네트워크 관리를 위한 이동 에이전트의 성능평가

권 혁 찬† · 김 흥 환†† · 유 관 종†††

요 약

본 논문에서는 네트워크 관리 시스템에서 SNMP(Simple Network Management Protocol) 프로토콜을 이용한 중앙 집중형 방식과 이동 에이전트(mobile agent)를 이용한 분산방식 각각의 성능을 평가하기 위한 성능평가 모델을 제시한다. 먼저 균등 네트워크 상에서 적용 가능한 모델을 제시하고 실제 실험을 통하여 각 파라미터 값의 변화에 따른 네트워크 수행 시간을 비교한다. 다음으로 비균등 네트워크에 적용 가능하도록 모델을 확장하고, 제시한 모델을 응용하여 실제 가상의 시나리오 하에서 네트워크 소요시간을 줄일 수 있는 방안과 그에 대한 실험 결과를 제시한다. 본 성능평가 모델은 차후 네트워크 관리시스템을 개발할 때에, 효율적인 패러다임(paradigm)과 수행 구조를 선택하는데 도움이 될 수 있을 것이다.

키워드 : 이동 에이전트, RPC, 네트워크 관리, SNMP

A Performance Evaluation of Mobile Agent for Network Management

Hyeok-Chan Kwon† · Heung-Hwan Kim†† · Kwan-Jong Yoo†††

ABSTRACT

This paper mentions a centralized approach based on SNMP protocol and distributed approach based on mobile agent in network management system. And it presents a few quantitative models for systematically evaluating those two different approaches. To do this, we propose model that is applicable under a uniform network environment, and compare network execution times of each paradigms based on parameters from simulation. The model is then refined to take into account non-uniform networks. We show that it can reduce overall network execution times by determining the best interaction patterns to perform network management operations from this model. We believe that the model proposed here should help us to decide appropriate paradigms and interaction patterns for developing network management applications.

Key word : mobile agent, RPC, network management, SNMP

1. 서 론

네트워크 관리 시스템은 네트워크의 효율성과 생산성을 최대화하기 위해 복잡한 네트워크를 통제하는 시스템이다. 네트워크 관리에 사용된 기존의 방식은 중앙 집중형 방식으로 대표적인 프로토콜로 SNMP(Simple Network Management Protocol), CMIP(Common Management Information Protocol) 등이 있다. 이중 SNMP[1]는 TCP/IP에 기반한 RPC(Remote Procedure Call) 방식의 네트워크 관리 프로토콜로서, 구조가 단순하며 구현도 용이해 현재 가장 널리 사용되는 프로토콜이다. 그러나 이러한 중앙 집중형 네트워크 관리는 네트워크 관리자에게 관리 작업이 집중화 되므로, 관리자의 처리 부하와 네트워크 트래픽이 증가한다는 문제가 있다. 이러한 문제점을 보완하기 위해 최근 들어서는 네트워크 관리를 위해 이동 에이전트를 이용하고자 하는 연구가 활기를 띄고 있다[2-6].

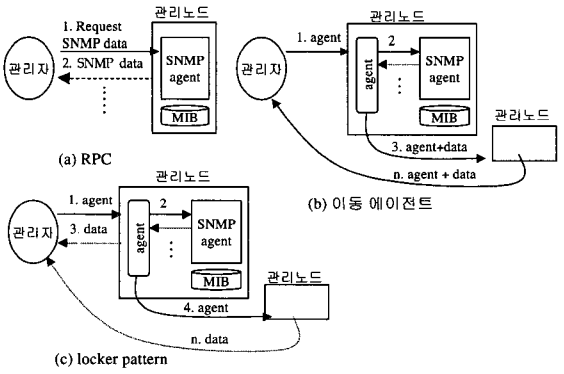
기존의 RPC나 멀티 에이전트가 원격 통신(remote communication)에 의해 작업을 수행했던 반면, 이동 에이전트[2, 13] 시스템은 에이전트 자신이 서버를 돌아다니며 주어진 작업을 수행하는 구조를 갖는다. 네트워크 관리의 경우에는 이동 에이전트가 관리되는 네트워크 관리 대상자들로 이동하며 관리 작업을 수행한다. 이러한 이동 에이전트의 이동성이라는 특성은 원격 통신비용을 줄일 수 있다는 장점이 있기는 하지만, 노드 방문 시 획득한 데이터들이 계속 누적되기 때문에 오히려 네트워크 소요시간이 증가될 수 있다는 단점도 있다. 따라서 네트워크 관리를 위한 이동 에이전트의 정확한 성능을 비교 평가하기 위해서는 직관적인 평가보다 실제 수행 능력을 수치적으로 평가할 수 있는 성능평가 모델이 필요하다.

[9-10]에서는 이동 에이전트를 이용하여 간단한 네트워크 관리 시스템을 개발하였으나 이동 에이전트를 이용한 경우와 기존의 SNMP 프로토콜을 이용한 중앙 집중형 방식에 대한 비교, 평가는 명확히 이루어지지 않았다. 실제 성능을

† 준 회 원 : 한국전자통신연구원 선임연구원
†† 정 회 원 : 서원대학교 컴퓨터정보통신학부 교수
††† 정 회 원 : 충남대학교 정보통신공학부 교수
논문접수 : 2000년 9월 6일, 심사완료 : 2001년 1월 26일

평가한 경우도 간단한 네트워크 관리 시나리오에 대해 특정 요구에 대한 응답시간을 측정하여 비교한 정도이다. 본 논문에서는 네트워크 관리를 위한 이동 에이전트와 기존의 RPC 방식의 성능을 양적으로 평가하기 위한 수학적 평가 모델을 작성한다. 이 평가 모델을 사용하여 네트워크 관리 시스템을 개발할 때에 적합한 분산 패러다임과 수행 구조를 선택할 수 있을 것이다.

이동 에이전트는 데이터를 누적시키며 이동하는 전형적인 구조와 locker pattern이 적용된 구조[5] 각각에 대해 모델링 하였다. locker pattern은 RPC의 경우 통신 횟수가 많다는 단점과 이동 에이전트의 경우 누적되는 데이터 양이 많다는 단점을 일부 극복하기 위한 모델로서, 이동 에이전트가 방문한 노드에서 수집한 데이터를 누적시켜 이동하는 것이 아니라, 방문한 노드의 임시 보관소에 데이터를 보관하고 다른 노드로 이동하는 방식이다. 이 데이터는 추후 RPC 방식으로 클라이언트로 전달된다. (그림 1)에서는 네트워크 관리에서 RPC와 이동 에이전트 그리고 locker pattern의 수행 방식을 보여준다. 실제 분산 시스템의 성능은 네트워크 소요 시간, 노드의 처리부하(processing load), 보안 등의 문제를 함께 고려하여 평가해야 하지만, 본 연구에서는 네트워크 소요시간만을 고려하였으며, 기타 요소들은 추후에 고려할 예정이다.

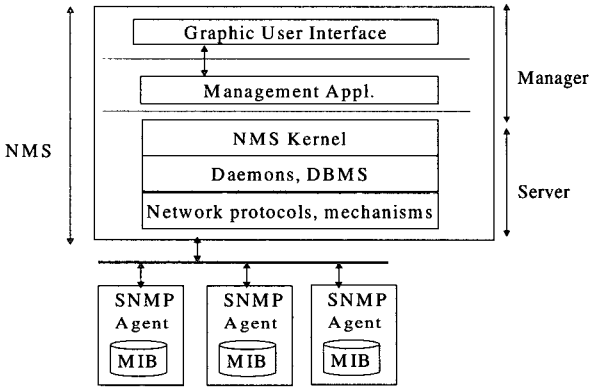


(그림 1) 네트워크 관리 시스템을 위한 3가지 패러다임

본 논문의 구성은 다음과 같다. 먼저 2장에서 네트워크 관리 시스템의 기본구조에 대해 살펴보고 3장에서는 균등 네트워크 환경을 고려한 성능평가모델을 제안하고 그에 대한 실험 결과를 보인다. 4장에서는 비균등 네트워크를 위한 확장된 모델을 제시하고, 주어진 모델을 기초로 하여 네트워크 소요시간을 줄일 수 있는 방안도 함께 제시한다. 5장에서 향후연구 과제와 결론을 제시한다.

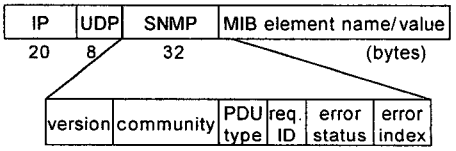
2. 네트워크 관리

일반적인 네트워크 관리 시스템[1]의 구조는 (그림 2)와 같다. NMS(Network Management Station)는 네트워크 관리자이다. 네트워크 관리자(NMS)와 SNMP 에이전트 사이에 특정한 정보를 주고 받는 것이 네트워크 관리의 기본이다. SNMP 에이전트는 일반 호스트(host)와 서버(server), 게이트웨이(gateway), 라우터(router), 브릿지(bridge), 허브(hub)와 같은 네트워크 구성요소에 설치되어 있는 소프트웨어로서 네트워크 관리자로부터의 요구에 응답하는 기능을 갖고 있다. SNMP 에이전트는 크게 관리자와의 인터페이스 부분과 MIB(Management Information Base)와 인터페이스하는 부분으로 나눌 수 있다. SNMP를 이용해서 관리되는 특정한 정보와 자원 등을 모아놓은 집합체가 MIB (Management Information Base)이다. 최근 사용되는 MIB-II의 경우 171개의 객체를 포함하고 있다. 네트워크를 관리한다는 것은 관리 대상인 장비들이 제공하는 MIB 중에서 특정 값을 얻어와 그 장비의 상태를 파악하거나 그 값을 변경함을 의미한다.



(그림 2) 전형적인 네트워크 관리 구조

NMS와 SNMP 에이전트간에 통신을 위해 사용되는 프로토콜이 SNMP이며 get-request, get-response, get-next-request, set-request, trap 등의 5가지 primitive가 제공된다. get-request는 네트워크 관리자가 SNMP 에이전트에게 관리 정보를 요구하기 위한 것이며, SNMP 에이전트가 요구받은 관리 정보를 관리자에게 보내주기 위한 메시지가 get-response이다. get-next-request는 다음 관리 정보를 읽기 위한 메시지이며, set-request는 네트워크 관리자가 SNMP 에이전트의 관리 정보를 변경하기 위한 메시지이다. trap은 네트워크 관리자에게 특정 상황이 발생했음을 알려주기 위한 메시지이다. SNMP는 데이터 전송 시 UDP 방식을 사용한다. SNMP 메시지의 형식은 (그림 3)과 같다.



(그림 3) SNMP 메시지 형식

3. 균등 네트워크 환경을 고려한 성능평가 모델

3.1 성능평가 모델

본 절에서는 균등 네트워크(uniform network) 환경에서의 성능평가 모델을 제시한다. 균등 네트워크란 모든 노드 간의 네트워크 대역폭이 동일한 네트워크 환경을 의미한다. 제공되는 수식은 네트워크 관리를 위해 SNMP 변수들을 관리되는 장치로부터 가져오는 get-request와 get-response 작업에 대한 것이다. 네트워크 관리에서 가장 큰 비중을 차지하는 작업이 get-request와 get-response를 처리하는 부분이기 때문에 이 부분에 주안점을 두어 모델링하였다[1, 7, 8]. 비균등 네트워크 환경에 대한 모델은 4장에서 제시한다. 멀티캐스트(multicast)는 고려하지 않았다. 모델링을 위해 사용되는 파라미터는 <표 1>과 같다.

〈표 1〉 모델링을 위한 파라미터

파라미터	설 명
N	관리되는 노드의 수
R_i	i번 노드로의 request 횟수
S_{nr}	n번 노드에서 r번째 request할 때 요구되는 SNMP 변수의 개수
L_V	SNMP variable name의 크기
L_S	SNMP 변수들의 전송을 위한 메시지 헤더의 크기 (IP+UDP+SNMP header)
L_T	TCP 전송을 위한 메시지 헤더의 크기 (IP+TCP header)
L_{VB}	variable binding의 크기 (SNMP variable name + value)
σ	네트워크 지연시간(delay)
β	네트워크 대역폭(bandwidth)
S_A	이동 에이전트의 크기

RPC[4]는 대표적인 클라이언트 서버 모델이다. 네트워크 관리자는 관리노드의 상태파악을 위해 필요한 SNMP 변수의 리스트를 관리노드로 보낸다(get_request). 관리노드는 요구받은 SNMP 변수의 값을 관리자에게 반환하여 준다(get_response). 관리자와 서버는 작업 수행 시 원격 통신을 사용하게 된다. RPC에 대한 전체 네트워크 로드(network load)는 식 (1)과 같다.

$$L_{RPC}=\sum_{n=1}^{N}\sum_{r=1}^{R_n}(2L_S+(L_V+L_{VB})S_{nr}) \tag{1}$$

N번째 관리노드로 r번째 요구에 대해, 관리자가 관리노드로 SNMP 변수들을 요구하는데 $L_S+L_VS_{nr}$, 관리노드가 관리자에게 요구받은 정보를 반환하는데 $L_S+L_{VB}S_{nr}$ 만큼의 네트워크 로드가 발생한다.

식 (2)는 RPC에 대한 전체 네트워크 소요시간에 대한 수식이다. RPC 방식의 경우 N개의 관리되는 노드로 각각 Rn번의 request와 response가 발생하므로, $2\sum_{n=1}^{N}\sum_{r=1}^{R_n}\delta$ 만큼의 네트워크 지연시간이 추가로 소요된다.

$$T_{RPC}=2\sum_{n=1}^{N}\sum_{r=1}^{R_n}\delta+\frac{L_{RPC}}{\beta} \tag{2}$$

이동 에이전트의 경우, 각각의 노드를 방문할 때 검색한 SNMP변수들과 그 값은 이동 에이전트의 데이터 영역에 계속 누적 시키고 이동한다. 이동 에이전트의 경우 전체 네트워크 로드는 식 (3), 전체 네트워크 소요시간은 식(4)와 같다.

$$L_{MA}=\sum_{n=0}^{N}(S_A+\sum_{i=1}^{n}\sum_{r=1}^{R_n}S_{nr}L_{VB}) \tag{3}$$

$$T_{MA}=(N+1)\delta+\frac{L_{MA}}{\beta} \tag{4}$$

식 (3)의 $\sum_{i=1}^{n}\sum_{r=1}^{R_n}S_{nr}L_{VB}$은 이동 에이전트가 n번째 노드 방문 시 누적되는 데이터의 양을 나타낸다. 식 (3)에서 S_A는 실제 네트워크 상에 전송되는 이동 에이전트의 크기로서 식 (5)와 같다.

$$\begin{aligned} S_A &= M_A+(M_A/payload)*(IPheader+TCPheader)\\ M_A &= ATP\_header+M\\ M &= M\_state+M\_code+M\_data \end{aligned} \tag{5}$$

이동 에이전트의 실제 크기는 이동 에이전트의 코드와 상태와 데이터 부분을 합한 M과 같다. 이동 에이전트의 전송을 위해서 M에 ATP(Agent Transfer Protocol) 헤더가 붙는다. ATP 헤더가 붙은 M_A는 TCP메시지로 분할된다. 식 (5)의 payload는 TCP payload이다. 분할된 각각의 조각들은 TCP 헤더와 IP 헤더로 사여 TCP/IP 방식으로 전송된다[11].

locker pattern의 경우, 에이전트는 각각의 노드를 방문할 때 검색한 SNMP 변수들과 그 값은 누적 시키지 않고 관리자에게 전송하여 주고 다음의 노드로 이동한다. locker pattern에 대한 전체 네트워크 로드는 식 (6), 전체 네트워크 소요시간은 식 (7)과 같다.

$$L_{LOC}=\sum_{n=0}^{N}(S_A+L_T+\sum_{r=1}^{R_n}S_{nr}L_{VB}) \tag{6}$$

$$T_{LOC}=2N\delta+\frac{L_{LOC}}{\beta} \tag{7}$$

n번째 노드에서 관리자에게 반환하여 주는 데이터의 양은 $\sum_{r=1}^{R_n}S_{nr}L_{VB}$과 같으며, 이 데이터를 전송하기 위해 붙는 TCP 헤더의 크기는 L_T이다. 식 (7)의 수식과 같이, 각각의 노드에 대해, 이동 에이전트의 이동과 수집된 결과의 전송을 위해 2번의 네트워크 지연시간이 소요된다.

3.2 실 험

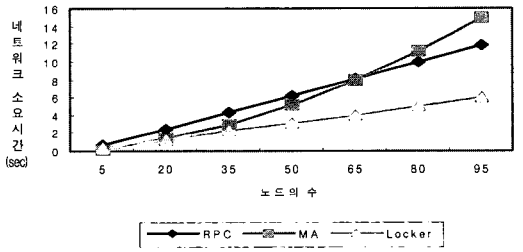
본 절에서는 실제 제안한 모델에 샘플 데이터를 적용하여 실험한 결과를 제시한다. 실험에 사용한 파라미터는 <표

2>와 같다. <표 2>에서 각 노드로의 request 횟수인 R_i 는 각각의 노드에 대해 동일한 평균 수치를 갖는 것으로 가정하였고, n번 노드로 r번째 request 할 때 요구되는 SNMP 변수의 수 S_{nr} 도 각각의 노드와 각각의 request에 대해 동일한 평균 수치를 갖는 것으로 가정하였다. 네트워크 지연시간 는 TCP를 사용하는 이동 에이전트와 locker pattern의 경우 20ms로, UDP 방식을 사용하는 RPC 방식의 경우엔 12ms로 하였다. UDP의 경우는 TCP 방식에 비해 메시지 전송의 신뢰성(reliability)은 떨어지는 반면, 네트워크 지연시간은 적게 소요된다[12]. locker pattern에서 반환하는 데이터는 TCP방식으로 전송되므로 메시지에 IP, TCP 헤더가 붙는다. 반면 SNMP 메시지의 경우는 IP, UDP, SNMP헤더가 붙기 때문에 메시지가 TCP방식보다 약 20 bytes 정도 더 크게 된다[1, 12].

〈표 2〉 실험을 위해 가정한 파라미터 값

파라미터	값
N	20개
R_i	5회
S_{nr}	15개
L_V	5bytes
L_S	60bytes
L_T	40bytes
L_{VB}	10bytes
σ	20ms(TCP)
	12ms(UDP)
β	330KB/sec
S_A	6KB

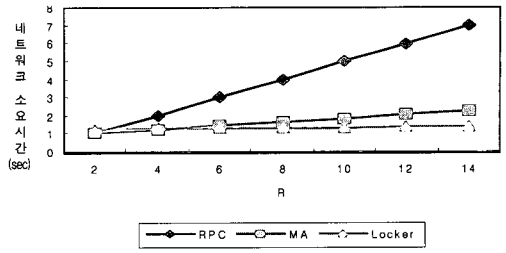
(그림 4)에서는 관리되는 노드 수에 대한 네트워크 소요시간을 비교하였다.



(그림 4) 관리되는 노드 수에 대한 네트워크 소요시간 비교

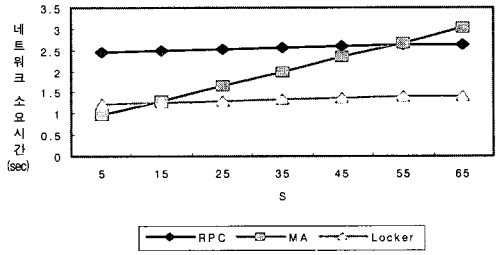
(그림 4)의 결과에서 볼 수 있듯이 locker pattern의 경우 RPC나 이동 에이전트 방식에 비해 네트워크 소요시간이 적음을 볼 수 있다. 이동 에이전트의 경우에는 노드의 수가 약 65개 이상이 되면 RPC보다 더 많은 네트워크 시간이 소요됨을 볼 수 있다. 이동 에이전트의 경우 노드의 수가 증가함에 따라 누적되어 네트워크 상에서 전송하는 데이터 양이 급격히 증가하기 때문에 이러한 결과를 보인다. (그림 5)는 네트워크 관리자가 각 노드로 SNMP 데이터를 요구하는 횟수에 대한 네트워크 소요시간을 비교하였다.

(그림 5)의 결과를 보면, 요구 횟수가 많을수록 locker



(그림 5) 각 노드로 요구하는 횟수에 대한 네트워크 소요시간 비교

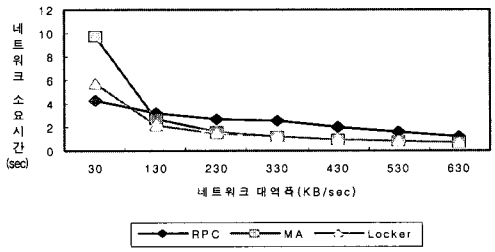
pattern과 이동 에이전트의 소요시간이 RPC 방식에 비해 더 적어짐을 확인할 수 있다. RPC의 경우 각 노드로의 요구횟수가 늘어날수록 네트워크 지연시간 발생 횟수가 locker pattern이나 이동 에이전트보다 많아지기 때문이다. (그림 6)에서는 각 노드에 한번의 request 메시지를 보낼 때 요구되는 SNMP 변수의 수에 대한 네트워크 소요시간을 비교하였다. 파라미터는 이전과 동일하다.



(그림 6) 각 노드로 한번 요구 시 필요로 하는 SNMP 변수의 수에 대한 네트워크 소요시간 비교

(그림 6)의 결과를 보면, 이동 에이전트의 경우가 S가 14 이내의 경우 가장 적은 소요시간을 나타냈으나, S가 54 이상의 경우에는 가장 많은 네트워크 시간이 소요됨을 볼 수 있다.

(그림 7)에서는 네트워크 대역폭의 변화에 따른 네트워크 소요시간을 비교하였다. 네트워크 대역폭이 630KB/sec인 경우 TCP를 이용하는 이동 에이전트와 locker pattern의 네트워크 지연시간은 11ms로 하였으며, 대역폭이 100KB/sec 증가할 때마다 3ms정도 네트워크 지연시간이 증가된다고 가정하였다. 또한 UDP를 이용하는 RPC방식은 TCP 방식의 60% 정도의 네트워크 지연시간을 갖도록 하여 실험을 하였다.



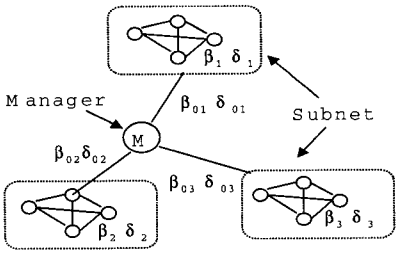
(그림 7) 네트워크 대역폭에 대한 네트워크 소요시간 비교

(그림 7)의 결과를 보면, 전반적으로 locker pattern의 경우 가장 적은 소요시간을 나타내며, 이동 에이전트의 경우 약 430KB/s 이상의 대역폭의 네트워크 환경에서 좋은 성능을 보이고 있다. RPC의 경우엔 대역폭이 약 90KB/sec 이하에서 가장 적은 네트워크 소요시간을 갖는다. 이동 에이전트와 locker pattern의 경우 네트워크 대역폭이 30KB/sec에서 130KB/sec으로 증가할 때 네트워크 소요시간이 급격히 감소한 이유는 이 두 가지 패러다임이 RPC에 비해 누적되는 데이터의 양이 상대적으로 많기 때문이다. 따라서 네트워크 대역폭에 매우 민감한 반응을 보이는 것이다. 그러나 (그림 7)의 결과에 의하면, 네트워크 대역폭이 130KB/sec 이상이 되면 누적된 데이터의 전송시간보다 네트워크 지연시간이 전체 네트워크 소요시간에 더 많은 영향을 준다는 점을 확인할 수 있다. 그 이유는 네트워크 관리의 경우 정보검색이나 데이터 마이닝과는 달리 수집되는 데이터의 크기와 양이 상대적으로 적기 때문이다. 본 논문에서 제안한 모델에 여러 가지 다양한 파라미터 값을 적용해 봄으로 네트워크 소요시간을 예측해 볼 수 있을 것이다.

4. 비균등 네트워크 환경을 고려한 성능평가 모델

4.1 성능평가 모델

본 절에서는 비균등 네트워크 환경에서의 성능평가 모델을 제시한다. 비균등 네트워크란 각 노드간의 네트워크 대역폭이 동일하지 않은 네트워크 환경을 의미한다. (그림 8)은 각기 다른 네트워크 대역폭을 갖는 세 개의 서브네트워크로 구성된 비균등 네트워크의 예를 보인다. 각 서브네트워크의 네트워크 대역폭은 $\beta_1, \beta_2, \beta_3$이며, 네트워크 지연시간 $\delta_1, \delta_2, \delta_3$이다. 네트워크 관리자와 대역폭이 β_1인 서브네트워크 간의 대역폭은 β_{01}이다. 실제 네트워크 관리자가 대역폭이 β_1인 서브네트워크에 존재한다면 $\beta_{01} = \beta_1$일 것이다. 비균등 네트워크 환경을 고려한 추가의 파라미터는 <표 3>과 같다.



(그림 8) 비균등 네트워크 환경

〈표 3〉 추가되는 파라미터

파라미터	설 명
N_N	각각의 서브 네트워크에 존재하는 노드의 수
N_S	서브네트워크의 수

각각의 서브네트워크에 존재하는 노드의 수는 실제로는 다양하겠지만, 수식의 단순화를 위해 동일하다고 가정하였다. 비균등 네트워크 상에서 RPC에 대한 전체 네트워크 소요시간은 식 (8)과 같다.

$$T_{RPC} = \sum_{j=1}^{N_S}(((\sum_{n=1}^{N_N}\sum_{r=1}^{R_n}(2L_S + (L_V + L_{VB})S_{nr}) \\ (1/(\min(\beta_{0j}, \beta_j)) + 2\sum_{n=1}^{N_N}\sum_{r=1}^{R_n}\max(\delta_{0j}, \delta_j)) \quad (8)$$

min 함수는 파라미터 중 최소값을 반환하는 함수이다. 실제 비 균등 네트워크의 경우 각각의 서브 네트워크에 속하는 노드들간의 네트워크 대역폭은 각 서브 네트워크의 대역폭 중 최소의 값을 갖기 때문이다. max 함수는 파라미터 중 최대값을 반환하는 함수이다. 식 (8)에서 $L_S + L_{VB}$ S_{nr}은 n번째 노드에서 r번째 요구에 대해 관리자에게 반환하여 주는 데이터의 양이다.

식 (9)는 이동 에이전트에 대한 전체 네트워크 소요시간이다. 식 (9)에서 $\sum_{i=1}^{j*N+n}\sum_{r=1}^{R_n}(S_{nr}L_{VB})$은 j번 서브 네트워크의 n번째 노드 방문후에 누적된 데이터 양이다.

$$T_{MA} = \sum_{j=0}^{N_S-1}(\sum_{n=1}^{N_N}(S_A + \sum_{i=1}^{j*N_N+n}\sum_{r=1}^{R_n}(S_{nr}L_{VB})) \\ (1/\min(\beta_{0j}, \beta_j)) + \max(\delta_{0j}, \delta_j)) \quad (9)$$

식 (10)은 locker pattern에 대한 전체 네트워크 소요시간이다. locker pattern의 경우, 하나의 서브 네트워크에 대한 네트워크 지연시간은 $2N_N\max(\delta_{0j}, \delta_j)$이며, 총 네트워크 지연시간은 $N_S \times 2N_N\max(\delta_{0j}, \delta_j)$이다.

$$T_{MA(L)} = \sum_{j=1}^{N_S}(\sum_{n=1}^{N_N}(S_A + L_T + \sum_{r=1}^{R_n}S_{nr} * L_{VB}) \\ (1/\min(\beta_{0j}, \beta_j)) + 2\max(\delta_{0j}, \delta_j))) \quad (10)$$

4.2 성능평가 모델의 응용

가상의 시나리오 하에서 본 논문에서 제안한 모델을 기초로 네트워크 관리 작업에 최소의 네트워크 시간이 소요되는 패러다임과 수행패턴을 선택할 수 있다. 예를 들어 3개의 서브 네트워크로 구성된 <표 4>와 같은 환경에서 네트워크 관리를 수행한다고 하자. 기타의 파라미터 값들은 3.2절의 실험에 사용한 값과 같으며, 네트워크 관리자(NMS)는 서브 네트워크 1에 있다고 가정한다. <표 4>의 네트워크 지연시간은 TCP에 대한 것이며, UDP의 경우는 각각 4, 16, 12ms이다.

〈표 4〉 가상 시나리오 환경

서브 네트워크	네트워크 대역폭	네트워크 지연시간	관리장치
1	1024 KB/sec	6ms	10개
2	56 KB/sec	27ms	10개
3	300 KB/sec	20ms	10개

이번 실험에선 이동 에이전트의 경우 네트워크 소요시간을 줄이기 위해 네트워크 대역폭이 적은 서브 네트워크부터 방문하도록 하였다. 성능평가 모델을 기초로 계산된 전체 네트워크 소요시간은 RPC만을 이용한 경우 3.492sec, 이동 에이전트만을 이용한 경우 3.173sec, locker pattern만을 이용한 경우 2.595sec이었다. 따라서 가정한 시나리오에서는 locker pattern을 이용하여 네트워크 관리를 하는 것이 가장 적은 네트워크 소요시간을 갖는다는 점을 알 수 있다.

네트워크 관리를 위해 3가지 패러다임을 혼합하여 수행하는 것도 가능하다. 혼합된 패러다임의 수행 패턴을 선택하기 위한 알고리즘은 다음과 같다.

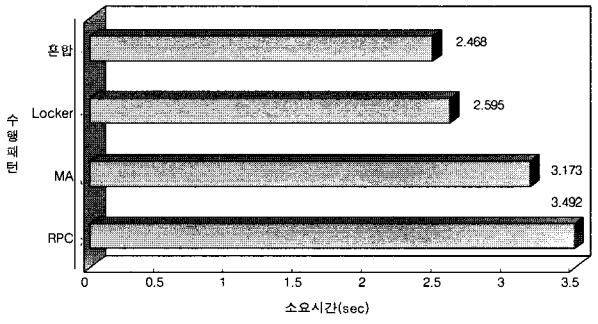
1. For each sub network, calculate network execution times of three paradigms from model for non-uniform network
2. For each sub network, choose one paradigm which has minimum network execution time

제안한 알고리즘에 의해 결정된 수행 패턴 <표 5>와 같다.

〈표 5〉 알고리즘에 의해 결정된 수행패턴

Sub net. 1	Sub net. 2	Sub net. 3
이동 에이전트	Locker pattern	이동 에이전트

(그림 9)에서는 RPC, 이동 에이전트, locker pattern만 이용한 경우와 알고리즘에 의해 결정된 <표 5>와 같은 혼합 패러다임을 이용한 경우에 대한 총 네트워크 소요시간을 비교하였다. 혼합 패러다임이 최소의 네트워크 시간이 소요됨을 확인할 수 있다.



(그림 9) 수행 패턴에 대한 네트워크 소요시간 비교

5. 결론 및 향후연구과제

본 논문에서는 네트워크 관리 시스템에서 SNMP프로토콜을 이용한 중앙 집중형 방식과 이동 에이전트를 이용한 분산방식 각각의 성능을 평가하기 위한 수학적인 모델을 제시하고, 실제 실험을 통하여 각 파라미터 값의 변화에 따른 네트워크 수행 시간을 비교하였다. 균등 네트워크 환경과 비균등 네트워크 환경 각각에 대해 성능평가 모델링을 수행하였으며, RPC와 이동 에이전트의 단점을 보완한 패러다임으로 locker pattern에 대한 모델도 함께 제시하였다. 또한 실제 가상의 시나리오 하에서 네트워크 소요시간을 줄이기 위해 적합한 분산 패러다임과 수행 구조를 선택하기 위한 방안과 그에 대한 실험 결과도 제시하였다. 네트워크 관리시스템을 개발할 때에, 시스템 개발 이전 단계에서 구현하고자 하는 시스템의 환경 정보를 본 모델의 적용하여 평가해 본다면 최소의 네트워크 소요시간을 갖는 분산 패러다임과 수행 구조를 선택할 수 있을 것이다.

향후연구 과제로서 분산 패러다임의 성능 평가를 위해 네트워크 소요시간뿐 아니라 노드의 처리부하, 보안 등의 요소를 함께 고려하는 작업이 필요하며, 모델에 사용된 파라미터 정보를 어떻게 계산할 것인지도 고려해 보아야 할 사항이다. 또한 본 논문에서 제안한 이론적인 모델이 얼마나 실세계와 일치한지를 실제 분산 애플리케이션을 개발하여서 검증하는 작업도 필요하다.

참 고 문 헌

[1] W. Stallings, SNMP, SNMPv2 and RMON, Addison Wesley, 1996.

[2] A. Bieszczad, T. White and B. Pagurek, "Mobile Agents for Network Management," IEEE Communications Surveys, Sep. 1998.

[3] G. P. Picco, M. Baldi, "Evaluating tradeoffs of Mobile Code Design Paradigms in Network Management Applications," Proc. of the 20th International Conference on Software Engineering(ICSE98), Japan, 1998.

[4] A. Sahai and C. Morin, "Mobile Agents Enhanced Thin Client Approach to Network Management," Proc. of the IEEE SI CON98, Singapore, Jul. 1998.

[5] A. Sahai and C. Morin, "Towards Distributed and Dynamic Network Management," Proc. of the IEEE/IFIP Network Operation and Management Symposium(NOMS), New Orleans, Lousiana, USA, Feb. 1998.

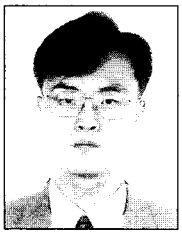
[6] G. Susilo, A. Biezczad and B. Pagurek, "Infrastructure for Advanced Network Management Based on Mobile Code," Proc. of the IEEE / IFIP Network Operation and Management Symposium (NOMS), New Orleans, Lousiana, USA, Feb. 1998.

[7] T. White, B. Pagurek and A. Biezcza, "Network Modeling for Management Applications Using Intelligent Mobile Agents," Journal of Network and Systems Management, Sep. 1999.

[8] D. L .Tennenhouse, M. S. Jonathan, W. D. Sincoskie, D. J. Wetherall and G. J. Minden, "A survey of Active Network Research," IEEE Comunications, pp.80-86, Jan. 1997.

[9] H. Ku, el al. "An Intelligent Mobile Agent Framework for Distributed Network Management," Globecom'97 Phoenix, Nov. 1997.

[10] M. G. Rubinstein, O. C. M. B. Duarte, G. Pujolle, "Improving Management Performance by Using Multiple Mobile Agents," Proc. 4th International Conference on Autonomous Agents, pp.165-166, 2000.
[11] D. B. Lange, M. Oshima, Programming and Deploying Java Mobile Agents with Aglets, Addison-Wesley, 1998.
[12] W. Stallings, Data and Computer Communications(4th edition), Macmillan Publishing Company, 1994.
[13] C. G. Harrison, D. M. Chess, A. Kershenbaum, Mobile Agents : Are they a good idea?, IBM Watson Research Center, Mar. 1995.
[14] RFC, "Management Information Base for Network Management of TCP/IP-based Internets : MIB-II," Mar. 1991.
[15] 권혁찬, 유관종, "이동 코드를 이용한 네트워크 관리 모델링", 한국정보과학회추계 학술발표회, 제26권 제2호(C), pp.676-678, 1999.
[16] H. C. Kwon, S. Sohn, W. J. Yoo, S. I. Lee, T. G. Kang, K. J. Yoo, "A Selection Algorithm for an Efficient Interaction Pattern out of Paradigms," Proc. of the Workshop on Agents in Industry at the 4th International Conference on Autonomous Agents 2000, Barcelona, Spain, Jun. 3, 2000.



권 혁 찬

e-mail : hckwon@etri.re.kr
1994년 서원대학교 전자계산학과 졸업 (공학사)
1996년 충남대학교 전산학과(이학석사)
2001년 충남대학교 컴퓨터과학과(이학박사)
2001년~현재 한국전자통신연구원 인터넷 보안연구팀 선임연구원
관심분야 : 에이전트 시스템, 네트워크 보안, 병렬처리, IPsec

김 흥 환

e-mail : khh@seowon.ac.kr
1985년 서울대학교 계산통계학과 졸업 (이학사)
1987년 서울대학교 계산통계학과(이학석사)
1990년 서울대학교 계산통계학과(이학박사)
1990년~현재 서원대학교 컴퓨터 정보통신학부 부교수
1996년~1997년 콜로라도 주립대학 전산과학과 객원교수
1997년 한국전자통신연구원 초빙 연구원
1998년~1999년 건국대학교 전산과학과 부교수
관심분야 : 프로그래밍 언어, 컴파일러, 병렬 및 분산처리 시스템, 웹컴퓨팅, 원격교육

유 관 종

e-mail : kjyoo@cs.cnu.ac.kr
1976년 서울대학교 계산통계학과 졸업 (이학사)
1978년 서울대학교 대학원 전산학 전공 (이학석사)
1985년 서울대학교 대학원 전산학 전공 (이학박사)
1987년 충남대학교 전산학과 학과장
1987년~1989년 충남대학교 전자계산소 소장
1990년 캘리포니아 대학(Irvine) 방문교수
1979년~현재 충남대학교 공과대학 정보통신공학부 교수
1995년~현재 한국정보과학회 이사
관심분야 : Agent, Scalable Coding, 멀티미디어 응용, VOD, 병렬처리, 컴파일러 설계