애드혹 유비쿼터스 컴퓨팅 환경에서의 컨텍스트 인지 기반 상호작용 지원을 위한 효율적인 Publish/Subscribe 기법*

문상철, 이경민, 이동만
한국정보통신대학교
{sangchul, kmlee, dlee}@icu.ac.kr

An Efficient Publish/Subscribe Scheme for Supporting Context-aware Interactions in Ad-hoc Ubiquitous Computing Environments

Sang-chul Moon, Kyungmin Lee, and Dongman Lee
Information and Communications University

요 약

애드혹 유비쿼터스 컴퓨팅 환경에서 상호작용 미들웨어는 컨텍스트의 변화에 따라 애플리케이션간의 상호작용을 재구성할 수 있도록 지원해야 한다. 또한 미들웨어는 컨텍스트 인지 상호작용을 지원하는데 있어 컨텍스트 종류나 애플리케이션들 혹은 환경의 다양성에 따라 컨텍스트 변화에 대한 적응성을 동적으로 변화시킬 수 있도록 기능을 제공해야 한다. 하지만 기존 연구들은 컨텍스트 변화에 대한 적응성을 변화시키는 이러한 요소들에 대해 고려하지 않으며 그로 인해 컨텍스트 변화에 대한 적응 정도를 정적으로 설정함으로써 컨텍스트 인지 상호작용을 지원하는데 있어 경우에 따라 오버헤드가 발생하거나 컨텍스트 변화에 빠르게 대응하지 못한다. 따라서 본 연구는 애드혹 유비쿼터스 컴퓨팅 환경에서 동작하는 Publish/Subscribe 기반의 상호작용 미들웨어에서 이벤트 중계의 정확성과 컨텍스트 변화 적응성간의 관계를 기반으로 다양한 컨텍스트 종류나 사용자, 환경에 따라 이벤트 중계의 정확도를 동적으로 조절하는 기법을 제안한다. 그리고 이를 기반으로 이벤트 중계 정보를 동적으로 재구성하여 컨텍스트에 맞춰 효율적으로 컨텍스트 인지 상호작용을 구성할 수 있는 기법을 제안한다. 비교 분석 결과 기존 연구에 비해 컨텍스트 변화에 대한 적응성에서 더 나은 결과를 보여주었다.

1. 서 론

유비쿼터스 컴퓨팅은 컴퓨팅 및 네트워킹 능력을 갖춘 다양한 지능 객체들이 서로 상호작용(interaction)하여 사용자가 원하는 서비스를 제공하는 것을 목표로 한다. 또한 사용자 및 환경의 컨텍스트(context)를 능동적으로 반영하여 가장 적절한 방식으로 서비스를 제공하려 한다. 이를 위해 유비쿼터스 컴퓨팅 환경에서의 애플리케이션은 컨텍스트 변화에 따라 지능 객체 간의 상호작용이 끊임없이 변경되어야 한다. 이러한 상호작용의 변경은 미리 예측할 수 없으며 자연발생적(spontaneous)이다 [1]. 예를 들어, 이어폰을 통해 음악을 듣고 있던 사용자가 집안으로 들어왔을 때 이 사용자는 거실에 위치한 스피커를 통해 음악을 듣는 것을 선호한다고 가정할 경우, 음악을 재생하는 애플리케이션은 이 사용자의 위치 (컨텍스트 상태) 에 따라 이어폰 혹은 스피커와 상호작용을 해야 한다.

위와 같은 특징을 갖는 유비쿼터스 컴퓨팅 환경에서의 상호작용을 지원하는데 있어서 기존 많은 연구들이 Publish/Subscribe 모델을 채택하고 있다. Publish/Subscribe 모델은 이벤트 생산자와 소비자의 중간에 이벤트 브로커를 둠으로써 이벤트 생산자와 소비자 간 의존성을 없앴다 [2]. 따라서, 이벤트 생산자와 소비자는 동시에 실행될 필요도, 서로의 위치를 인지하고 있을 필요도 없게 되어 변경이 잦은 자연발생적 상호작용을 쉽게 지원할 수 있다.

애드혹 (ad-hoc) 유비쿼터스 컴퓨팅은 고정된 인프라스트럭처가 존재하지 않는 환경에서의 유비쿼터스 컴퓨팅을 의미한다[3]. 따라서, 애드혹 유비쿼터스 컴퓨팅 환경에서는 Publish/Subscribe 모델을 적용하는데 있어 고정된 이벤트 중계자(Event Mediator)를 가정할 수 없다. 이와 같은 제약점을 해결하는 방법에 있어 기존의 애드혹 유비쿼터스 컴퓨팅 환경을 위한 Publish/Subscribe 상호작용 미들웨어에 관한 연구들은 크게 세가지 형태로 나뉜다. 하나는 중앙의 이벤트 중계자를 가정하지 않고 이벤트 중계 기능을 애드혹 환경 내 모든 노드에 완전히 분산시킨 형태이다[4]. 다른 하나는 Publisher와 Subscriber 사이의 이벤트 전달 기능으로써 멀티캐스트 기법을 이용하는 형태이다[5]. 마지막으로 애드혹 환경 내의 특정 노드를 선택하여 이 노드가 이벤트 중계자의 역할을 수행하도록 하는 형태이다[6]. 하지만, 기존 연구들은 애드혹 유비쿼터스 컴퓨팅 환경에서 컨텍스트 인지 상호작용이 원활하게 제공되기 위해서 필요한 몇 가지 요소들을 고려하지

* 본 연구는 21세기 프론티어 연구개발사업의 일환으로 추진되고 있는 정보통신부의 유비쿼터스컴퓨팅및네트워크원천기반기술개발사업과 정보통신부 및 정보통신연구진흥원의 디지털미디어연구소 지원사업의 지원에 의한 것임

않았다. 먼저 상호작용 미들웨어는 변화할 가능성이 있는 컨텍스트 상태들을 미리 예측하여 이러한 컨텍스트 상태를 가진 애플리케이션과 미리 상호작용을 수행할 준비를 해야 컨텍스트 변화에 빠르게 대처할 수 있다. 하지만 이러한 과정은 이벤트 중계의 정확성을 감소시켜 이벤트의 낭비를 초래하게 된다. 만일 이벤트의 낭비를 줄이기 위해 중계의 정확성을 높이게 될 경우 컨텍스트 변화에 빠르게 적응하지 못하게 된다. 따라서 컨텍스트 인지 상호작용을 원활하게 제공하기 위해서 상호작용 미들웨어는 컨텍스트 변화에 대한 적응성과 이벤트 중계의 정확성 사이의 트레이드 오프 (trade-off) 관계를 고려하여 동적으로 이들의 수준을 결정할 수 있어야 한다. 하지만 기존 연구들은 이러한 고려를 하지 않았거나 혹은 동적으로 컨텍스트 변화에 따라 변경하지 못하는 단점이 있다. 또한 컨텍스트 변화에 대한 적응성은 컨텍스트 종류나, 사용자의 개인 정보나 선호, 혹은 주변 환경에 따라 다르게 된다. 따라서 이러한 요소들을 고려하여 컨텍스트 변화에 대한 적응성의 정도가 동적으로 결정되는 것은 컨텍스트 인지 상호작용을 원활하게 지원하는데 중요한 요소이다. 하지면 기존 연구들은 역시 이와 같은 요소를 고려하고 있지 않기 때문에 경우에 따라 네트워크 오버헤드를 가중시키거나 컨텍스트 변화에 빠르게 대처하지 못하게 된다는 문제점이 있다.

본 연구는 애드혹 유비쿼터스 컴퓨팅 환경에서 컨텍스트 인지 상호작용을 효율적으로 지원하기 위해 이벤트 중계의 정확성과 컨텍스트 변화에 대한 적응성간의 트레이드 오프 관계에 대해 설명한다. 그리고 이러한 컨텍스트 변화에 대한 적응성의 수준을 결정하기 위해 컨텍스트 종류나 사용자 정보나 선호와 같은 다양한 고려 요소들을 기반으로 적정 수준의 이벤트 중계의 정확도를 동적으로 계산하는 기법과 이렇게 계산된 정확도를 토대로 이벤트 중계 정보를 동적으로 재구성하는 기법을 제안한다.

본 논문은 다음과 같이 구성되어 있다. 2장에서는 애드혹 환경에서의 상호작용 미들웨어에 관한 기존 연구들을 분석하며 이들 연구가 가지는 문제점을 설명하고 3장에서는 문제 해결을 위한 디자인 고려사항 및 알고리즘에 대해 설명하며 4장에서는 기존 연구와의 비교 분석을 통해 제안한 알고리즘에 대해 평가한다. 5장에서는 결론 및 향후 작업을 설명하며 맺는다.

2. 관련 연구 분석

EMMA[6]는 애드혹 네트워크 환경에서의 Publish/Subscribe에 대한 연구를 통해 기존 JMS (Java Messaging Service)를 애드혹 네트워크 환경에 이식하는 연구이다. JMS의 토픽(Topic)은 Publisher와 Subscriber사이에서 이벤트를 중계해주는 역할을 하는데, 이 연구에서는 Publisher와 Subscriber 사이에 어느 특정 네트워크 노드(Node)가 이러한 중계 노드로써의 역할을 담당한다. 그리고 그 노드는 해당 토픽에 관심 있는 Subscriber 목록을 유지하면서 Publisher의 이벤트를 수신해서 Subscriber 목록을 따라 Subscriber에게 이벤트를 전달한다. 그러나 이러한 EMMA를 컨텍스트 인지 상호작용을 지원하도록 확장하게 될 경우 오버헤드가 초래되는 문제점이 있다. 유비쿼터스 컴퓨팅에서는 서로 관심 있는 토픽을 공유하는 노드들 간의 상호작용뿐만 아니라 이러한 토픽과 컨텍스트를 함께 공유하는 노드들 간에도 상호작용이 필요하기 때문에, 한 토픽에 대해 컨텍스트 상태 및 종류에 따라 이벤트 중계가 별도로 존재해야 한다. 게다가 컨텍스트가 변할 때마다 Publisher 및 Subscriber는 서로 다른 이벤트 중계 노드에 이벤트를 발행하거나 혹은 이벤트 수신을 위한 등록해야 하므로 컨텍스트의 잦은 변화 시 이벤트 중계 노드를 교체하는 오버헤드가 발생하게 된다.

STEAM[5]은 이동 애드혹 네트워크(Mobile Ad-hoc Network) 환경에서 동작하는 상호 협동 가능한 애플리케이션의 개발을 지원하는 이벤트 기반의 미들웨어이다. STEAM은 Publisher와 Subscriber사이에 이벤트 중계자를 없애고 필터링 기능을 Publisher와 Subscriber 측에 분산시켰다. 그리고 멀티캐스트 그룹과 이벤트의 전달 범위 개념이 결합된 그룹통신방법인 Proximity group communication을 이용해 유저가 정한 위치와 공간 내에서 이벤트를 송수신할 수 있도록 하였다. 하지만 STEAM은 유비쿼터스 컴퓨팅 환경에서의 상호작용 미들웨어로 사용되기에는 오버헤드가 심하다는 문제점이 있다. STEAM은 컨텍스트로서 위치 변화만을 고려하였지만, 유비쿼터스 컴퓨팅에서는 위치뿐만 아니라 다른 수많은 컨텍스트가 존재하며, 만일 이러한 각각의 컨텍스트에 따른 상호작용을 모두 지원하게 된다면 Proximity group의 수가 따라서 증가하게 된다. 게다가 컨텍스트의 변화가 잦아지게 되면 Proximity group에 Join/Leave하게 되는 회수가 늘어남에 따라 멀티캐스트 그룹을 관리하는데 드는 오버헤드가 증가하게 된다.

[4]은 Content 기반 Publish/Subscribe 미들웨어에서 논리적 이동성을 어떻게 지원할 것인지에 관한 연구이다. 이 연구에서는 이벤트 중계를 위해 Subscription Forwarding 방식을 쓴다. 이는 Publisher와 Subscriber들 사이의 중간 이벤트 중계 노드들이 Subscriber가 관심 있는 이벤트에 대해 기술한 Subscription 정보를 내부에 저장해둔 뒤, Publisher로부터 이벤트가 전달되었을 때 저장해둔 Subscription 정보를 토대로 이벤트를 중계하는 방식을 말한다. [4]에서 설명하는 논리적 이동성이란 컨텍스트의 변화를 의미하며, 결국 [4]는 컨텍스트의 변화가 발생하였을 때 중간 이벤트 중계 노드들의 변화에 적응하기 위해 Subscription 정보를 어떻게 바꿔야 하는지에 관한 연구이다. 이 연구에서는 컨텍스트 변화에 적응하는데 걸리는 지연 시간으로 인한 메시지 유실 문제를 풀기 위해 단위 시간 변화에 따른 Subscriber의 예측 가능한 위치를 고려하여 이를 이동 그래프(Movement graph)로 만들었다. 그리고 이 그래프 정보를 토대로 단위 시간에 따라 변화가 예상되는 위치들을 계산하는 함수(loc)를 만들었다. 그래서 Subscriber에서부터 Publisher까지 사이의 중간 노드들은 loc 함수를 이용하여 Subscriber의 예측 가능한 위치를 계산하여 이를 Subscription 정보에 반영시켜 단계적으로 필터링을 수행하도록 하였다. 즉, Subscriber에서 가까운 노드는 현재 Subscriber의 위치만을 기준으로 필터링하며 Subscriber에서 먼 중간 노드들은 예상 위치에서 발행되는 이벤트까지 수신하도록 Subscriber에서 가까운 노드에서 먼 노드까지 점진적으로 Subscription 정보를 구성하였다. 이를 통해 Subscriber의 컨텍스트(위치)가 변하게 되더라도 다시 Subscription을 할 때 새로운 Subscription 정보가 전달 되어야 하는 중간 노드들의 범위를 축소시켜 컨텍스트 변화에 적응하는 시간을 단축시켰다. 하지만 유비쿼터스 컴퓨팅

환경에서 컨텍스트 상태 간의 관계가 복잡해지게 되면 이들의 방법은 오버헤드를 초래하게 된다. 빌딩 내부 공간에서의 위치라는 컨텍스트로 예로 들 경우, 빌딩의 입구를 들어서게 되면 빌딩의 로비를 통해 빌딩 내 각 방이나 엘리베이터, 화장실 등 로비에서 그 다음으로 이를 수 있는 공간은 매우 많다. 이러한 공간에 사용자가 위치할 수 있는 상태들을 모두 컨텍스트 상태로 관리할 경우 사실상 하나의 컨텍스트 상태에서 다른 상태로의 변화할 수 있는 경우의 수가 크게 늘어나기 때문에 이를 예측하는 것이 힘들게 된다. 따라서 이러한 경우에 이동 그래프가 복잡해지게 되어 단위 시간 별로 사용자의 위치를 예측하는 것은 Subscriber가 컨텍스트를 공유하는 모든 위치의 Publisher로부터 이벤트를 수신하게 된다. 이는 이벤트를 Flooding한 후 Subscriber측에서 필터링하는 것과 차이가 없어지게 되어 Subscriber들에 쓸모 없는 이벤트들의 전달비율이 크게 늘어나게 되고 이는 네트워크의 오버헤드를 가중시킨다.

게다가 [4]에서 정의한 이동 그래프의 목적은 Subscriber가 하나의 컨텍스트 상태에서 단위 시간 후에 도달할 수 있는 컨텍스트 상태를 이동 그래프 및 loc 함수를 통해 예측하여 이들 컨텍스트 상태의 Publisher에서 발행한 이벤트들을 중간 이벤트 중계자 노드들이 미리 수신하고자 하는 것이다. 이를 통해 컨텍스트 상태가 바뀌더라도 새로운 Subscription 정보의 전달 범위를 최소화 시키기 위함이다. 이는 이벤트 중계의 정확성을 감소시키는 대신 컨텍스트의 변화에 빠르게 대처하는 것으로 요약할 수 있다. 따라서 컨텍스트 변화에 대처하는 시간, 즉 컨텍스트 변화 적응성과 이벤트 중계의 정확성은 서로 트레이드 오프 관계에 있음을 알 수 있다. 하지만 애드혹 유비쿼터스 컴퓨팅 환경에서 다양한 컨텍스트의 종류나 애플리케이션들 그리고 환경의 차이로 인해 이러한 이벤트 중계의 정확성은 동적으로 변화해야 한다. 예를 들어, 장시간 회의중인 참석자는 확률상 회의장 외의 다른 장소로 이동할 확률이 낮으며, 따라서 이런 경우 이벤트 중계의 정확성을 희생하면서 컨텍스트 변화에 대처하는 시간을 줄이게 되면 위에서 설명한 것과 같이 쓸모 없는 이벤트의 전송이 많아지게 되어 오버헤드가 초래된다. 하지만 [4]는 이와 같이 컨텍스트 종류나 환경에 따라 동적으로 컨텍스트 변화에 적응하는 시간을 조절하지 못한다.

3. 설계

3.1. 설계 시 고려 사항

애드혹 유비쿼터스 컴퓨팅 환경에서의 상호작용 미들웨어는 컨텍스트의 종류나 애플리케이션, 그리고 환경에 따라 이벤트 중계의 정확도를 동적으로 조절할 수 있는 메커니즘을 제공하여야 하며, 이를 통해 컨텍스트 인지 상호작용을 지원하는데 있어 경우에 따라 발생할 수 있는 오버헤드를 줄일 수 있는 방법을 제시해야 한다. 이를 위해 설계 시 다음과 같은 두 가지 사항을 고려해야 한다.

- 이벤트 중계의 정확도를 동적으로 계산하기 위해 고려될 수 있는 요소들은 무엇이 있는가? 그리고 이들을 이용해 어떻게 적절한 정확도를 계산할 수 있는가?
- 이벤트 중계의 정확도에 따라 Publisher와 Subscriber 사이의 중간 이벤트 중계자들은 Subscription 정보를 어떻게 변환시킬 것인가?

이러한 사항을 기반으로 먼저 컨텍스트 인지 상호작용을 지원하는데 있어 컨텍스트 종류나 환경과 같은 요소들을 고려하여 컨텍스트 변화의 적응성을 조절하기 위해 이벤트 중계의 정확도 값을 동적으로 계산하는 방법을 제안한다. 그리고 Publisher와 Subscriber사이의 중간 이벤트 중계 노드들이 새로이 계산된 정확도 값을 바탕으로 예상 가능한 컨텍스트의 값들을 동적으로 계산하여 이를 Subscription 정보에 반영하는 알고리즘을 제안한다.

3.2 알고리즘

본 연구에서는 먼저 이동 그래프[4] 개념을 확장하여 컨텍스트 전이 그래프 개념을 구상하였다. 이 그래프는 컨텍스트 종류별로 존재하며, 해당 컨텍스트 종류의 고유의 특성을 고려하여 컨텍스트 전이 그래프 생성 함수를 통해 동적으로 생성된다. 그리고 사용자의 개인 정보나 주변 환경 혹은 애플리케이션의 특성을 고려하여 이벤트 중계의 정확도 값을 계산하고, 이렇게 계산된 정확도 값을 컨텍스트 전이 그래프 조회 함수에 인자로 하여 각 이벤트 중계 노드별로 예상 가능한 컨텍스트 값들을 조회한다. 각 이벤트 중계 노드들은 이 컨텍스트 값들을 기반으로 Subscription 정보를 변경하여 전체적으로 해당 컨텍스트에 대해 컨텍스트 변화 적응성을 동적으로 조절하게 하였다.

3.2.1 컨텍스트 전이 그래프

기존 이동 그래프가 단위 시간별 예측 가능한 위치를 표현하는데 반해 이를 다양한 컨텍스트를 지원할 수 있도록 컨텍스트 전이 그래프 개념으로 확장하였다. 이 컨텍스트 전이 그래프는 각 컨텍스트의 종류별로 존재하며 이 그래프는 다음과 같은 세 가지 요소로 구성되어 있다.

- 그래프 노드 : 컨텍스트 값
- 연결(Edge) : 두 그래프 노드(컨텍스트 값) 사이의 전이를 표현한다.
- 가중치 : 두 그래프 노드 사이의 연결에 부여되는 가중치를 뜻한다. 이러한 가중치는 두 위치간의 거리나 사용자의 움직임 패턴을 기반으로 한 확률값, 혹은 사용자의 움직임 기록등을 토대로 매겨질 수 있다.

컨텍스트 전이 그래프는 다음과 같이 표현될 수 있다.

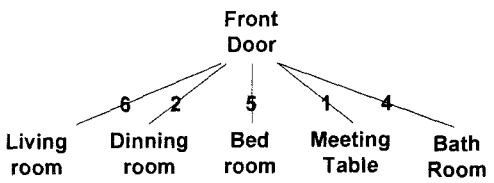

그림 1. Active Surroundings 데모룸의 컨텍스트 전이 그래프

현재 이 그래프는 FrontDoor위치에서 집안 환경의 각 위치간의 연결만을 표현하였으며, 두 그래프 노드 사이 연결의 가중치를 두 위치간의 거리를 이용해 계산하였다. 여기서 거리의 단위는 두 위치간을 이동하는데 드는 단위 시간의 양이다.

3.2.2 컨텍스트 전이 그래프 생성 함수

컨텍스트 전이 그래프를 동적으로 생성하기 위해 사용되는 함수 *gen*를 다음과 같이 정의하였다.

gen(vl, ei) = 컨텍스트 전이 그래프

이 함수에서 vl은 그래프 노드로 사용될 컨텍스트 값들을 의미하고 ei는 그래프 노드들간의 연결 정보 및 그들의 가중치를 의미한다. 이 함수를 통해 해당 컨텍스트 종류에 대한 컨텍스트 전이 그래프가 동적으로 생성된다.

3.2.3 컨텍스트 전이 그래프 조회 함수

컨텍스트 전이 그래프로부터 특정 단위 시간이 지난 후에 예상 가능한 컨텍스트 값들을 조회하는 함수이다. 기존 연구[4]에서 제안한 이동 그래프를 조회하는 loc 함수를 기반으로 확장하였으며, 다음과 같이 정의하였다.

ret(cv, tu, ac) = {예상 가능한 컨텍스트 값의 집합}

이 함수에서 cv는 현재 컨텍스트 값, tu는 예측하고자 하는 단위 시간, 그리고 ac는 정확도를 의미한다. 여기서 정확도는 0보다 커야 한다. 이 함수를 통해 인자로 정의한 단위 시간이 지난 후에 현재 컨텍스트 값에서 어떤 다른 컨텍스트 값으로 전이가 가능한지 예측할 수 있다. 여기서 정확도는 0과 1 사이의 소수점으로 표시된다. 예측 가능한 컨텍스트 값의 목록을 계산하는 방법은 다음과 같다.

- 단위시간을 정확도로 나눈 값을 threshold 값으로 정의한다. 단위시간이 2이고 정확도가 0.75일 경우 2/0.75가 되어 threshold 값은 2.66이 된다. 이 의미는 컨텍스트 값간의 연결에 부여된 가중치가 2.66보다 작은 컨텍스트 전이를 모두 선택하겠다는 의미이다. *Threshold* 값이 가중치와 같은 경우 선택되지 않는데, 그 이유는 컨텍스트 전이로 인한 처리 시간을 고려해야 하기 때문이다.
- 컨텍스트 전이 그래프의 연결에 부여된 가중치들 중 위에서 계산한 threshold 값보다 작거나 같은 연결들을 선택한다. 이때 복수의 연결들의 가중치를 합한 경우도 고려한다. 즉, 컨텍스트 상태가 A, B, C가 있다고 가정하고 A-B, B-C간 연결의 가중치가 각각 1이라고 하였을 때 A-C간 직접 연결이 존재하지 않는다고 하면 A-C 연결의 가중치는 2가 된다. 위에서 계산한 Thredhold 값이 2.66이므로 이와 같이 A-C 처럼 B 상태를 거쳐서 계산된 가중치의 연결도 예측 가능한 컨텍스트의 상태로써 반환된다.
- 정확도가 1일 경우 이벤트 중계의 정확성을 높여서 해당 단위 시간 안에 허용되는 컨텍스트 값들 만을 반환하며, 정확도가 낮아질수록 허용되는 단위 시간의 범위가 넓어지게 되어 좀 더 많은 컨텍스트 값을 반환하게 된다.

3.2.4 이벤트 중계 정확도 값 계산

이벤트 중계 정확도를 동적으로 계산하기 위해 두 가지 요소를 고려해야 한다. 첫 번째는 컨텍스트 변화의 빈도이다. 즉, 컨텍스트 변화가 빈번하게 되면 컨텍스트 변화 적응도를 높여야 하므로 이러한 경우 이벤트 중계의 정확성을 줄여 다양한 컨텍스트 변화에 대응할 수 있도록 해야 한다. 두 번째는 이벤트 중계 노드들의 단위 시간 당 이벤트 수신량이다. 컨텍스트의 변화 없는 상태에서 Publisher의 이벤트 발행이 잦아지게 되면 필요 없는 이벤트에 대한 수신을 줄여야 하므로 이벤트 중계의 정확성을 높여야 한다.

그러므로 이벤트 중계의 정확성(AC)을 동적으로 계산하기 위해 다음과 같은 수식을 설계하였다..

$$NE_{avg\_t0} = 1 \qquad AC_0 = 0.5 \tag{1}$$

수식 (1)에서 NE는 1 단위 시간 동안 이벤트 중계 노드가 수신한 이벤트의 수이며 초기 정확도 값은 0.5로 설정한다.

$$NE_{avg\_t} = \sum_{i=1}^{t} NE_i \Big/ t \tag{2}$$

수식 (2)는 단위 시간 1부터 t 사이의 NE의 평균값을 계산하는 수식이다.

$$AC_{t+1} = AC_t + \alpha\, NE / NE_{avg\_t-1} \tag{3}$$

수식 (3)은 다음 단위 시간에 적용될 수 있는 이벤트 중계 정확성을 계산하는 방법을 설명하고 있다. 단위시간 1부터 t-1까지 한 단위시간 동안 수신된 이벤트 수의 평균값을 구하고, 이를 현재 시간 t사이에 수신된 이벤트의 수로 나눈다. 그리고 이 값을 정확도의 증가율로써 계산하여 현재 정확도에 더한다. 만일 현재의 단위 시간 동안 평균치보다 더 많은 이벤트들이 수신되었을 경우, 이 수식을 통해 정확도는 현재 값보다 증가하게 된다.

$$AC_{t+1} = AC_t \times \beta\, NC / NC_{avg\_t-1} \tag{4}$$

수식 (4)는 컨텍스트 변화 이벤트를 수신함에 따라 정확도를 어떻게 변경시킬 것인지에 관해 설명하고 있다. NC는 한 단위 시간 동안 수신한 컨텍스트 변화 이벤트의 수를 뜻한다. 수식 (4) 역시 수식 (3)의 경우와 마찬가지로 1 부터 t-1 시간 사이에 이벤트 중계자가 수신했던 컨텍스트 변화 이벤트들의 개수의 평균값을 구하고, 이 값으로 현재 단위 시간 동안 수신한 컨텍스트 변화 이벤트 수를 나눈다. 이 값을 토대로 현재 컨텍스트 변화 이벤트의 발생율을 계산하여 그 비율만큼 정확도를 감소시킨다. 이 경우 β 를 0과 1 사이의 적절한 값으로 설정하여 컨텍스트 변화 이벤트 발생 건수 당 정확도의 감소량을 조절한다.

4. 비교 분석

위에서 제안한 아이디어를 [4] 논문에서 제시한 환경에 적용하여

비교해보았다. 그림 1의 컨텍스트 전이 그래프를 [4]에서 제시한 loc 함수를 통해 예상 가능한 컨텍스트 값을 계산해보면 그 결과가 다음과 같다.

loc(FrontDoor, 1) = {FrontDoor}
loc(FrontDoor, 2) = {FrontDoor, Livingroom, Dinningroom, Bedroom, Meeting table, Bathroom}

이와 같이 2 단위 시간 이후에는 환경 내에 모든 위치가 예상 가능한 컨텍스트 값으로써 반환된다. 따라서 애드혹 유비쿼터스 컴퓨팅 환경에서 Subscriber의 바로 이웃 노드는 모든 위치에 존재하는 Publisher로부터 발생한 이벤트를 모두 수신하고 Subscriber에서 필터링이 수행됨을 알 수 있다. 이러한 결과는 loc함수 및 이동 그래프가 각 컨텍스트 값들 사이의 관계에 예상되는 가중치를 고려하지 않았기 때문이다. 반면 그림 1의 컨텍스트 전이 그래프와 아래 그림 2의 컨텍스트 값 사이의 연결에 가중치 정보를 토대로 본 연구에서 제안한 ret함수를 이용할 경우 결과는 다음과 같다.

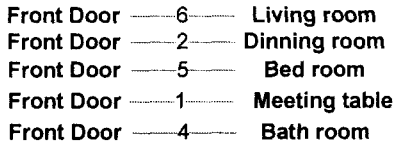

그림 2. 컨텍스트 값 사이 연결과 가중치 그림

ret(Location, FrontDoor, 1, 1) = {FrontDoor}
ret(Location, FrontDoor, 2, 1) = {FrontDoor, Meeting table}

첫 번째 예에서 현재 위치는 FrontDoor이고 2 단위 시간 후의 예상 가능한 위치를 계산하였을 때 결과는 {FrontDoor, Meeting table}이 된다. 이와 같은 결과는 컨텍스트 값 사이의 연결에 두 컨텍스트 값의 관계를 수치화 하여 가중치를 부여하고 이를 이용해 단위 시간 후 예상 컨텍스트 값을 계산하였기 때문으로 해석할 수 있다. 따라서 Subscriber의 바로 이웃 노드가 수신하는 이벤트의 수가 기존 연구에 비해 크게 줄어듦을 알 수 있다.

또한 컨텍스트 전이 그래프 조회 함수에 정확도라는 인자를 추가하여 컨텍스트의 종류나 애플리케이션의 특성, 혹은 주변 환경에 따라 정확도의 값을 조절하여 동적으로 예상 컨텍스트 값들 재계산 할 수 있도록 하였다. 다음은 정확도 값이 달라짐에 따라 예상되는 컨텍스트 값들의 변화이다.

ret(Location, FrontDoor, 2, 1) = {FrontDoor, Meeting table}
ret(Location, FrontDoor, 2, 0.75) = {FrontDoor, Meeting table, Dinning room}
ret(Location, FrontDoor, 2, 0.5) = {FrontDoor, Meeting table, Dinning room, Bathroom}

이와 같이 정확도의 값을 변경함에 따라 예상 컨텍스트 값 또한 달라짐을 알 수 있다. 따라서 컨텍스트 변화의 정도가 서로 다른 환경이나 애플리케이션, 컨텍스트 종류에 따라 서로 다른 정확도 값을 이용하여 컨텍스트 인지 상호작용을 지원하기 위해 중간 이벤트 중계 노드들이 미리 수신해야 하는 이벤트의 수를 조절할 수 있게 되었다.

5. 결론 및 향후 계획

본 연구에서는 애드혹 유비쿼터스 컴퓨팅 환경에서 기존 연구들이 가진 문제점으로써 컨텍스트의 종류가 다양하고 변화가 잦은 환경에서 컨텍스트 인지 상호작용을 지원하는데 있어 발생하는 오버헤드를 지적하였다. 이를 해결하기 위해 컨텍스트 인지 상호작용을 지원하는데 있어 이벤트 중계의 정확성과 컨텍스트 변화 적응성간의 트레이드 오프 관계가 있음을 설명하였으며, 컨텍스트의 종류나 애플리케이션 또는 환경에 따라 이벤트 중계의 정확성을 통해 컨텍스트 변화 적응성을 조절하여 이러한 오버헤드를 최소화 시킬 수 있는 방법을 제시하였고 현재 구현이 진행중이다.

향후 연구로써 본 연구에서 제시한 개념을 위치뿐만 아니라 다양한 컨텍스트 종류들을 포함할 수 있도록 확장할 계획이며, 또한 위에서 제시한 이벤트 중계의 정확성을 동적으로 계산하는데 있어 효율성을 높이기 위해 적절한 상수값들에 대해 연구할 계획이다.

7. 참고문헌

[1] T. Kindberg, and A. Fox, "System Software for Ubiquitous Computing", IEEE Pervasive Computing, 2002

[2] P. Eugster, P. Felber, R. Guerraoui, and A.-M. Kermarrec. "The Many Faces of Publish/Subscribe". ACM Computing Surveys, 2003.

[3] S. Chetan, J. Al-Muhtadi, R. Campbell, and M.D. Mickunas, "Mobile Gaia: A Middleware for Ad-hoc Pervasive Computing", Consumer Communications and Networking Conference, IEEE, 2005

[4] L. Fiege, F.C. Gartner, O. Kasten, and A. Zeidler. "Supporting Mobility in Content-based Publish/Subscribe Middleware". In Proc. of the 4th ACM/IFIP/USENIX Int. Middleware Conference, 2003.

[5] R. Meier and V. Cahill. "STEAM : Event-based Middleware for Wireless Ad hoc Network", In ICDCSW '02: Proceedings of the 22nd International Conference on Distributed Computing Systems, 2002. IEEE

[6] M. Musolesi, C. Mascolo, and S. Hailes. "Adapting asynchronous messaging middleware to ad hoc networking", In Proceedings of the 2nd workshop on Middleware for pervasive and ad-hoc computing, ACM Press, 2004