개인 지능형 공간에서의 상황정보 추론을 위한 작업 분배 기법*

A Task Decomposition Scheme for Context Aggregation in Personal Smart Space

유호석, Hoseok Ryu*, 박인석, Insuk Park*, 현순주, Soon J. Hyun*, 이동만, Dongman Lee*, 김정선, Kim Jeong Seon **

* 한국정보통신대학교 공학부, ** KT 네트워크 인프라 연구소

요약 상황 인지 컴퓨팅에서 상황정보 추론 기능은 상황정보 관리를 위해 중요한 기능 중의 하나이다. 상황정보 추론 기능은 하위 레벨의 상황정보들로부터 사용자의 상황을 나타내는 상위 레벨의 상황정보를 제공한다. 인프라 기반 지능형 공간에서 중앙 집중 형의 상황정보 관리 시스템은 상황정보 추론을 위한 자원 소모를 고려할 필요가 없었다. 하지만 자원이 제약된 장치들로만 구성된 개인 지능형 공간에서는 공간 내 전체의 자원 소모뿐만 아니라 상황정보 관리자 역할을 하는 장치 (coordinator)들의 자원 소모가 최소화 되어야 한다. 본 논문에서는 중앙 집중적인 상황정보 추론 작업을 분배하여 개인 지능형 공간 내의 다른 장치들에게 작업을 분산시키는 상황정보 추론 작업 분배 기법을 제안한다. 제안된 분배 기법은 건강정보, 환경정보, 지리정보 같이 상황정보가 자주 발생하는 환경에서 더 효율적이다. 상황정보 추론작업을 분배 함으로써 상황정보 추론을 위한 개인 지능형 공간의 전체의 처리량을 크게 증가시키지 않으면서 코디네이터의 처리량을 줄일 수 있다. 본 논문의 작업분배 기법은 상황정보 추론의 역할을 하는 코디네이터와 분산된 로컬 상황정보 추론기능을 제안한다. 본 논문에서는 제안된 상황정보 추론 기능을 개인 지능형 공간을 구성하는 장치들에 각각 구현하고 상황정보 추론을 위한 처리부하를 측정하여 제안된 기법의 실행 가능성을 보였다.

핵심어: Ubiquitous computing, Context awareness, Personal smart space, Context aggregation, Task distribution

1. 서론

상황정보 관리 시스템들은 지능형 홈이나 지능형 오피스 같은 인프라 기반의 지능형 공간에서 상황정보 서비스를 제공한다. 상황정보 관리 시스템은 다양한 상황정보 제공자들로부터 상황정보를 수집하고, 처리하여 상황정보에 관심 있어하는 서비스들에게 상황정보를 전달한다. 상황 인지 서비스들은 상황정보 관리 시스템으로부터 상황정보를 요청 하거나 상황정보의 변화에 대하여 반응하게 된다.

최근, 사용자들이 여러 개의 모바일 장치들을 가지게 되면서 장치들끼리 개인 지능형 공간 이라는 새로운 지능형 공간을 구성하게 되었다 [2]. 개인 지능형 공간에서는 자원이 제약적인 모바일 장치들이 때에 따라 서로 연결되며 사용자에게 상황인지 서비스를 제공하기 위해 서로 상호작용한다. 하나의 개인 지능형 공간은 하나의 코디네이터 (coordinator) 장치와 하나 이상의 클라이언트 (client)장치들로 구성된다. 각각의 장치는 다양한 센서들과 그에 따르는 상황정보 제공자 (context provider)들을 포함한다. 클라이언트에 비해 비교적 풍부한 자원들을 가지는 코디네이터는 규칙 기반의 상황정보 추론 기능, 상황정보관리, 상황정보 제공자 관리 등의 추가적인 기능들을 제공한다.

로직 기반의 추론 같은 기존의 상황정보 추론 기법들은 많은 자원의 소모를 필요로 한다 [4], [5]. 만약 코디네이터가 상황정보 추론을 위하여 모든 기능을 담당한다면 코디네이터의 상황정보 추론의 부하를 증가시켜 코디네이터의 중요한 자원중의 하나인 배터리가 고갈되는 상황이 발생 할 수 있다. 그 결과 다른 모바일 장치들이 동작함에도 불구하고 개인 지능형 공간의 서비스들을 더 이상 유지할 수 없게 된다.

추론의 부하를 감소시키기 위한 방법으로 상황정보 추론 기능을 분산시키는 다양한 접근방법들이 제안되었다. 상황정보 추론 기능을 분산 시킴으로 EDCI는 추론을 이용한 기법에 비하여 상황정보 추론 처리에 걸리는 시간을 감소시키고 Solar 에서는 상황정보 제공자의 재사용성을 증가시킨다 [6], [7]. 하지만 위의 접근 방법들은 자원이 제약적이고 모바일 에드혹 환경을 고려하지 못하였다. 또한 Contory 는 자원이 제약적인 모바일 환경에서의 상황정보 제공을 위한 미들웨어를 제안하였다 [3]. 하지만 이 연구에서도 코디네이터에 집중되는 상황정보 추론을 위한 자원의 소모에 대해 다루지 못하고 있다.

본 논문에서는 하나의 상황정보 추론작업을 상황정보 제공자의 배치에 따라 여러 개의 부분 작업으로 나누어 여러 장치들에 분산시키는 효율적인 상황정보 추론 기법을 제안한

* 본 연구는 KT-ICU 공동 연구 센터와 정보통신부 및 정보통신연구진흥원의 디지털미디어연구소 지원사업의 지원에 의한 것임

다. 제안된 기법은 같은 장치에 있는 상황정보 제공자들이 제공하는 상황정보를 부분적으로 추론 함으로써 하나의 코디네이터에 부하가 집중되는 것을 방지한다. 이러한 접근방법은 상황정보 추론의 부하를 나머지 클라이언트장치들에 분산시켜 코디네이터에 집중되는 부하를 줄일 수 있으며, 불필요한 통신횟수를 줄일 수 있다. 실험결과는 상황 정보 추론을 위한 개인 지능형 공간의 전체 처리량을 크게 증가시키지 않으면서 코디네이터의 처리량을 50% 가량 줄여주는 것을 보여준다.

본 논문의 나머지 부분은 다음과 같이 구성된다. 2장에서는 연구 동기와 예제 시나리오를 설명한다. 그리고 3장에서는 개인 지능형 공간을 위한 상황정보 관리 구조와 제안된 분배 기법을 설명한다. 4장에서는 제안된 기법의 상세 구현을 설명하고 있다. 5장에서는 우리 접근방법의 성능평가를 보여준다. 6장에서는 관련연구를 소개하고 마지막으로 7장에서는 결론을 맺는다.

2. 동기

본 장에서는 상황 인지 서비스 들을 가능하게 하기 위한 개인 지능형 공간의 기술적인 특징들을 소개한다. 개인 지능형 공간의 대표적인 특징은 단말기들간의 무선 에드혹 연결성과 단말기들의 자원 제약성이다. 본 장에서는 개인 지능형 공간에 대한 소개와 운동 시나리오에서 본 논문에서 제안하는 기법을 필요로 하는 두 가지 예제 상황을 보여준다.

2.1 개인 지능형 공간

무선 이동 컴퓨팅 기술의 발달하면서 한 명의 사용자를 중심으로 하는 개인화된 네트워크를 형성하는 개인 지능형 공간의 개념이 등장하게 되었다. 개인 지능형 공간에서는 개인의 장치들간에 서비스들과 자원들을 공유하는 것을 필요로 한다. 기존의 연구 중에서 개인 지능형 공간의 개념으로서 Mobile Gaia가 있다 [2]. Mobile Gaia에서는 하나의 코디네이터장치를 중심으로 에드혹 네트워크를 형성한다. Mobile Gaia에서는 장치들의 역할이 코디네이터(coordinator)나 클라이언트(client) 두 가지로 정해진다. 코디네이터는 클라이언트보다 더 많은 기능을 담당한다. Mobile Gaia에서는 하나의 디바이스가 코디네이터로 결정될 때 장치들의 역할이 결정되고, 이에 따라 필요한 서비스 컴포넌트들만 해당 장치에서 실행된다. 이를 위해, Mobile Gaia는 코디네이터를 선정하는 메커니즘과 동적으로 컴포넌트를 로드하는 메커니즘을 제공한다. 하지만 Mobile Gaia는 상황 정보 서비스에 대하여는 장치들의 역할에 따른 컴포넌트들을 정의하지 않고 있다. 따라서, 상황 정보 서비스는 코디네이터를 중심으로 중앙 집중적인 방식으로 제공된다. 중앙 집중 방식의 상황정보 관리는 상황정보 추론 규칙을 만족시키는지 확인하기 위하여 코디네이터에 처리 부하를 집중시킨다.

본 논문의 개인 지능형 공간에서는 사용자가 MP3 플레이어나 지능형 시계 또는 다른 지능형 장치들을 휴대하거나 몸에 지내고 다닌다. 각 장치들의 역할은 코디네이터와 클라이언트에 대하여 각각 정해진다. 클라이언트와 코디네이터는 둘 다 하나 또는 그 이상의 상황정보 제공자와 다양한 서비스들을 포함한다. 코디네이터는 위의 기본적인 기능들뿐만 아니라 규칙 기반의 상황정보 추론 기법, 상황정보 및 상황정보 제공자 목록을 관리하는 등 추가적인 기능을 담당한다. 그림 1은 개인 지능형 공간에서 제공되는 상황정보들의 예를 보여주고 있다. 핸드폰에서의 light, vibration, noise, 지능형 시계에서의 sweat, temperature, pulse, blood pressure 등은 센서로부터 얻어지는 상황정보들이다. 그리고 user status는 규칙으로부터 추론 되는 상위 레벨의 상황정보이다. 마지막으로 PDA에 있는 user preference나 schedule정보는 장치에 미리 저장되어 있는 상황정보이다.

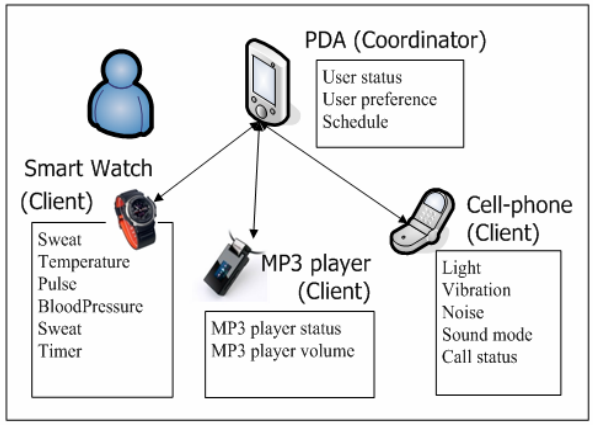

그림 1. 개인 지능형 공간에서의 상황정보 예제

2.2 예제 시나리오

우리는 개인 지능형 공간에서의 운동 시나리오를 바탕으로 두 가지 예제 상황을 제시한다. 운동 시나리오에서는 김씨가 핸드폰, 지능형 시계, MP3 플레이어, PDA등의 개인화 장치들을 지니고 다닌다. 각각의 장치들은 그림 1처럼 다양한 센서들과 그 센서들의 정보를 상황정보로 사용하기 위한 상황정보 제공자들을 가진다. 이 시나리오에서의 두 가지 상황은 다음과 같다.

김씨는 MP3 플레이어로 큰소리로 음악을 들으며 운동을 하고 있다. 그리고 그때 김씨의 전화벨이 울린다. 전화벨이 울릴 때 MP3 플레이어의 볼륨이 70이상이라면 김씨는 듣고 있는 음악소리 때문에 전화가 온 것을 알아차리지 못할 수도 있다. 이러한 상황은 김씨가 전화를 놓치게 만들 수도 있다. 그러므로 전화벨이 울릴 경우 PDA의 상황인지 애플리케이션은 김씨의 선호도 정보에 따라 MP3 플레이어의 볼륨을 줄이거나 MP3 플레이어를 잠시 멈춘다.

또한 운동하는 가운데 김씨의 맥박이나 혈압이 김씨의 정상적인 수치를 넘어서는 경우(맥박의 경우 140 pulse rate, 혈압의 경우 160 mmHg)가 발생한다. 만약 김씨의 상황이 정상적인 수치를 넘어서고 김씨가 땀을 많이 흘리고 있다면 김씨는 무리해서 운동하는 상태에 있다. 그러므로 김씨의 PDA는 김씨에게 현재의 운동량을 조절하는 경고메시지와 함께 현재의 건강 정보들을 보여준다.

위의 두 가지 상황을 가능하게 하기 위해서는 두 가지의 상황정보 추론 규칙과 ECA규칙이 필요하다. PDA의 상황정보 추론자(Context Aggregator)에 있는 상황정보 추론 규칙

과 상황 인지 애플리케이션 안에 들어 있는 ECA 규칙은 표1과 2에 각각 나타낸다.

표 1. 운동 상태 상황정보 추론 규칙들

Condition rules:	
a) CallStatus(Kim, Ringing) ^	cell-phone
b) hasVolume(MP3 player, over 50) ^	MP3 player
c) Status(MP3 player, On) ^	MP3 player
d) -> Status(Kim, InterruptableCalling)	PDA
Condition rules:	
a) Vibration(Kim, Running) ^	cell-phone
b) Pulse(Kim, Over 140) ^	smart watch
c) BloodPressure(Kim, Over 160) ^	smart watch
d) Sweat(Kim, Wet) ^	smart watch
e) -> Status(Kim, OverRunning)	PDA

표 2. 운동 보조 애플리케이션의 ECA 규칙들

```
On(Status(Kim, InterruptableCalling))
  If(true)
    Do( start(service set1) )
      -> Service set1 = {
          Pause the music play of MP3 player
          according to Kim's preference          };
```

```
On( Status(Kim, OverRunning))
  If(true)
    Do( start(service set2) )
      -> Service set2 = {
          Alert to Kim for adjusting amount of exercise,
          Show the health information on PDA     };
```

위에서의 두 가지 상황정보 추론 규칙처럼 각 장치들은 하나의 상황정보 추론 작업을 위한 부분적인 규칙들을 포함하게 된다. 이러한 경우 중앙 집중 형의 상황정보 추론은 상황인지 서비스를 지연시키거나 코디네이터에 처리 부하가 집중되어 코디네이터가 중단되게 할 수 있다.

3. 개인 지능형 공간 상황정보 관리 구조

본 장에서는 개인 지능형 공간을 위한 디자인 고려사항과 상황정보 관리 구조를 제시한다. 기존의 지능형 공간과는 달리 개인 지능형 공간에서의 상황정보 추론은 작은 처리 부하와 작은 네트워크 부하 그리고 동적으로 변화하는 환경에 대한 유연성을 필요로 한다. 이러한 요구사항들을 만족시키기 위하여 상황정보 추론 작업이 분산되어야 하며 상황정보 이벤트 subscription 그래프가 동적으로 변할 수 있어야 한다. 그러므로 하나의 상황정보 추론 작업이 네트워크 상황에 따라 여러 개의 작은 부분 작업으로 나누어져야 한다. 그리고 개인 지능형 공간의 상황정보 관리 구조는 상황정보 제공자의 배치 상태에 따라 상황정보 추론 작업을 분배하는 메커니즘을 제공한다.

3.1 설계 고려사항

자원이 제약적인 개인 지능형 공간에서의 상황 정보 수집 기능을 제공하기 위해서는 해결해야 할 요구사항들이 있다. 첫째, 자원 제약적인 모바일 장치 들의 상황 정보 수집기능은 작은 처리 부하를 가져야 한다. 둘째, 수집 처리 기능은 또한 코디네이터 장치에 처리부하가 집중 되어 코디네이터가 정상적으로 동작 하지 않는 경우를 피할 수 있도록 해야 한다. 셋째, 네트워크 부하를 최소화 하기 위하여 필요 없는 네트워크 전송을 최소화 해야 한다. 넷째, 동적으로 변화하는 네트워크의 특징은 다양한 시간과 공간에서 동적으로 상황정보 이벤트 subscription을 유연하게 바꾸어 줄 수 있는 메커니즘을 필요로 한다.

이러한 요구사항들을 만족시키기 위하여 하나의 상황정보 추론 작업이 네트워크 환경이 변할 때 마다 이용 가능한 상황정보 제공자들을 고려하여 여러 개의 부분 작업으로 분배하여 효율적인 상황정보 추론 기능을 제공해야 한다. 추론 작업의 분해는 같은 장치에 있는 상황정보 제공자들이 제공하는 상황정보를 로컬에서 추론할 수 있도록 한다. 또한 모든 상황을 고려한 부분 추론 작업들을 모두 정의 하는 데는 어려움이 있기 때문에 네트워크 변화에 동적으로 추론 작업을 분배 할 수 있는 기능을 제공한다.

3.2 시스템 구조

상황정보 관리를 위한 기본적인 기능 들은 상황정보를 모으고, 추론하며, 전달하는 것이다. 우리는 상황정보 관리 미들웨어를 위하여 다섯 가지 컴포넌트들을 정의 하였다. **상황정보 위젯 (context widget)**은 센싱 정보들을 추상화 하여 상황정보의 형태로 나타낸다. **상황정보 추론자 (context aggregator)**는 상황정보 추론 규칙에 따라 여러 하위 레벨의 상황정보로부터 상위 레벨의 상황정보를 제공한다. 우리의 구조에서는 작은 처리부하를 제공하기 위하여 단순한 복합 이벤트 감지 기법을 이용하여 추론 기능을 대신한다 [11]. **상황정보 해석자 (context interpreter)**는 사용자가 관심 있어 하는 상황정보를 감지하여 애플리케이션에 알려주는 역할을 한다. **상황 인지 애플리케이션 (context aware application)**은 상황정보를 이용하여 ECA 규칙을 가지고 상황정보에 따라 적절한 서비스를 제공한다. **상황정보 관리자 (context manager)**는 상황정보의 저장소 역할을 하며 최소한의 온톨로지 정보를 가진다.

개인 지능형 공간에서의 작업 분배 기법과 동적인 네트워크 환경에 대한 유연성을 제공하는 효율적인 상황정보 추론 기능을 제공 하기 위해서는 위의 컴포넌트에 추가적인 컴포넌트들을 필요로 한다. 추가 적인 컴포넌트들의 세부 내용들은 다음과 같다.

상황정보 레지스트리 (Context Registry)는 상황정보 제공자 들의 목록을 관리하고 모든 상황정보 제공자들은 자신을 상황정보 레지스트리에 등록한다. 그리고 필요에 따라 상황정보의 이름과 형태를 가지고 원하는 상황정보 제공자를 찾을 수 있는 기능을 제공한다.

분배 관리자 (Decomposition Manager)는 새로운 상황정보 제공자가 개인 지능형 공간에 새롭게 들어올 때 상황정보 추론 작업의 분배 수행을 요청한다. 그리고 작업 분배 알고리즘을 적용하여 상황정보 이벤트 subscription 트리와 부분 작업 규칙 트리를 만들어낸다. 두 가지의 트리가 구성 되면 분배 관리자는 부분 작업 규칙을 로컬 추자에게 전달하고 복합 이벤트 규칙을 상황정보 추론자에 각각 전달한다.

로컬 추론자 (Local Aggregator)는 분배관리자를 통하여 만들어진 작은 부분 추론 작업 규칙들을 등록한다. 로컬 추

론자는 등록된 부분 추론 작업의 규칙이 만족되는 것을 발견한다. 그리고 그 상황정보를 원하는 상위 레벨의 상황정보 추론자에게 부분적으로 만족된 상황정보를 알려준다. 개인 지능형 공간내의 모든 장치 들은 하나의 로컬 추론자를 가지고 있다.

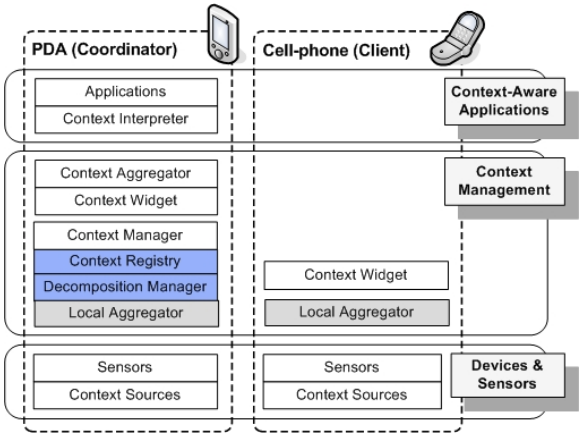

그림 2. 상황정보 관리 구조

그림 2는 개인 지능형 공간에서 장치들의 역할에 따른 상황정보 관리 구조를 보여주고 있다. 코디네이터는 상황정보 관리를 위한 모든 컴포넌트를 포함할 수 있으며 클라이언트의 경우에는 자원의 제약 때문에 상황정보 위젯과 로컬 추론자만을 포함할 수 있다.

3.3 분배 알고리즘

작업 분배 알고리즘은 상황정보 추론자의 이름을 입력 값으로 사용한다. 개인 지능형 공간에서는 여러 개의 상황정보 추론자가 존재할 수 있다. 하나의 상황정보 추론자는 하나 또는 그 이상의 상황정보 추론 규칙을 가지고 있다. 개인 지능형 공산에 새로운 상황정보 제공자가 등장하거나 상황정보 추론자가 작업 분배를 요청할 때 분배관리자는 상황정보 추론자로부터 추론 규칙을 받아온다. 그리고 상황정보 레지스트리로부터 개인 지능형 공간에 현재 이용 가능한 장치들의 목록을 받아온다. 이 알고리즘에서는 두가지 종류의 트리를 생성한다. 하나는 상황정보 이벤트 subscription 트리이고 또 다른 하나는 부분 작업을 위한 규칙 트리이다. 분배 알고리즘에서는 먼저 상황정보 추론자의 이름으로 상황정보 이벤트 subscription 트리의 루트 노드를 생성한다. 그리고 각각의 장치들의 이름들을 새로운 노드들로 추가한다. 그리고 각각의 상황정보 추론 규칙에 대하여 다음과 같은 작업을 수행한다.

하나의 상황정보 추론 규칙을 위해서 분배 알고리즘은 상황정보 추론 규칙의 결과 규칙을 루트 노드로 하여 부분 작업 규칙 트리를 초기화한다. 그리고 상황정보 제공자의 주소 값을 가지고 부분 작업 규칙을 생성한다. 각각의 부분 작업 규칙은 RDF triple형태의 결과 규칙을 가진다. 결과 규칙은 SubTaskRule(ip, random value)와 같이 상황정보 제공자의 주소 값을 주어로 가지고 랜덤 값을 목적어로 가진다. 예를 들면 210.107.250.173의 주소 값을 가지는 장치에서 SubTaskRule(210.107.250.173, 24212)과 같은 부분 작업 규칙이 만들어진다. 부분 작업 규칙이 만들어지면 알고리즘은 부분 작업 규칙 트리에 부분 작업의 결과 규칙의 값을 가지는 새로운 노드를 추가한다. 그리고 상황정보 추론 작업 규칙의 조건 규칙 들을 확인한다.

조건 규칙의 상황정보를 제공하는 상황정보 제공자를 찾기 위해서 상황정보의 이름과 형태를 가지고 상황정보 제공자를 찾고 상황정보 레지스트리로부터 주소 값을 가져온다. 그리고 상황정보 제공자 노드를 상황정보 이벤트 subscription 트리에 추가하고 조건 규칙 노드를 부분 작업 규칙 노드에 추가한다. 그리고 하나의 상황정보 추론 규칙에 대하여 알고리즘은 부분 작업 규칙 테이블에 부분 작업 규칙 트리를 저장한다.

위의 과정을 모두 마치면 모든 상황정보 추론 작업 규칙들에 대한 부분 작업 규칙들이 생성되고 부분 작업 규칙 테이블에 저장된다. 그리고 상황정보 추론자를 위한 상황정보 이벤트 subscription 트리가 생성된다. 부분 작업 규칙 트리와 상황정보 이벤트 subscription 트리의 예제가 그림 3과 4에 각각 나타나있다. 그림 3은 과도한 운동 상태를 위한 부분 작업 규칙 트리를 보여준다. 그림 3에서는 하나의 복합 이벤트 규칙과 두 개의 부분 작업 규칙이 있다. 복합 이벤트 규칙은 상황정보 추론자에 있으며 SubTaskRule (210.107.250.80, 24212) ^ SubTaskRule (210.107.250.173, 18721) -> Status (Kim, OverRunning)로 표현된다. 그리고 두개의 부분 작업 규칙들은 각각 Sweat (Kim, Wet)^Pulse (Kim, Over 140)^BloodPressure (Kim, Over 160) -> SubTaskRule (210.107.250.180, 24212) 과 Vibration (Kim, Running) -> SubTaskRule (210.107.250.173,18721)로 표현된다.

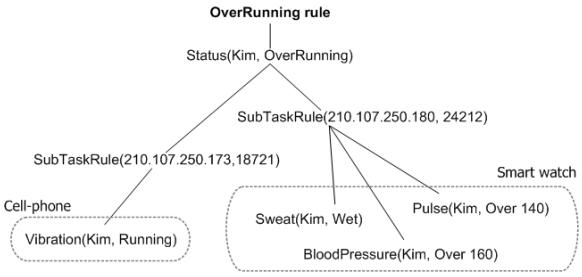

그림 3. 과도한 운동 상태의 부분 작업 규칙 트리

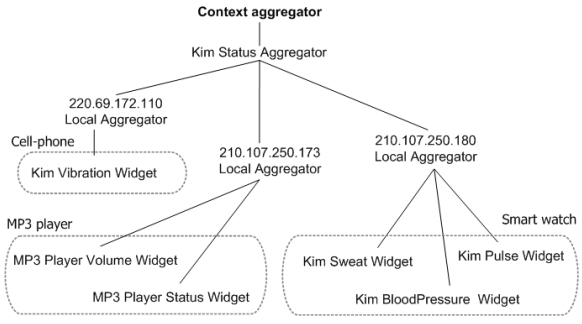

그림 4. 상황정보 이벤트 subscription 트리

표 3은 작업 분배를 위한 알고리즘을 보여준다. 이 알고리즘에서 우리는 상황정보 레지스트리가 같은 종류의 상황정보를 제공 할 경우 가장 적합한 상황정보 제공자를 선택할 수 있는 기능을 가지고 있다고 가정하고 있다.

표 3. 작업 분배 알고리즘

```
Operations
  getDeviceList(): get device list in space from context registry
  getAggregationRules(): get aggregation rule set
  initTree(tree): init subscription tree
  getConditionRules(): get condition rules from aggregation rule
  getReferenceOf(prvName): get reference of a context provider
  generateSubTaskRule(): generate sub task rule
       with reference of context provider and random value
  AddsubTaskRuleToSubRulesTable():Add sub task rules to
    SubRulesTable;

Function Decomposition(aggName) returns void

Inputs:
  aggName, aggregator name

Variables:
  aggRules, aggregation rules in an aggregator
  deviceList, list of devices in personal smart space
  subscriptionTree: subscription tree for decomposition
  subTaskRulesTable: list of sub task rules
  subTaskRuleTree: sub task rule tree
  cRules: conditions rules for an aggregation rule

deviceList = getDeviceList()
aggRules = getAggregationRules()
subscriptionTree.initTree(aggregatorName)
rootNode=subscriptionTree.getRootNode()
rootNode.addNodes(deviceList)

for  i, 1 to  aggRules.size()
{
  subTaskRuleTree.initTree(resultRule[i])
  resNode= subTaskRuleTree.getRootNode()
  subtaskResultRule = generateSubTaskRule();
  resNode.addNodes(subtaskResultRule)

   conRules[i] = getConditionRules()
   for j, 1 to cRules.size())
   {
      provider=lookup(cRule[j].cName,cRules [j].cType)
      ref = getReferenceOf(provider)
      subscriptionNode = subscriptionTree.search(ref)
      subTaskRuleNode = subTaskRuleTree.search(ref)
      subscriptionNode.addNode(provider)
      subTaskRuleNode .addNode(cRules[i])
   }
   Add subTaskRuleToSubRulesTable()
}
```

4. 구현

우리는 제안된 상황정보 관리 구조를 Active Surroundings이라는 유비쿼터스 컴퓨팅 미들웨어의 한 부분으로 구현하였다 [10]. Active Surroundings의 상황정보 관리 컴포넌트들은 PDA에 구현하였으며 CDC를 지원하는 IBM J9 에서 실행시켜 보았다. 본 장에서는 우리가 제안한 구조에서의 컴포넌트들간의 상호 작용을 보여주고 2장에서 설명한 과도한 운동 상황의 실행 순서를 설명한다.

4.1 컴포넌트 간의 상호 작용

그림 5는 하나의 상황정보 추론 작업을 여러 개의 부분 작업으로 나누기 위한 컴포넌트간 상호작용을 보여주고 있다. 컴포넌트 간의 상호 작용은 두 가지 단계로 나뉜다. 첫 번째는 상황정보 제공자들을 등록하는 단계이고 두 번째는 상황정보 추론 작업을 분배하는 단계이다. 등록 단계에서는 모든 상황정보 위젯과 추론자들 그리고 각 장치의 로컬 추론자를 상황정보 레지스트리에 등록한다. 분배 단계에서는 분배 관리자가 분배 알고리즘을 수행하여 하나의 상황정보 추론 작업을 여러 개의 장치들로 분산시킨다. 컴포넌트 간의 전체 상호작용은 다음과 같다.

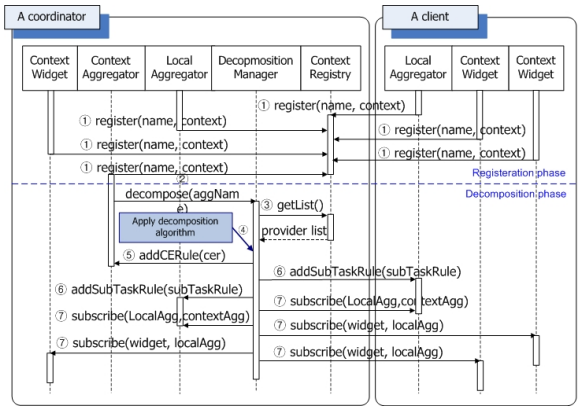

그림 5. 작업 분배를 위한 컴포넌트 간 상호작용

① 모든 상황정보 위젯과 추론자들 그리고 각 장치의 로컬 추론자들을 상황정보의 이름과 종류와 함께 상황정보 레지스트리에 등록한다.

② 등록된 상황정보 추론자는 분배 관리자에게 작업의 분배를 요청한다.

③ 분배 관리자는 상황정보 레지스트리로부터 상황정보 제공자들의 목록과 그 주소 값을 받아온다.

④ 분배 관리자는 상황정보 제공자의 배치를 고려하여 상황정보 이벤트 subscription 트리와 부분 작업 추론 규칙 테이블을 구성한다.

⑤ 분배 관리자는 복합 이벤트 규칙을 상황정보 추론자에게 전달한다.

⑥ 분배 관리자는 부분 작업 규칙을 로컬 추론자에 전달한다.

⑦ 분배 관리자는 subscription 트리를 따라 정해진 로컬 추론자에 부분 작업 규칙을 구독한다.

다른 상황정보 제공자가 나타나게 되면 분배 관리자는 분배 알고리즘을 수행하게 된다. 그리고 위의 과정을 반복하게 된다.

4.2 예제 시나리오를 위한 실행 순서

우리는 앞에서 제시한 운동 시나리오를 어떻게 구현했는지를 보여준다. 상황 인지 서비스를 제공하기 위해서는 각각의 애플리케이션은 관심 있는 상황정보를 그림 3과 같이 구

독해야 한다. 일단 상황 인지 애플리케이션이 관심 있는 상황정보들을 구독하면 애플리케이션은 상황정보에 반응하여 실행될 수 있다. 상황정보 관리 시스템은 상황정보 subscription 트리를 따라서 상황정보를 제공한다. 예를 들면 과도한 운동 상태 예제의 경우 vibration, pulse, blood pressure 위젯들은 센싱 정보들을 추상화하여 상황정보 형태로 제공하고 상황정보가 변화할 경우 각 장치의 로컬 추론자는 부분 작업 규칙이 만족 되었는지를 확인한다. 그리고 부분 작업 규칙이 만족 되면 로컬 추론자는 다른 장치에 있는 사용자 상태 추론자에게 상황정보를 전달한다. 사용자 상태 추론자는 사용자가 과도한 운동상태인지를 확인하고 조건이 만족되었을 경우 과도한 운동상태 해석자가 상황이 만족되었음을 인지하여 상황 인지 애플리케이션인 건강 정보 뷰어에 운동량을 조절하라는 경고메시지와 함께 현재의 건강 정보를 보여준다. 그림 6에서는 위의 예제 상황의 실행 순서를 보여주고 있다.

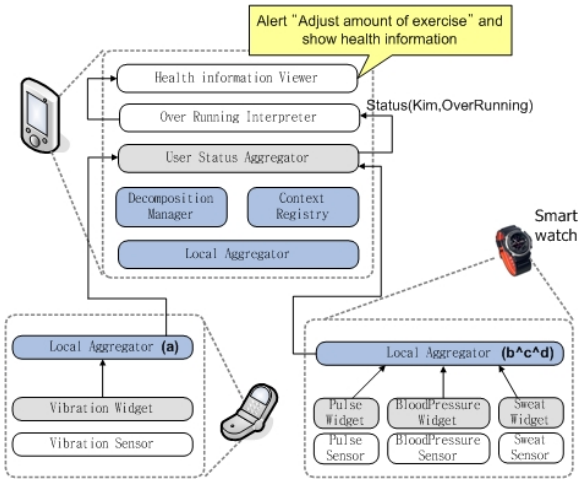

그림 6. 과도한 운동상태 예제 실행 순서

5. 성능 평가

본 장에서는 우리가 제안한 작업 분배 기법이 얼마나 코디네이터에 집중되는 처리부하를 줄일 수 있는지를 보여준다. 우리는 작업 부하를 분산시킴으로 집중된 상황정보 추론 처리 부하를 개인 지능형 공간에 있는 여러 디바이스들에게 나누어 줄 수 있다는 것을 예상한다. 이러한 예상을 증명하기 위하여 우리는 제안한 시스템을 가지고 실험을 하였다. 우리의 실험에서는 코디네이터와 클라이언트에서의 처리 시간을 각각 측정한다. 그리고 그 결과를 분산시키지 않은 기존의 방법과 처리 시간을 비교한다.

우리의 실험에서는 각각의 상황정보 위젯들이 난수 값을 가지는 100개의 상황정보 이벤트를 1초에 하나씩 생성한다. 그리고 세가지 상황에 대하여 비교한다. 하나는 기존의 방법이며 나머지 둘은 우리가 제안한 방법이다. 우리가 제안한 방법에서는 개인 지능형 공간에서의 전체 처리 시간과 코디네이터 측면에서의 처리 시간을 각각 측정한다. 전체 처리 시간은 클라이언트들 중의 최대 처리 시간과 코디네이터의 처리 시간의 합으로 만들어낼 수 있다. 전체 처리 시간을 구하기 위한 공식은 다음과 같다.

$$PT(total) = PT(coordinator)+\max(PT(client1),PT(client2)) \quad (1)$$

우리의 실험 환경으로서는 두 개의 PDA를 사용하였다. 사용한 PDA는 HP rx3715(Processor speed: 400MHz, Installed RAM: 152MB)이며 운영체제는 Microsoft Windows Mobile Pocket PC 2003이다. 우리가 사용한 JVM은 CDC를 지원하는 IBM J9이다.

위의 실험으로 우리는 처리부하 측면에서의 성능 향상을 보여줄 수 있다. 그림 7은 위의 세가지 상황에 따른 상황정보 추론 처리 시간을 비교한 것을 보여준다. 그림에서 보여지는 것처럼 실험 결과는 세가지로 나타난다. PT(prev coor)는 기존의 접근방법에서의 처리 시간을 PT(coor)와 PT(total)은 코디네이터에서의 처리 시간과 전체 처리 시간을 각각 나타낸다.

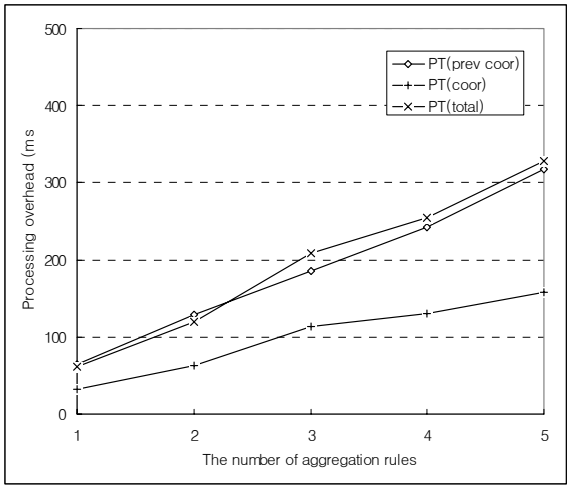

그림 7. 상황정보 추론 처리 시간 비교

PT (prev coor)과 PT (total)는 비슷한 처리시간을 보여준다. 하지만 코디네이터 측면에서의 PT (coor)는 기존의 접근방법과 비교하였을 때 반정도의 처리 시간이 걸린다. 이 결과는 처리 해야 할 작업의 부하가 전체 처리량의 증가 없이 여러 장치들로 분산되어 코디네이터에 집중되는 처리량을 줄일 수 있음을 보여준다.

6. 관련 연구

대부분의 기존 연구들은 지능형 공간에서의 인프라 기반의 상황정보 관리 구조를 제시하고 있다. SOCAM이나 Gaia의 상황 인지 미들웨어, CoBrA와 같은 온톨로지 기반의 상황 인지 미들웨어 접근 방법은 자원이 풍부한 환경에서의 상황정보를 제공한다 [3], [4], [13], [14]. 그러나 개인 지능형 공간이라는 개념이 나타나면서 상황 인지 시스템이 장치들의 제약된 자원을 고려할 필요가 생기게 되었다.

자원이 제약적인 장치에서 상황정보를 제공하기 위한 연구들 가운데 하나인 Contory에서는 지능형 전화기에서의 상황정보 제공을 위한 미들웨어를 제시하고 있다 [5]. Contory는 에드혹한 환경에서의 분산된 상황정보 제공을 포함한 세가지 종류의 상황정보 제공방법을 제공하며 세가지 제공방법을 네트워크 환경에서의 자원 제약사항을 고려하여 동적으로 적당한 상황정보 제공 방법을 선택하는 메커니즘을 제공한다. 하지만 이 연구에서는 상황정보 추론을 위하여 코디네이터에 처리 부하가 집중되는 상황을 고려하지 않고 있다.

그리고 추론작업의 기능을 분산하여 효율적인 상황정보 추론방법을 위한 몇몇의 연구들이 있다. EDCI는 이벤트 기반의 분산된 상황정보 추론 기능을 제공하고 있다 [6]. 이 연구에서는 분산된 상황정보 제공자들이 복합 이벤트 형태로 상황정보를 처리 함으로서 상황정보 추론에 비하여 빠른 상황정보 처리 시간을 제공한다. 추론 작업 분산을 위한 다른 접근 방법으로 Solar시스템이 있다 [7], [8]. 그래프 기반의 추상화는 상황정보를 모으고, 수집하고 처리하는데 도움을 준다. 이러한 접근 방법은 현재 존재하고 있는 상황정보 추론자들의 재사용 성을 증가시켜준다. 재사용 가능한 상황정보 제공자들은 불필요한 네트워크 전송 횟수를 줄여 주면서 추론된 상위레벨 상황정보를 제공할 수 있게 한다. 하지만 보다 빠른 추론 시간을 제공하고 재사용 성을 높인다 하더라도 이러한 연구들은 코디네이터만을 의존하는 중앙 집중 형의 상황정보 추론 작업을 수행 한다는 한계가 있다.

이러한 한계점을 해결하기 위하여 우리는 네트워크의 환경 변화에 따라 하나의 상황정보 추론 작업을 여러 개의 부분 작업으로 분배하는 효율적인 상황정보 추론 기법을 제공한다. 이러한 점들을 고려하여 우리는 개인 지능형 공간에서의 효율적인 상황정보 추론 기능을 제공하는 상황정보 관리 구조를 제시하고 있다.

7. 결론

우리는 개인 지능형 공간에서 자원 제약을 고려한 효율적인 상황정보 추론 기법을 제안하였다. 우리는 미리 정의된 부분 작업 규칙 없이 하나의 상황정보 추론 규칙을 여러 개의 부분 작업 규칙으로 분산 시킬 수 있도록 하는 상황정보 관리 구조를 제시한다. 우리의 접근 방법으로 우리는 코디네이터에만 집중된 상황정보 추론 처리 부하를 줄여준다.

본 연구는 복합 이벤트 감지 기법을 이용하여 가벼운 상황정보 처리 기법을 지원하는 상황 인지 시스템을 가진다. 이러한 방법이 상황정보 처리를 위한 처리 시간을 줄일 수는 있어도 풍부한 표현력을 제공하지는 못한다. 가벼운 상황정보 추론자는 시맨틱한 상황정보를 제공할 수 없는 단점을 지니고 있다.

이 연구의 확장으로서는 코디네이터가 없는 에드혹 환경에서의 다중 사용자를 지원하기 위한 상황정보 관리 구조를 제시하는 것이다. 현재 개인 지능형 공간을 위한 상황정보 관리 구조는 개인 지능형 공간이 하나의 코디네이터 역할을 하는 장치를 가지고 있다. 또한 개인 지능형 공간이 하나만 있을 때를 가정한다. 여러 개의 개인 지능형 공간과의 상호작용과 기존의 지능형 공간과의 상호작용은 다루지 않고 있다. 우리는 다중 사용자를 지원하는 환경에서의 새로운 사용자 시나리오를 제시하고 그에 맞는 에드혹 환경에서의 상황정보 관리구조를 제시할 계획을 가지고 있다.

참고문헌

[1] Anind K. Dey and Gregory D. Abowd, "Towards a Better Understanding of Context and Context-Awareness", Conference on Human Factors in Computing Systems (CHI 2000)

[2] Shiva Chetan, Jalal Al-Muthadi, Roy Campbell, and M. Dennis Mickunas, “Mobile Gaia: A Middleware for Ad-hoc Pervasive Computing”, In IEEE Consumer Communications & Networking Conference (CCNC 2005), Jan. 2005.

[3] O. Riva, Contory: A Middleware for the Provisioning of Context Information on Smart Phones, in the Proceedings of the ACM/IFIP/USENIX 7th International Middleware Conference (Middleware'06)

[4] T. Gu, H. K. Pung, D. Q. Zhang. "A Service-Oriented Middleware for Building Context-Aware Services." Journal of Network and Computer Applications (JNCA), Vol. 28, Issue 1, pp. 1-18, January 2005.

[5] Anand Ranganathan, Roy H. Campbell. "An Infrastructure for Context-Awareness based on First Order Logic."Personal and Ubiquitous Computing 7 2003

[6] Joo Geok Tan, Daqing Zhang, Xiaohang Wang, and Heng Seng Cheng, "Enhancing Semantic Spaces with Event-Driven Context Interpretation", PERVASIVE 2005

[7] Guanling Chen and David Kotz, “Context Aggregation and Dissemination in Ubiquitous Computing Systems”, Dartmouth Computer Science Technical Report TR2002-420

[8] Guanling Chen, Ming Li, David Kotz, "Design and Implementation of a Large-Scale Context Fusion Network", Proceedings of the First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous 2004)

[9] Anind K. Dey, Daniel Salber and Gregory D. Abowd, "A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications", Anchor article of a special issue on context-aware computing in the Human-Computer Interaction (HCI) Journal, 2001

[10] Dongman Lee, Seunghyun Han, Insuk Park, SaeHoon Kang, Kyungmin Lee, Soon J. Hyun, Young-Hee Lee, and Geehyuk Lee, “A Group-Aware Middleware for Ubiquitous Computing Environments”, ICAT 2004

[11] Peter R. Pietzuch, Brian Shand, and Jean Bacon "Composite Event Detection as a Generic Middleware Extension". IEEE Network Magazine, Special Issue on Middleware Technologies for Future Communication Networks, 2004.

[12] Samuel R. Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong, “The Design of an Acquisitional Query Processor for Sensor Networks,” SIGMOD, June 2003, San Diego, CA.

[13] Harry Chen, Tim Finin, and Anupam Joshi, "An Ontology for Context-Aware Pervasive

Computing Environments", The Knowledge Engineering Review 2003

[14] Carl-Fredrik Sorensen, Maomao Wu, Thirunavukkarasu Sivaharan, Gordon S. Blair, Paul Okanda, Adrian Friday, Hector Duran-Limon, "A Context-Aware Middleware for Applications in Mobile Ad Hoc Environments", Proceedings of the 2nd workshop on Middleware for pervasive and ad-hoc computing, 2004

[15] G. Judd and P. Steenkiste. "Providing Contextual Information to Pervasive Computing Applications.", In 1st IEEE Conference on Pervasive Computing and Communications (PerCom) , pages 133-142, Fort Worth, March 2003.