한국컴퓨터정보학회 하계학술대회 논문집 제18권 제2호 (2010. 7)

환경 변화에 동적으로 적응하는 멀티에이전트 시스템을 위한 우선순위 기반 메시지 스케줄링 기법

강병길°, 이강렬*, 유한구*, 윤희용*,
°*성균관대학교 전자전기컴퓨터공학과
e-mail : {bk0820, Intensity}@skku.edu, gksrnek@gmail.com, youn@icc.skku.ac.kr

A Priority-based Message Scheduling Scheme for Multi-agent System Dynamically Adapting to the Environment Change

Byung Kil Kang°, Kang Lyul Lee*, Han Ku Yoo*, Hee Yong Youn*
°*Dept. of Computer Engineering, Sungkyunkwan University.

● 요 약 ●

멀티 에이전트 시스템은 공통의 목표를 위해 함께 일하기로 합의한 협업 에이전트로 구성된다. 에이전트 시스템에서 메시지를 스케줄링 할 때, 동일한 우선순위가 역할의 중요도를 고려하지 않고 모든 에이전트에게 동일하게 할당된다면 성능은 극대화되지 않을 것이다. 본 논문에서 우리는 새로운 메시지 기반 우선순위 스케줄링 기법을 제안한다. 그것은 에이전트 시스템으로 하여금 에이전트와 메시지 작업 처리량의 중요도에 기반으로 메시지의 우선순위를 적절하게 모델링 함으로써 환경 변화에 신속히 적응할 수 있게 한다. 에이전트 시스템의 성능은 저자들에 의해 개발 된 에이전트 플랫폼을 통해 실행된 범죄 예방 시스템에 대한 사례 연구를 이용한 제안된 계획에 의해 상당히 개선된 것으로 보여준다.

키워드: Agent community, importance of agent, multi-agent system, ubiquitous system

I. 서론

다양한 어플리케이션을 지원하는 유비쿼터스 컴퓨팅 시스템은 본 시스템 전체에 걸쳐 분산된 에이전트를 통해 효과적으로 실행될 수 있으며, 이는 사용[1]자를 위해 자발적으로 작업한다. 에이전트는 커뮤니티를 조직하며 개개인의 사용자의 문맥에 맞게 만들어지고 정보 처리 기능을 가진 서비스를 제공하기 위해 서로 협업한다.

멀티-에이전트 시스템에는 다양한 커뮤니티 조직 모델이 존재한다[1-2]. 에이전트 시스템에 의해 차용된 커뮤니티 조직 모델은 시스템의 성능 상에 중대하고 정량적으로 예측 가능한 영향을 줄 수 있다. 이로 인해 최근에는 많은 연구들이 커뮤니티 조직의 방법론에 초점을 맞춘다. 하지만, 그 연구들은 에이전트의 역할을 아직 적절히 고려하고 있지 못하다. 사용자들을 위한 모든 컨텍스트와 서비스가 항상 높은 중요도를 필요로 하는 것은 아니기 때문이다. 최근의 연구들은 컨텍스트와 주위 환경에 관련된 에이전트의 중요도와 필요성 대신 커뮤니티 조직의 방법에 초점을 맞춘다. 특히, 일부 연구들은 에이전트 사이에서 교환되는 FIPA[3] ACL(Agent Communication Language) 메시지에 사용된 일련의 통신 행위 및 행동 라벨을 고려한다[4-9]. 하지만, 이러한 접근 방법은 모든 에이전트에게 적용하는데 있어 효율적이지 않다.

본 연구에서 우리는 본 시스템의 성능을 보다 더 극대화시키기 위해 새로운 우선순위 기반의 메시지 스케줄링 기법을 제안한다. 제안된 본 메시지 스케줄링 기법에서 스케줄링은 에이전트의 중요도에 의해 결정되며 그것은 정해지는 것이 아니라 에이전트 커뮤니티 속 에이전트 사이에서 전송된 메시지의 양을 주기적으로 감시함으로써 업데이트된다. 제안된 계획을 통해 본 에이전트 시스템은 에이전트 및 메시지 작업 처리량의 중요도를 기반으로 메시지의 우선순위를 적절하게 계획함으로써 사용자 선호도와 시스템 환경의 변화를 빨리 적응할 수 있다. 에이전트 시스템의 성능은 범죄 예방에 대한 사례 연구를 이용하는 제안된 계획 에 의해 상당히 개선된 것을 보여준다.

본 연구의 나머지 부분은 다음과 같이 구성된다. Section 2에서는 관련 연구에 대해 설명하고, Section 3은 본론으로 제안된 계획에 대해 보여준다. 제안된 접근법의 성능은 Section 4에서 평가되며, Section 5는 여러 소건을 통해 본 연구의 결론을 내린다.

II. 관련 연구

메시지들은 메시지의 길이, 그 시스템 내에 있었던 시간의 양, 그리고 메시지에 제공된 서비스의 양 등과 같은 다양한 조건에 근

본 연구는 서울시 산학연 협력사업(CR070019) 지원으로 수행되었습니다.

거하여 일정이 스케줄링 될 수 있다.

간단하고 직관적인 스케줄링기법이 FCFS 계획이다. 그러나, FCFS는 긴 메시지 뒤에 기다리는 짧은 메시지에 대하여 불공평함이라는 명백한 단점을 가진다.

라운드 로빈(round-robin) 기법을 FCFS의 단점을 극복하는데 짧은 메시지들이 완료되어야 하는 긴 메시지의 전송을 기다려야 하는 단점을 보완한다. 하지만, 그것은 시스템 내에 다수의 메시지가 있을 때 불필요하게 큰 지연이 발생하는 단점을 가진다.

더 나은 성능을 나타내는 것으로 알려진 SMP의 약간의 변화란 우선 가장 짧은 남은 처리(이 경우에는 전송) 시간에 대한 메시지를 수행하는 것이다. 본 계획을 통해 더 짧은 메시지들이 메시지가 수행되고 있을 때 도착한다면 완료까지 소요되는 남은 시간의 양이 더 짧은 메시지보다 더 작은 경우 더 긴 메시지는 그 서비스를 계속 받을 것이다.

다이나믹한 스케줄링 기법은 본 시스템 내에서 얼마나 시간이 걸리고 있으며 그것이 얼마나 많은 서비스를 받고 있는지를 바탕으로 메시지의 우선순위를 변경한다. "우선순위" 기능은 메시지에 우선순위 값을 할당하는데 사용된다. 알고리즘의 각각의 반복에 있어서 스케줄러는 가장 높은 우선순위 메시지의 하나의 cell를 수행하며, 모든 메시지의 우선순위를 다시 계산한다.

III. 본론

우리는 멀티-에이전트 시스템을 위한 제안된 메시지 스케줄링 기법을 소개한다. 메시지를 스케줄링 할 때 에이전트에게 특수한 컨텍스트에서 자신의 역할을 고려하지 않고 동일한 우선순위가 할당되는 경우 에이전트 시스템의 성능 가 극대화되지 않을 것이다. 중요한 역할을 하는 에이전트에 대해 정해진 메시지가 지연될 때 사용자에게 서비스를 공급하는 것 또한 지연될 것이다. 사용자들이 다른 서비스 도메인으로 이동할 때 그들의 모바일 장치들 내에 포함된 에이전트는 그들과 협업한다. 이 경우 에이전트 또한 메시지 전송 지연 문제를 가진다. 그러므로 대상 서비스에 대해 할당된 에이전트역할의 중요도에 따라 우선순위를 사용하는 에이전트 커뮤니티의 메시지를 스케줄링 할 필요가 있다.

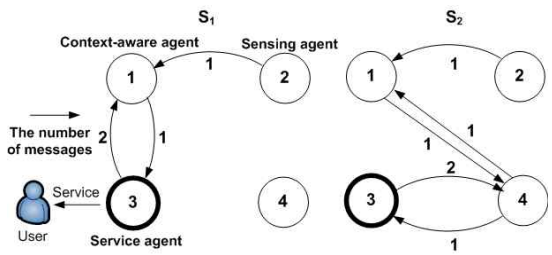

그림 1. 에이전트 커뮤니티의 예.

에이전트 시스템을 설계하는데 있어 사용자에게 최적의 서비스를 제공하기 위해 어떤 커뮤니티가 어떤 에이전트에 의해 조직되는지, 그리고 에이전트 사이에서 어떤 종류의 메시지가 교환되는지를 예측하는 것은 중요하다. 그림 1에서는 2개의 서비스, Service-1 (S1)와 Service-2 (S2)를 제공하는 4개의 에이전트로 구성되는 에이전트 커뮤니티에 대한 예를 보인다. 여기서 굵은 원의 에이전트-3이 서비스를 제공하는 에이전트이다.

에이전트-1, 에이전트-2 및 에이전트-3은 Service-1을 제공하기 위해 메시지를 교환할 필요가 있는 그림에서 관찰 할 수 있다. 이와 유사하게 Service-2는 4명의 에이전트 모두가 서로 협업하는 것을 필요로 한다. 각각의 링크와 연합된 수는 전송이 요구되는 메시지의 수를 나타낸다.

에이전트 커뮤니티는 다양한 종류의 서비스를 제공할 수 있다. Service-k(Sk)를 제공하기 위해 커뮤니티의 t 에이전트 사이에서 교환되는 메시지의 수가 다음과 같이 표시된다.

$$M_{S_k} = \begin{bmatrix} 0 & m_{1,2} & m_{1,3} & \dots & m_{1,t} \\ m_{2,1} & 0 & m_{2,3} & \dots & m_{2,t} \\ m_{3,1} & m_{3,2} & 0 & \dots & m_{3,t} \\ \dots & \dots & \dots & \dots & \dots \\ m_{t,1} & m_{t,2} & m_{t,3} & \dots & 0 \end{bmatrix}$$

여기서, mi,j는 에이전트 커뮤니티의 에이전트-i로부터 에이전트-j까지 전송되는 메시지 전송 수를 나타낸다. n개의 다른 서비스를 제공하는 에이전트 커뮤니티 C1 의 전체 메시지 전송 수는 다음과 같이 표시된다. 여기서 w는 각각의 서비스의 가중치를 나타낸다.

$$M_{C_1} = w_{S_1} M_{S_1} + w_{S_2} M_{S_2} + \dots + w_{S_n} M_{S_n}, \ \sum w_{S_i} = 1$$

에이전트 커뮤니티에서 각각의 에이전트는 다른 중요한 역할을 한다. 커뮤니티 에이전트의 중요도가 다음과 같이 표시된다.

$$I = \begin{bmatrix} 1 & 2 & \dots & i \end{bmatrix}, \ \sum i = 1$$

여기서 I는 에이전트-i의 중요도를 나타낸다. 에이전트 커뮤니티 에이전트의 중요도의 합은 1이 된다. 에이전트자신이 속하는 에이전트 커뮤니티에 따라 다른 중요도를 가진다. 에이전트 시스템을 설계할 때 이를 따라 시스템 설계자는 에이전트 커뮤니티를 고려하여 에이전트의 중요도를 결정한다. 우리는 높은 중요도의 에이전트에 의해 발생된 메시지를 똑같이 중요한 것으로 간주한다. 그러면 에이전트 커뮤니티 T의 메시지 작업 처리량은 다음과 같이 에이전트 중요도(I)에 대한 행렬과 메시지 주기(M)에 대한 행렬의 곱으로 정의된다.

$$T = I \times M$$

T의 각각의 요소는 각각의 에이전트의 상관 메시지 작업 처리량을 나타낸다. 제안된 계획에서 높은 중요도의 에이전트의 메시지에게 높은 우선순위가 주어진다. 동일한 중요도의 에이전트의 경우, 더 높은 메시지 작업 처리량 한 개에는 더 높은 우선순위가 주어진다. 이는 메시지 중요도가 사용자에게 서비스를 제공하는데 있어 작업 처리량보다 더 중대하기 때문이다. 그림 1의 커뮤니티

에이전트에 대해 할당된 다음의 중요도 값과 가중치를 가정한다.

$$I = \begin{bmatrix} 0.2 & 0.1 & 0.5 & 0.2 \end{bmatrix}$$

$$W = \begin{bmatrix} 0.3 & 0.7 \end{bmatrix}$$

그러면, 우리는 다음을 얻는다.

$$T = I \times \left(0.3 \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} + 0.7 \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 \\ 1 & 0 & 1 & 0 \end{bmatrix} \right)$$
$$= \begin{bmatrix} 0.54 & 0 & 0.2 & 0.84 \end{bmatrix}$$

그림 1의 에이전트 커뮤니티의 모든 에이전트가 메시지를 동시에 보낸다고 가정한다. 그림 2에서는 스케줄링 순서와 함께 6개의 전송을 보인다.

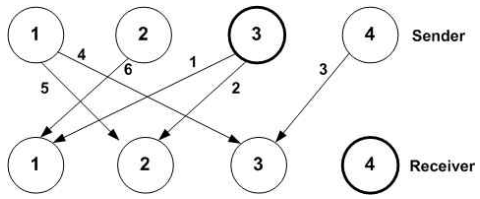

그림 2. 그림 1의 예를 위한 메시지 스케줄링.

여기서 I[3]은 0.5 중 가장 큰 값이므로 에이전트-3로부터의 2개의 전송이 우선 스케줄링 된다. 둘 중에서 에이전트-1에 대한 것은 에이전트-2의 0보다 더 높은 0.54의 작업 처리량을 허용하며 이로 인해 그것이 먼저 스케줄링 된다. 에이전트-1 및 에이전트-4의 중요도가 같더라도 에이전트-4의 작업 처리량은 에이전트-1 보다 더 높으며, 이로 인해 에이전트-4의 2개의 전송은 에이전트-1 보다 더 앞서 스케줄링 된다. 에이전트-2로부터의 전송이 마지막으로 스케줄링 된다.

앞선 경우에서 에이전트 커뮤니티의 우선순위는 다음과 같이 얻어지는 CT (Community Throughput)에 의해 결정된다.

$$CT_k = \sum_{i=1} T_k[i], \ (k = 1, 2, 3...)$$

k 와 i 는 각각 개별 커뮤니티 내의 에이전트 커뮤니티의 수와 에이전트의 수를 나타낸다. 뒤의 경우에서 실제로 우선순위 CT는 다음과 같이 결정된다.

$$CT_k = \sum_{i=1} w_{C_k} T_k[i], \ \sum_{i=1} w_{C_i} = 1, \ (k = 1, 2, 3...)$$

I 와 M 행렬은 시스템 개발 시 에이전트 시스템 설계자에 의해 결정된다는 것에 유념해야 된다. 하지만, M행렬은 에이전트 시스템 작동 동안 변화할 수 있다. 따라서, 작동 중 정확한 T 행렬을 얻기 위해 에이전트 사이에서 메시지 교환을 감시함으로써 주기적으로 업데이트될 필요가 있다. M은 사용자 선호도의 변화, 서비스 에이전트의 증가 등으로 초래된 시스템 환경 변화로 인해 업데이트될 필요가 있다. 에이전트 시스템에서 에이전트는 학습을 통해 자신들의 기능성을 증가하고 확장할 수 있다. 결과적으로 새로운 서비스가 에이전트 커뮤니티에 의해 제공될 수 있다. M행렬을 업데이트 하는 것은 이 현상을 반영한다. 장기적으로 I 와 T 행렬을 이용하는 제안된 메시지 스케줄링 기법은 효과적으로 에이전트 시스템으로 하여금 시스템 환경 변화에 적응하도록 할 수 있다.

Ⅳ. 성능 평가

제안된 계획의 유효성을 평가하기 위해 우리는 우리가 개발한 에이전트를 사용하는 전과자로부터 아이들을 보호하는 에이전트 기반의 범죄 예방 시스템을 실행하였다. 전자팔지를 찬 전과자가 아이들에게 다가올 때 범죄 예방 시스템은 아이에게 이러한 상황을 GPS로 추적하여 경고한다. 에이전트기반의 범죄 예방 시스템의 상황 수준을 표1에서 보인다. 에이전트기반의 범죄 예방 시스템은 범죄 수준이 1과 5사이에 있을 때 상황 인지 서비스를 시작한다.

표1. 범죄 예방 시스템의 상황 수준

위험수준	설명
1	아이가 요청한 긴급 호출
2	전과자 와 아이 사이의 거리는 50미터 미만이다.
3	전과자와 어린이 사이의 거리는 100미터 미만이며, 전과자는 사전 설정된 제한 시간보다 본 조건하에서 더 긴 상태로 머무른다.
4	전과자는 제한된 영역으로 들어간다.
5	정상 조건

본 성능평가 시스템은 그림 3에서 보이는 바와 같이 5가지 유형의 에이전트, 즉 센싱 에이전트, 어린이 에이전트, 관리자 에이전트, 제어 센터 에이전트, 그리고 상황 인지 에이전트로 이루어지는 동일한 에이전트 커뮤니티에 의해 지원된다.

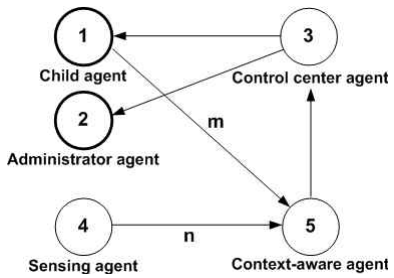

그림 3. 범죄 예방을 위해 형성된 커뮤니티.

실험은 서비스를 제공하기 위해 소요되는 시간을 고려하여 수행되며, 제안된 메시지 스케줄링 기법의 서비스 시간이 FCFS 스케줄링 기법의 서비스 시간과 비교된다. 커뮤니티 내의 에이전트가 주기적으로 서로 통신한다고 가정한다. 그림 4은 본 시스템 내

에 단지 하나의 에이전트 커뮤니티가 존재할 때 메시지 증가의 수로서 서비스 완료 시간들을 비교한다. 제안된 스케줄링 기법은 훨씬 더 적은 서비스 시간을 FCFS 스케줄링의 약 1/3만큼 허용한다. 그것은 또한 훨씬 더 안정한 동작이다.

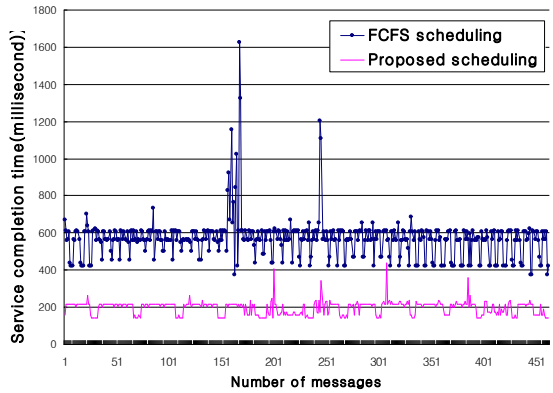

그림 4. 서비스 완료 시간 비교.

V. 결론

본 연구에서 우리는 모바일 환경 내에서 context-aware services 를 위해 여러 중요한 특성들을 반영하는 멀티-에이전트 시스템을 위한 메시지 스케줄링 기법을 보이고 있다. 제안된 계획은 에이전트 시스템으로 하여금 에이전트와 메시지 처리 작업량의 중요도를 기반으로 한 메시지의 우선순위를 적절하게 계획함으로써 환경 변화에 빠르게 적응할 수 있게 한다. 성능 평가는 중요도의 순서에 의해 메시지 스케줄링 컨텍스트에 따라 즉시 서비스를 제공하는데 매우 중요하다는 것을 보여준다. 또한 제안된 접근법이 FCFS 스케줄링보다 훨씬 더 빠른 서비스를 허용한다는 것 또한 증명된다.

제안된 메시지 스케줄링 기법이 적용되었으며 단지 동일한 에이전트 커뮤니티를 이용해서 검사되었다. 우리는 그것을 복수 에이전트 커뮤니티의 더욱 복잡한 시스템을 효과적으로 다루기 위해 더욱 확대해 나갈 것이다.

참고문헌

[1] M. Luck, P. McBurney, O. Shehory and S. Willmott, “Agent Technology: Enabling Next Generation Computing,” AgentLink community, 2003.

[2] J. Wenpin, S.Zhongzhi “Adynamicarchitectureformulti-agentsystems”, TechnologyofObject-OrientedLanguagesandSystems, TOOLS31.Proceedings, pp.253-260, 1999

[3] FIPA: The Foundation for Intelligent Physical Agents, http://www.fipa.org/

[4] K.A. Gouda, J. Cheng, K. Ushijima, “DASS: a discovery agent supporting system” IEEE Systems, Man, and Cybernetics, Vol.5, pp.894-899, Oct. 1999

[5] Naumenko, S. Nikitin and V. Terziyan. “Service matching in agent systems”, Lecture Notes in Computer Science, Vol. 25, pp.223-237, Oct. 2006

[6] JADE: Java Agent Development Frame work, http://jade.tilab.com/

[7] J.P. Charles, “Agent-Based Engineering, the Web, and Intelligence”, IEEE Expert 1996, Vol. 11 No. 6., pp.24-29.

[8] V.A. Pham and A. Karmouch, “MobileSoftwareAgents: AnOverview”, IEEECommunicationMagazine, Vol.36, No.7, pp.26-37July1998.

[9] J. Baumann, et. al., “Communication Concepts for Mobile Agent Systems”, Lecture Notes in Computer Science, Vol. 1219, Springer-Verlag, 1997.