에는 본 논문에서 제안한 세그먼트 클러스터링 방식의 대역폭 사용량이 가장 적음을 알 수 있었다.

참 고 문 헌

[1] William Stallings, SNMP, SNMPv2, and RMON, Second Edition, Addison-Wesley, 1996.

[2] W.J. Buchanan, M.Naylor, A.V.Scott, "Enhanced Network Management using Mobile Agents," Seventh IEEE International Conference and Workshop, pp.218-216, 2000 .

[3] M.Zapf, K.Herrmann, K.Geihs, "Decentralized SNMP Management with Mobile Agents", 1999 .

[4] J. Case, M.Fedor, M. Schoffstall, J. Davin, A Simple Network Management Protocol (SNMP), RFC1157, 1990 .

[5] K.McCloghrie, M. Rose, Management information base for netwokr management of TCP/IP-based internets: MIB-II, RFC1213.

[6] Yemini Y., Goldszmidt G., Yemini S., "Network Management by Delegation", Proceedings of the 2nd International Symposium on Integrated Network Management, April 1991.

[7] Andrzej Bieszczad, Bernard Pagurek, Tony White, "Mobile Agent for Network Management," IEEE Communications Survey, Fourth Quarter 1998, Vol 1, No 1.

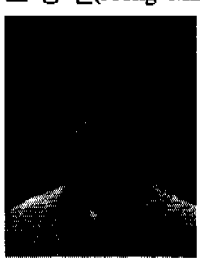
[8] Damianos Gavalas, "Advanced network monitoring applications based on mobile/ intelligent agent technology," *Computer Communications* Vol 23, pp. 720-730, 2000 .

[9] Aglet, http://www.trl.ibm.co.jp/aglets/ .

[10] ATP(Agent Transfer Protocol), http://www.trl.ibm.co.jp/aglets/atp/atp.htm .

[11] Abstract Syntax Notation One, ITU-T Recommandation, X.680 .

노 정 민(Jeong Min Noh)



1995년 3월~1999년 2월 : 한국항공대학교 항공통신정보공학과 졸업 (공학사)

1999년 3월~2001년 2월 : 한국항공대학교 항공통신정보공학과 대학원 졸업 (공학석사)

현재 : 머큐리 연구원

<주관심 분야> 망 관리 시스템, 이동 에이전트, VoIP 기술, 실시간 OS

윤 종 호(Chong Ho Yoon)

1977년 3월~1984년 2월 : 한양대학교 전자공학과 졸업 (공학사)

1984년 3월~1986년 2월 : 한국과학기술원 전기 및 전자공학과 졸업 (공학석사)

1986년 3월~1990년 8월 : 한국과학기술원 전기 및 전자공학과 졸업 (공학박사)

1995년 8월~1996년 8월 : University of Arizona 방문교수

1991년 8월~현재 : 한국항공대학교 항공통신정보공학과 부교수