모바일 미들웨어를 위한 시뮬레이션 툴의 설계와 구현

류준희° 박은정 신현식
서울대학교 컴퓨터공학과
{jhryu°, ejpark, shinhs}@cslab.snu.ac.kr

Design and Implementation of a Simulation Tool for Mobile Middleware

Junhee Ryu° Eunjeong Park Heonshik Shin
School of Computer Science and Engineering, Seoul National University

요 약

모바일 디바이스의 성능이 향상되고 무선 네트워크에 기반한 유비쿼터스 컴퓨팅이 새로운 패러다임으로 각광을 받고 있다. 하지만, 모바일 디바이스는 여전히 프로세서의 처리 능력, 메모리 크기, 그리고 네트워크 대역폭 면에서 제약이 많기 때문에 이를 극복하기 위한 미들웨어의 개발이 필요하다. 본 논문에서는 미들웨어의 효과적인 개발을 위한 시뮬레이션 툴을 설계하고 ns2(network simulator 2) 상에서 구현하였다. 이 시뮬레이션 툴은 유비쿼터스 환경에서 디바이스의 자원 관리, 프로세서와 네트워크 상에서의 소요 전력 측정, 네트워크의 가용 대역폭 측정, 상황 인식(context-aware) 및 코드 마이그레이션 기능을 제공하며 이를 통하여 다양한 모바일 미들웨어를 시뮬레이션 할 수 있다.

1. 서 론

최근 유비쿼터스 컴퓨팅에 대한 관심이 고조됨에 따라 무선 네트워크와 모바일 디바이스에 대한 연구가 활발히 진행되고 있다. 이러한 연구는 주로 무선 매체와 사용자의 이동에 따른 불안정한 통신 상태와 모바일 디바이스의 자원 부족 문제를 해결하는데 중점을 두고 있으며, 모바일 에이전트 시스템은 이러한 문제에 대한 해결책의 하나로 제시되고 있다[1].

특히 상황을 인식하는 모바일 에이전트는 주변 호스트의 자원 및 네트워크 상황을 모니터링 하여 부하분산, 고장감내, 응답시간 향상 등의 다양한 효과를 위하여 응용될 수 있다[1].

지금까지 모바일 에이전트 시스템 제작에 대한 연구가 활발하였지만 모바일 에이전트 시스템의 시뮬레이션에 대한 연구는 부족하다. 본 논문에서는 무선 환경의 모바일 에이전트 시스템을 시뮬레이션 할 수 있는 툴을 설계하고 ns2[3] 상에서 구현함으로써 네트워크 토폴로지 및 스케쥴링 기법을 자유롭게 재구성할 수 있도록 하였다. 또한 프로세서와 메모리, 배터리 등의 가용 시스템 자원과 네트워크 대역폭을 인식하여 각종 스케쥴링 및 시스템 성능 분석에 사용할 수 있도록 하였다.

2. 관련 연구

모바일 에이전트 시뮬레이터에 관련된 연구로는 Lamar University의 Kunal shah가 만든 "Mobile agent extension to NS"가 있다[5]. 이 시뮬레이터는 MOLE 모바일 에이전트 시스템[10]을 모델로 하고 Aglets의 인터페이스[4]를 기반으로 작성되었으며 클라이언트/서버 패러다임과 모바일 에이전트 패러다임의 비교를 목적으로 제작되었다. 하지만 하나의 디바이스 내에 하나의 모바일 에이전트의 동작만을 지원하고 디바이스의 상태나 네트워크의 정보는 고려하고 있지 않다. 따라서 상황 인식을 통한 코드 마이그레이션이나 부하분산 등의 시뮬레이션에 사용될 수 없다는 단점이 있다.

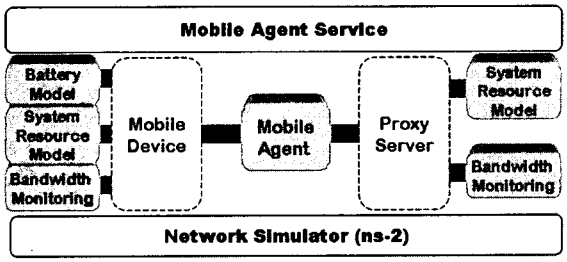
본 논문에서는 각 디바이스 내에서 여러 모바일 에이전트를 동작시킬 수 있고 상황 인식, 시스템 시뮬레이션 및 소비 전력 측정이 가능한 시뮬레이션 툴을 설계 및 구현하였다.

3. 시뮬레이션 툴의 설계 및 구현

3.1. 시뮬레이션 툴의 구조

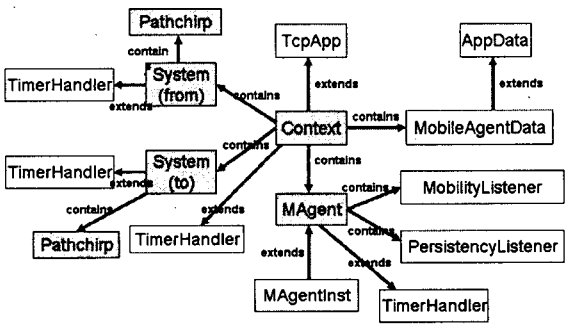
본 논문의 시뮬레이션 툴은 네트워크 부분의 신뢰도를 높이기 위하여 ns2를 기반으로 제작하였다. 시뮬레이터의 구성 요소는 [그림 1]에 나타나 있다. 전체 구조는

크게 1)네트워크 토폴로지 구축을 위한 기본적인 네트워크 계층, 시스템 자원 및 모바일 에이전트의 동작을 시뮬레이션 하는 미들웨어 계층, 그리고 3)마이그레이션 스킴과 같이 미들웨어의 동작을 정의하는 모바일 에이전트 서비스 계층으로 구성된다.



[그림 1] 시뮬레이션 툴의 구조

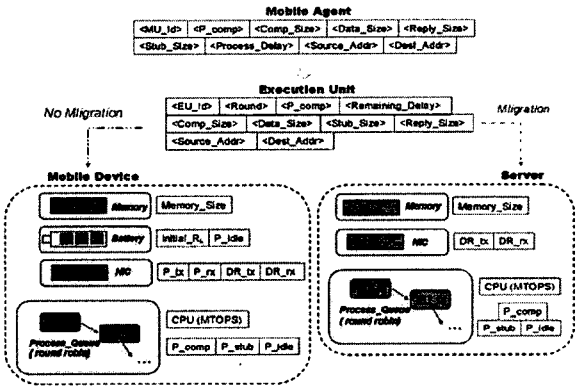
미들웨어 계층은 시스템 자원 관리, 소비 전력 측정, 네트워크 대역폭 측정, 그리고 모바일 에이전트 부분으로 세분화 된다. [그림 2]는 MAgent 중심의 클래스 상속도를 보여준다. 자원관리 및 소비 전력 부분은 System 클래스에, 네트워크 대역폭 측정은 Pathchirp 클래스에, 코드 마이그레이션 관련 부분은 Context 클래스에 구현하였다. Context는 모바일 에이전트가 이동하기 위한 시스템 사이의 TCP 연결 및 에이전트의 이동과 관련된 메소드들을 제공한다. 각 System 클래스는 시스템의 자원을 관리하고 이를 상황 인식에 사용할 수 있도록 하는 메소드들을 정의 하였고 네트워크 대역폭 측정을 위해 각 System 클래스에 Pathchirp 클래스를 포함시켜서 시스템 간의 가용 네트워크 대역폭 측정을 가능하도록 하였다. 코드 마이그레이션에 소모되는 시간 요소들은 3.3장의 타이머 모델링에 따른다.



[그림 2] MAgent 클래스 중심의 클래스 상속도

3.2. 시뮬레이션 툴의 파라미터 및 동작

시뮬레이션 공간에서 MAgent의 처리 흐름은 [그림 3]과 같다. 모바일 에이전트의 속성을 나타내는 파라미터로는 코드 크기(Comp_Size), 데이터 크기(Data_Size), 응답 패킷의 크기(Reply_Size), 실행 스택의 크기(Stub_Size), 에이전트를 수행시키는데 걸리는 시간(Process_Delay) 등이 있다. 한 시스템 내에 여러 개의 모바일 에이전트가 있을 경우 각 모바일 에이전트는 라운드로빈 방식으로 처리된다. 시스템 정보와 관련된 파라미터로는 프로세서의 성능을 나타내는 MTOPS와 메모리 사이즈(Memory_Size)가 있고 프로세서의 상태에 따라 소비되는 전력 파라미터(P_comp, P_stub, P_idle)에 따라 각 타임슬라이스에서 디바이스가 소비한 전력과 네트워크의 전력 관련 파라미터(P_tx, P_rx)를 통하여 송수신할 때의 전력 소비를 측정하도록 구현 하였다. 이러한 소비 전력은 배터리 관련 파라미터(Initial_R_k, P_Idle)와 함께 현재 디바이스의 가용 전력값을 갱신한다. 시스템 내의 모든 모바일 에이전트가 요구하는 메모리 크기보다 물리 메모리가 작을 경우 페이지 폴트로 인한 성능 저하[8]를 반영하고, Pathchirp[9] 모듈을 이용하여 시스템 간의 네트워크 대역폭을 측정할 수 있다.



[그림 3] 시뮬레이션 파라미터

3.3. 타이머 모델링

동기화를 위해 MAgent, Context, System 클래스에서 모두 TimerHandler 클래스를 상속받았다. 그리고 모바일 에이전트의 파라미터 중 프로세서 처리를 요구하는 시간 값은 모두 프로세서의 CTP(composite theoretical performance) 값이 5333에서 소모되는 시간 값을 나타낸다. 이는 모바일 에이전트가 어느 프로세서에서 수행될 지 알 수 없으므로 기준값이 필요하기 때문이다. 이 표준 CTP와 모바일 에이전트가 수행되는 시스템의 CTP 값의 비율에 따라 실제 처리 시간이 결정된다. 본 시뮬레이터에 사용된 시간과 관련된 수식은 MOLE 모바일 에이전트 시스템을 따른다[5].

코드 이동시 필요한 데이터의 크기 B_{MIG}는 코드와 데이터, 실행 스택의 합인 수식으로 표현할 수 있다.

$$B_{MIG} = B_{CODE} + B_{DATA} + B_{STATE} \quad (1)$$

원격지 서버가 모바일 에이전트를 실행 중 또는 실행 후에 중간 결과 또는 결과 데이터를 클라이언트로 보내는데, 이 크기를 B_{REP}라고 하고 원격지 서버에서 실행될 때 증감되는 결과 패킷의 비율을 selectivity_로 가정하면 원격지 서버로 이동할 때의 총 데이터 크기 B_{MR}을 다음의 수식으로 표현할 수 있다.

$$B_{MR} = B_{CODE} + B_{DATA} + B_{STATE} + (1\text{-selectivity\_}) * B_{REP} \quad (2)$$

(1), (2)번 수식을 바탕으로 클라이언트에서 원격지 서버로 이동하는 시간 T_{MIG}을 다음의 수식으로 표현할 수 있다.

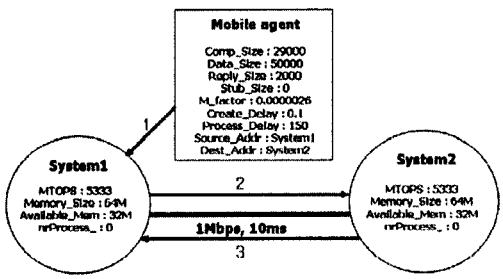
$$T_{MIG} = \delta + [(1/\gamma) + 2*M_{FACTOR}] * (B_{CODE} + B_{DATA} + B_{STATE}) \quad (3)$$

여기서, δ는 네트워크의 지연시간을, γ는 네트워크의 대역폭을 나타낸다. 그리고 M_{FACTOR}는 1바이트를 마샬링(marshalling) 또는 언마샬링(unmarshalling)하는데 소비되는 시간을 나타낸다. 여기에 실제로 프로세스가 수행되는 시간 $T_{PROCESS}$를 고려하면 클라이언트에서 서버로 모바일 에이전트를 이동시켜서 실행한 후 다시 클라이언트로 돌아와서 종료하는데 소비되는 총 응답 시간 T_{MR}은 다음의 수식으로 표현할 수 있다.

$$T_{MR} = T_{MIG} + [(1/\gamma) + 2*M_{FACTOR}] * B_{MR} + T_{PROCESS} \quad (4)$$

3.4. 시뮬레이터의 정확도 검증 실험

시뮬레이터의 정확도를 검증하기 위하여 파라미터가 [그림 4]와 같은 모바일 에이전트를 System1에서 System2로 이동시켜서 실행시키고 System1으로 돌아와서 종료하는 시나리오로 실험을 하였다.



[그림 4] 정확도 검증 시나리오

Process_Delay 파라미터가 시뮬레이션 시간의 대부분을 차지하기 때문에 Process_Delay를 고려한 것과 고려하지 않았을 경우의 오차도 비교하였다. 타이머 모델링에 따른 수학적 모델과 시뮬레이션 툴을 통한 실험 결과의 비교는 [표 1]과 같으며 오차는 0.1% 미만이다.

	Process_Delay를 고려한 비교	Process_Delay를 제외한 비교
수학적 모델(초)	152.352	2.352
시뮬레이션 결과(초)	152.354	2.354
오차(%)	0.0013	0.0849

[표 1] 정확도 검증 실험 결과

4. 결론 및 향후 과제

본 논문에서는 여러 모바일 에이전트의 동시 수행과 시스템 시뮬레이션, 네트워크 가용 대역폭 측정과 프로세서, 네트워크에서의 소모 전력 측정을 지원하여 모바일 에이전트의 마이그레이션과 관련된 알고리즘을 평가할 수 있는 시뮬레이터의 구조를 설계 및 구현하였다.

앞에서 언급하였듯이 본 논문에서 제작한 시뮬레이션 툴은 크게 1)시스템 자원 시뮬레이션 2)네트워크 대역폭 측정 3)상황 인식 4)코드 마이그레이션 5)NS2의 다른 모듈들과의 인터페이스 부분으로 구현하였다.

향후 과제로는 NS2의 시뮬레이션 커널과 SystemC의 시뮬레이션 커널의 통합을 통하여 네트워크 시뮬레이션과 디바이스 시뮬레이션의 동기화를 시키는 것을 계획중에 있다.

참고 문헌

[1] Cecilia Mascolo, Licia Capra and Wolfgang Emmerich, "Mobile computing Middleware," *Advanced lectures on networking*, pp.20-58, 2002.

[2] Mahmoud, "MobiAgent: A mobile agent-based approach to wireless information systems," *Proceedings of the International Bi-Conference Workshop on Agent Oriented Information Systems*, 2001.

[3] The Network Simulator ns-2 Documentation, http://www.isi.edu/nsnam/ns/ns-documentation.html.

[4] Aglets: Mobile Java Agents, http://www.trl.ibm.com/aglets.

[5] Performance Analysis of Mobile Agents in Wireless Internet Applications Using Simulation, http://www.geocities.com/kunal_u_shah.

[6] Jain, Ravi, Farooq Anjum, and Amjad Umar, "A comparaison of mobile agent and Client-Server paradigms for information retrieval tasks in virutal enterprises," *AiWoRC Workshop*, 2000

[7] NS by Example, http://nile.wpi.edu/NS.

[8] A Page Fault Equation for Modeling the Effect of Memory Size, http://www.math.nus.edu.sg/~mattyc/pagefault.pdf.

[9] Vinary Ribeiro, Rudolf Reidi, Richard Baraniuk, Jiri Navratil, Les Cottrell, "pathChirp: Efficient Available Bandwidth Estimation for Network Paths," *Passive and Active Measurement Workshop (PAM)*, 2003

[10] J.Baumann, F.Hohl, K. Rothermel and M.Strasser, "Mole - Concepts of a Mobile Agent System," *World Wide Web, Vol 1, Nr 3*, pp.123-137, 1998