UPnP 구조에서 규칙 기반 적응형 디바이스 컨트롤

박동훈[o1], 정우성[1], 김희천[2], 우치수[1]
[1]서울대학교 컴퓨터공학부 소프트웨어 공학 연구실
{donghun, wsjung, wuchisu}@selab.snu.ac.kr
[2]한국방송통신대학교 컴퓨터과학과
hckim@knou.ac.kr

Rule-based Adaptive Device Control in UPnP Architecture

Donghun Park[o1], Woosung Jung[1], Heechern Kim[2], Chisu Wu[1]
[1]School of Computer Science & Engineering, Seoul National University
[2]Dept. of Computer Science, Korea National Open University

요 약

홈 네트워크 기술의 발전과 더불어 관련 미들웨어 기술의 중요성이 높아지고 있다. 미들웨어 기술 중 하나인 UPnP로 네트워크를 구성할 때 유사한 서비스를 제공하는 디바이스들이 여럿 존재하는 경우라면 사용자의 의도에 적합한 선택이 가능해야 한다. 본 연구에서는 규칙 언어에 기반을 둔 적응형 아키텍처를 통해 이러한 문제를 해결한다. 제안하는 적응형 디바이스 컨트롤은 기존의 UPnP 구조를 변형 시키지 않고 새로운 규칙 관리자를 도입하여 호환성을 유지하였다. 프린터기를 이용한 가상 시나리오를 구상하고, 이에 맞춰 UPnP 네트워크를 구현함으로써 아키텍처의 타당성을 검증해 본다.

1. 서 론

홈 네트워크는 가정안의 모든 가전기기 및 시스템이 서로 또는 외부와 네트워크를 통해 연결하여 원격 접근과 제어가 가능하고, 다양한 콘텐츠를 사용할 수 있도록 양방향 통신 서비스가 제공되는 환경을 제공한다.

인터넷 기반의 정보화가 점차 생활 속에 자리 잡아 가는 면에서 홈 네트워크 기술은 미래를 선도하는 기술이 될 것이다. 고속 인터넷 서비스뿐만 아니라 와이브로와 같은 차세대 모바일 서비스, IP-TV와 같은 디지털 방송 서비스처럼, 홈 네트워크 기술은 다양한 영역을 통합하는 기술로 그 중요성이 확대 되고 있다. 홈 네트워크 시장이 확대되어 갈수록 다양한 기기들이 홈 네트워크에 접속 될 것이고 이를 감지하고 제어하기 위한 미들웨어 기술의 중요성도 높아진다.

홈 네트워크를 지원해주는 미들웨어 기술로는 마이크로소프트가 제안한 UPnP(Universal Plug and Play) 기술과, 소니와 여러 AV업체가 함께 제안한 HAVi(Home Audio Video interoperability) 기술, 삼성이 제안한 VHN (Versatile Home Network) 기술, 선 마이크로 시스템이 제안한 Jini 기술 등이 있다. 이 중 UPnP가 홈 네트워크를 위한 미들웨어 기술 중 가장 대표적인 기술로 평가받고 있다. UPnP가 다른 기술과 차별되는 점은 표준 프로토콜을 사용한다는 점이다[1]. UPnP는 표준 프로토콜을 사용함으로써 개발환경이나 구현 언어에 상관없이 다양한 디바이스들이 네트워크 안에 참여 할 수 있게 해주어, 다른 미들웨어 기술보다 환경에 독립적인 특성을 가지고, 이로 인해 서로 다른 업체의 기기들이 존재하는 홈 네트워크 환경에서 이종 디바이스들의 협업을 보다 쉽게 실현할 수 있다. UPnP는 마이크로소프트, 소니 등 대표적인 홈 네트워크 장비 생산 업체들이 주축이 되어 설립된 DHWG(Digital Home Working Group) 에서 엔터테인먼트 분야의 표준 기술로 선택 되는 등 그 적용 분야가 확대 되어 가고 있다.

본 논문은 UPnP를 사용하여 디바이스들을 제어할 때, 사용자의 의도를 반영할 수 있는 방법과 이를 구현하기 위한 아키텍처를 제안한다. 홈 네트워크 시장이 확대 될수록 유사한 서비스를 제공 하는 다양한 디바이스들이 존재 하게 될 것이다. 유사한 기능과 인터페이스를 가진 디바이스의 증가는 사용자가 원하는 최적의 서비스를 찾는데 드는 비용을 증가시키게 될 것이다. 이는 지능형 홈 네트워크를 구축하는데 있어서는 중요한 문제이다. 뿐만 아니라. 사용자의 요구가 언제나 같을 수는 없다. 경우에 따라 조금씩 다른 요구를 하는 사용자의 의도에 적응해 가장 최적의 디바이스를 자동으로 연결하여 줄 수 있다면 보다 나은 홈 네트워크의 구성에 중요한 기술이 될 수 있을 것이다. 본 연구는 사용자의 의도를 판단하기 위해 규칙 기반 언어를 사용하였다. 규칙에 기반하여 디바이스가 제공하는 역할과 사용자의 의도를 토대로 추론하여 가장 적절한 디바이스를 선택하여 서비스를 제공하는 것을 목표로 하고, 실제 기술에 적용, 확장 가능한 아키텍처를 제안한다.

이 논문은 다음과 같이 구성되어 있다. 2장에서는 이미 존재하는 선행 연구들에 대해 알아본다. 3장에서는 UPnP의 기술에 관해 기본적인 소개와 논문과 연관된 기술을 위주로 소개하고, 4장에서는 가상 시나리오를 만들고, 시나리오에 부합하는 UPnP 아키텍처를 설명한다. 그리고 아키텍처를 바탕으로 작동하는 시스템을 구현 한다. 5장에서는 논문에 대한 결론을 맺는다.

+ 본 연구는 한국과학재단 특정기초연구(R01-2006-000-11150-0)지원으로 수행되었음

2. 관련 연구

홈 네트워크 환경에서 다양한 디바이스를 적응형으로 제어하기 위한 연구는 서로 다른 프레임워크 아래에서 진행되어 왔다. 대부분의 연구들은 홈 네트워크에 존재하는 다양한 자원들을 보다 적응형으로 사용하는 것에 초점을 맞추고 있다.

JVM(Java Virtual Machine) 위에 독립적 서버 시스템을 구성하여, 디스크 공간, CPU 파워, 메모리 등 네트워크상에 존재하는 자원들을 적응형으로 사용해 보다 좋은 서비스를 제공할 수 있는 디바이스로 부터 서비스를 받고, 디바이스들 간의 균형적인 사용이 가능하게 하는 연구가 있다[2]. UPnP 기술을 사용하여 여러 개의 디바이스가 존재하는 환경아래에서 각 디바이스들의 조정을 연구한 논문도 있다[3]. 이 논문에서는 여러 개의 디바이스가 존재하는 상황에서 미리 정의 한 시나리오에 따라, 분산된 환경에 존재하는 디바이스들이 서로 조정, 협력하는 상황을 연구하였다. 필요에 따라 각 시나리오 상의 디바이스들은 자동으로 재설정이 가능한 아키텍처를 제공한다. UPnP 기술에서 여러 디바이스를 찾기 위해 컨트롤 포인트가 사용하는 멀티캐스트 쿼리에 관한 연구도 있다[4]. 컨트롤 포인트는 자신이 원하는 디바이스를 찾기 위해 네트워크에 후보 디바이스를 찾는 질의를 멀티캐스트로 전송하게 되는데 이때 아무런 제약이 없다면 질의가 서비스가 가능한 지역을 벗어나 오버런(Overrun)하는 문제가 생길 수 있다. 이 문제를 해결하기 위해 일정한 경계를 제시하고 상황에 따라 적응해서 경계 값을 바꾸는 연구가 이루어졌다.

AV(Audio Visual) 디바이스의 자원을 적응형으로 사용하기 위해 QoS(Quality of Service) 아키텍처를 사용하는 연구도 있다[5]. 멀티미디어 응용프로그램은 QoS를 사용해 일정 수준이상의 서비스 제공을 보장 할 수 있다. 이 연구는 UPnP에 QoS를 지원하는 아키텍처를 제안해서 자원을 적응형으로 사용해서 일정 수준의 서비스가 제공 될 수 있다는 것을 실험을 통해 입증하였다. 나아가 QoS를 통한 승인 제어가 가능한 것도 보여준다. UPnP 기술에 QoS를 접목 하는 시도는 UPnP 포럼에서도 지원하는 것으로 연구에서는 실제 실험을 통해 논문에서 제시한 아키텍처로 테스트를 해보았더니 상당히 안정적인 서비스가 이루어지는 것을 확인 할 수 있다고 밝히고 있다.

3. UPnP 기술 소개

UPnP는 마이크로 소프트가 주축이 되어 제안한 홈 네트워크 미들웨어 기술로 정보가전, 무선통신장치, PC관련 장비 등 여러 장소에 분산되어 있는 디바이스와 서비스간의 편리한 통신을 제공하는 미들웨어기술이다. 이는 마이크로 소프트의 운영체제인 윈도우의 PnP(Plug and Play)를 보다 다양한 장치에 적용할 수 있게 확장한 것으로, 네트워크상에서 관리자의 직접적인 개입이 없거나, 사전 정보가 없는 상태에서도 장치들의 연결이 가능하도록 해준다. UPnP는 인터넷 표준 프로토콜(TCP/IP)을 사용하여 구성된다. 이 프로토콜 위에서 XML기반으로 SOAP(Simple Object Access Protocol)과 GENA(General Event Notification Architecture), SSDP(Simple Service Discovery Protocol) 를 사용해 정보를 주고 받는다. 따라서 구현된 프로그램의 언어나 기반 시스템에 독립적이다. 그리고 웹 브라우저를 통해 디바이스를 제어하는 사용자 인터페이스를 제공한다.

그림 1 . UPnP의 계층 구조

그림 1에서는 UPnP의 계층 구조를 여섯 개의 단계별로 나누어 나타내고 각 계층에서 사용하는 기술들을 나타내고 있다.

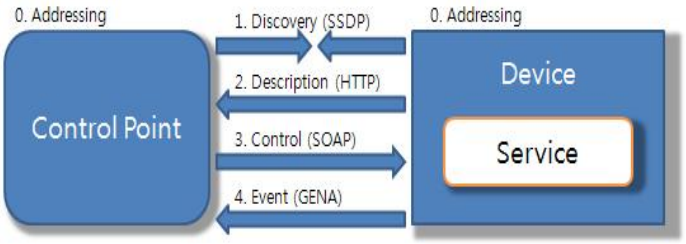

그림 2 . UPnP의 동작 과정

UPnP의 동작은 디바이스와 컨트롤 포인트를 중심으로 설명 할 수 있다. 디바이스는 사용자가 필요로 하는 서비스를 제공하는 장치로 컨트롤 포인트의 명령을 받아들여 서비스를 제공하는 역할을 한다. 컨트롤 포인트는 명령을 만드는 장치로 필요에 따라 디바이스에게 서비스를 요청 한다.

그림 2에서는 UPnP의 동작 과정을 설명하고 있다. 그림 1에서 언급한 계층들이 어떤 방향에서 디바이스와 컨트롤 포인트 관계에서 동작하는지를 나타내고 있다.

- Addressing

  디바이스와 컨트롤 포인트는 처음 네트워크에 참여

하게 되면 자기 고유의 주소를 가지게 된다. 이 주소에 기반 하여 상대방을 인식하고 메시지를 전달하게 된다.

- Discovery
  디바이스와 컨트롤 포인트가 서로를 인식하는 과정이다. 네트워크에 컨트롤 포인트가 먼저 자리 잡고 있는 상태에서 디바이스가 자신을 알리는 '광고'를 할 수도 있고, 반대로 새로 추가된 컨트롤 포인트가 네트워크에 미리 존재하던 디바이스를 찾는 '검색' 과정을 통해서도 서로의 존재를 발견 할 수 있다.

- Description
  서로의 존재를 인식한 후 디바이스가 컨트롤 포인트에게 자기소개를 하는 과정이다. 이 과정을 통해 컨트롤 포인트는 디바이스가 제공하는 여러 가지 서비스를 보다 자세히 알 수 있다.

- Control
  디스크립션 을 통해 알게 된 디바이스의 서비스를 사용하기 위해 컨트롤 포인트는 SOAP 메시지를 통해 디바이스를 제어 하게 된다.

- Event
  제어액션을 실행한 결과 값을 컨트롤 포인트에게 알려주기 위해 또는 디바이스 스스로 자신의 상태 변화가 생겼을 때 디바이스는 GENA를 사용해 컨트롤 포인트에 변화를 알린다.

- Presentation
  프레젠테이션은 사용자 친화적인 환경을 웹 브라우저를 기반으로 제공하는 역할을 한다.

4. 규칙 기반 적응형 아키텍처 소개와 구현

4.1 개요

UPnP 미들웨어 기술 시장이 확대 될수록 UPnP를 지원하는 제품들도 많이 등장 할 것이다. 이들 제품들을 구분하고 관리하기 위해, UPnP 포럼에서는 UPnP 포럼 템플릿(Forum Templates)을 두어 비슷한 서비스를 제공하는 벤더(Vendor)들이 같은 템플릿을 사용하여 미리 결정한 형식을 가지게 한다.

그림 3을 보면 UPnP 포럼 템플릿의 역할을 알 수 있다. 컨트롤 포인트에서 자신이 필요로 하는 디바이스를 검색할 때 이 템플릿을 구분자로 사용하여 같은 템플릿을 가지는 디바이스들만 발견 할 수 있다. 이렇게 발견된 디바이스들의 집합 중 컨트롤 포인트가 원하는 서비스를 제공 할 수 있는 최상의 디바이스를 적응형으로 선택할 수 있는 아키텍처를 제공하는 것이 본 연구의 목적이다.

4.2 아키텍처 정의

그림 4에서 논문에서 제시하는 규칙 기반 적응형 아키텍처의 모습을 볼 수 있다. 기존 UPnP는 컨트롤 포인트와 디바이스가 직접 연결되어 (그림 4의 점선) 작동 한다. 새로운 구조에서는 이들 둘 사이에 규칙 관리자(Rule Manager)를 두어 적응형 구조를 만들었다. 규칙 관리자는 기존 UPnP 구조에서 사용되는 컨트롤 포인트와 디바이스 그리고 규칙 처리자(Rule Handler)를 포함한 구조로 구성되어 있다. 기존의 컨트롤 포인트에게는 규칙 관리자의 구성요소 중 디바이스가 연결되어 마치 서비스를 직접 제공해주는 디바이스인 것처럼 보이게 작동한다. 기존의 디바이스에는 규칙 관리자의 구성요소 중 컨트롤 포인트가 연결되어 또 다른 하나의 컨트롤 포인트로 인식되게 해준다.

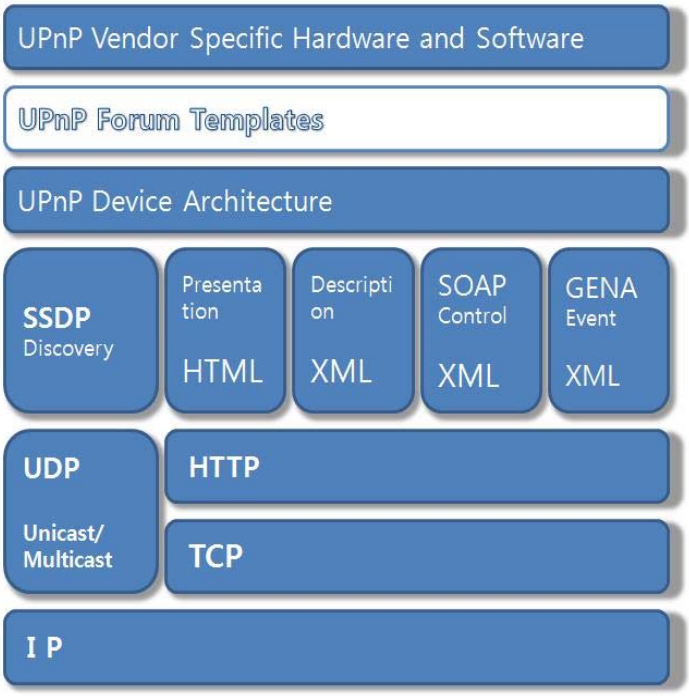

그림 3 . UPnP 프로토콜 계층

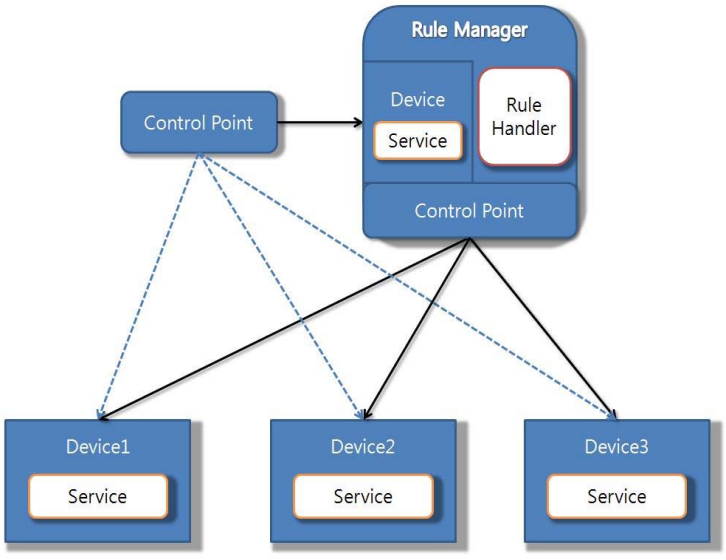

그림 4 . 규칙 기반 적응형 아키텍처

```
- <scpd>
  + <specVersion>
  + <serviceStateTable>
  - <actionList>
    - <action>
        <name>CreateJob</name>
      - <argumentList>
        + <argument>
        - <argument>
            <name>DocumentFormat</name>
            <direction>in</direction>
            <relatedStateVariable>DocumentFormat</relatedStateVariable>
          </argument>
        - <argument>
            <name>MediaSize</name>
            <direction>in</direction>
            <relatedStateVariable>MediaSize</relatedStateVariable>
          </argument>
        - <argument>
            <name>MediaType</name>
            <direction>in</direction>
            <relatedStateVariable>MediaType</relatedStateVariable>
          </argument>
        - <argument>
            <name>PrintQuality</name>
            <direction>in</direction>
            <relatedStateVariable>PrintQuality</relatedStateVariable>
          </argument>
        </argumentList>
      </action>
    - <action>
        <name>CancelJob</name>
      - <argumentList>
        - <argument>
            <name>JobId</name>
            <direction>in</direction>
            <relatedStateVariable>JobId</relatedStateVariable>
          </argument>
        </argumentList>
      </action>
    </actionList>
  - <clips_rule>
    - <![CDATA[


        ;;=======================================================
        ;; printer HPK-1800 rule
        ;; check lease condition
        ;; check leave condition
        ;;=======================================================

        (deftemplate attribute
           (slot DeviceId)
           (slot PrinterState (default idle))
           (slot DocumentFormat)
           (slot MediaType)
           (slot PrintQuality (default 5)))

        (defrule check_lease_condition
          ; status check
          (PrinterState is idle)
          ; printer quality check
          (test (> PrintQuality ?RequestQuality))
          ; support media type check
          (eq MediaType ?RequestMediaType)
          =>
          (assert (lease this printer)))

        (defrule check_leave_condition1
           (PrinterState is stopped)
           (or (PrinterStateReasons is media-jam)
               (PrinterStateReason is media-empty))
           =>
           (assert (find another printer)))

        (defrule check_leave_condition2
           (test (< PrinterQuality ?RequestQuality))
           =>
           (assert (find another printer)))

        ....


      ]]>
    </clips_rule>
  </scpd>
```

그림 5 프린터 디바이스 서비스 디스크립션

4.3 동작 과정 설명

컨트롤 포인트가 자신이 원하는 디바이스를 찾는 검색 질의를 규칙 관리자에게 보낸다. 규칙 관리자는 컨트롤 포인트의 질의를 보고, 찾고자 하는 템플릿을 가진 디바이스를 전체 네트워크에서 찾는 멀티캐스트 메시지를 보낸다. 규칙 관리자에 있는 컨트롤 포인트가 보낸 메시지를 받은 디바이스는 자신의 상세 정보를 규칙 관리자에게 보낸다. 이때 전달되는 XML 문서는 그림 5에서 나타나듯 자신의 서비스와 관련한 정보와 서비스 판단 내용이 들어있는 규칙을 같이 보낸다. 규칙 관리자의 컨트롤 포인트는 디바이스들로부터 받은 XML 문서를 파싱(parsing)해서 디바이스의 정보를 나타내는 부분은 규칙 관리자의 디바이스 부분으로 보내고, 규칙에 관련된 부분은 규칙 처리자에게 보낸다. 규칙 관리자에 있는 디바이스는 이렇게 전달된 디바이스 정보를 종합하여 처음 메시지를 보냈던 컨트롤 포인트에게 자신의 XML 문서로 SSDP를 통해 전달한다.

컨트롤 포인트가 디바이스를 제어 하고 싶어 규칙 관리자의 디바이스에 SOAP을 통해 메시지를 보내면 규칙 관리자는 규칙 처리자에게 주어진 메시지를 토대로 정보를 구성해 가장 적절한 디바이스를 선택하여 그 디바이스에게만 자신이 가진 컨트롤 포인트를 사용해 SOAP을 통한 메시지를 보내게 한다. 디바이스로부터 전해지는 이벤트 값은 그대로 컨트롤 포인트에게 전달한다.

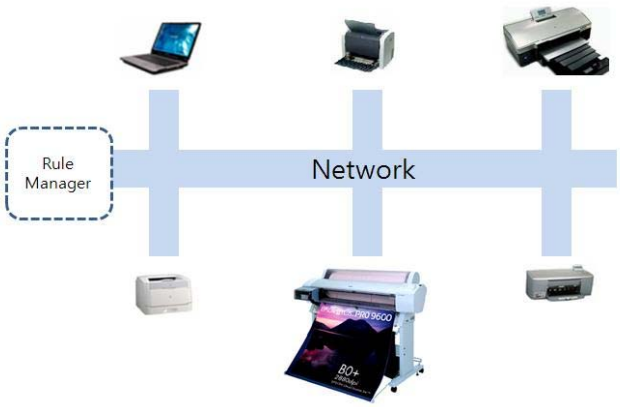

그림 6 UPnP 네트워크 가상 배치도

4.4 가상 시나리오

새로 산 노트북을 가지고 연구실에 들어왔다. 연구실은 UPnP를 사용해 구성된 네트워크가 구성된 환경이다. 작업한 문서를 출력하고 싶어 문서 편집기의 인쇄 버튼을 눌렀고, 프로그램이 내장된 프린터 컨트롤 포인터를 실행 하였다. 노트북에 설치된 컨트롤 포인트가 연구실에서 존재하는 프린터를 모두 검색하였고, 자동으로 그 중 한 가지 프린터를 연결해 주었다. 출력을 하였는데 칼라출력이 지원되지 않는 프린터였고 내가 작성한 문서에는 사진이 들어있어서 결국 원하는 모습대로 나오지 않았다. 이번엔 다른 문서를 출력하고 싶었는데 내가 원하는 출력은 B4 용지에 출력 하는 것, 혹은 OHP 필름에 출력하는 것이다. 매번 이런 경우 마다 직접 컨트롤 포인터에 연결되는 디바이스를 선택하고 변경하는 작업을 하는 것이 너무 힘들다. 사용자의 개입 없이 이루어지는 UPnP와도 맞지 않는 것 같다. 조금 더 편리하게 내 의도에 적응하는 시스템을 만들고 싶다. 내가 출력하고자 하는 문서에 담긴 정보를 토대로 가능한 프린터 후보들 중 최적의 후보가 선택되어 자동으로 출력이 된다면, 그리고 그 후보들의 구성이 UPnP를 통해 동적으로 구성된

다면 가장 좋을 것 같다.

4.5 구현

4.4의 가상시나리오를 기초로 구현을 해보았다. 구현은 리눅스의 한 종류인 ubuntu[6] 커널 2.6 운영체제 아래에서 이루어졌다. UPnP 네트워크를 구성하기 위해서 오픈소스로 이루어지는 Portable SDK for UPnP Device[7]를 사용하였다. 프로그램을 컴파일 하기 위해 사용한 gcc의 버전은 4.1.2 이다.

컨트롤 포인트와 디바이스를 위해서는 SDK에서 제공하는 API를 사용하였다. 가상 시나리오에서 이야기한 디바이스를 구성하기 위해서 UPnP 포럼에서 제공하는 프린터 디바이스 템플릿 1.01을 사용하였다. 규칙 관리자를 만들기 위해서 SDK에서 제공하는 API를 사용하였고, 규칙 처리자 부분을 구성하기 위해서는 clips[8]를 사용하였다. 디바이스가 가지는 변수들의 값은 FACT 정보로 규칙 처리자에 들어가게 되고, 디바이스의 디스크립션 단계에서 핸들러에 구성된 규칙에 의해 추론이 단계에서 사용된다.

구현을 위해 만들어진 시스템은 한 개의 컨트롤 포인트와 하나의 규칙 관리자, 그리고 두 개의 프린터 디바이스로 하나는 흑백 프린터, 하나는 컬러 프린터로 구성하였다.

표 1. 규칙 관리자 구동 모습

```
Initializing UPnP Sdk with
        ipaddress = (null) port = 0
UPnP Initialized
        ipaddress= 147.46.216.146 port = 49152
Specifying the webserver root directory -- ./web
Registering the RootDevice
        with desc_doc_url: http://147.46.216.146:49152/rulemanagerdesc.xml
RootDevice Registered

Init virtual Device..
Advertisements Sent

Init virtual Control Point
Registering Control Point
Control Point Registered
```

컨트롤 포인트와 디바이스가 각각 하나씩 등록된다.

표 2 컨트롤 포인트 구동 모습

```
Intializing UPnP with ipaddress=(null) port=0
UPnP Initialized (147.46.216.146:-16383)
Registering Control Point
Control Point Registered
```

표 3 디바이스 구동 모습

```
Initializing UPnP Sdk with
        ipaddress = (null) port = 0
UPnP Initialized
        ipaddress= 147.46.216.146 port = 49153
Specifying the webserver root directory -- ./web
Registering the RootDevice
        with desc_doc_url: http://147.46.216.145:49153/gray_printer.xml
RootDevice Registered

Advertisements Sent
```

표 4 규칙 관리자 작동 모습

```
============================================================
------------------------------------------------------------
UPNP_CONTROL_ACTION_REQUEST
ErrCode     =  0
ErrStr      =
ActionName  =  CreateJob
UDN         =  uuid:Upnp-PrintEmulator-1_0-1234567890001
ServiceID   =  urn:upnp-org:serviceId:printercontrol1
ActRequest  =  <u:CreateJob
xmlns:u="urn:schemas-upnp-org:service:printercontrol:1"></u:CreateJob>

ActResult   =  <u:CreateJobResponse
xmlns:u="urn:schemas-upnp-org:service:printercontrol:1">
<Copies>1</Copies>
<MediaType>color</MediaType>
<PrintQuality>high</PrintQuality>
</u:CreateJobResponse>
------------------------------------------------------------
============================================================

print color media type document message received!
use HPK-1800



============================================================
------------------------------------------------------------
UPNP_CONTROL_ACTION_REQUEST
ErrCode     =  0
ErrStr      =
ActionName  =  CreateJob
UDN         =  uuid:Upnp-PrintEmulator-1_0-1234567890001
ServiceID   =  urn:upnp-org:serviceId:printercontrol1
ActRequest  =  <u:CreateJob
xmlns:u="urn:schemas-upnp-org:service:printercontrol:1"></u:CreateJob>

ActResult   =  <u:CreateJobResponse
xmlns:u="urn:schemas-upnp-org:service:printercontrol:1">
<Copies>2</Copies>
<MediaType>gray</MediaType>
<PrintQuality>low</PrintQuality>
</u:CreateJobResponse>
------------------------------------------------------------
============================================================
print gray media type message received!
leave HPK-1800
use HPG-1200
```

칼라문서를 1개 인쇄하기 위해 CreateJob 명령어를 실행하면 규칙 관리자에서 요청을 확인하고 프린터 디바이스 중 HPK-1800을 사용하게 한다. 그리고 연이어 흑백 문서 2장의 인쇄를 요청하면 기존 프린터 디바이스 인 HPK-1800보다 HPG-1200을 사용하는 것이 더 적절하다는 규칙 처리자의 판단에 따라 HPG-1200을 사용하는 요청을 보낸다.

5 결론

본 논문에서는 UPnP로 구성한 홈 네트워크에서 유사한 기능을 가진 디바이스가 존재할 때 사용자의 의도에 맞는 최적의 디바이스를 상황에 따라 적응하여 선택하는 방법에 대해 생각해 보았다. 사용자의 의도를 파악하기 위해 본 논문에서는 룰 기반 언어인 clips를 사용하였다. 기존 UPnP의 특징 중 하나인 디스크립션에 디바이스의

룰을 넣어서 사용자의 의도를 파악하는데 사용하는 시도가 본 연구의 가장 큰 특징이라 할 수 있고, 간단한 구현을 통해 아키텍처의 타당성을 증명 할 수 있었다. 규칙 기반 언어의 특성상 다양한 확장과 자유로운 표현, 추론이 가능하다는 장점을 가지지만, 전문가 시스템의 특징인 실시간 환경에는 부적합 하다는 한계도 가지고 있다. 향후에는 규칙 언어의 장점을 그대로 유지 하면서 실시간으로 동적 적응을 제공하는 규칙 처리자에 대해 연구할 계획이다. 또한 보다 실용적인 사례를 통해 실생활에 적용 하여 복잡한 경우에 대해서도 유용함을 검증하고, 이를 모델화함으로써 적응형 UPnP 프레임워크를 설계하기 위한 연구를 진행할 계획이다.

6. 참고문헌

[1] UPnP, http://www.upnp.org/

[2] H. Okamura, "Adaptive resource management system for home-area networks," *IEEE Distributed Computing Systems Workshop*, Apr. 2001.

[3] Y. Tomloka, "Architecture for presentation coordination under a multiple device environment," *IEEE Consumer Communications and Networking Conference*, 2004.

[4] K. Mills, C. Dabrowski, " Adaptive jitter control for UPnP M-search," *IEEE Communications,* vol.2, 2003.

[5] M. Ditze, T. Bresser, F. Berger, "Resource Adaptation for Audio-Visual Devices in the UPnP QoS Architecture," *IEEE AINA,* vol.2, pp.543-547, 2006.

[6] Ubuntu, http://www.ubuntu.com/

[7] Portable SDK, http://pupnp.sourceforge.net/

[8] Clips, http://www.ghg.net/clips/CLIPS.html