라이브 비디오 스트리밍을 지원하는 RTC 기반 홈 게이트웨이의 설계 및 구현

김 혜 선[†] · 황 기 태[††]

요 약

본 논문은 홈 내부의 Non-SIP 기반 비디오 카메라로부터 홈 외부의 SIP 기반 단말기로 비디오 스트림을 전송할 수 있는 홈 게이트웨이의 설계 구현에 목적이 있다. 홈 게이트웨이는 OSGi 프레임워크를 기반으로 하였으며 홈 내부의 기기로부터 홈 외부의 이동 중인 모바일 사용자를 연결하기 위해 SIP 스택을 내포하는 마이크로소프트 사의 RTC 기술을 이용하였다. 홈 게이트웨이 상에, SIP 세션의 연결 등을 관리하는 RTC 번들을 개발하였으며, 홈 내부의 Non-SIP 기반 비디오 카메라로부터 비디오 스트림을 읽기 위해 가상 비디오 캡처 드라이버를 개발하였다. 최종적으로 설계 구현된 홈 게이트웨이를 테스트하기 위해 SIP 프로토콜을 탑재한 윈도우 메신저를 홈 외부의 단말기로 설정하고 AXIS 2100 UPnP 비디오 카메라를 홈 네트워크에 연결하여 비디오 카메라로부터 이동중인 윈도우 메신저로의 세션 연결과 라이브 비디오의 스트리밍을 실험하고 확인하였다.

키워드 : 비디오 스트리밍, 홈 게이트웨이, SIP, RTC, 홈 네트워킹

A Design and Implementation of a Home Gateway based on the RTC Technology Supporting Live Video Streaming

Hye Sun Kim[†] · Kitae Hwang[††]

ABSTRACT

The objective of this paper lies in the design and implementation of a home gateway supporting live video streaming which flows from the Non-SIP video camera in home to the mobile SIP device outside. We developed the home gateway on the OSGi framework and employed the RTC technology which embeds an SIP stack so that the multimedia session can be established from the home device to the mobile user outside. And also we developed an RTC bundle to manage the session and a virtual capture device driver to read the video stream from the Non-SIP video camera in the home network, and installed them on the home gateway. Finally, we constructed the experimental environment that has the windows messenger as the SIP mobile device and an AXIS 2100 UPnP video camera as a video source, and then tested if the session establishment to the mobile user from the camera and live video streaming work well between them.

Key Words : Video Streaming, Home Gateway, SIP, RTC, Home Networking

1. 서 론

홈 네트워크는 홈 내의 PC, 가전 기기 등을 유무선 네트워크로 연결하여 홈 내 혹은 홈 외부에서 홈 내의 가전 기기들을 감시 제어할 수 있는 통신 서비스 환경이다[1]. 더욱이 최근 모바일 기술의 발전과 병행하여 모바일과 홈 네트워킹 기술을 조합하여 가전 기기들을 제어하고 감시하는 기능 외에 홈 내의 비디오 소스로부터 멀티미디어 스트림을 홈 외부의 모바일 사용자에게 전송하는 기능이 구현 가능하게 되었다[6]. 본 논문에서는 all-IP 기반 무선 인프라 상의 모바일 단말기를 가진 후자의 서비스에 초점을 맞추었으며 이동중인 모바일 단말기 사용자를 단순히 모바일 사용자라고 부른다.

실제 홈 네트워크와 모바일 사용자 사이에 멀티미디어 스트림을 주고 받는 서비스가 실현되기 위해서는 두 가지의 문제가 선행적으로 해결되어야 한다. 첫째, 모바일 사용자의 위치를 홈 내의 장치들이 어떻게 찾아서 연결하며 이동중인 사용자와의 연결을 계속 유지할 수 있는가 하는 것이다. SIP(Session Initiation Protocol)[2]은 IETF(Internet Engineering Task Force)에서 채택된 프로토콜로서 최근에 모바일 통신을 위한 3GPP의 표준으로 채택되는 등 이 질문에 대한 적합한 하나의 답이다. SIP는 최근에 RTP(Real-time Transfer Protocol)와 통합하여 VoIP(Voice over IP), V2oIP(Video Voice over

※ 본 연구는 2005년도 한성대학교 교내연구비 지원 과제임
† 준 회 원 : 한성대학교 컴퓨터 공학과 석사과정
†† 정 회 원 : 한성대학교 컴퓨터 공학과 교수
논문접수 : 2005년 1월 25일, 심사완료 : 2005년 6월 20일

IP) 등 멀티미디어 네트워크 응용들에 사용이 확산되고 있다. 특히 모바일 네트워크에서 SIP와 RTP를 결합하는 많은 연구 과제들이 최근까지 진행되어 왔으며 대표적으로는 미국 Colombia 대학의 CINEMA[3], Vovida 사의 SIPSET[4], 마이크로소프트사의 RTC[5] 등이 있다. 본 논문에서 RTC 기술을 활용하였다.

둘째, 모바일 사용자와 홈 기기들 간의 연결과 그들 사이의 멀티미디어 스트리밍을 위해 홈 게이트웨이 소프트웨어 아키텍쳐를 어떻게 구성하느냐 하는 문제이다. 이 문제를 해결하기 위해 홈 게이트웨이와 홈 네트워크를 제안하고 구현한 연구 사례 들이 있다. 대부분의 기존 연구들은 멀티미디어 스트리밍 보다는 홈 기기들을 제어하거나 홈 내의 스트리밍에 초점이 맞추어져 왔으며[9, 10], 실제 상황에서의 구현 없이 논리적 설계에 머무르거나[6, 7, 8], SIP와 RTP 기능을 모두 갖춘 홈 기기들과 같이 시장에서 일반화 되어 있지 않는 홈 기기들을 기반으로 하는 등 현실성이 부족한 문제점을 가지고 있다[6, 8].

본 논문에서는 홈 내의 Non-SIP 비디오 카메라로부터 홈 외부의 SIP 기반의 모바일 사용자에게 라이브 비디오 스트리밍을 지원하는 홈 게이트웨이의 설계 및 구현에 관해 기술하고 테스트한 결과를 보인다. 본 논문의 주 기여 사항은 홈 내에서 Non-SIP 기반의 비디오 소스로부터 홈 외부에서 이동중인 SIP 모바일 사용자에게 라이브 비디오를 전송하는 아키텍쳐를 실제 구현한 것과 이를 위해 마이크로소프트 사의 RTC 기술을 활용하는 방법론을 설계한 것이라고 할 수 있다.

본 논문은 다음과 같이 구성된다. 2장에서는 연구의 관련 기술들을 설명한다. 3장에서는 본 논문에서 제안한 홈 게이트웨이의 프로토타입 설계와 구현에 관해 상세히 기술한다. 4장에서는 실험 및 결과를 보이며 5장에서 결론을 맺는다.

2. 연구 배경

2.1 홈 네트워킹

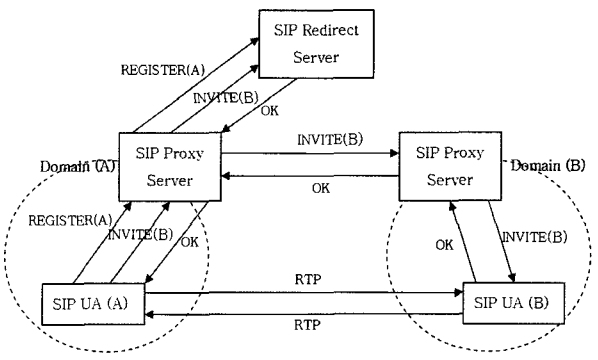
홈 네트워킹 기술은 크게 홈 기기들간에 연결/연결해제 등과 같은 통신 기술과 홈 게이트웨이 기술을 필요로 한다. 주목할 만한 통신 기술로는 Bluetooth, HomeRF, HAVi, IEEE 802.11, IEEE1394, Jini, X10, HomePNA, UPnP 아키텍쳐 등이다[12]. 이들 기술 중 홈 네트워킹의 표준을 정하는 범 세계적인 가전 업체 단체인 DLNA(Digital Living Network Appliance)[13]에서 표준으로 채택된 바 있는 UPnP는 운영체제, 프로그래밍 언어, 물리적인 네트워크 매체 등에 독립적인 장점과, 표준화된 기술만으로 구현할 수 있는 아키텍쳐 등으로 인해 UPnP가 가장 우세한 미들웨어라고 믿는다. UPnP는 서비스를 제공하는 디바이스와 서비스를 이용하는 컨트롤 포인트의 두 요소로 구성된다. 컨트롤 포인트는 디바이스를 발견하고 디바이스가 지원하는 서비스에 대한 내용을 얻고 이를 토대로 디바이스에게 서비스의 개별적인 액션(action) 실행을 호출한다. 한편 디바이스는 호출된 서비스를 처리하며 자신의 상태 값이 변할 때마다 컨트롤 포인트에게 이를 알리는 방식으로 동작한다[11].

홈 네트워크 상에 근본적으로 존재하는 다양한 기기들, 그리고 이질적인 인터페이스 및 네트워크를 제어하기 위해 다양한 여러 표준화 기관에 의해 홈 게이트웨이 프레임워크가 정의되고 있다. OSGi(Open System Gateway Initiative)[14]는 홈 게이트웨이를 위한 대표적인 프레임워크 아키텍쳐로서 현재 많은 기업이나 연구소에서 연구 개발 이용되고 있다. OSGi 프레임워크는 자바(Java) 기반 아키텍쳐이며 홈 네트워크에 번들(bundle)이라고 불리는 서비스를 관리하고 실행시키는 기능을 지원한다. 번들은 기본적으로 OSGi 아키텍쳐에서 명시된 자바 API로 작성되지만 JNI(Java Native Interface)를 이용하여 다른 언어로 작성된 코드를 결합하기도 한다.

2.2 SIP

SIP은 멀티미디어 통신을 위한 호(call) 설정 및 관리를 위한 응용 계층의 시그널링 프로토콜로서 일대일 혹은 일대다로 세션을 생성하고, 수정, 종료하는 기능을 지원한다. 이러한 세션은 인터넷 상에서 멀티미디어 컨퍼런스, 인터넷 전화 호출, 원격 교육, 기타 멀티미디어 통신을 포함한다. SIP를 사용하는 장점은 기존의 세션 설정 프로토콜인 H.323 보다 간단하여 구현 및 포팅이 쉽다는 점과 HTTP나 SMTP와 유사한 텍스트 기반의 메시지를 사용한다는 점, 높은 확장성, 그리고 이동성을 지원한다는 점이다.

SIP 시스템은 1개 이상의 SIP 서버와 SIP UA 들로 구성된다. SIP UA는 주로 단말기에서 실행되는 것으로서 상대방 SIP 주소를 이용하여 상대방 SIP UA에게 세션을 연결하는 응용 소프트웨어이다. (그림 1)은 SIP를 이용하는 전형적인 시스템 모델과 세션 설정 과정을 보여준다. SIP UA인 A와 B는 각각 REGISTER 메시지를 이용하여 자신의 SIP 주소와 현재 IP 주소를 SIP Register 서버에 등록한다. SIP 주소란 전자우편(e-mail)과 비슷한 형식으로서 논리적인 주소이다. A는 B와 세션을 연결하기 위해 자신의 SIP Proxy 서버에게 B의 SIP 주소를 담은 INVITE 메시지를 보낸다. A의 SIP Proxy 서버는 Redirect 서버로부터 B의 SIP Proxy 서버를 찾고 B의 SIP Proxy 서버에게 INVITE 메시지를 보낸다. B의 SIP Proxy 서버는 B에게 INVITE 메시지를 보내고 B의 UA는 세션 설정을 받아들이는 200 OK 메시지를 보내고 다



(그림 1) SIP 시스템 모델 및 세션 설정 과정

시 A의 UA는 ACK 메시지를 보내 세션이 설정된다. 세션이 설정되면 A와 B는 상호 RTP를 이용하여 원하는 멀티미디어 통신을 수행한다.

2.3 SIP 기반 통신을 위한 홈 게이트웨이

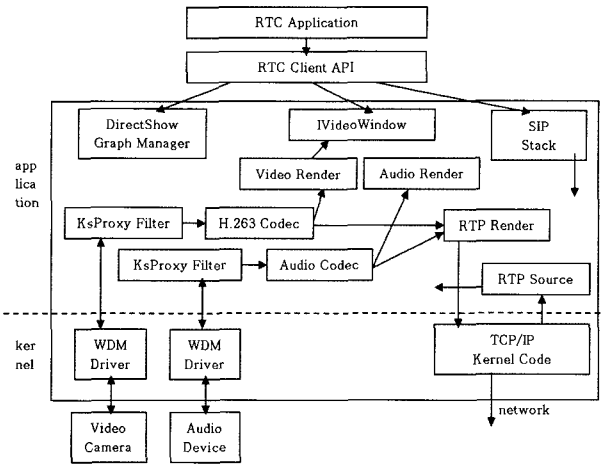
홈 네트워크와 외부의 인터넷 상에서 SIP 기반의 멀티미디어 통신을 지원하는 홈 게이트웨이에 대한 연구는 크게 홈 내의 기기들이 SIP 기반인 경우와 Non-SIP 기반인 경우로 나뉜다. SIP 기반 기기는 기기 내부에 물리적으로 SIP 프로토콜 스택을 구현한 SIP UA를 내장한 것을 말하지만 Non-SIP 기반 기기는 SIP UA를 가지지 않는 경우이다. Non-SIP 기기가 인터넷 상의 SIP 기기들과 통신하기 위해서는 홈 게이트웨이 상의 한 번들이 SIP UA 기능을 가지고 이 기기에 대해 외부에 SIP 기기인 것처럼 에뮬레이션 할 필요가 있다. [6, 7]에서는 SIP 기반 혹은 Non-SIP 기반의 모든 기기들은 구분 없이 SIP 기기로서 외부와 통신할 수 있도록 지원하는 SIP 서비스 아키텍처와 OSGi 번들 인터페이스를 제안하였으며, [8, 9]에서는 인터넷 상의 SIP 기기에서 홈 내의 UPnP 기기들을 제어할 수 있는 홈 네트워킹 모델을 제안한 바 있다. [10]에서는 홈 내에 SIP, Non-SIP 기기 들이 각각 인터넷 상의 SIP 기기와 통신하기 위한 홈 게이트웨이 모델은 제안한 바 있다. 그러나 이들은 공통적으로 세션 연결 이후 멀티미디어의 실시간 통신에 대해서는 거의 다루고 있지 않다.

본 논문은 현존하는 표준 기술을 이용하여 Non-SIP 기반의 홈 기기들이 외부 인터넷 상의 SIP 기기들과 세션을 연결하고 RTP를 이용하여 멀티미디어 통신을 수행하는 프로토타입을 설계 구현하는데 초점을 두었다.

2.4 RTC : 실시간 멀티미디어 스트리밍 플랫폼

마이크로소프트 사는 데스크 탑과 내장형 컴퓨터 상에서 실행되는 윈도우 운영체제 상에서 SIP, SIMPLE(SIP Instant Messaging and Presence Language Extensions), RTP 프로토콜과 관련 기술을 사용하여 음성 및 비디오 통신, 인스턴트 메시징 등과 같은 멀티미디어 통신을 위한 실시간 통신 플랫폼인 RTC(Real-time Communication)를 개발하였다[5]. 현재 Windows XP와 Windows CE 상에서 실행되는 Windows Messenger는 RTC를 이용하는 대표적인 응용 프로그램이다. RTC 플랫폼은 내부적으로 SIP 스택을 사용하여 멀티미디어 세션을 설정하고 로컬 컴퓨터의 비디오 캡쳐 장치로부터 비디오 스트림을 처리하기 위해 DirectShow 미들웨어[15]를 이용한다. 그리고 RTP를 사용하여 DirectShow에 의해 처리된 스트림을 상대편으로 전송한다. 물론 RTC는 네트워크를 통해 전송되는 RTP 스트림을 받아 자기 컴퓨터의 화면에 출력하는 스트림 처리도 수행한다. (그림 2)는 RTC 아키텍처를 사용할 때 비디오 스트리밍을 위한 소프트웨어 구성 요소들을 보여 준다.

RTC 플랫폼은 실행 초기에 시스템에 부착된 미디어 장치들을 탐색하고 미디어 입력과 출력, 미디어 데이터의 변환 등에 적합한 DirectShow 필터 그래프를 구성한다. 예를 들어



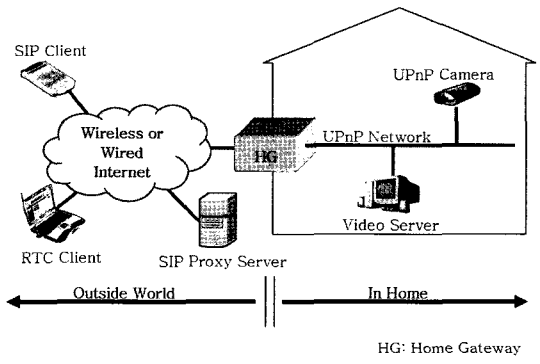
(그림 2) RTC를 사용할 때 소프트웨어 구성 요소

비디오 캡쳐 드라이버를 발견하면 이 드라이버를 호출할 수 있는 소스 필터를 필터 그래프에 삽입하고 비디오를 H.261로 변환하는 인코딩 필터를 연결하고 RTP 패킷으로 전송하는 RTP sender 필터를 파이프라인 방식으로 다시 연결한다. RTC 플랫폼은 마이크나 카메라 등과 같은 미디어 소스 장치와 스피커, 모니터 등과 같은 미디어 출력 장치를 요구하며 이 둘 중 하나는 반드시 필요하다.

3. 홈 게이트웨이 설계

3.1 홈 네트워크 모델

본 논문은 (그림 3)과 같이 UPnP 미들웨어에 기반한 홈 네트워크를 연구의 대상으로 설정하였다. PC 한 대를 비디오 서버로 사용하였다. 비디오 서버의 기능은 홈 네트워크 속의 비디오 소스 장치나 소프트웨어 모듈로부터 홈 게이트웨이로 비디오 스트림을 공급하는 것이다. UPnP 비디오 카메라(AXIS 2100 네트워크 카메라)를 라이브 비디오 소스로 설치하였다. 실제에서는 많은 UPnP 장치 들이 네트워크에 연결되겠지만 연구 초점을 벗어나기 때문에 본 모델에서 생략하였다. 현재 설치된 UPnP 비디오 카메라는 사설 IP를 가진 Non-SIP 장치이다.



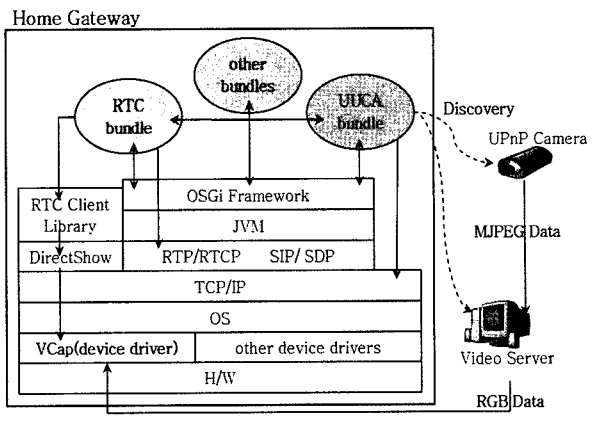
(그림 3) 홈 네트워크 모델

홈 네트워크의 외부의 인터넷은 유선 무선을 구별하지 않으며 모두 가능하다. 인터넷 상에는 모바일 사용자들이 소유하는 RTC 기반의 UA나 SIP UA 들이 존재한다고 가정하며 테스트 시에는 실제 이들을 무선 인터넷에 설치하여 실험을 수행하였다. 그들은 홈 게이트웨이의 도움을 받아 Non-SIP 기반의 UPnP 비디오 카메라로부터 연결되고 비디오 스트림을 받게 된다.

3.2 홈 게이트웨이 소프트웨어 구조

본 논문에서 제안하는 홈 게이트웨이의 소프트웨어 아키텍쳐는 (그림 4)와 같다. 홈 게이트웨이 플랫폼으로 OSGi 프레임워크를 선택하였으며 두 개의 핵심적인 번들을 제작하였다. 그 중 UUCA(Unified UPnP Control Agent) 번들은 UPnP 컨트롤 포인트의 확장한 것으로서 홈 네트워크 상에 존재하는 모든 UPnP 장치들은 관리한다. UUCA는 본 연구팀의 이전 연구에서 이미 개발되었다. 본 논문의 핵심적인 작업으로서 RTC 번들은 홈 외부의 SIP UA 들에게 홈 네트워크 상의 Non-SIP 비디오 카메라를 대신하여 SIP UA로서의 역할을 대신한다. 제작된 RTC 번들은 두 부분으로 구성된다. 자바 언어 부분은 OSGi 프레임워크와 접목하기 위한 것이며 실제 많은 부분은 C++ 언어로 구현되었다. 후자의 코드들은 RTC API를 호출하여 홈 외부의 모바일 사용자와 SIP 세션을 연결하고 홈 내의 비디오 소스로부터 비디오 스트림을 실시간으로 전송한다.

비디오 프레임은 UPnP 비디오 카메라 즉, 비디오 소스 장치로부터 홈 게이트웨이 상의 RTC 번들로 직접적으로 스트림 될 수 없는 몇 가지 이유가 존재한다. RTC 번들은 (그림 2)에서 보인 바와 같이 DirectShow source filter(KsProxy filter)로부터 비디오 프레임을 공급 받는다. 그리고 이 source filter는 로컬 컴퓨터 상에 설치된 비디오 캡쳐 드라이버로부터 RGB 포맷의 프레임을 요청하여 공급 받는다. 그러나 UPnP 비디오 카메라는 홈 게이트웨이의 로컬 장치가 아니라는 데 문제가 있다. 이 문제를 풀기 위해 VCap(Virtual Capture Driver)와 비디오 서버를 본 프로토타입에서 구현하였다. VCap은 가상 캡쳐 드라이버로서 비디오 서버에 접속하여 비디오 프레임을 공급 받는다. VCap은 근본적으로 실수 연산을 수행하지 못하도록 금지되어 있는 커널 모드 코드이기 때문에 전송 받은 비디오 프레임의 포맷을 변형하는 등의 작업은 할 수 없다. 그러므로 VCap이 요구하는 형태로 비디오 프레임을 전송할 수 있는 비디오 서버 등과 같은 중재 장치나 소프트웨어가 필요하다. 본 논문에서 사용한 AXIS 2100 UPnP 카메라는 MJPEG 포맷의 비디오 프레임을 출력하므로 VCap이 원하는 RGB 포맷으로 변형하기 위해 (그림 4)와 같이 비디오 서버가 존재한다. 비디오 서버는 굳이 하나의 독립된 컴퓨터에 따로 설치되어 존재할 필요가 없다. 홈 게이트웨이 상에서 하나의 응용프로그램으로 존재할 수도 있으며 홈 서버에 탑재되어 실행될 수도 있다.



VCap : Video Capture driver

(그림 4) 홈 게이트웨이 소프트웨어 구조

3.3 홈 게이트웨이 서비스

3.3.1 UPnP 기기 관리 서비스 : UUCA 번들

UPnP 프로토콜의 탐지(Discovery) 과정을 통해 UPnP 컨트롤 포인트는 네트워크 상에 존재하는 UPnP 디바이스들의 리스트와 각 디바이스의 서비스 목록 그리고 개별적인 액션(action)을 모두 파악할 수 있다. 원래 디바이스를 제공하는 업체에서 디바이스를 제어하기 위한 소프트웨어로 컨트롤 포인트를 함께 제공하는데 이렇게 되면 홈 게이트웨이에는 UPnP 디바이스의 개수만큼의 번들이 실행되는 복잡성이 존재한다. 본 논문 연구의 사전 연구를 통해 홈 네트워크 상에 존재하는 모든 UPnP 디바이스를 제어할 수 있는 통합 컨트롤 포인트인 UUCA를 개발하였으나 본 논문에서는 구체적인 아키텍쳐를 다루지 않는다. UUCA 번들은 홈 네트워크 내에 존재하는 모든 UPnP 디바이스의 리스트와 서비스, 액션들에 관한 정보 DB를 유지 관리하며 새로운 UPnP 디바이스 홈 네트워크에 연결되거나 어떤 UPnP 디바이스가 홈 네트워크로부터 분리되면 빠른 시간 내에 정보 DB를 갱신하며 다른 번들이 이 정보들을 접근할 수 있도록 한다.

3.3.2 비디오 스트리밍 서비스 : RTC 번들

RTC 번들의 한 인스턴스는 인터넷 상의 SIP 시스템에서 하나의 가상적인 SIP UA로서의 역할을 수행한다. 즉, 홈 네트워크 상에서 멀티미디어 스트리밍의 능력을 갖춘 UPnP 기기가 인터넷 상의 SIP UA와 멀티미디어 통신을 수행하기 위해 홈 게이트웨이 상에서 에이전트로서의 역할을 하는 것이다.

UPnP 비디오 카메라는 MJPEG 프레임을 출력하며 이 프레임들은 비디오 서버를 거쳐 RGB 프레임들로 변환된다. RGB 프레임들은 홈 게이트웨이의 VCap를 통해 읽혀 지고 홈 게이트웨이의 내부의 DirectShow의 KsProxy filter에게 공급된다. 이 필터로부터 다시 RGB 프레임은 H.261 코덱을 거치면서 변환되고 RTP 스트림화 되어 상대편 SIP UA에게 전송된다. <표 1>은 RTC 번들이 외부에 노출한 인터페이스를 보여주며 (그림 5)는 RTC 번들을 구성하는 RTC_Thread의 알고리즘을 기술한다.

〈표 1〉 RTC 번들의 인터페이스

번들 인터페이스	기 능
start()	RTC 번들의 서비스를 OSGi 프레임워크에 등록하고 RTC_Thread를 생성하며 RTC_Thread에게 WM_ RTC_LOGON 메시지를 보내 RTC 번들의 SIP 주소를 SIP Proxy 서버에 등록하게 한다.
RTC_Activate()	RTC_Thread에게 WM_RTC_INVITE 메시지를 보내 상대편 SIP UA와 세션을 연결하고 비디오를 전송하도록 지시한다.
RTC_Bye()	RTC_Thread에게 WM_RTC_BYE 메시지를 보내 비디오 전송을 중단하고 세션을 종료한다.
stop()	RTC 번들을 종료하고 RTC_Thread에게 WM_ RTC_STOP 메시지를 보내 스레드를 종료시킨다.

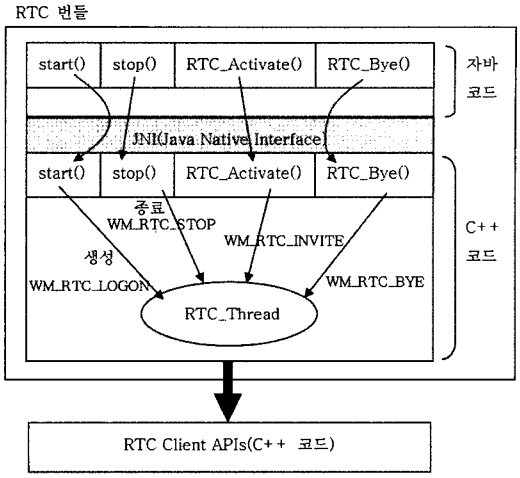
```
Algorithm RTC_Thread
input: none
output: none
{
        RTC Client API를 사용할 수 있도록 초기화한다.;
        자신의 SIP 포트로 사용하는 TCP 포트 5060 번을 연다.;
        bool RTC_run_state = false;
        while(true){ /* RTC_Thread message handler */
                메시지 큐에서 메시지를 message 변수로 가져온다.;
                switch(message){
                case WM_RTC_LOGON: // SIP Proxy 서버에 로그인
                        SIP Proxy 서버에게 REGISTER 메시지를 보내
                        자신의 SIP 주소를 등록한다.;
                        break;
                case WM_RTC_INVITE: // 상대방에게 콜을 요청
                        if(자신의 SIP 주소가 등록되었다면){
                                상대방 SIP 단말기에게 INVITE 메시
                                지를 보내 세션을 생성한다.;
                                RTC_Thread 자신에게 WM_RTC_
                                RUNNING 메시지를 보냄;
                        }
                        else {
                        등록완료를 위한 약 300ms의 시간을 지연한다.;
                        자신에게 WM_RTC_INVITE 메시지를 보낸다.;
                        }
                        break;
                case WM_RTC_RUNNING: // 비디오 전송
                        RTC_run_state = true; break;
                case WM_RTC_BYE: // 상대방에게 콜 해제를 요청
                        상대방 SIP 단말기에게 BYE 메시지를 보내 세션
                        을 종료한다.;
                        RTC_run_state = false; break;
                case WM_RTC_STOP: // RTC_Thread를 종료
                        RTC_Thread를 종료한다.; break;
                }
        }
}
```

(그림 5) RTC 번들의 RTC_Thread 알고리즘

3.3.3 RTC 번들의 구현

마이크로소프트 사에서 제공하는 RTC Client API 들이 C++ 코드로 작성되어 있기 때문에 RTC 번들은 크게 자바 코드 부분과 C++ 코드 부분으로 나누어 작성되었으며 이들은 JNI(Java Native Interface)로 연결되었다. 번들이 동작하는 구조는 (그림 6)과 같다.

<표 1>에 나열된 인터페이스 중 start()와 stop()은 Bundle Activator라는 자바 추상 클래스의 추상 메소드(abstract method)로 정의된 멤버이며 이들은 OSGi에서 표준으로 정의



(그림 6) RTC 번들의 구조

되어 있다. 번들을 초기에 실행 시킬 때 번들 내의 start() 메소드를 호출하며 번들을 종료 시킬 때 stop() 메소드를 호출하는 등 OSGi 프레임워크와 번들의 관계는 매우 단순하다. 번들을 구현하는 사용자는 기본적으로 start()와 stop() 메소드를 구현하여야 하며 그 외 멤버들은 사용자의 필요에 의해 추가된다.

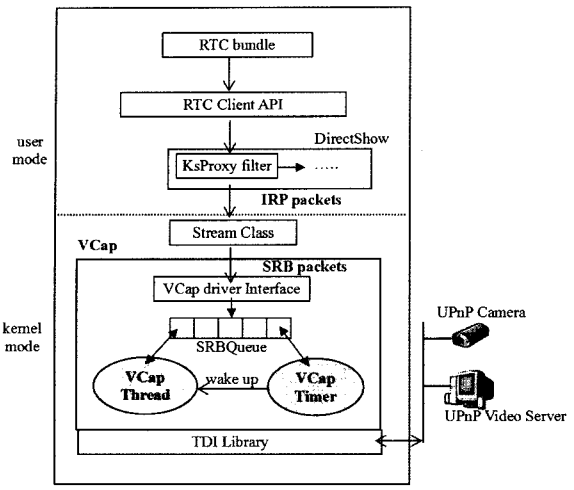
본 연구에서는 번들을 작성하기 위한 기본 클래스인 Bundle Activator 클래스를 상속받아 RTCBundleActivator 클래스를 작성하고 start(), stop() 메소드 뿐만 아니라 추가적으로 RTC_Activate(), RTC_Bye() 메소드를 구현하였다. 이 메소드들은 <표 1>에 설명된 내용과 같은 기능을 수행하며 (그림 6)과 같이 JNI를 통해 각각 C++ 함수를 호출하며 다시 이 C++ 함수들은 RTC Client API들을 호출한다.

RTC 번들은 초기에 RTC_Thread를 생성하고 다른 번들에 의해 RTC_Activate() 메소드가 호출되었을 때 즉, 홈 외부의 SIP 단말기에 연결하고 비디오 스트리밍을 시작할 시점이 되었을 때 RTC_Thread가 RTC Client API 들을 호출하며, 이때부터 외부 SIP 단말기와의 세션 연결 및 비디오 스트리밍은 RTC Client와 DirectShow에 의해 자동으로 움직인다.

3.4 가상 비디오 캡쳐 드라이버(VCap)

VCap은 윈도우 운영체제의 드라이버 중 스트림 클래스 미니 드라이버(Stream Class Mini driver) 타입으로 구현되었다. VCap이 시스템에 설치될 때 운영체제는 이 드라이버로부터 몇몇 정보를 얻어 KsProxy 필터라는 DirectShow의 소스 필터를 자동 생성한다. 이 필터는 실행 시간에 VCap에게 주기적으로 비디오 스트림을 요구한다.

(그림 7)은 VCap과 상위 계층의 소프트웨어 모듈 간의 관계를 보여주며 (그림 8)은 VCap 내의 루틴들의 알고리즘을 기술한다. 운영체제나 응용프로그램은 IRP(IO Request packet) 패킷을 통해 디바이스 드라이버에게 입출력(IO)을 요청하며 커널 내의 Stream Class에서 IRP 패킷을 SRB(Stream Request Block) 패킷으로 변환하여 VCap에 전달한다. VCap은 SRB 패킷을 분석하여 요청 IO의 종류를 결정하고 해당 요청을 서비스한다.



(그림 7) 가상 비디오 캡쳐 드라이버(VCap)의 구조

```
int VideoFrameRate; /* 프레임수/초 */

Algorithm VCapIF
input: SRB 패킷
output: none
{
        SRB 타입을 분석한다.;
        switch(SRB 타입) {
                case SRB_INITIALIZE :
                        VCapThread 를 생성하고 실행시킨다.;
                        VideoFameRate 을 결정한다.;
                        break;
                case PLAY_MODE : // 재생모드로 변환
                        VCapTimer를 호출한다.;
                        break;
                case SRB_READ_DATA : // 재생 중에서 비디오 데이
                        터 요구
                        VideoSRBQueue에 SRB를 삽입한다.;
                        break;
                case … // 다른  SRB에 대해서는 생략한다.
        }
}

Algorithm VCapTimer
input: none
output: none
{
        while(true) {
                if(VideoSRBQueue 가 empty) break;
                else VCapThread를 깨운다.;
        }
        1/VideoFrameRate 후에 VCapTimer가 호출되도록 커널 타이머
        를 설정한다.;
}

Algorithm VCapThread
input: SRB 패킷
output: none
{
        while(true) {
                if(VideoSRBQueue 가 empty가 아니라면) {
                        VideoSRBQueue로부터 SRB를 하나 얻어온다.;
                        SRB를 분석한다.;
                        TDI 인터페이스를 호출하여 비디오 서버로부터
                        비디오 프레임을 가져온다.;
                        SRB 패킷에 연결된 버퍼에 비디오 프레임을 복
                        사한다.;
                        SRB 패킷에 대한 처리가 끝났음을 커널에게 알
                        린다.;
                }
                누군가 깨울 때까지 잠잔다.;
        }
}
```

(그림 8) VCap 드라이버 루틴 들의 알고리즘

VCap은 크게 VCap 드라이버 인터페이스(VCapIF), 타이머 루틴(VCapTimer), 캡쳐 스레드(VCapThread) 등 3 부분으로 구성된다. VCapIF는 SRB 패킷을 VCap 내부의 SRB 큐에 삽입하고, 만일 잠자고 있다면 VCapThread를 깨운다. 드라이버 실행 초기에 VCapTimer은 비디오 프레임의 전송률과 같은 주기로 콜백(callback)되도록 설정된다. VCapTimer은 SRB 큐에 하나 이상의 SRB 패킷이 존재하는 한 VCap Thread를 깨워 SRB 패킷을 처리하게 한다. VCapThread는 초기에 설정된 비디오 서버와의 TCP 연결 상에서 비디오 프레임을 읽는다. 그리고 SRB 패킷이 가리키는 버퍼에 저장한다. SRB 큐가 비어 있으면 VCapThread는 잠을 잔다.

VCap이 커널 모드에서 실행되므로 이에 따른 두 가지 제약 사항을 가진다. 첫째 실수 연산을 실행할 수 없다. 이 때문에 VCap은 압축 또는 인코딩된 비디오 데이터를 디코딩하는 등의 작업을 수행할 수 없다. 그러므로 비디오 서버로부터 반드시 YUV나 RGB 등과 같이 디코딩된 형태의 비디오 프레임을 받아야 한다. 둘째 TCP 통신시 소켓(socket) 라이브러리를 이용할 수 없다. 그러므로 비디오 서버와 TCP 통신을 위해 커널의 TCP 스택을 직접 호출할 수 있는 TDI(Transport Driver Interface) 커널 라이브러리를 사용하였다.

3.5 비디오 서버

본 논문에서 설계 구현한 비디오 서버는 UPnP 카메라로부터 MJPEG 스트림을 받아 프레임 단위로 RGB 형태로 변환하여 VCap 드라이버에 TCP 소켓을 통해 전송한다. 비디오 서버는 UPnP 카메라와 VCap 드라이버와 접속을 위해 각 각 소켓을 연다. UPnP 카메라로부터 도착하는 MJPEG 스트림은 JPEG 단위로 분리되어 내부적인 JPEG 큐에 삽입된다. 하나의 JPEG이 하나의 프레임에 해당한다. 그리고 각 프레임은 RGB 형식으로 변환되어 RGB 버퍼 큐에 삽입된다. 이 두 큐의 길이는 제한적이다. 비디오 서버는 VCap으로부터의 전송 요청이 시작되면 전송 중단이 요청될 때까지 RGB 버퍼 큐에 있는 RGB 프레임을 계속적으로 전송한다.

4. 프로토타입 테스트

4.1 프로토타입 구현 요소

본 논문에서 구현된 프로토타입 시스템은 <표 2>와 같은 요소를 이용하였다.

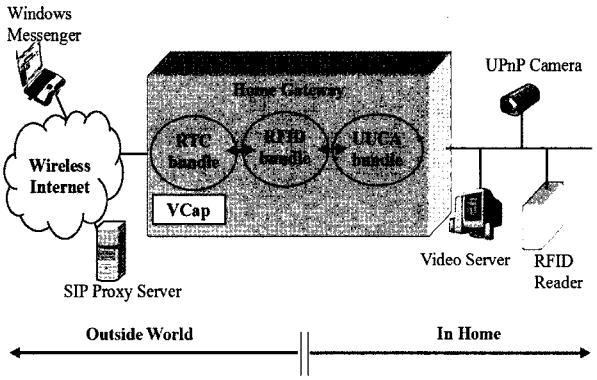
4.2 RFID를 이용한 테스트 시나리오

본 논문에서 구현된 홈 게이트웨이와 홈 네트워크를 테스트하기 위해 (그림 9)와 같이 RFID 판독기를 홈 네트워크에 부착하고 홈 게이트웨이에서 RFID 번들을 제작하여 설치하였다. RFID 번들은 RFID 판독기로부터 읽혀진 태그 ID를 전송 받아 그 유효성을 확인한다.

테스트를 위한 시나리오는 다음과 같다. 홈 게이트웨이와 홈 네트워크 시스템이 설치된 사무실이나 개인 집을 가정하여 보자. 사무실 입구나 현관에는 무인 카메라가 설치되어 감

〈표 2〉 구현 요소

요 소	내 용
홈 네트워크	UPnP 미들웨어, Ethernet
홈 카메라	AXIS2100, UPnP 비디오 카메라
OSGi	OSGi 3.0 IBM SMF
비디오 서버	홈 네트워크 내의 독립된 PC 상에 설치
UPnP 스택	인텔 사 제공 공개 소스를 수정
RTC 라이브러리	마이크로소프트 사의 RTC Client API 1.2
모바일 단말기	무선 인터넷에 연결된 노트북
모바일 SIP UA	마이크로소프트 사의 Windows Messenger
번들	자바와 C++ 혼용. JNI 인터페이스로 연결



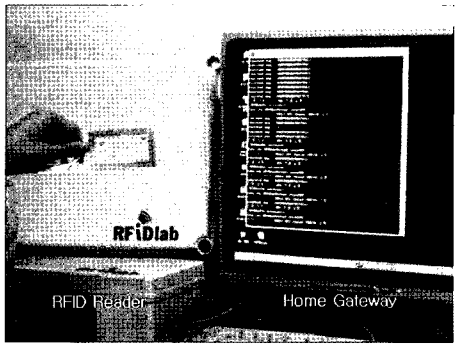
(그림 9) RFID 리더기를 가진 홈 게이트웨이 실험 환경

시하고 있지만 무인 카메라로부터의 비디오를 늘 지켜보고 있지 않고서는 외부로부터 침입 순간을 포착하기 어렵다. 누군가 허락되지 않은 RFID 태그 ID를 가지고 RFID 리더기에 출입을 시도하는 경우를 가정해보자. RFID 번들은 허락되지 않는 태그 ID를 발견하면 RTC 번들을 호출한다. RTC 번들은 집주인의 SIP 주소를 가지고 SIP Proxy 서버를 통해 집주인의 현재 IP 주소를 알아내고 Windows Messenger와 같은 모바일 SIP UA나 RTC UA를 가진 바깥의 집 주인과 SIP 세션을 연결한다. 그리고 나서 RTC 번들은 RTC Client API를 호출하여 DirectShow의 필터 그래프를 구성하고 VCap으로부터 비디오 프레임들을 공급 받는다. 이는 결국 UPnP 비디오 카메라로부터 비디오 프레임을 공급 받게 되며, 집 주인에게 RTP를 이용하여 최종적으로 비디오 프레임을 전송한다.

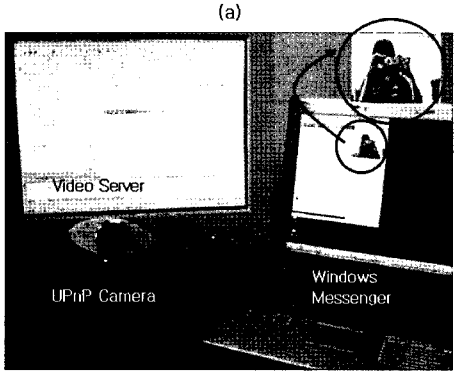
홈 게이트웨이는 집주인이 어디에 있든지 간에 SIP 주소를 이용하여 집주인에게 무인 카메라의 비디오를 집주인에게 전송하게 된다.

4.3 테스트

(그림 10)은 시연하는 모습을 보여준다. (그림 10) (a)에서 RFID 판독기와 홈 게이트웨이, 비디오 서버, 그리고 UPnP 기반의 AXIS2100 네트워크 카메라를 볼 수 있으며 (그림 10) (b)에는 인터넷 상에서 비디오 스트림을 받는 사람의 SIP 단말기로서 Windows Messenger 를 탑재한 컴퓨터를 볼 수 있



(a)



(b)

(그림 10) RFID 판독기를 이용한 비디오 스트리밍 작동 시연

다. 유효하지 않은 RFID 태그를 단말기에 갖다 대자 홈 게이트웨이는 원격 SIP 단말기와 세션을 연결하고 AXIS2100 카메라의 비디오 스트림을 전송하는 모습을 보여준다.

5. 결 론

본 논문은 홈 내의 Non-SIP 비디오 카메라로부터 홈 외의 SIP 기반 모바일 사용자에게 비디오를 전송하기 위해 OSGi 프레임워크와 마이크로소프트 사의 RTC 기술을 이용하며 홈 게이트웨이를 설계 구현한 사례를 기술하였다. UPnP로 홈 네트워크를 구성하고 홈 게이트웨이 상에 가상 비디오 캡쳐 드라이버를 구현하였으며 Non-SIP 비디오 카메라로부터 SIP UA에 비디오 스트림을 실시간으로 전송하는 RTC 번들을 OSGi 상에서 구현하고 RFID 판독기와 Windows Messenger를 이용하여 홈 게이트웨이와 홈 네트워크를 테스트하였다. OSGi나 UPnP 등과 같이 본 개발에서 사용된 기술들은 모두 표준화되어 있기 때문에 비록 현재 개발된 시스템은 윈도우 XP에서 작동하지만 윈도우 CE와 같은 내장형 시스템에서 쉽게 포팅될 수 있다.

참 고 문 헌

[1] Moyer S., Marples D., and Tsang S., "A protocol for wide area secure networked appliance communication", *IEEE Communications*, Vol.39, No.10, pp.52-59, Oct., 2001.

[2] SIP Forum, www.sipforum.com
[3] CINEMA project, www.cs.Columbia.edu/IRT/cinema
[4] SIPSET, www.vovida.com
[5] RTC, msdn.microsoft.com
[6] D. Bushmitch, W. Lin, A. Bieszczad, A. Kaplan, A. Papageorgiou, A. Pakstas, "A SIP-based Device Communication Service for OSGi Framework," *Proc. of CCNC 2004* pp.453-458, 5-8 Jan., 2004.
[7] Pavlin Dobrev, David Famolari, Christian Kurzke, and Brent A. Miller, "Device and Service Discovery in Home Networks with OSGi," IEEE Communications Magazine, pp.86-92, Aug., 2003.
[8] Tsang, S., Marples, D., Moyer, S, "Accessing networked appliances using the session initiation protocol", *Proc. of IEEE International Conference on Communications*, Vol.4, pp.1280-1285 June, 2001.
[9] 김동균, 전병찬, 윤홍수, 이상정, "SIP와 UPnP를 이용한 광대역 인터넷망에서의 정보가전 제어", 한국정보과학회 02 가을학술발표논문집(3) pp.283-285, 10월 2002년.
[10] Takeshi Saito, Ichiro Tomoda, Yoshiaki Takabatake, "Gateway Technologies for Home Network and Their Implementation", *Int. Conf. Dist. Comp. Systems Workshop*, pp.175-180, 2001.
[11] UPnP, www.upnp.org
[12] Henrik Wessung, Home Access Technology
[13] DLNA, www.dlna.org
[14] OGSi, www.osgi.org
[15] DirectShow, msdn.microsoft.com

김 혜 선

e-mail : happyi00@hansung.ac.kr
2004년 한성대학교 컴퓨터공학과(학사)
2004년~현재 한성대학교 대학원 컴퓨터공학과 석사과정
관심분야 : 모바일 컴퓨팅, 유비쿼터스 컴퓨팅

황 기 태

e-mail : calafk@hansung.ac.kr
1986년 서울대학교 컴퓨터공학과(학사)
1988년 서울대학교 대학원 컴퓨터공학과 (공학석사)
1994년 서울대학교 대학원 컴퓨터공학과 (공학박사)
2000년~2001년 University of California, Irvine,의 방문 교수
1994년~현재 한성대학교 컴퓨터시스템공학부 교수
관심분야 : 모바일 컴퓨팅, 유비쿼터스 컴퓨팅