유비쿼터스 환경에서 문맥을 고려하여 대체 서비스를 찾아주는 기법

이재익[O] 이해준 김우현 김영준 고보정 이동만
한국정보통신대학교
{humanjack[O], haejun, woorung, jykim412, bjko, dlee }@icu.ac.kr

A Mechanism to Look up Alternative Services for a Context-Aware Service Discovery System in a Pervasive Computing Environment

Jae Ik Lee[O], Haejun Lee, Woohyun Kim, Youngjun Kim, Bojung Ko, and Dongman Lee
Information and Communication Univ.

요 약

유비쿼터스 컴퓨팅 환경에서는 사용자가 움직이고 주변환경이 변화하면서 문맥이 변화한다. 사용자가 원하는 서비스를 찾기 위해서는 이러한 변화가 있을 때마다 변화한 부분을 알고 새롭게 요청을 해야 한다. 이러한 사용자의 추가적인 작업을 최소한으로 하기 위해서는 서비스 찾기 시스템(Service Discovery System)에서 문맥을 인식하여 새로운 환경에서 기존에 사용자가 원했던 요청을 새로운 환경에서도 받아들일 수 있는 방법이 필요하다. 본 연구에서는 서비스 제공자가 서비스를 알릴 때나 사용자가 처음 요청을 할 때 대체할 서비스에 대한 기술을 추가함으로써 문맥이 바뀌거나 주위환경이 바뀌어 사용자가 본래 원하던 서비스가 존재하지 않더라도 대체할 서비스를 찾아주는 방법을 고안하고 이에 대한 고려사항과 방법을 설명한다. 실험결과에서 제안하는 방법이 기존의 방법보다 변화되는 환경에서 대체 서비스를 찾기 위한 지연시간을 현저히 줄인다는 것을 보여준다.

1. 서 론

유비쿼터스 컴퓨팅 환경의 가장 큰 목표 중 하나는 사용자가 컴퓨팅 환경을 사용하기 위한 부가작업을 최소한으로 것이다[1]. 이러한 목표를 이루기 위해서는 컴퓨팅 환경이 문맥 인식을 할 필요가 있는데, 그 이유는 사용자가 움직이고 주변 환경이 변화면서 문맥이 변화하고, 문맥이 변화 될 때 마다 사용자가 사용하던 서비스를 계속적으로 사용하기 위해서는 사용자의 추가적인 작업이 필요하기 때문이다. 그러므로 컴퓨팅 환경이 문맥의 변화를 인식하여 사용자의 추가적인 작업을 최소한으로 줄이는 일은 유비쿼터스 환경에서 매우 중요한 일이다[1].

서비스를 찾을 때에도 위에서 설명한 바와 같이 문맥의 변화를 인식하여 추가적인 작업 없이 사용자가 원하는 서비스를 계속해서 찾아줄 필요가 있다. 기존의 서비스 찾기(Service Discovery) 연구들이 서비스의 움직임[2]나 문맥의 변화 대한 인식[3]을 하려는 연구들을 해오고 있으나, 문맥의 변화로 사용자가 찾는 서비스가 없을 경우 대체되는 서비스를 찾는 고려는 많이 하고 있지 않다[4].

본 연구에서는 주위 환경의 변화에 따라 사용자가 원하는 서비스를 찾을 수 없을 경우에 대체 되는 서비스를 찾아주기 위한 고려사항들을 찾고 이를 INS를 기반으로 구현함으로써 그 효용성을 살펴볼 것이다.

본 논문의 구성은 다음과 같다. 2장에서는 서비스를 찾아주기 위한 관련 연구를 소개하고 3장에서는 주위 환경의 변화에 따라 사용자가 원하는 서비스를 찾을 수 없을 경우에 대체 되는 서비스를 찾아주기 위한 고려사항들을 찾아 보고 그 방법을 고안 한다. 4장에서는 제안하는 방법의 구현과 성능 평가 결과를 보이고 5장에서 결론을 말하고자 한다.

2. 관련 연구

Jini[5], UPnP[6], SLP[7]등 서비스 찾기에 대한 연구들이 많이 있지만 유비쿼터스 환경에서 서비스와 서비스의 이름 사이의 바인딩이 자주 바뀌는 것에 대한 고려는 하고 있지 않다.

INS[2]는 이동(Mobile)환경에서 서비스와 이름이 바뀌는 것에 대한 고려를 늦은 바인딩을 통해서 하고 있고, 주위 환경 변화에 따른 서비스들의 상태 변화는 느슨한 서비스 관리(soft state)를 통해서 한다. 하지만 사용자가 요청한 서비스가 없을 경우에 대체할 서비스를 찾아주지는 않는다.

Context Attributes[8]에서는 서비스의 상태변화를 문맥으로 하고, 서비스 상태를 사용자가 서비스를 요청한 시간에 확인 함으로써 사용자의 요청에 가장 적합한 상태에 있는 서비스를 찾아주려는 기법을 고안하였다. 하지만, 적합한 상태에 있는 서비스가 없을 경우에는 서비스를 찾는데 실패하게 된다.

Solar[3]는 문맥 변화가 생겨도 동일한 이름으로 서비스를 찾도록 해주지만, 주위 환경의 변화에 따로 요청한 서비스를 찾을 수 없을 경우에 대체 서비스를 찾아주지는 못 한다.

3. 제안하는 기법

3.1 설계시의 고려사항

① 대체 서비스를 찾기 위한 기준들

대체 서비스가 사용자가 찾으려고 했던 서비스와 기능적으로 일치하지 않는다면, 대체 서비스는 의미가 없다. 그러므로, 기능적 일치 성은 대체 서비스를 찾기 위한 가장 중요한 기준이 된다. 위치와 상태 또한 중요한 요소이다. 기능이 일치하더라도 사용자가 접근할 수 없거나 원하지 않는 위치에 있는 서비스를 찾아준다면 사용자가 쓸 수 없기 때문이다. 또한 찾아준 서비스에 부하가 많이 걸려있으면 역시 사용할 수 없으므로 서비스의 상태 또한 중요한 기준이 된다. INS[2], Solar[3], Context Attribute[8] 등의 기존 네이밍 시스템 역시 기능, 위치, 상태 등을 서비스를 찾기 위한 중요 속성으로 사용하고 있다.

② 서비스 사이의 기능적 유사성의 정의 방법

대체 서비스가 사용자가 찾으려 하는 서비스와 기능적으로 유사하다는 것을 시스템이 알려면 그에 대한 기술이 있어야 한다. 서비스의 기능적 호환성은 서비스의 배포자가 가장 잘 알므로 이에

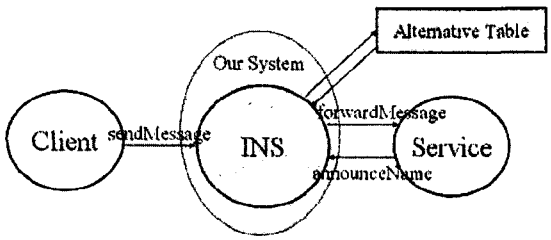

그림 1. 시스템 구조

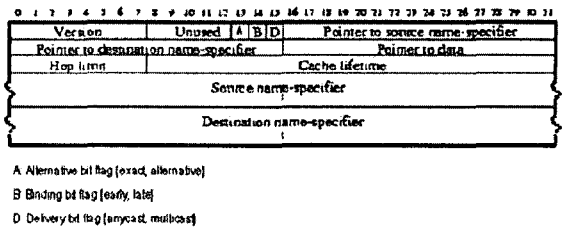

그림 2. INS의 변경된 메시지 헤더

대한 기술을 서비스 배포 자에게 맡기고 시스템에서는 기술 방법만을 제공한다.

③ 사용자가 대체 서비스를 원하는 지에 대한 정책

대체 서비스를 찾아 주는 것은 선택적인 일이고 사용자가 대체 서비스를 원하지 않을 수도 있으므로 대체 서비스를 찾아 주길 원하는 가에 대한 선택을 할 수 있는 방법을 제공해야 한다.

④ 대체 서비스가 사용자가 원하는 서비스와 호환성이 있다는 것을 보증하는 방법

두 서비스가 기능적으로 유사하다 하더라도 완전하게 호환성을 가진다고 할 수는 없다. 예를 들어, TV는 모니터와 기능적으로 유사하므로 대체서비스가 될 수 있지만, 일반 TV는 컴퓨터로부터의 신호를 해석할 수 없다. 이를 해결하기 위해 대체 서비스를 찾을 때 데이터 형식 등을 비교하는 방법을 사용하거나 신호를 보내 보고 처리가 불가능 하면 다른 서비스를 찾는 등의 방법을 사용할 수 있을 것이다. 본 논문에서는 간단이 하기 위해 서비스 제공자가 기능적으로 유사한 서비스를 기술할 때 호환성에 대한 고려가 있다고 가정한다.

⑤ 대체 서비스를 찾기 위한 정보를 동적으로 갱신하는 방법

유비쿼터스 환경에서는 서비스들이 하나의 네트워크에 들어오고 나가는 일이 빈번하게 일어 날 수 있다. 따라서, 대체 가능한 서비스들도 계속적으로 변할 수 있으므로 이에 대한 최신의 정보를 유지할 수 있도록 시스템에서 지원해야 한다.

3.2 시스템 구조

본 연구에서는 대체 서비스를 찾는 기법을 지원하기 위해서 앞은 논문에서 참조되고 있는 서비스 찾기 시스템인 INS[2]를 변경하였다. 그림 1에서처럼 INS와 같은 구조를 갖지만, 서비스를 알리거나 찾을 때 대체 서비스에 대한 정보를 저장하고 찾기 위해 대체 서비스 목록(Alternative Table)이 추가 되었다. 대체 서비스를 찾기 위해 대체 서비스 목록(Alternative Table)에 기능성과 위치를 기반으로 서비스간의 유사성을 정의하였고, 사용자가 요청한 서비스를 찾지 못할 경우에 이 서비스 목록에 기반해서 대체 서비스를 찾아 준다.

3.3 제안하는 기법

제안하는 기법의 서비스 알림, 서비스 요청방법은 기본적으로 INS의 방법을 사용하였고, 대체 서비스 기술 방법을 INS 기반으로 설명하였다.

3.3.1 서비스 제공자가 대체 서비스를 기술하는 방법

서비스 제공자가 대체 서비스에 대한 기술을 하려면 이름을 네이밍 시스템에 알리는 단계에서 대체 가능한 서비스를 알려야 한다. 예를 들어,' [service=printer,plotter][building=engineering[floor=6[room=617]' 으로 서비스의 이름을 알릴 경우에 본래의 서비스는 ' printer' 이고 대체 가능한 서비스는 ' plotter' 가 된다. 이때 비슷한 종류의 서비스라 하더라도 제공하는 기능이 다르므로 이러한 대체 서비스에 대한 것을 서비스 단위로 한정한다. 즉, 같은 TV라 하더라도 어떤 TV는 모니터를 대신 할 수 있고 어떤 TV는 그렇지 않으므로, 각 서비스 개체 별로 대체 가능한 서비스 리스트를 따로 관리한다는 의미이다.

3.3.2 사용자의 대체 서비스 요청 방법

INS는 기본적으로 대체 서비스를 찾아 주지 않으므로 이를 지원하기 위해, INS의 메시지 헤더에 대체 서비스를 찾아주는 기법을 사용하겠다는 것을 표시 하는 부분을 추가하였다. 그림 2에서처럼 ' A' 비트를 설정함으로써 제안하는 기법을 사용할 수 있게 된다. 서비스를 요청할 때는 서비스 제공자가 서비스를 알릴 때와 같은 방식으로 요청하면 된다. 예를 들어, ' [service=printer, plotter][building=engineering[floor=6[room=617.618]' 이라고 요청하면 printer가 요청한 서비스가 되고, 요청한 위치에 printer가 없으면 plotter를 찾게 된다. 또한, 617에서 대체 서비스도 찾지 못하면 618에서 찾는다.

Alternative Table	
Service	Alternatives
Monitor	TV, Projector
TV	Projector, Monitor
Printer	Plotter
...	...
...	...

그림 3. 대체 서비스 목록

3.3.3 대체 서비스를 찾는 방식

사용자의 정책에 따라 서비스를 찾는 방식이 달라진다. 사용자가 대체 서비스를 찾길 원하지 않는다면, INS와 서비스를 찾는 방식이 같다. 사용자가 대체 서비스를 찾기 원할 경우, 우선 요청한 서비스들을 찾는다. 요청한 서비스를 찾지 못했을 경우에 대체 서비스 목록의 정보를 이용해서 모든 가능한 대체 서비스들의 이름을 만들고, 그 이름 순으로 서비스들을 찾는다.

3.3.4 대체 서비스 목록

대체 서비스 목록은 각 서비스 별로 대체 가능한 서비스들에 대한 정보를 가지고 있다. 대체 가능한 서비스를 찾기 위해서는 ' Ontology' 와 ' Semantic Search' 등이 사용될 수 있지만[9], 제안하는 기법에서는 서비스들의 문자열만 비교하므로 그림 3과 같이 간단하게 서비스 별 대체 가능한 서비스들의 목록만을 가지게 된다. 이 목록은3.3.2에 기술한 것과 같은 방법으로 서비스 제공자에 의해서 갱신된다.

3.3.5 대체 서비스 사용에 대한 3가지 정책

① 대체 서비스를 사용할지에 대한 결정

대체 서비스를 사용할지를 결정하기 위해서 사용자는3.3.1에서 기술한 것과 같이 메시지 헤더의 값을 정해야 한다. 대체 서비스를 찾으려면 ' A' 비트를 1로 하고 그렇지 않으면 0으로 해야 한다.

② 대체 서비스를 찾기 위해 사용할 속성

사용자는 모든 속성에 대체할 속성을 지정할 수 있다. 예를 들어 그림 4에서 ' room=613' 대신 ' room=613, 614' 라고 요청하면 613에 서비스가 없으면 614에 있는 서비스를 찾아달라는 의미가 된다. 또한 사용자는 특정 속성에 대한 것은 대체 서비스를 원하지 않을 수도 있다. 예를 들어, 지정한 위치에서만 서비스를 찾고 싶다면 그림 4에서와 같이 해당 속성 앞에 ' !' 표시를 한다. 이럴 경우, 똑 같은 서비스가 다른 위치에 있다 하더라도 사용자가 요청한 위치의

```
![building=engineering[floor=6th[room=613]]],
[service=TV[resolution=1024*768,size=30inch]]
```

그림 4. 사용자는 대체할 속성을 정할 수 있다.

서비스만 찾게 된다.

③ 대체 서비스 찾을 때의 우선순위

사용자는 여러 개의 대체 속성을 줄 수 있고 이중 우선순위를 두고 싶어할 것이다. 예를 들어, TV가 없으면 monitor와 projector를 원하지만 projector보다는 monitor를 우선 찾고 싶을 경우, 사용자는 순서대로 기술하면, 순서대로 우선순위가 부여된다. 즉, ' service=TV, monitor, projector' 라고 기술하면 TV가 요청하는 서비스고 TV가 없을 경우 monitor, projector의 순으로 찾아 달라는 의미가 된다.

4. 구현 및 성능 분석

4.1 구현

제안하는 방법은 Java로 구현한 INS 2.0에 기반해서 구현되었다. 우선 대체 서비스를 찾아 줄 것인가를 표시하기 위해 INS의 메시지 헤더 중 헤더 변경을 위해 예약된 부분을 수정하였다. 두 번째로 사용자가 대체 서비스를 요청하거나 서비스 제공자가 대체 서비스를 알릴 때의 이름을 INS가 이해 할 수 있도록 이름 해석 부분을 수정하였다. 마지막으로 대체 서비스 목록과 목록을 이용해서 모든 대체 서비스들의 이름을 만드는 부분, 그리고 그 이름들을 이용해서 대체 서비스를 찾는 부분이 추가 되었다.

4.2 성능 분석

4.2.1 실험 환경

제안하는 방법을 평가하기 위해서 두 대의 컴퓨터를 설정하였다. 하나는 펜티엄4 2.4G CPU, 512 RAM, Windows XP, 10Mbps 유선 랜 환경의 데스크 탑이고, 다른 하나는 펜티엄 4 모바일 2.2G CPU, 1G RAM, Windows XP, 11M 무선 랜 환경의 노트북이다. 데스크 탑은 네이밍 서버로 사용되었고, 노트북은 서비스 제공자와 사용자로 사용되었다. 이 실험은 제안하는 방법의 계산 부하와 사용자가 서비스를 찾기 위해 걸리는 시간만을 평가 기준으로 사용하므로 위의 실험 환경으로 충분히 두 가지 기준을 평가할 수 있다.

4.2.2 계산 부하

4.1에서 설명한 것과 같이 제안된 방법을 구현하기 위해서 사용자의 요청이 들어 왔을 때 사용자가 본래 원했던 서비스를 찾지 못하면 모든 가능한 대체 서비스의 이름들을 만든 후 순서대로 이름을 찾는다. 그러므로, 제안된 방법을 사용함으로 해서 대체 서비스들의 이름을 만드는 계산의 부하와 대체 서비스들을 찾는 부하가 생긴다. INS의 경우 이름을 해쉬 함수를 사용해서 찾아주므로 이름을 찾는 부하는 거의 무시할만하다. 그래서, 이름을 만드는 부하만을 고려하였는데, 그 결과는 그림 5와 같다. 그림에서와 같이 전체의 계산 부하 중 이름을 생성하는 부하가 대부분을 차지한다. 따라서 제안하는 방법을 개선하기 위해서는 이름을 생성하는 알고리즘을 개선하고 모든 가능한 이름이 아닌 일부만을 생성하는 방법이 고안 되어야 할 것이다.

4.2.3 서비스 검색 시간

계산 부하만을 생각한다면 제안하는 방법이 비효율적으로 보이지만 제안하는 방법의 가정처럼 사용자가 요청한 서비스를 사용할 수 없게 되는 경우 사용자는 원하는 서비스를 찾을 때까지 계속 다시 시도를 해야 한다. 이럴 경우 사용자가 다시 요청을 하 는 지연시간과 네트워크로 요청이 전송되는 지연이 추가된다. 이 두 지연시간이 계산 부하에 의한 지연보다 훨씬 크기 때문에 제안하는 방법이 효율성을 가지게 된다. 그림 6은 그 결과를 보여준다. 대체 서비스를 찾는 회수가 늘어나더라도 제안하는 방법은 지연 시간의 증가에 큰 변화가 없지만 제안하는 방법을 사용하지 않을 경우에는 극적으로 지연시간이 늘어나게 된다.

5. 결론

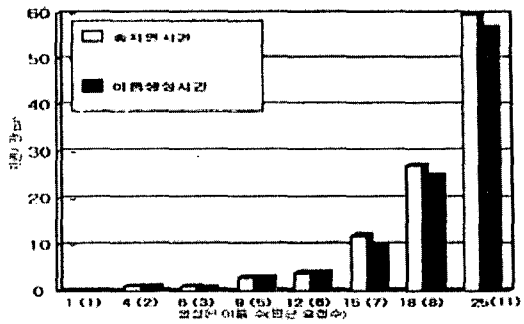

그림 5. 계산 부하

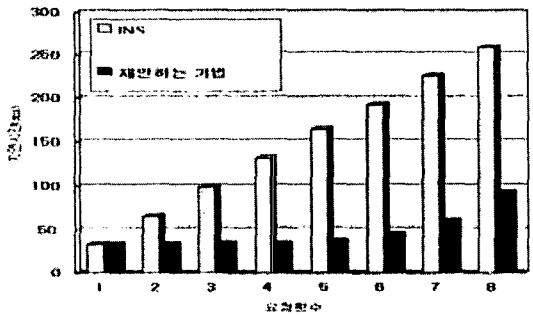

그림 6. 제안된 방법을 사용할 경우와 사용하지 않을 경우의 비교

성능 분석 결과에서 보듯이 대체 서비스를 찾아 줌으로써 사용자가 원하는 서비스를 찾아주는 지연시간은 크게 단축되고, 환경의 변화가 심해서 서비스들의 상태변화가 심할수록 더 극적으로 제안된 방법이 좋은 결과를 낸다는 걸 볼 수 있다.

이 논문에서는 사용자가 문맥의 변화에 대한 기술을 명시적으로 서비스의 이름에 기술해야 했다. 향후 과제로서 사용자가 문맥의 변화에 대한 입력을 주지 않더라도 시스템에서 변화하는 부분을 대신 찾아주어 사용자가 원하는 서비스를 찾아주는 연구를 계속 할 것이다.

참고 문헌

[1] M. Weiser. " The computer for the 21st century." , Scientific American, 265(3):66-75, Jan. 1991.

[2] William Adjie-Winoto, Elliot Schwartz, Hari Balakrishnan, and Jeremy Lilley, " The design and implementation of an intentional Naming System" , MIT Laboratory for Computer Science, ACM 1999.

[3] Guanling Chen and David Kotz, " Context-Sensitive Resource Discovery" , IEEE PerCom03.

[4] F. Zhu, M. Mutka, and L. Ni, Classification of ServiceDiscovery in Pervasive Computing Environments, MSU-CSE-02-24, Michigan State University, East Lansing, 2002.

[5] Arnold, K., Wollrath, A., OSullivan, B., Scheifler, R., Waldo, J.: The Jini Specification. Addison-Wesley (1999)

[6] UPnP Forum: Universal Plug and Play Connects Smart Devices. http://www.upnp.org/

[7] Guttmann, E., Perkins, C., Veizades, J., Day, M.: Service Location Protocol, Version 2. IETF Internet Draft, RFC 2608 (1999)

[8]C. Lee and A. Helal, "Context Attributes: An Approach to Enable Context-awareness for Service Discovery," Proceedings of the Third IEEE/IPSJ Symposium on Applications and the Internet, Orlando, Florida, January 2003

[9] Robert E. McGrath, Anand Ranganathan, Roy H. Campbell, M. Dennis Mickunas, " Use of Ontologies in Pervasive Computing Environments" , Department of Computer Science, University of Illinois, Urbana-Champaign, April, 2003.