유비쿼터스 컴퓨팅 환경에서 추상적 Context 지원을 위한 연구 방안

차창호° 고영배 김재훈
chcha@dmc.ajou.ac.kr°, {youngko, jaikim}@ajou.ac.kr

A Research to support implicit Context in Ubiquitous Computing Environment

Chang-Ho Cha°, Young-Bae Ko , Jai-Hoon Kim
Graduate School of Information and Communication, Ajou University

요 약

유비쿼터스 컴퓨팅 환경에서 가장 중요한 이슈중의 하나는 상황인지(Context-Aware)가 가능한 환경을 구축하는 것이다. Context-Aware 환경이 구축되면 주변의 상황을 감지하여 특정 어플리케이션을 실행한다거나, 시스템을 재구성하는 등의 일을 수행할 수 있다. 그동안 이런 상황인지가 가능한 환경을 Context-Aware 미들웨어를 통해 구현하려는 연구가 많이 수행되었다. Context-Aware 미들웨어를 구현하기 위해 무엇보다도 중요한 것은 실제 세계의 다양한 종류의 상황을 컴퓨팅 환경에 적용시키는 것이다. 그러나 현실 세계에서 Context의 종류는 거의 무한하다고 할 수 있다. 기술이 발전하게 되면서 인지 가능한 Context의 종류도 무한정 늘어나게 될 것이다. 또한 실제 세계에서의 Context들은 추상적인 경우가 많이 있다. 그러나 Context가 추상적이라 해도 다른 Context 정보를 이용해서 구체화 할 수 있다. 이런 과정을 위해 Context들을 계층적으로 관리해야 할 필요가 있다. 본 논문에서는 Context-Aware 미들웨어를 구현할 때 Context들의 이런 여러 가지 특성들을 고려해서 Context Type을 관리할 수 있는 하나의 객체를 제안하고, 다른 방법들과 비교, 분석해 보았다.

1. 서 론

유비쿼터스 컴퓨팅 환경에 대한 연구 중 Context-Aware는 대단히 중요한 주제이다. 최근의 연구에 의하면 Context 정보에 따라 자원을 관리하기 위한 Context-Aware Middleware[1], 필요한 Context만 이용하도록 도와주는 Context-Aware Middleware[1], Context-Aware하고 이기종 간의 호환성을 보장해주는 Digital Home Environment[2]를 구축하려는 노력이 있었다. 이런 여러 가지 Context-Aware한 미들웨어를 구축하는 데 있어서, 가장 중요한 것 중 하나가 실제 세계의 Context를 Computing 환경에 맞게 적용하는 것이다. 그러나, 실제 세계의 추상적인 Context를 컴퓨팅 환경에 적용시키는 것은 매우 힘든 일이다. 기존의 Context-Aware 미들웨어에서는 명시적인 Context에 대한 인지만을 고려해 왔다. 하지만, 추상적인 Context까지도 Application에서 사용할 수 있게 된다면, 보다 현실 세계에 가까운 프로그래밍이 가능해 질 것이다. 예를 들어 "날씨"에 따라 음악을 연주하는 "날씨 적응형 음악 연주기" 어플리케이션을 생각해 보자. 어플리케이션 레벨에서는 "날씨" 라는 Context에 맞는 음악을 연주하는 것이 본래의 목적이다. 그러나 기존의 Context-Aware 미들웨어에서 "날씨"와 같은 추상적인 Context를 적용하는 것은 불가능하다. 하지만, 그림 1에서 처럼, "날씨" 라는 Context는 몇 가지 명확한 Context들(예를 들어, 온도, 빛의 양, 습도 등)을 통해서 객관적인 값으로 이끌어 낼 수 있는 정보이다. 이처럼 계층적으로 Context를 표현함으로써 추상적인 Context를 구체적으로 표현할 수 있다.

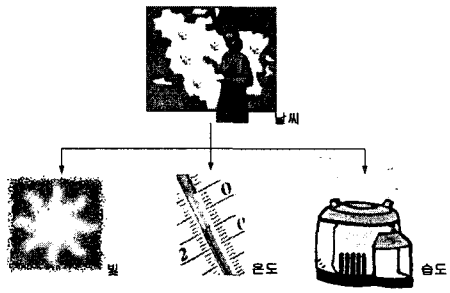

그림 1 계층화된 Context 구조의 예

* 본 논문은 과학기술부의 유비쿼터스 컴퓨팅 및 네트워킹 원천기반 기술개발의 프론티어사업에 의해 연구됨

또, 이렇게 계층적으로 관리하게 되면, 추상적인 Context뿐만 아니라, 현재 사용할 수 없는 Context에 대한 요청이 오더라도 그 Context의 하위 개념의 Context를 이용해 도출해 낼 수 있다. 그렇게 되면 Context가 존재하지 않아서 Service가 불가능한 경우가 훨씬 줄어들게 될 것이다.

본 논문에서는 이러한 Context의 특성에 의한 문제점들을 해결하기 위해 여러 Context 관리 방법에 대해서 알아보고, 문제점을 보완하는 방법으로 Context 관리 서비스를 제안한다.

2. 연구 배경 및 관련 연구

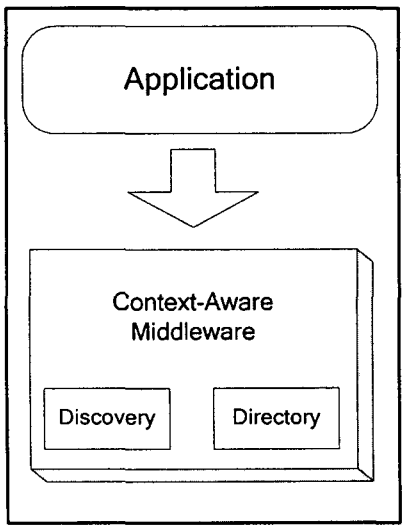

그림 2 기존의 미들웨어 모델

Context-Aware 미들웨어의 기능은 어플리케이션으로부터 요구 받은 Context에 대해 알맞는 기능을 수행하는 것이다. 기존의 미들웨어에서는 그림 2에서처럼 Context에 대한 요청이 있을 때, Local에서 제공 가능한 Context는 Discovery Facility, Global에서 제공 가능한 Context는 Directory Facility를 통해 얻어 이를 적당히 병합하여 사용한다[1]. Context의 구조와 모니터링 된 값은 미들웨어 내의 Metadata Manager에서 Metadata의 형태로 관리한다[1]. 하지만 이 모델은 명시적인 Context에 대해서만 올바르게 작동한다. "날씨 적응형 음악 연주기" 를 이 환경에서 작성하기 위해서는 어플리케이션 작성 단계에서 "날씨" Context를 명시적이게 임의의 Level로 정의해야 하고 결국은 Middleware에게는 "날씨"가 아닌 임의로 정의한 Context들을 요구해야 한다. 그러나, 미들웨어 차원에서 이 추상적인 Context를 지원해 준다면 어플리케이션을 작성하는 입장에서는 임의의 정의를 할 필요 없이 단지 "날씨"에 대한 어플리케이션만 작성하면 되므로 보다 투명성을 보장할 수 있다.

3. Context Type Manager

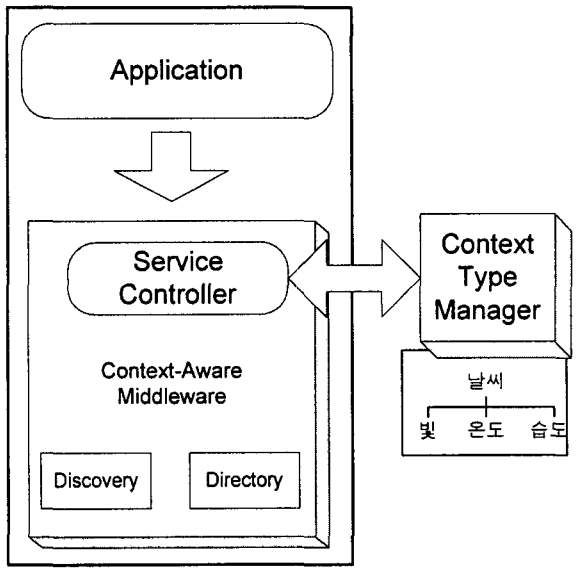

그림 3 Context Type Manager를 추가한 모델

그림3은 앞에서 제기된 문제를 개선하기 위해 본 논문에서 제안하고자 하는 Context Type Manager가 포함된 모델이다. 어플리케이션에서 미들웨어에 "날씨" 라는 Context를 요구하면, 미들웨어에서는 Context Type Manager로부터 "날씨"라는 Context의 계층화된 구조를 얻어온다. 이 구조를 이용해 미들웨어에서는 날씨라는 Context를 구성하여 Application에게 적당한 음악을 플레이 하도록 할 수 있다. Context Type Manager에서는 "날씨" 값이 갖게 되는 값을 레벨화해서 정의해둬야 한다.

Context의 계층적인 구조를 구성할 때 이런 독립된 서비스형태가 아닌 기존의 Metadata를 수정하는 것으로도 어느 정도의 표현은 가능하지만 충분히 표현하기는 힘들다. 왜냐하면 실제 세계에서의 Context의 체계적인 구조는 무한히 확장 가능하고 그만큼 복잡한 구조를 갖게 된다. Metadata로는 이렇게 복잡한 구조를 표현하기에는 한계가 있다. 또, Context Type이 변화할 때 기존의 모델에서는 각 미들웨어가 Metadata를 갖게 되므로 즉각적으로 반응하기 힘들다. 하지만, Context Type Manager라는 독립된 객체로 구성하면 Context의 변화에 대해 동적으로 대응할 수 있다.

4. 구현방법

Context Type Manager 는 각 노드의 사이에 존재하게 된다. 구현하는 방법으로 미들웨어의 사이에 위치하는 Mobile Agent의 형태로써 Context의 구성을 관리하는 방법이 있다[3]. Context Type Manager는 Agent로써 Context의 구조를 기억하고, 변화가 필요할 경우 구조를 확장한다. 각 미들웨어에서는 Context가 요구될 때마다 Context Type Manger에게 질의를 통해 해당 Context의 계층화된 구조를 전달해 준다.

또 다른 구현 방법으로는 Context 구조의 Updater를 Agent의 형태로 구성하고 Context 구조는 각 미들웨어에 저장하는 방법이다. Context의 구조가 자주 update되지 않는다면 더 효율적인 방법이다.

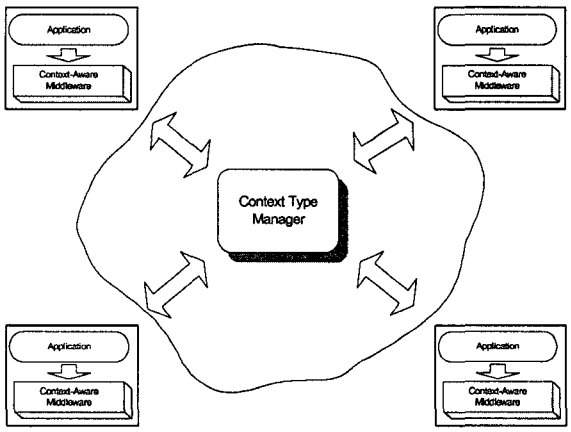

그림 4 Context Type을 Agent의 형태로 관리할 경우

5. 결론 및 향후 연구방향

본 논문에서는 Context-Aware 미들웨어에서 실제 세계의 Context를 Computing 환경에 적용시키는 방법에 대해서 다시 살펴보고, Context의 Type들을 관리하는 방법으로 정적인 Metadata형식이 아닌 동적인 서비스 형태를 제안하였다. Context의 Type을 정적인 Metadata로 관리하게 될 경우 미들웨어가 구조적으로 간단해진다는 장점이 있지만, 미들웨어 각각에 종속적이게 되어 Context Type들의 변화에 대한 대응이 힘들고, 복잡한 Context들의 관계를 나타내기 힘들다. 서비스의 형태로 관리하게 되면, 구조적으로는 미들웨어에서는 서비스로 연결하는 부분이 필요하게 되고, 서비스 자체도 구현해야 하는 등 구조적으로는 복잡해질 수도 있다. 또, Context를 요구할 때마다 질의를 통해 Context의 구조를 파악해야 하므로 효율이 떨어질 수도 있다. 하지만, Context의 특성상 Context의 Type의 변화는 항상 일어날 수 있고, 보다 복잡한 Context의 구조를 지원할 수 있도록 Context Type Manager는 독립된 형태로 존재해야 한다.

향후 연구에서는 위에서 예를 든 "날씨에 따른 음악 연주기" 시나리오를 위한 Context-Aware 미들웨어를 구현하고, Context Type 관리 서비스를 구현하여 위에서 구현된 미들웨어에 적용하도록 한다. 각각의 구현 방법에 대해서도 장단점을 비교해 보도록 한다. 또한, Context Type이 계층화된 구조로써 복잡하게 얽힘으로써 발생할 수 있는 문제를 방지하기 위해 구조를 정의하는 정책을 마련하도록 한다.

5. 참고문헌

[1] Bellavista, P., Corradi, A., Montanari, R., and Stefanelli, C. "Context-Aware middleware for resource management in the wireless Internet", Software Engineering, IEEE Transactions on, Vol. 29, No. 12, Dec. 2003.

[2] Kyeong-Deok Moon, Young-Hee Lee, Chae-Kyu Kim, "Context-Aware and Adaptive Universal Home Network Middleware for Pervasive Digital Home Environment", Consumer Communications and Networking Conference, 2004. CCNC 2004. First IEEE , 5-8 pp. 721 - 723, Jan. 2004.

[3] P. Bellavista, A. Corradi, and C. Stefanelli, "Mobile Agent Middleware for Mobile Computing," Computer, vol. 34, no. 3, pp. 73-81, Mar. 2001.