주 제

Intentional 네임 시스템의 구조 및 서비스

한서대학교 이재용, 김홍윤

차례

I. 서 론

컴퓨터의 하드웨어 기술의 발전은 생활 속에 수많은 사물에 컴퓨터를 내장하게 되어 엠베디드 시스템의 발전을 이끌었으며 생활 속의 수많은 컴퓨터가 분산되어 존재하게 되었다. 더욱이 HCI(Human Computer Interaction) 기술의 발전은 컴퓨터 언어를 통한 대화, 키보드와 마우스의 처리 및 관리등, 기계중심의 처리에서 인간중심의 처리로 바꿀 수 있게 하였다. 민간 기업들과 대학, 연구소에서는 각기 다른 다양한 형태의 모습으로 미래 컴퓨팅 환경을 연구하고 있다.

미래의 컴퓨팅 환경 구축을 위해 진행되는 프로젝트들을 크게 서버중심의 접근방법과 어플리케이션 중심의 접근방법으로 구분할 수 있다. 서버중심의 컴퓨터 내재성을 강조하는 프로젝트로는 IBM社의 Pervasive Computing, HP社의 Cool Town 프로젝트와 켈리포니아 대학에서 진행하고 있는 Smart Dust 프로젝트가 대표적이다[1,2,3]. 어플리케이션 중심의 접근으로는 MS社의 Easy Living 프로젝트[4]와 MIT에서 진행하고 있는 Oxygen 프로젝트가 있다[5].

이와 같이 미래의 컴퓨팅 환경은 수많은 컴퓨터가 도처에 혼재 하는 상황이다. 이러한 환경에서 컴퓨터를 정확히 인지하는 것이 무엇보다도 중요하다. 현재 가장 널리 사용되는 네이밍 시스템이 DNS이다. DNS는 기본적으로 호스트기반 서비스이다.

현재 연구가 진행되고 있는 차세대 네이밍 시스템으로 대표적인 것들은 썬 마이크로 시스템즈社의 Jini[6], 버클리대학의 서비스 발견 시스템[7], MIT의 INS(Intentional Name System)[8]등이 있다. 이러한 서비스들은 기본적으로 어플리케이션 중심 서

비스들이다.

어플리케이션 중심의 네이밍 서비스가 가능하기 위해서는 응용서비스들이 디바이스를 찾기 위해서 네트워크 내에서 high-level 서비스들과 상호대화 할 수 있는 메커니즘이 존재해야 한다. 다시 말해서, 관련 서비스를 발견하고 이용하기 위해 네트워크계층 연결성이 확보되어야 하는 것이다. 예를 들어서, 네트워크에 연결되어 있는 최소부하 프린터나 가장 가까이에 있는 네트워크 웹 카메라의 위치를 찾도록 하기위한 디바이스의 연결성이 필요한 것이다.

현재 구현 단계에 있는 가장 유력한 서비스가 INS(Intentional Name System)이다. "Intentional" 이란 자원의 표현과 이름이 하나이고 같은 것이란 뜻이다. INS안에 있는 네임이 자원의 표현과 같이 보이는 것이다. INS는 Oxygen 프로젝트의 일환으로 연구되고 있으며 주된 구성요소는 주택의 지하실, 사무실 벽, 차의 트렁크 등에 심어지는 컴퓨터, 음성만으로 대화할 수 있어 어디서나 사용자 의사 소통 및 컴퓨터 이용을 지원하는 디바이스, 주변 환경 변화에 맞게 스스로 설정이 가능한 자율 네트워크 환경, 사용자 요구의 변화에 맞는 적절한 서비스를 지원하는 기능이다[5]. 본고에서는 차세대 네임 시스템으로 유력한 INS의 근간이 되는 Oxygen 프로젝트에 대하여 살펴보고, INS의 구성, 동작원리, 현재까지의 연구 결과 등을 살펴본다.

II. Oxygen 프로젝트

Oxygen 프로젝트는 사용자와 시스템 기술들을 조합하여 인간 중심형 컴퓨팅 기술을 가능하게 하며, 음성, 비전 기술들을 이용하여 우리가 다른 사람들과 말하는 것처럼 시간과 노력을 절약하기 위한 목적으로 진행되고 있다[5].

2.1. Oxygen 개요

지난 40년 동안, 사람을 돕기 위해 만들어진 컴퓨터에 오히려 인간이 맞추어져 사용되어 왔다. 인간들은 냉난방 장치가 있는 방에서 컴퓨터를 애지중지 하며 보호했다. 사용자는 컴퓨터 언어로 이야기하고 키보드 와 마우스 등 특정한 장치를 사용하여 컴퓨터와 상호 작용하여야 했으나, 컴퓨터는 사용자가 방에 있는지 조차 인지하지 못한다. 따라서 미래의 컴퓨터 환경은 인간 중심적이어야 하며, 사용자가 보다 적은 노력으로 많은 일을 할 수 있도록 도울 수 있어야 할 것이다.

미래의 컴퓨터는 공기속의 산소처럼 그리고 주변에서 쉽게 보는 건전지와 전원 소켓 같이 환경 속에 녹아 들어가 있게 될 것이다. 또한 사용자가 필요할 때 언제든지, 어디에 있던지 쉽게 정보를 이용할 수 있어야 하며, 임의의 컴퓨터를 이용할 때에도 개별 사용자의 개인적 성향에 맞출 수 있어야 하고, 개인 정보가 누출 되지 않도록 적절한 보안 조치가 이루어 져야 할 것이다. 또한 컴퓨터와의 상호 작용은 새로운 컴퓨터 특수 용어를 타자 치거나, 배울 필요 없이 일상적인 자연언어와 제스처를 이용하여 이루어질 것이다.

새로운 시스템은 일상적으로 반복되는 작업들의 자동화, 필요한 정보의 검색, 그리고 시간과 공간의 제약을 극복하여 다른 사람과 효율적인 협동 작업을 도와주어 사람들만이 할 수 있는 창조적인 일의 생산성을 향상시킬 것이다.

2.2. Oxygen 관련 기술

Oxygen 프로젝트의 목적을 이루기 위하여 사용자 및 시스템 기술을 공동으로 이용하는 다음의 방법들을 제안하고 있다.

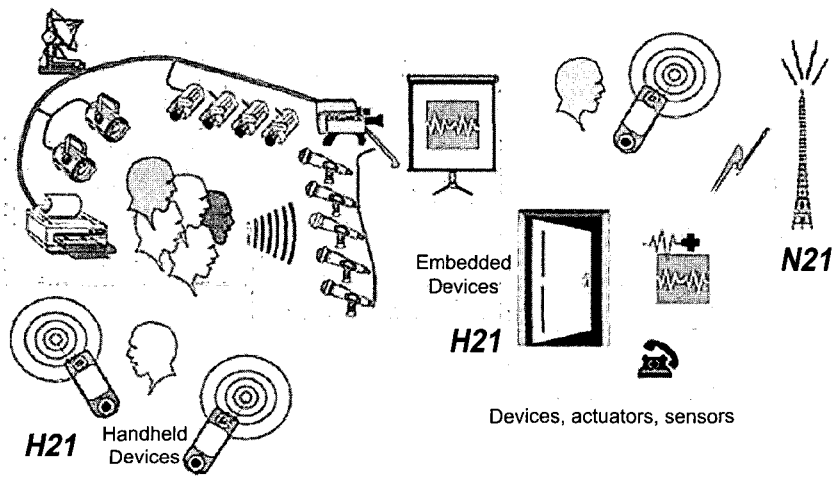
Oxygen 사용자 기술 : 여러 가지 사용자 기술들은 Oxygen의 막대한 계산, 통신, 인지 자원을 이용한다. Oxygen의 시스템 기술은 자동화 기술, 협력 기술, 지식 접근 기술들을 포함한다. 시맨틱 웹(Semantic Web) 기술을 이용하여 메타데이터 관리 및 이용을 통한 개인화 정보 관리 및 협동을 제공한다.

Oxygen 인지 기술 : 마우스나 키보드 대신에 얼굴 표정, 입술 운동 및 시선을 인식하는 기술을 중심으로 하는 상호 작용 방식이다. 음성, 비전 등을 통합하여 인지 기술의 효율성을 증가 시켰다.

Oxygen 소프트웨어 기술 : Oxygen 소프트웨어 환경은 변화를 지원하기 위하여 만들어졌다. 변화란 사용자들을 위한 커스터 마이징 하거나, 사용자들의 요구, 응용 프로그램의 요구, 현재 작동조건, 새로운 소프트웨어, 업그레이드, 또는 다른 여러 가지 원인에 의하여 발생할 수 있다.

Oxygen 디바이스 기술 : (그림 1)에서 E21s라고 불리는 엠베디드 디바이스들은 가정에서, 사무실 자동차등에 지능화 공간을 만든다. E21s는 많은 양의 엠베디드 컴퓨팅 계산을 발생 시키며, 카메라, 디스플레이, 스크린 등의 디바이스에 인터페이스 역할을 하게 된다. H21s라고 불리는 소형 장치는 우리가 어디에 있든 통신하며, 컴퓨팅 기능을 수행하며, PDA의 음성 인식 및 합성 기능을 한다. H21s는 E21s에 의하여 조절되는 지능형 공간 안 또는 밖에서 사용자들을 위한 모바일 액세스 포인트를 제공한다.

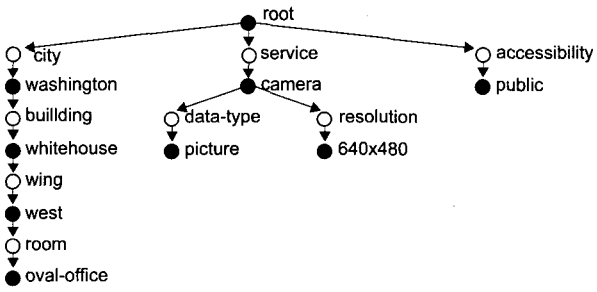
- Oxygen 네트워크 기술 : (그림 1)에서 N21s라고 불리는 자율설정 네트워크는 동적이며 우리가 도달하기 원하는 사람, 서비스 및 자원들에 도달하도록 도와주는 시스템이다. N21s는 저전력의 점대점 통신기능, 대학 캠퍼스 규모를 처리할 수 있는 다중통신 프로토콜들을 제공한다. Oxygen 프로젝트에서는, 이동할 수 있는 자원 발견 문제의 해결책으로 움직이는 객체에 대한 네임의 속성을 연결하는 INS을 제안했다. INS는 주택의 지하실, 사무실 벽, 차의 트렁크 등에 심어지는 컴퓨터를 이용하여, 음성만으로 대화할 수 있어 어디서나 사용자 의사소통과 컴



(그림 1) Oxygen 디바이스 기술

퓨터 이용을 지원하는 디바이스, 주변 환경 변화에 맞게 스스로 설정이 가능한 자율 네트워크 환경, 사용자 요구의 변화에 맞는 적절한 서비스 지원 기능을 목표로 하고 있다.

Oxygen 프로젝트에 사용되는 INS의 연산은 네트워크가 가능하게 된 전문화한 장치들 (예를 들면, 사진기, 커피 기계, 감지기, 등등)까지 미치고, 환경이 동적이기 때문에 구현하기가 쉽지 않다. INS에 있는 네임은 자원과 자료의 네트워크 위치와 속성의 모양으로 기술한다. 즉, INS에서 자원들의 이름은 네트워크 주소보다는 자원의 속성에 의하여 기술된다.



(그림 2) 네임 식별자의 예에 대한 그림 표현

III. INS 시스템

Intentional 네임 서비스가 뜻하는 것은 자원의 표현과 이름이 하나이고 같은 것이라는 의미이다. INS 안에 하나의 네임은 자원의 표현처럼 보인다는 것이다. 이장에서는 INS의 설계 목적 및 동작원리 시스템 구조, 동작 알고리즘등을 살펴본다[8].

3.1. 설계 목적과 구성

미래의 네트워크 환경은 범용 컴퓨터 뿐만 아니라 다양한 자동차, 무선장치들로 특징 지워질 것이다. 이러한 환경은 성능에 있어서의 빠른 변화뿐 아니라, 노드와 서비스의 이동성으로 인하여 유선 네트워크에서 모바일 네트워크로 급속히 변화되고 있다. 이러한 이기종의 모바일 네트워크에서 서비스를 지원하기 위해서 미리 환경설정을 하지는 못한다.

그동안 모바일 네트워크에서 라우팅 문제는 광범위하게 연구되었으나, 자원의 발견 및 서비스 위치에 관한 기능들은 최근에 관심을 가지게 되었다. 그러나 네트워크 인프라에서 가격을 낮추는 것은, 소프트웨어와 서비스관리를 통해서 가능하기 때문에 오히려 자원 발견 및 서비스가 더 중요해지는 것이다. 즉 "최소부하 프린터의 검색"이나 모바일 검색이 필요로 하는 "복제된 서버 발견", 건물의 특정 위치에 있는 모바일 카메라로부터 이미지의 검색 등을 수행하고자 하는 경우에 "효과적인 서비스 발견"이 비용을 감소시킬 것이다.

많은 자원 발견 설계들이 작은 규모의 네트워크[6,9,10]에서나 상대적으로 일반적이지 않거나 드물게 발생하는 동적인 갱신이 발생하는 네트워크[11,12,17,18]에서 설계되었다. 따라서, 많은 자원이 발생하는 경우나 일상적으로 갱신이 일어나는 네트워크에서는 잘 동작하지 않는다[13].

이러한 점들을 고려하여 INS가 중요한 설계 목적으로 하는 것은 다음과 같다.

- Expressiveness : 네이밍 시스템은 디바이스와 서비스들의 폭넓은 변화를 취급하기 위해서는 융통성이 있어야 한다.
- Responsiveness : 네이밍 시스템은 최상의 네트워크 위치에서, 종단 노드와 서비스 모빌성, 성능 변화, 서비스의 변화에 신속히 적응해야 한다.
- Robustness : 네이밍 시스템은 다른 레솔버의

내부 불일치와 서비스 실패에 대하여 탄력이 있어야 한다.

- Easy Configuration : 네임 레솔버는 자신을 구성하기 위해서 최소 수동 작업만을 하여야하며, 서비스의 수동 등록을 요구하지 않아야 한다. 시스템 구성 결과는 레솔버들 사이에서 레솔루션 부하를 자동 분배해야 한다.

```
[city = washington    [building = whitehouse
                       [wing = west
                       [room = oval-office]]]]
[service = camera     [data-type = picture
                          [format = jpg]]
                       [resolution = 640x480]]
[accessibility = public]
```

(그림 3) (그림 2)의 네임 식별자에 대한 연결된 표현

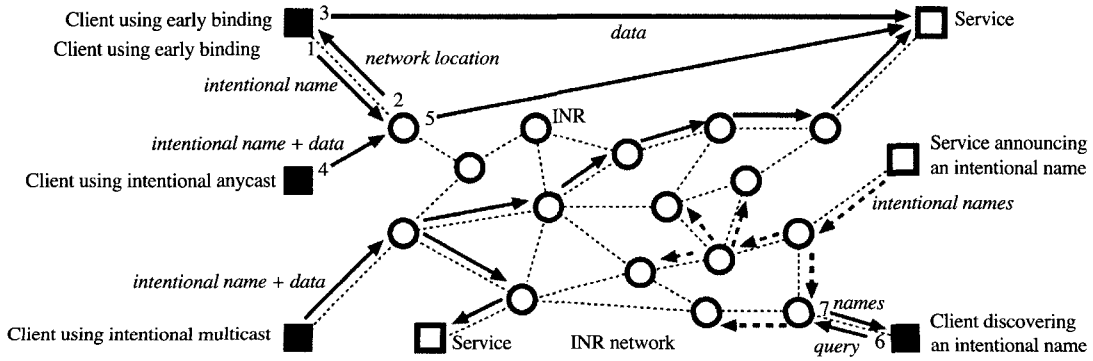
3.2. INS의 구조

INS 어플리케이션은 서비스들이나 클라이언트들이 될 수 있다. 서비스는 기능이나 데이터를 제공한다. 클라이언트들은 기능과 데이터를 요청하고 엑세스한다. INR(Intentional Name Resolver)는 클라이언트가 적절한 서비스에게 요청해야할 경로를 찾는다. 또, 느슨하게 연결된 장치도 찾을 수 있도록 하였으며, 간단한 알고리즘과 프로토콜로 구현되었다.

3.2.1 네임 명세

INS는 attribute와 value의 계층 구조에 기반을 둔 intentional 네임 언어를 사용하여 표현한다. 클라이언트는 자신의 헤더에 있는 네임 식별자를 사용하여 요청한 종착지에 지시한다. 네임 식별자는 간단하고 쉽게 구현되도록 설계되었다. 네임식별자의 의미있는 부분이 attribute와 value의 두쌍이다. 예를 들어 색깔은 attribute이고 빨강은 value이다. Attribute와 관련된 value를 Attribute-Value 쌍, 또는 AV쌍이라고 한다. 네임 식별자는 AV쌍의 계층적 정열로써 부모/자식 관계이다. 예를 들어서 (그림 2)에서 보이고 있는 whitehouse라고 불리우는 건물은 washington 시라는 문맥 안에서만 의미가 있다. AV쌍 building = whitehouse은 AV쌍 city = washington에 종속적이다. 형제관계에 있는 AV쌍은 data-type=picture와 resolution=640x480이다.

네임식별자의 표현은 (그림 3)에서 보이고 있다. 이 표현은 메시지 헤더에 포함된다. 이 문자기반 표현은 HTTP나 NNTP와 같이 문자기반 프로토콜들처



(그림 4) INS의 구조

럼, 디버깅하는데 도움이 될 수 있도록 쉽게 읽을 수 있도록 하였다. Nesting을 표현하기 위해서는 bracket([와])을 사용하였고, AV값은 등호(=)기호로 표시하였다. AV 문자열 중간을 제외하고는 공백을 허락하고 있다..

3.2.2 INR의 동작

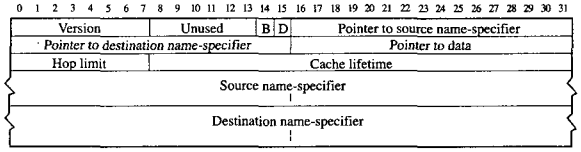
INR은 Ad hoc 네트워크의 임의의 장치나 컴퓨터가 레솔버로써 동작할 수 있으며, 레솔버들이 협력하여 네트워크 전체의 자원 발견 서비스를 제공한다. 메시지가 INR에 도착하면 종착지 이름에 기반을 두어 해석하여 서비스를 레솔브할지, 전달할지를 결정한다. 만일 어플리케이션이 early binding을 선택한다면, INR은 네임을 따라오는 IP 주소의 목록으로 응답한다. 이것은 DNS가 응답하는 것과 유사하며 서비스가 상대적으로 정적일때 유용하다. 다중 IP 주소들이 네임에 따라 올때는 클라이언트가 레솔루션 요청의 결과로부터 가능한 최소한의 매트릭으로 종단노드를 선택할 수 있다.

Early binding은 early-binding flag을 이용하여 전달한다. Late binding 옵션은 두 개를 사용한다. 하나는 intentional anycast와 intentional multicast이다.

(그림 4)에서는 INS의 구조를 보면 어떻게 어플리케이션과 INR들이 상호 대화하는지 보이고 있다. 네임 레솔버는 INR 네트워크 내에서 경로배정을 위해서 서비스의 서술과 네트워크 사이의 연결을 유지한다. 하나의 어플리케이션이 early binding을 사용한 것이 그림의 왼쪽 위에 있다. (1)에서 어플리케이션은 INR에게 레솔브하도록 요청하는 intentional 네임을 보낸다. (2)네트워크의 위치를 수신한다. (3)종착지 어플리케이션에게 데이타를 직접 보낸다. 중앙 좌측에 하나의 어플리케이션이 intentional anycast를 사용한 예이다. (4)어플리케이션이 intentional 네임과 데이터를 INR에게 보낸다. (5)최소 매트릭을 가지는 종착지로 데이터를 보낸다. 아래 왼쪽에는 어플리케이션이 intentional mutlicast를 사용한 예이다. 어플리케이션이 intentional 네임과 데이터를 INR에게 보내면 INR은 INR 네트워크를 통해서 다른 모든 응용들에게로 포워드(forward)한다. 오른쪽 아래에는 하나의 어플리케이션이 INR에게 intentional 네임을 알리는 과정을 보여주고 있다. Intentional 네임은 INR 네트워크를 통해서 지연이 발생하고 있다. (6)네임을 발견한 어플리케이션은 INR에게로 질의를 보내고 (7)네임이 일치하는 네임쌍을 질의어로 수신한다. 송수신 되는 패킷의 구조는 (그림 5)에서 보이고 있다.

3.2.3 네임의 조사및 검출

INR의 주요임무는 해당 네임식별자를 네트워크 위치로 레솔브하는 것이다. 메시지가 INR에 도착하면 해당 네임트리 안에 있는 종착지 네임 식별자를 조사(lookup) 검출하여 네임레코드를 반환한다. 그 네임레코드에는 종착지 IP 주소를 포함하고 있고, 다음 홉 경로의 집합과 네임을 브로드케스팅한다. 네임트리는 네임식별자와 네임레코드 사이의 대응하는 것을 저장하는 자료구조이다. 네임레코드의 정보는 다음 홉으로 가는 경로와 중간 노드의 IP 주소, 경로의 매트릭, intentional anycast의 종착노드 매트릭, 네임레코드의 만기시간이다. (그림 6)에서는 네임트리 T로부터 특별 네임식별자을 위한 네임레코드을 검출하는데 사용되는 NAME-LOOKUP 알고리즘을 보



(그림 5) INS 패킷 형식

이고 있다. 알고리즘의 주된 아이디어는 연속적인 순환호출이 leaf-value 노드가 지시하는 레코드인, 네임레코드의 집합을 가로채는 방법으로 후보 네임레코드 집합 S을 줄인다. 알고리즘이 종료될때, S는 검출네임레코드만을 가지고 있게 된다.

알고리즘은 모든 가능한 네임레코드들의 집합을 초기화하는 S로 시작된다. 그리고 나서, 네임식별자의 각 AV쌍에 대하여 네임트리 안에 있는 일치하는 attribute 노드을 찾는다. 만일, AV쌍안에 있는 값이 와일드카드이면, 일치하는 attribute 노드에 붙어 있는 부트리(subtree)안에 있는 모든 네임레코드를 조합함으로써 S' 을 계산하고, S' 으로 S를 교체한다. 와일드카드가 아니면, 네임트리 안에 일치하는 value노드를 찾는다. 만일 네임식별자나 네임트리의 leaf에 도달하면, 알고리즘은 일치하는 value노드에 네임레코드로 S를 치환한다. 네임식별자나 네임트리의 leaf에 도달하지 않으면, 일치하는 value노드의 부트리의 상대적인 집합을 계산하기 위해서 순환호출을 만든다.

인접한 INR을 갱신하기 위해서는 INR은 네임트리로부터 네임식별자를 받아와서 전송할 필요가 있다. 따라서 네임트리는 INR이 알고 있는 모든 네임식별자를 겹쳐놓고, 네임식별자를 검출한다. GET-NAME 알고리즘이 (그림 7)에서 보이고 있다. 네임트리 T로부터 특별한 네임레코드 r을 위해서 네임식별자를 검출하는데 사용된다. 네임식별자의 구축은 네임레코드의 부모로부터 네임트리의 root로 추적해서 올라가는 동안에 할 수 있다는 것과 이미 구축된 네임식별자의 부분으로 접목될 수 있다는 것이 이 알고리즘의 주된 아이디어이다. TRACE는 함수로 구현되는데, 존재하는 네임 인식자에 접목해 갈 수 있을 때까지 leaf-value로부터 추적해서 올라간다.

```
LOOKUP-NAME(T,n)
  S ← the set of all possible name-records
  for each av-pair p := (n_a, n_v) in n
    T_a ← the child of T such that
          T_a's attribute = n_a's attribute
    if T_a = null
      continue

    if n_v = *                    ▷ wild card matching
      S' ← ∅
      for each T_v which is a child of T_a
        S' ← S' ∪ all of the name-records in the
                  subtree rooted at T_v
      S ← S ∩ S'
    else                          ▷ normal matching
      T_v ← the child of T_a such that
            T_v's value = n_v's value
      if T_v is a leaf node or p is a leaf node
        S ← S ∩ the name-records of T_v
      else
        S ← S ∩ LOOKUP-NAME(T_v, p)
  return S∪ the name-records of T
```

(그림 6) LOOKUP-NAME 알고리즘

```
GET-NAME(T,r)
  n ← a new, empty name-specifier
  T.PTR ← n
  for each T_v which is a parent value-node of r
    TRACE(T_v, null)
  reset all PTRs that have been set to null
  return n

TRACE(T_v,n)
  if T_v.PTR ≠ null          ▷ something to graft onto
    if n ≠ null              ▷ something to graft
      graft n as a child of T_v.PTR
  else                       ▷ nothing to graft onto; make it
    T_v.PTR ← a new av-pair consisting of
              T_v's value and its parent's attribute
    if n ≠ null              ▷ something to graft
      graft n as a child of T_v.PTR
    TRACE(parent value-node of T_v, T_v.PTR)
```

(그림 7) GET-NAME 알고리즘

3.2.4 레솔버 네트워크 : DNS++

서비스와 클라이언트에게 데이터를 포워드하거나 갱신하기 위해서는 INR이 반드시 네트워크에 연결되어서 구축되어야 한다. 이 어플리케이션 수준에서 중첩된 네트워크는 INR과 INR의 왕복 레이턴시를 반영하는 매트릭에 기반을 둔 spanning tree의 형식으로 INR 자동 구성을 함으로써 분산된 방법으로 구성된다. 이 매트릭을 얻기 위해서 INR-ping이 INR에 의해서 진행되며, INR들 사이에 작은 네임을 보내고, 이 메시지가 처리하고 응답하는 시간을 측정한다.

Active INR와 후보 INR의 목록은 DSR(Domain Space Resolver)라는 시스템에서 알려진 목록들로

관리된다. DSR은 현재 유지하고 있는 DNS 서버를 확장하는 것이 순조로울 것다. 고장진단에 대비하기 위하여 복사본을 운영할 수 있어야 한다. DSR은 한 도메인 안에서 현재 active되고 후보된 INR에게 질의를 돌려주는 것이 가능해야 한다. 이 새로운 형태의 도메인 네임서비스를 DNS++라고 부른다.

INR이 깨어나면 DSR로부터 현재 active된 INR의 목록을 얻는다. 새로운 INR은 active되어 있는 INR에게 INRping을 이용하여 최소시간에 응답하는 INR을 찾아낸다. 각 INR이 이와 같이 하면 결과적으로 토폴로지는 spanning tree가 된다.

IV. 네임 레솔버의 구현

Intentional Name Resovler의 구현은 INS/Twine으로 이루어졌다. 이 장에서는 INS의 자원 표현 방법에 대하여 자세히 알아보고, 수치 키들, 배포정보, 해석 질의들에 대한 자원의 표현을 변형하기 위한 방법을 알아보고, INS/Twine 시스템의 구조에 대하여 알아본다[13].

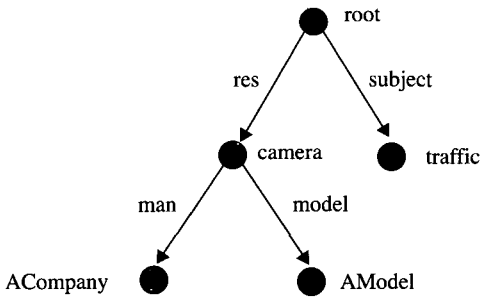
4.1. INR의 구조및 네임 식별자

INS/Twine은 모든 자원이 똑같이 유용하다는 조건에서, 크기조절이 가능한 자원 발견을 가능하게 하기 위해서 설계되었으며, 인터넷을 통해서 세계 전체의 모든 자원을 대상으로 한다는 가정하에 만들어졌다. 다른 도시의 날씨를 보기 위한 카메라나 파일 서버가 될 수 있다.

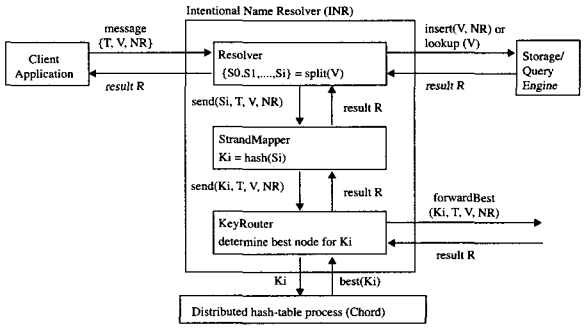
INS/Twine에서의 자원들은 XML이나 INS 네임 식별자와 같이 편리한 언어를 이용하여 AV쌍의 계층 구조로 표현된다. (그림 8)와 (그림 9)에서에서는 간단한 자원의 표현 예를 보이고 있다. 자원은 carmera이고 ACompany에서 제작되었으며 적은 트래픽을 가지고 있다. 보다 편리하게 표현하여 AVTree라고 한다. 각 자원은 자원의 현재 네트워크 위치에 관한 정보를 담고 있는 name-record 쌍으로 표현되며, IP 주소, 전송/응용 프로토콜, 전송계층 포트 번호를 포함하고 있다. 따라서 0이나 잘려진 AV쌍인 질의에 의한 AVTree 형식이 원래의 AVTree의 표현과 같다면 자원은 질의와 일치하는 것이다. 즉, <res>camera<man>ACompany</man></res>이나 <res>camera</res>나 같은 디바이스로 일치된다.

```
<res>camera
    <man>ACompany</man>
    <model>AModel</model>
</res>
<subject>traffic</subject>
```

(그림 8) 간단한 자원의 표현예



(그림 9) 간단한 자원표현의 AVtree의 표현



(그림 10) INR의 3계층 구조

INS/Twine은 정확히 알려진 자원을 표현하는 완벽한 질의와 부분적 질의를 지원한다.

INR는 3개의 계층구조로 구성된다. 레솔버는 클라이언트 어플리케이션으로부터 메시지를 수신한다. 이러한 메시지는 type T(브로트케스트/질의), AVTree V, 네임 레코드 NR을 포함하고 있다. 그 후, V부터의 모든 strand를 추출해낸 다음 각 쌍 하나를 StrandMapper가 키 Ki를 계산한다.

KeyRouter는 마지막으로 Chord와 같이 분산된 해시 테이블 프로세스를 사용하여 INR에 있는 각 키들과 일치하는지 확인한다. 그 후 그 메시지는 레솔버에게 포워드한다. 각 레솔버들의 질의에서는 어플리케이션이 있는 원래의 INR에게로 결과를 다시 돌려보낸다.

4.2. Resolver 계층

INS/Twine 레솔버의 핵심은 표현으로부터 strand를 추출하기 위한 strand 분할 알고리즘이다. 알고리즘의 목표는 의미있는 조각으로 표현을 쪼개내고, 쪼개진 것은 서술 구조를 표현하고 있어야 하며 부분질의를 지원해야 한다.

간단한 strand 검출 방법은 AVTree로부터 AV 쌍을 추출할 수 있어야 하며, 키로서 독립적으로 사상되어야 한다. 따라서 이 규약은 계층적 표현의 장점을 잃어버리게 되며 표현적 질의를 허락하지 않게 된다.

Strand 분할을 위한 Twine 알고리즘은 표현구조를 보호하고 부분질의를 가능하게 한다. (그림 11)에서와 같이 브로드케스트/질의의 표현으로부터 AV의 유일한 접두 부분 수열(prefix subsequence)를 검출한다. 각 부분수열은 strand라고 부른다. 각 strand는 부분 키를 산출하는데 사용된다.

```
Input strand: res-camera-man-ACompany
h1 = hash(res-camera)
h2 = hash(res-camera-man)
h3 = hash(res-camera-man-ACompany)
Output keys: h1, h2 and h3
```

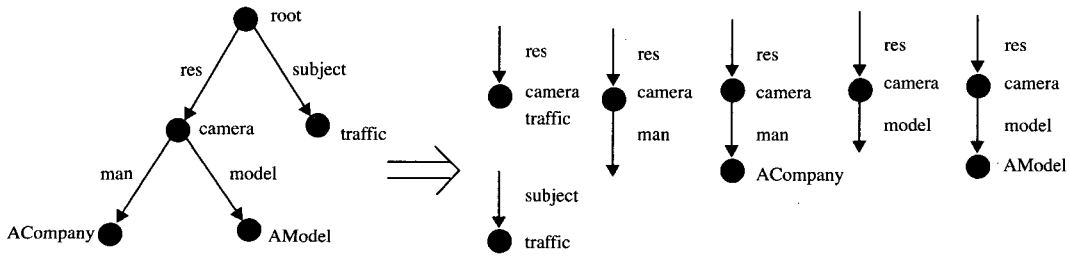
(그림 12) Strand의 부분 추출

4.3. StrandMapper 계층

표현으로부터 검출되는 각 strand는 완벽한 자원의 표현과 질의와 함께 독립적으로 StrandMapper 계층으로 전달된다. StrandMapper는 각 strand의 연관 수치 키를 이용하여 응답한다. 이는 strand의 AV를 합쳐서 하나의 스트링으로 만들고 스트링으로부터 128비트 MD5 해시를 계산한다.

4.4. KeyRouter 계층

StrandMapper 는 KeyRouter 계층에게 키를 전달한다. KeyRouter 계층은 다른 레솔버가 다른 자



(그림 11) 분할을 통한 strand을 이용한 자원의 표현

원으로부터 정보를 저장할 수 있어야 하며 질의를 풀어내는데 참가할 수 있다. KeyRouter를 위한 적절한 설계는 INS/Twine 성능의 제한된 요소가 쉽게 될 수 있도록 정밀해야 한다.

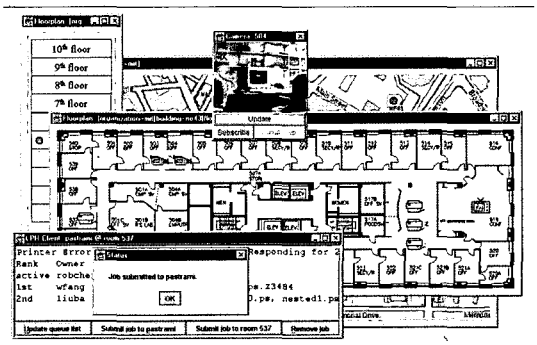
KeyRouter는 분산 해시 테이블로 생각할 수 있으며 네트워크 상에 각 노드는 동적으로 연결된 키의 영역내에서 key-value 바인딩을 유지해야 한다. 이를 수행할 수 있는 여러 개의 peer-to-peer 알고리즘이 제안되어 있으며 그것이 CAN[14], Chord[15], Pastray[16]이다. 키가 주어지면 이 시스템은 네트워크의 노드에서 따라오는 값을 저장할 수 있다. Chord와 Pastray는 모순 없는 해싱과 변하기 쉬운 것의 처리에 기반을 두었으며, 현 네트워크에서 가장 가까이에서 지시하는 노드 인식자와 노드 인식자 사이에서, 인식이 실패한 인식자를 위해서 모든 키에 대하여 응답할 수 있다. 따라서 노드가 시스템에 합쳐지거나 떠나가는 일이 발생하면 시스템의 오동작이 일어 날수 있다. CAN에서도 유사하게 동작한다.

V. 응용 서비스들

INS 유틸리티의 구현을 위하여 세 개의 어플리케이션을 구현하였다. *Floorplan*은 위치 종속 서비스를 제공하기 위한 지도기반 서비스 발견 도구이다. *Camera*는 원격감시를 위한 모바일 카메라 네트워크이며, *Printer*는 사용자의 프린트 요청을 최적의 프린터에게 보내기 위한 부하균형기능을 가진다[8].

5.1. *Floorplan*

*Floorplan*은 INS을 사용하여 발견할 수 있는 다양한 위치 기반 서비스를 보여줄 수 있는 서비스발견도구이다. 사용자가 새로운 영역으로 옮겨가면 그 지역의 지도인 건물 평면도가 화면에 뜬다. *Floorplan*은 INR에게 발견 메시지를 보냄으로써 새로운 서비스에 대하여 기억한다. 이 메시지는 필터로서 사용할 수 있는 네임 식별자를 가지고 있으며 일치하는 모든 네임 식별자는 어플리케이션에게 되돌려 보낸다. 위치와 각 서비스의 형식을 추론하기 위해 돌려받은 네임 식별자 안에 포함된 위치와 서비스 정보를 사용하여 적절한 아이콘을 디스플레이한다.



(그림 13) 응용 서비스들의 실행

*Floorplan*의 중요한 요소가 위치 서버인 Locator이다. *Floorplan*은 Locator로부터 지역 지도를 찾아낸다. 서비스가 예고되거나 시간이 경과되면 새로운 아이콘이 나타나거나 사라진다. 아이콘을 클릭하면 아이콘이 표현하는 적절한 어플리케이션을 깨운다. *Floorplan*은 네트워크 카메라, 프린터나 TV/MP3 플레이어를 위한 디바이스 제어를 포함하는 다양한 서비스들이 발견하도록 한다.

5.2. *Camera*

INS가 사용하는 모바일 카메라 서비스는 *Camera*로 구현된다. *Camera*에는 송신기와 수신기, 두 개의 항목으로 구성된다. 수신기는 intentional 네임으로 표현된 요청을 보냄으로써 *Floorplan*에서 사용자가

선택한 카메라로부터 이미지를 요청한다. 이러한 요청들은 INR에 의해서 *Camera* 송신기로 포워드되며 사진 응답으로 되돌려 보낸다.

카메라 송수신 사이에는 두 개의 전송모드가 있다. 하나는 요청-응답 모드이고, 다른 하나가 응모-형식 대화모드이다. 응모-형식 대화 모드는 intentional multicast를 사용한다. 요청-응답 모드에서 수신기는 적절하게 이름 지워진 전송기로 이미지 요청을 보낸다. 상응하는 송신기가 반대로 수신기로 요청한 이미지를 되돌려 보낸다. 요청자에게 이미지를 되돌려 보내기 위해서 송신기는 유일하게 지시를 해주는 수신기 식별자를 사용한다. *Camera*는 노드나 카메라의 모빌성의 존재를 매끄럽게 연속해서 사용한다.

5.3. *Printer*

*Printer*는 부하 균등 프린터 유틸리티이다. 프린터 클라이언트 어플리케이션은 사용자가 *Floorplan*에 디스플레이되고 있는 프린터 아이콘을 클릭함으로써 실행된다. 프린터 클라이언트 어플리케이션은 여러 개의 특징을 갖는다. 프린터 큐 안에 있는 프린트 작업 목록을 검출하는 기능, 선택한 작업의 제거, 사용자의 퍼미션을 제공하는 기능, 사용자가 파일을 프린터로 의뢰하는 기능이 있다. 파일을 프린터로 의뢰하는 기능은 위치와 부하특징에 따라서 최적의 프린터를 발견하기 위한 intentional anycast를 사용하는 것에 따라 두 가지로 나눌 수 있다.

첫 번째 제출 모드는 직접적으로 "submit job to name"하는 방법이다. 이때, name은 프린터의 intentional 네임이다. 두 번째 모드가 사용자의 위치에 기반을 둔 작업을 의뢰한 것으로 하루단위로 사용할 수 있다. 프린터 서버는 변화된 매트릭을 INR에게 주기적으로 계정의 에러상태, 대기 작업 수, 각 작업의 길이를 알린다. INR은 광고에 기반을 둔 최소부하 프린터에게 작업을 전달하고 선택한 프린터의 사용자에게 확인한다. *Printer*는 부하를 자동적으로 균형있게 조절하기 위해서 적은 부하의 프린터를 위해 적은 매트릭을 광고하며 intentional anycast를 사용한다.

VI. 결 론

미래의 컴퓨팅 환경은 수많은 컴퓨터가 도처에 혼재 하는 상황이다. 이러한 상황에서 원하는 호스트의 원하는 서비스를 발견하고 이용하기 위해서는 네트워크계층 연결성이 확보되어야 한다. Oxygen 프로젝트의 사용자기술, 인지기술, 디바이스 기술들 함께 연구되고 있는 INS는 이러한 네트워크계층의 연결성이 확보되어 이동 중에 최소부하 프린터나 찾고자 하는 적절한 위치의 카메라를 찾을 수 있었다. MIT에서는 intentional 네임 시스템의 설계를 위해서 네임 명세를 설계하였다. INS을 위해서 INR과 DSR이 구성된다. INR은 서비스와 호스트의 연결성을 유지하며 anycast와 multicast기술을 사용한다.

네임 명세의 기술법으로 XML의 사용도 가능할 것이다. INR의 다양한 구현을 위해서는 분산된 해시 테이블 프로세스들인 Chord, CAN, Pastray을 통하여 다양한 구현이 이루어 질것으로 기대된다. 또, DSR은 가존의 DNS에 추가 되어 DNS++라는 이름으로, 차세대 네임 시스템으로 보편적인 내용이 될 것이다. 현재 진행되고 있는 다른 네이밍 연구들도 더 활발히 연구될 것으로 기대된다.

[참 고 문 헌]

[1] Landay, J., A. and Davis, R., C., "Making sharing pervasive ; Ubiquitous computing for shared note taking", IBM Systems Journal, pp351-550, Vol. 38, No. 4, 1999.

[2] Cooltown Labs, Internet URL, http://www.hpl.hp.com/archive/cooltown/

[3] Smart Dust, Internet URL, http://www-bsac.eecs.berkeley.edu/archive/users/warneke-brett/SmartDust/

[4] Easy Living, Internet URL, http://www.research.microsoft.com/easyliving

[5] Oxygen Project, Internet URL, http://www.oxygen.lcs.mit.edu/

[6] Sun Microsystems, "Jini technology architectural overview" http://www.sum.com/jini/whitepaers/architecture.pdf, 1999.

[7] Czerwinski, B. S., Zhao, T., Joseph, A., and Katz, R., "An Architecture for a Secure Service Discovery Service", Proceeding of ACM/IEEE MOBICOM, pp 24-35, August, 1999.

[8] Adjie-Winoto, W., Schwartz, E., Balakrishnan, H., and Lilley, J., "The Design and implementation of an intentional naming system", Proceeding of The ACM Symposium on Operating Systems Principles, pp186-201, 1999.

[9] Guttman, E., and Perkins, C., "Service Location Protocol, Version2", RFC2608. http://www.ietf.org/rfc/rfc2608.txt, 1999.

[10] UPnP Forum, "Understanding Universal Plug and Play: A white paper", http://upnp.org/download/UPNP_UnderstandingUPNP.doc, 2000.

[11] Mockapetris, P., V., and Dunlap, K., J., "Development of the Domain Name System", Proceeding of the ACM SIGCOMM conference, Standford, CA, pp123-133, 1988.

[12] Yeon, W., "Lightweight Directory Access Protocol", RFC 1777, 1995.

[13] Balazinska, M., Balakrishnan, H., and Karger, D., "INS/Twine: A Scalable Peer-to-Peer Archietcture for Intentional Resource Discovery", http://nms.lcs.mit.edu/papers/twine-pervasive02.html

[14] Ratnasamy, S., Francis, P., Handley, M., Karp, R., and Shenker, S., "A scalable content-addressable network", Proceeding of the ACM SIGCOMM Conference, San Diego, CA, pp161-172, 2001.

[15] Stoica, I., Moris, R., Karger, D., kaashoek, M., F., and Balakrishnan, H., "Chard : A scalable peer-to-peer lookup service for Internet applications", Proceeding of the ACM SIGCOMM Conference, San Diego, CA, pp149-160, 2001.

[16] Rowstron, A., and Druschel, P., "Pastry : Scalable, distributed object location and routing for large-scale peer-to-peer systems", IFIP/ACM middleware, heidelberg, Germany, 2001.

[17] 이재용, "인터넷 네임 객체와 속성 관리를 위한 분산네임관리 시스템", 인하대학교, 2000.

[18] 이재용, 이균하, "한글 어순을 따르는 인터넷 한글 도메인 네임 시스템", 한국정보처리학회 논문지, Vol. 05, No. 07, pp1855-1862, 1988.

이재용

1985년 인하대학교 전자계산학과 (이학사)
1990년 인하대학교 전자계산학과 (이학석사)
2000년 인하대학교 전자계산공학과 (공학박사)
2000년 ~ 현재 한서대학교 인터넷공학과 부교수
관심분야 : 의료정보처리및 전송기술, 인터넷관리

김흥윤

1982년 인하대학교 전자계산학과 (이학사)
1984년 인하대학교 전자계산학과 (이학석사)
1996년 인하대학교 전자계산학과 (이학박사)
1995년 ~ 현재 한서대학교 인터넷공학과 부교수
관심분야 : 센서 네트워크, 의료정보처리,
정보 보안