보안성과 유연성을 갖는 피어-투-피어 인터넷 음성 통신 서비스의 구현

이주호° 정충교
강원대학교 컴퓨터정보통신공학과
alwaysshiny@gmail.com° ckjeong@kangwon.ac.kr

Implementation of a Secure and Flexible Peer-to-Peer VoIP Service

Juho Lee° Choongkyo Jeong
Dept. of Computer and Information Communications Engineering
Kangwon National University

요 약

중앙 집중형 음성 통신 시스템이 확장성, 신뢰성, 그리고 초기 비용의 문제를 가짐에 따라, 분산 아키텍쳐 기반의 피어 투 피어 음성 통신 서비스를 실현하기 위한 연구가 최근 활발히 이루어지고 있다. 그러나 대부분의 연구들은 완전 분산 시스템을 구현하지 못하고 사용자의 인증 및 보안 기능을 위해 중앙의 서버에 일부 의존하는 형태를 취하고 있다. 이러한 형태로 구현된 서비스는 완화되기는 했지만 확장성, 신뢰성, 초기 비용 등의 문제를 그대로 가지고 있으며 또 다양한 서비스로 유연하게 확장하기가 쉽지 않다. 이 연구에서는 사용자 인증 및 보안 기능까지도 완전한 분산 형태로 구현할 수 있는 피어 투 피어 음성 통신 서비스의 구조를 제안하고 구현 사례를 보였다. 또 제안된 서비스 구조가 유연한 확장성과 발전성을 갖는다는 것을 보이기 위해 웹 서비스와의 연동을 통한 확장된 서비스 사례를 보였다. 추가적으로 우리의 제안 구조가 기존의 중앙 집중형 음성 통신 시스템과 호환성을 갖출 수 있도록 하는 방안도 제시한다.

1. 서 론

최근 인터넷 망의 보급이 빠른 속도로 확산됨에 따라 인터넷 기반의 음성 통신 서비스에 대한 많은 관심이 모아지고 있다. 최근까지 인터넷 기반의 음성 통신 서비스에 관한 많은 연구가 진행되었으나 기반 아키텍쳐보다는 음성 데이터 송·수신을 위한 프로토콜의 연구에 그 초점이 모아졌고, 기반 아키텍쳐로는 중앙 집중형 아키텍쳐를 사용하였다. 그러나 이러한 중앙 집중형 아키텍쳐는 확장성, 신뢰성의 문제점을 가지며 많은 초기 비용 및 유지 비용을 필요로 한다. 이러한 까닭으로 최근에 분산 아키텍쳐 기반의 피어 투 피어 음성 통신 서비스에 대한 다양한 연구가 이루어졌지만, 사용자 인증 및 보안 기능 등을 위해서는 중앙 집중형 서버에 여전히 의존하고 있어 서비스의 유연한 확장 및 발전이 쉽지 않다.

이러한 상황에 착안하여 우리는 최근의 피어 투 피어 음성 통신 서비스가 갖는 제한적 분산구조의 문제점을 해결하고 나아가 서비스의 확장성 및 발전성을 갖는 완전 분산형 피어 투 피어 음성 통신 서비스를 제안하였으며 모델 시스템을 구현하였다.

아래는 우리가 연구의 목표로 삼은 속성들이다.

보 안 사용자 입장에서 보안은 매우 중요한 사항이다. 우리는 특정 서버에 의존하지 않고 보안 문제를 해결하고자 한다. 보안 문제는 그 속성상 중앙 집중형 서버의 도움을 받을 때 가장 구현하기 쉽다. 그러나 우리는 완전 분산형 구조를 목표로 하고 있으므로 보안 문제도 역시 전용 서버 없이 완전 분산형으로 구현하고자 한다.

유연성 및 발전성 우리의 시스템 구조는 인스턴트 메시징 서비스, 메일 서비스, 다자간 영상회의 등이미 대중적으로 많이 사용되고 있는 기존의 서비스를 수용할 수 있어야 하고 향후 추가로 개발되고 보급될 미래의 서비스에 대해서도 유연한 수용 능력을 갖추어야 한다.

확장성 및 신뢰성 제안 구조는 별도의 설정 및 관리 기능이 없이도 점진적으로 확장되어 대규모화할 수 있어야 한다. 늘어나는 사용자들이 기존 시스템에 투명하고 자연스럽게 수용되어야 한다. 또한 일부 컴퓨터에 결함이 생긴 경우에도 시스템의 전체 기능에는 이상이 생기지 않아야 한다.

우리는 이런 속성을 갖는 시스템을 위해 분산 해쉬 테이블을 기반으로 하는 오버레이 네트워크를 구성하는 방안을 제시한다. 분산 해쉬 테이블이 오버레이 네트워크를 자동적으로 구성하고 우수한 확장성 및 신뢰성을 갖는다는 것은 이전의 많은 연구들[1, 2, 3, 4]에 의해서 입증되었다. 확장성과 신뢰성은 이런 분산 해쉬 테이블의 속성을 그대로 이어받음으로써 달성한다. 또한 보안성을 위해서는 오버레이 네트워크에 사용자들의 인증서와 사용자에 관한 정적/동적 정보를 공개키 기반 암호화 기법을 활용하여 저장하고 필요시 이를 읽어 들여 사용하는 방법을 사용한다. 완전 분산형 구조와 지리적 (혹은 네트워크) 투명성은 그 자체가 유연성을 의미하며 우리의 연구에서는 이러한 유연성을 이용하여 다른 기존 응용 서비스와 연동되고 새로운 융합 서비스로 발전할 수 있음을 구현 사례를 통해 보인다.

2장에서는 본 논문의 배경이 되는 피어 투 피어 오버레이 네트워크에 대해서 설명하였고, 3장에서는 이전의 관련 연구들을 나타내었다. 4장에서는 서비스를 위한 디자인 및 구현에 대

해서 설명하고, 5장에서는 서비스의 실현 가능성을 입증하는 간단한 테스트를 나타내었다. 6장과 7장에서는 각각 결론과 참고 자료를 나타내었다.

2. 피어 투 피어 오버레이 네트워크

피어 투 피어 오버레이 네트워크는 하위 계층에 투명성을 갖도록 응용 계층에 논리적 토폴로지를 구성하여 구현되고 물리적 토폴로지에 독립적이다. 이러한 오버레이 네트워크는 구조적인 오버레이 네트워크와 비구조적인 오버레이 네트워크로 나뉜다.

구조적인 오버레이 네트워크는 분산 해쉬 테이블을 사용하여 논리적으로 트리(tree), 링(ring), 다면체(hypercube), 혹은 하이브리드(hybrid) 형상을 갖는다. 분산 해쉬 테이블이 갖는 논리적 형상에 따라 데이터를 검색 혹은 라우팅하는 알고리즘이 다르다. 이러한 구조적인 오버레이 네트워크는 우수한 확장성, 강인성, 그리고 결함에 대한 저항성을 갖는다.

비구조적 오버레이 네트워크는 Napster[5], Gnutella[6] 등의 서비스에서 사용된 기술로서 특정한 논리적 토폴로지가 존재하지 않고, 플러딩이나 중앙 서버 인덱싱을 사용하여 데이터를 저장하고 검색한다. Gnutella의 경우에는 많은 사람들이 관심이 있는 데이터의 검색에는 효율적(짧은 응답 시간)이나 그렇지 않은 데이터의 검색에는 많은 시간이 할애되거나 검색이 불가능한 경우가 발생하기도 한다. Napster는 중앙의 서버에서 데이터의 목록을 관리하는 방식이어서 분산형 시스템의 장점을 충분히 갖지 못한다.

앞으로 특별한 언급이 없는 한 이 논문에서 오버레이 네트워크는 구조적 오버레이 네트워크의 의미로 사용한다.

2.1 분산 해쉬 테이블

우리는 중앙 서버에 의존하지 않는 피어 투 피어 방식의 음성 통신 서비스 개발을 목표로 하기 때문에 피어 노드들이 유기적인 관계를 가지고 오버레이 네트워크를 구성해야 한다. 분산 해쉬 테이블은 오버레이 네트워크를 구성할 때 사용하는 핵심 컴포넌트로서 자동적으로 오버레이 네트워크를 구성할 수 있도록 한다. 이러한 자기구성력은 긴급 상황과 같은 특정 상황에서 빠른 시간 내에 서비스의 기반이 되는 오버레이 네트워크를 구성할 수 있도록 해 준다. 분산 해쉬 테이블을 사용하여 오버레이 네트워크를 구성할 경우 많은 사용자들이 오버레이 네트워크에 참여하여도 각 피어 노드의 유지 비용이 증가하지 않고 라우팅 비용은 사용자 수의 로그에 비례하기 때문에 확장성을 가지며 일부 노드의 결함에 대해서도 강인성을 갖기 때문에 신뢰성 있는 서비스를 가능하게 한다.

우리는 오버레이 네트워크를 구현하기 위해 Bamboo[1, 7] 분산 해쉬 테이블을 사용하였다. Bamboo 외에도 Pastry[2], Chord[3], CAN[4]과 같은 분산 해쉬 테이블 기술들이 사용될 수 있지만 Bamboo는 다른 기술에 비해 지리적 인접성을 고려할 수 있으므로 실시간성 서비스에 보다 적합하고 무선 환경에서 특히나 효율적이다. 또한 Bamboo는 Java로 구현되어 있어 플랫폼에 독립적이라는 장점을 가진다.

분산 해쉬 테이블을 사용하여 오버레이 네트워크 내에 참여하는 모든 노드들은 자신을 고유하게 구분 지을 수 있는 노드 아이디를 갖는다. Bamboo는 기본적으로 노드의 인터넷 주소 혹은 노드의 공개키를 해쉬 함수(SHA-1)에 적용시켜 생성한 160bit 크기의 데이터를 노드 아이디로 사용한다. 또한 분산 해쉬 테이블은 기본적으로 put/get 인터페이스를 통해 분산 스토리지 서비스를 지원하는데 put 인터페이스는 특정 개체를 오버레이 네트워크 내에 저장하고자 할 때 사용하며, get 인터페이스는 특정 개체를 읽어올 때 사용한다. 모든 개체는 (key, value) 형태로 나타내지며 오버레이 네트워크 내에서 key와 가장 가까운 노드 아이디를 갖는 노드에 저장된다.

Bamboo 오버레이 네트워크는 링 구조와 트리 구조가 혼합된 하이브리드 구조를 사용하는데, 오버레이 네트워크 내의 모든 노드는 고유한 노드 아이디를 가지고 전체 링에서 한 좌표를 차지한다. 각각의 노드는 논리적 토폴로지 상에서 자신과 근접한 이웃 노드와 정보를 교류하여 트리 구조의 라우팅 테이블을 생성하고 관리한다. 즉 네트워크의 여러 노드들이 모여 논리적 링형 구조를 형성하지만 각 노드의 라우팅 테이블은 트리 구조를 가진다.

그림 1은 Bamboo 오버레이 네트워크 위에 있는 노드 65a1fe에서 d46a1c의 키를 갖는 개체를 저장하고자 할 때 이 키와 가장 가까운 아이디를 갖는 노드 d467c4로 개체가 라우팅되는 과정을 보여준다. 매 단계마다 개체의 키와 노드의 아이디가 상위 비트의 더 많은 비트들에서 일치하도록 맞추어 나가므로 최대 log(N) 스텝에 목적지에 도달하게 된다. 여기에서 N은 노드의 수이다.

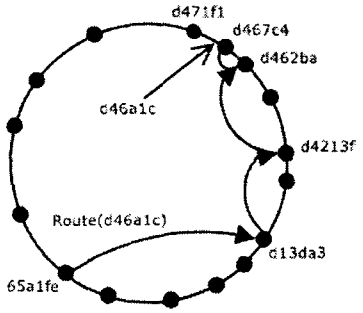

그림 1 노드 65a1fe에서 노드 d46a1c로의 메시지 라우팅

Bamboo는 기본적으로 노드 아이디를 사용하여 오버레이 라우팅을 통해 메시지를 전송하지만 상대의 아이피 주소와 Bamboo 시스템 데몬이 대기 중인 포트 번호를 알고 있는 경우에는 오버레이 라우팅 대신 인터넷 라우팅을 통해 직접적으로 메시지를 전송할 수도 있다.

3. 관련 연구들

연구 [8]에서는 분산 해쉬 테이블을 사용한 SIP(Session Initiation Protocol) 기반의 피어 투 피어 인터넷 텔레포니 서비스의 구현 방안을 개념적으로 제안하였으나 사용자 입장에서 매우 중요하게 다루어져야할 서비스 보안에 대해서는 고려하지 않았다. Skype[9, 10]는 많은 사용자들을 확보하고 있는 피어 투 피어 아키텍쳐 기반(Kazaa[11])의 인터넷 텔레포니 응용 프로그램이다. 그러나 Skype는 사용자가 서비스에 로그인할 때 중앙 집중형 로그인 서버를 사용하여 사용자 인증을 처리하기 때문에 완전 분산형 서비스라고 볼 수 없고, 음성 통신 및 인스턴트 메시징 서비스를 위한 특화된 아키텍쳐를 가지고 있어 서비스의 발전 및 유연한 확장이 쉽지 않다. 이에 반해 우리의

연구는 보다 안전한 피어 투 피어 음성 통신 서비스가 완전 분산 방식으로 구현이 가능함을 보이고 이렇게 할 경우 서비스가 다양한 장점을 가질 수 있다는 데 초점을 맞추었다.

연구 [12]에서는 오버레이 네트워크 내의 노드들을 인증하기 위하여 노드들의 인증서를 오버레이 네트워크에 저장하여 공개키 기반의 인증 및 암호화 통신이 가능하도록 하는 기법을 제시하고 그 활용 예로서 피어 투 피어 이메일 서비스를 구현하였다. 본 논문에서는 이와 유사한 기법을 사용하여 사용자들을 인증하고 사용자 데이터에 기밀성을 부여하였다.

연구 [13]에서는 Bamboo 오버레이 네트워크 상에서 효율적으로 멀티캐스트 트리를 구축하는 기법을 제시하였다. 이 연구는 우리의 서비스가 1:1 통신에서 1:N 통신으로 발전할 수 있다는 가능성을 긍정적으로 뒷받침 해준다.

4. 디자인 및 구현

오버레이 네트워크에는 우리가 구현한 서비스를 사용하는 사용자와 그렇지 않은 사용자들이 혼재되어 있다. 이 논문에서 사용자가 서비스에 로그인 한다는 말은 우리가 구현한 서비스를 지원하는 노드를 사용하여 오버레이 네트워크에 참여한다는 것을 의미한다. 사용자는 서비스에 로그인하기 위하여 자신의 아이디와 공개키/개인키 쌍을 필요로 한다. 이 연구에서는 자신을 고유하게 나타낼 수 있고 많은 상대에게 공개될 수 있는 이메일 주소를 사용자의 아이디로 사용하였다.

우리는 서비스 구현을 위해 아래에서 자세히 설명할 인증 개체, 사용자 상태 개체, 사용자 환경 개체를 분산 해쉬 테이블에 저장하도록 설계하였다.

4.1 분산 해쉬 테이블에 저장되는 개체들

1) 인증 개체

(h(e-mail), certificate): 인증 개체는 자신을 고유하게 구분지을 수 있는 이메일 주소의 해쉬 값을 키(key)로, 인증서를 값(value)으로 갖는다. h(•)기호는 적절한 해쉬 함수를 의미한다. 피어 투 피어 음성 통신 서비스를 원하는 사용자는 우선적으로 자신의 인증 개체를 put 인터페이스를 통해 오버레이 네트워크에 저장해야 한다. 인증서에는 사용자의 이름, 이메일 주소, 공개키, 그리고 인증서의 유효기간을 나타내는 TTL, 인증 기관의 전자서명이 포함되어 있다.

이메일 주소는 외부에 공개되어 있기 때문에 악의적인 공격자는 인증 개체의 인증서를 수정할 수 있다. 그러나 인증 개체의 키로 사용하는 이메일 주소와 인증서 내의 인증된 이메일 주소는 항상 일치해야하기 때문에 사용자들은 인증 개체의 유효성을 쉽게 확인할 수 있다.

이렇게 인증 개체를 오버레이 네트워크에 저장하고 그 안에 들어 있는 공개키를 사용하면 음성 통신의 상대편 신분을 사전에 확인할 수 있고 필요한 경우 여러 종류의 데이터에 기밀성을 부여할 수 있게 된다.

2) 사용자 환경 개체

(h(user_config_key), $E_{user\_config\_key}$(user_config)): 사용자는 서비스를 사용하는 동안 주기적으로 사용자 환경 개체를 오버레이 네트워크에 저장한다. $E_k(C)$는 k를 비밀키로 삼아 적절한 방식으로 데이터 C를 암호화한 결과를 의미한다. 사용자 환경 개체는 사용자 본인 외의 다른 사람이 그 내용을 볼 수 없도록 암호화한 user_config 데이터를 값으로 갖는다. user_config 데이터에는 사용자가 사용하는 응용 프로그램의 설정 정보, 친구 리스트 등 개별 사용자마다 관리되어야 할 환경정보들이 포함되어있다. user_config 데이터를 암호화할 때 사용한 비밀키 user_config_key는 다른 사람이 볼 수 없도록 사용자의 공개키에 의해 암호화된 상태로 아래에 설명할 사용자 상태 개체에 기록된다. 사용자 환경 개체는 user_config_key의 해쉬 값을 키로 갖는다. 우리가 구현한 시스템에서는 user_config 데이터를 해쉬 함수에 적용하여 생성한 메시지 다이제스트를 user_config_key로 삼았다.

3) 사용자 상태 개체

(h(pk), (IP+port, Flag, E_{pk}(user_config_key), Signature)): 사용자는 서비스에 로그인 할 때와 로그아웃할 때 사용자 상태 개체를 오버레이 네트워크에 저장하는데 이때 저장될 사용자 상태 중 가장 중요한 것은 사용자가 현재 로그인 상태에 있는지 여부와 로그인 상태에 있는 경우 현재 사용하고 있는 컴퓨터의 아이피 주소와 Bamboo 시스템 데몬(응용 프로그램)의 포트 번호이다. 사용자의 접속 여부 (on-line 혹은 off-line)은 Flag 필드에 표시된다. 사용자 상태 개체는 그 밖에도 사용자 환경 정보를 읽어 낼 때 사용하는 user_config_key를 사용자의 공개키 pk로 암호화한 내용을 포함한다. 또 이런 상태 정보의 위변조를 방지하기 위해 전체 내용에 대한 전자서명을 덧붙인다. 이 사용자 상태 개체의 키로는 사용자의 공개키 pk의 해쉬 값을 사용한다. 즉 누구든지 특정 사용자의 이메일 주소를 알면 그것의 해쉬 값을 키로 하여 그 사용자의 인증 개체를 읽어내고 그 안에 있는 사용자의 공개키를 얻을 수 있으며 이 공개키를 사용하여 상태 객체를 읽음으로써 그 사용자의 현재 상태를 알아 낼 수 있다.

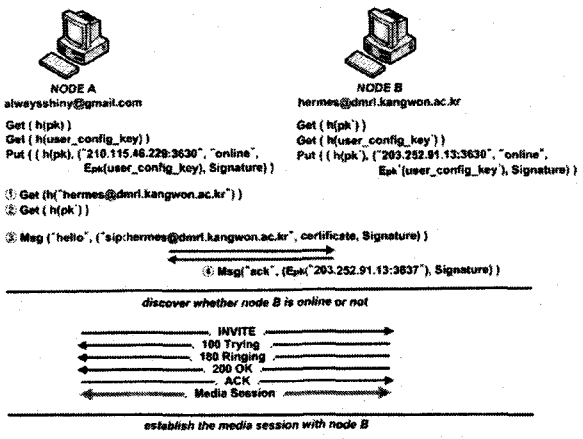

그림 2 음성 통신 세션 설립 과정

4.2 안전한 음성 통신 세션 설립 절차

그림 2는 발신자 alwaysshiny@gmail.com이 착신자 hermes @dmrl.kangwon.ac.kr과 음성 통신 세션을 설립하는 과정을 나타낸다. 이 둘은 각각 노드 A와 노드 B를 사용하여 서비스에 로그인 하는데, 로그인을 위해 자신의 사용자 상태 개체와 사용자 환경 개체를 읽어온다. 그리고 난 후에 현재 사용 중인 노드에 맞게 사용자 상태 개체를 수정하여 오버레이 네트워크에 재 저장한다. 사용자 환경 개체의 활용 과정은 그림에서 생략하였다. 과정 ①, ②는 발신자가 착신자의 인증 개체를 읽어와서 공개키를 확인하고 이 공개키를 사용하여 착신자의 사용자 상태 개체를 읽어오는 과정을 나타낸다. 그 후에 과정 ③,

④에서는 발신자가 착신자에게 로그인 확인 메시지를 전송하고 이에 대한 응답을 받는 것을 나타내는데, 착신자는 수신 메시지에 포함된 인증서와 전자서명을 통해 발신자의 신원을 확인하고 자신의 로그인 여부와 상대의 공개키로 암호화된 음성 통신 세션 관리 데몬의 포트 번호 그리고 이에 대한 전자서명을 응답 메시지에 포함시켜 응답한다. 이때 전자서명이 포함된 까닭은 착신자가 서비스에서 로그오프하고 다른 사용자가 노드 B를 사용하여 서비스에 로그인 한 경우에 발신자가 의도하지 않은 상대와 음성 통신 세션을 설립하지 않도록 하기 위해서이다. 즉, 발신자의 입장에서 응답 메시지를 전송한 상대가 자신이 의도한 착신자임을 확인하기 위해서이다.

과정 ①, ②에서는 Bamboo 오버레이 라우팅을 사용하여 메시지를 전달하고 그 이후에는 상대의 사용자 상태 개체에 기록되어 있는 아이피 주소, Bamboo 시스템 데몬의 포트 번호로 인터넷 라우팅을 통해 직접적으로 메시지를 전달한다. 그림의 하단부에는 발신자와 착신자가 음성 통신 세션을 설립하는 과정을 나타낸다.

우리는 음성 통신 세션을 설립하기 위하여 SIP(Session Initialization Protocol)을 사용하였으나 음성 통신 세션을 설립하는 절차와 이전에 사용자 록업 절차가 서로 독립적이므로 프로토콜에 종속적이지 않으며 SIP가 아닌 다른 적당한 프로토콜을 사용해도 된다.

정리하면 대화하고자 하는 상대의 이메일 주소를 알고 있을 경우 인증기관에서 발행한 인증서를 통해 그 상대를 신뢰할 수 있고 그 상대가 서비스에 로그인되어 있는 지의 여부 와 대기중인 음성 통신 세션 관리 데몬의 포트 번호를 안전하게 확인할 수 있기 때문에 신뢰성 있는 음성 통신 세션을 설립할 수 있다. 이후에 송·수신 하는 음성 데이터는 필요한 경우 서로의 공개키를 사용하여 교환된 세션키로 암호화할 수 있다.

4.3 호환성

이전의 중앙 집중형 음성 통신 시스템과의 호환성은 사용자 입장에서 중요한 요구사항이다. 우리는 오버레이 네트워크 내의 노드들과 중앙 집중형 서버에 의존하는 클라이언트들이 서로 음성 통신 세션을 개설할 수 있도록 하기 위하여 중앙 서버가 오버레이 네트워크에 참여하여 중간 매개체 역할을 하도록 할 것을 제안한다.

그림 3은 분산 아키텍쳐와 중앙 집중형 아키텍쳐가 혼합된 아키텍쳐로 중앙 서버가 오버레이 네트워크에 참여하고 있다. 과정 ①은 sip:someone@testsipclient.com이 노드 A를 사용하여 중앙 서버에 로그인 하는 과정을 나타낸다. 과정 ②, ③, ④는 중앙 서버가 sip:someone@testsipclient.com의 인증 개체, 사용자 환경 개체, 그리고 사용자 상태 개체를 오버레이 네트워크에 저장하는 과정을 나타내는데, 이때 중앙 서버는 자신이 중간 매개 역할을 담당하기 위하여 사용자 상태 개체에는 자신의 아이피와 Bamboo 시스템 데몬의 포트 번호를 작성한다.

가령 발신자 alwaysshiny@gmail.com이 sip:someone@testsipclient.com를 착신자로 하여 음성 통신 세션을 설립하고자 할 경우에, 발신자는 착신자의 인증 개체 및 사용자 상태 개체를 읽어 들인 후 중앙 서버에게 Msg("hello", ("sip:someone@testsipclient.com", certificate, Signature)) 메시지를 전송한다. 이에 대해 중앙 서버는 클라이언트의 음성 통신 세션 관리 데몬의 포트 번호를 포함시킨 Msg("ack", (E_{pk}("210.115.46.231:3637"), Signature)) 메시지를 작성하여 응답한다. 이렇게 할 경우에 오버레이 네트워크에 존재하는 발신자는 중앙 서버에 로그인되어 있는 클라이언트를 착신자로하여 음성 통신 세션을 설립할 수 있게 된다. 이와 반대로 발신자 sip:someone@testsipclient.com이 alwaysshiny@gmail.com을 착신자로 하여 통신하기를 원할 경우에는 중앙 서버가 착신자의 인증 개체 및 사용자 상태 개체를 읽어들여 착신자로부터 노드 B의 아이피 주소와 음성 통신 세션 관리 데몬의 포트 번호를 수신하고 이를 발신자에게 전달한다.

4.4 서비스의 발전 및 유연한 확장

일반적으로 음성 통신 서비스는 발신자가 착신자를 알고 있는 경우에 한하여 통신할 수 있다. 그러나 사용자들은 새로 알게 된 사용자를 착신자로하여 즉시 음성 통신하기를 원할 수 있다. 가령 웹 페이지에는 수많은 사용자들의 정보가 공개되어 있고 사용자들은 웹 서핑 중에 알게된 사용자와 음성 통신하고자 하는 경우가 있을 수 있다. 우리의 연구에서는 이러한 문제를 해결하고 제안된 시스템의 유연성을 입증하기 위하여 웹 서버를 오버레이 네트워크에 참여시켜 음성 통신 서비스를 웹 서비스로 확장시켰다. 이러한 상황에서 웹 서버는 음성 통신을 위한 중개자로서 기능할 수 있다.

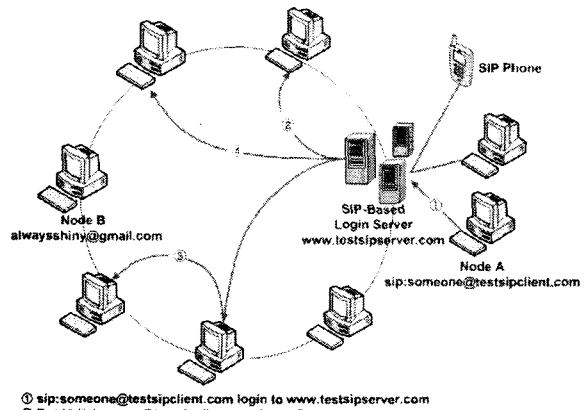

그림 3 중앙 집중형 시스템과의 연동

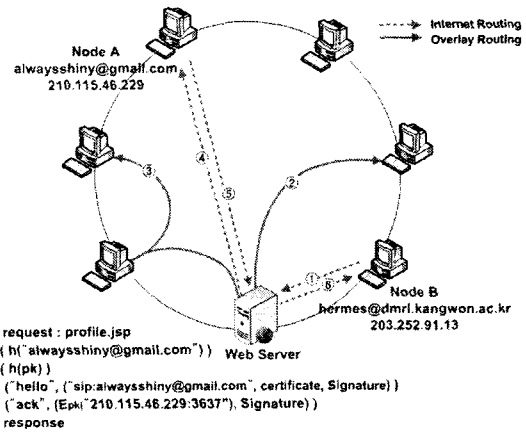

그림 4 웹 서비스로의 유연한 확장

그림 4는 웹 서버가 클라이언트 hermes@dmrl.kangwon.ac.kr로부터 이메일 주소 alwaysshiny@gmail.com이 포함된 웹 페이지를 요청 받았을 때 alwaysshiny@gmail.com가 서비스에 로그인되어 있는 지를 확인한 후 그 상태를 나타내는 아이콘을 웹 페이지에 추가하여 응답하는 과정을 나타낸다. ①은 클라이언트 hermes@dmrl.kangwon.ac.kr이 웹 서버에게 웹 페이지를 요청하는 과정을 나타내고 ②, ③은 각각 alwaysshiny@gmail.com의 인증 개체 및 사용자 상태 개체를 오버레이 네트워크로부터 읽어 들이는 과정을 나타낸다. ④, ⑤는 웹 서버가 alwaysshiny@gmail.com이 사용 중인 노드 A와 로그인 확인 메시지를 주고받는 과정을 나타내고, ⑥은 로그인 상태를 나타내는 아이콘을 추가하여 클라이언트에게 응답하는 과정을 나타낸다. 이에 관련된 테스트 화면은 5.2 절에서 확인할 수 있다.

이 외에도 우리가 구현한 서비스는 원하는 경우에 오버레이 네트워크에 데이터를 저장할 수 있고 특정 상대의 로그인 여부를 실시간으로 확인할 수 있으며 오버레이 네트워크 레벨에서 효율적인 멀티캐스트 트리를 구축할[13] 수 있으므로 인스턴트 메시징 서비스, 이메일 서비스 [12], 다자간 영상회의 등으로 쉽게 발전할 수 있다.

5. 테스트

우리가 구현한 음성 통신 서비스는 분산 해쉬 테이블로부터 상속한 확장성 및 신뢰성을 갖는다. 이러한 장점은 다수의 분산 해쉬 테이블 관련 연구들[1, 2, 3, 4]에서 확인되었으므로 이에 대한 시뮬레이션은 생략하였다. 우리는 서비스의 실현 가능성을 입증하기 위한 간단한 테스트를 아래의 그림 5에 나타나있는 테스트 오버레이 네트워크에서 실행하였다. 실제 오버레이 네트워크에는 우리의 서비스를 지원하는 노드들과 그렇지 않은 노드들이 혼재될 수 있지만, 테스트 오버레이 네트워크 내의 모든 노드들은 우리가 구현한 서비스를 지원하도록 하였다. Bamboo 튜토리얼 [14, 15]은 테스트를 수행하는데 있어 우리에게 많은 참고가 되었다.

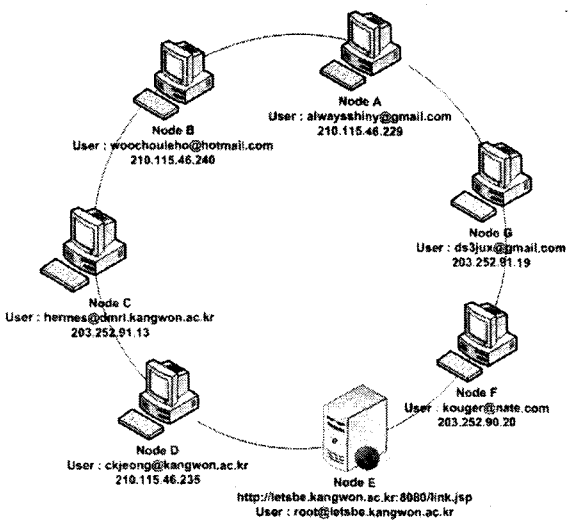

그림 5 테스트 Bamboo 오버레이 네트워크

5.1 메신저 형태의 서비스

그림 6은 alwaysshiny@gmail.com이 서비스에 로그인하여 친구 리스트 내에 포함되어 있는 hermes@dmrl.kangwon.ac.kr과 음성 통신 하는 것을 나타낸다.

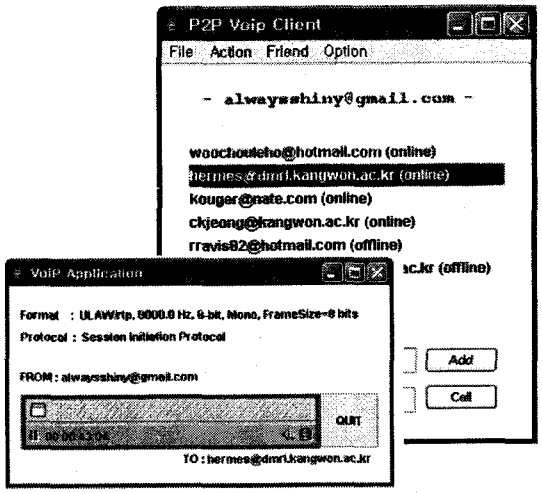

그림 6 메신저 형태의 음성 통신 응용 프로그램

alwaysshiny@gmail.com의 친구 리스트에서 현재 테스트 Bamboo 오버레이 네트워크에 참여하여 서비스에 로그인 한 친구는 이메일 주소 옆에 online이라는 표시를 하였고 음성 통신 진행 상황을 나타내는 응용 프로그램에는 음성 데이터의 포맷과 프로토콜의 정보를 추가적으로 나타내었다.

5.2 웹 서비스로의 유연한 확장

그림 7은 클라이언트 hermes@dmrl.kangwon.ac.kr이 오버레이 네트워크에 참여하고 있는 웹 서버에게 다수의 인물들의 프로필이 기록되어 있는 웹 페이지(http://letsbe.kangwon.ac.kr:8080/link.jsp)를 요청하였을 때 웹 서버가 클라이언트에게 응답하는 페이지를 나타낸다.

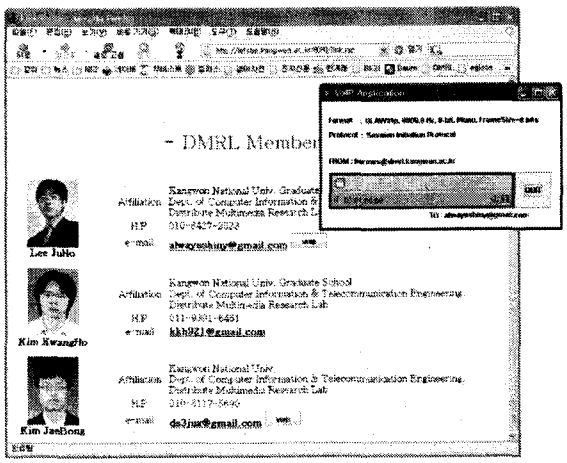

그림 7 웹 서비스로 확장된 음성 통신 서비스

응답된 웹 페이지에는 프로필 내의 이메일 주소 옆에 로그인 상태를 나타내는 애플릿 아이콘이 추가되어 있다. hermes@dmrl.kangwon.ac.kr이 통신하고자 하는 상대의 애플릿 버튼을 누르면 애플릿은 이 두 발신자와 착신자 사이에 음성 통신 세션이 개설되도록 한다.

웹 서비스로의 확장 가능성을 입증하는 이 테스트는 우리가 구현한 서비스가 다양한 서비스들로 유연하게 확장 할 수 있다는 가능성을 입증한다.

6. 결론

우리는 완전 분산 아키텍쳐를 기반으로 한 피어 투 피어 오버레이 네트워크를 제안하고 이를 구현하였다. 피어 투 피어 음성 통신에 관련한 이전의 연구들은 완전 분산 아키텍쳐를 기반으로 사용자의 인증 및 보안 문제를 해결하지 못하였다. 우리는 사용자의 인증서를 오버레이 네트워크에 저장하는 방식으로 이러한 문제를 해결하였다. 그러므로 우리 시스템은 더욱 높아진 확장성과 신뢰성 그리고 안정성을 갖추게 되었다.

또 우리의 음성 통신 서비스는 다양한 서비스로 유연하게 확장할 수 있다는 것을 간단한 테스트를 통해서 입증하였다. 우리가 구현한 서비스는 앞으로 인스턴트 메시징 서비스, 이메일 서비스, 다자간 영상회의 등으로도 쉽게 발전할 수 있다. 추가적으로 우리는 제안된 시스템과 현재 서비스 중인 중앙 집중형 음성 통신 시스템이 호환성을 갖출 수 있도록 하는 방안을 제시하였다.

우리가 구현한 음성 통신 서비스는 완전 분산 아키텍쳐를 그 기반으로 하기 때문에 애드-호크 네트워크에서 보다 효율적일 수 있을 것이다. 앞으로 우리는 이 연구를 애드-호크 네트워크로 확장할 계획이다.

7. References

[1] Sean Rhea, Dennis Geels, Timothy Roscoe, and John Kubiatowicz. "Handling Churn in a DHT". Proceedings of the USENIX Annual Technical Conference, June 2004.

[2] A. Rowstron and P. Druschel. "Pastry: Scalable distributed object location and routing for large-scale peer-to-peer systems". In Proc. MIDDLEWARE'01.

[3] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications", in Proceedings of ACM SIGCOMM'01, San Diego, September 2001

[4] S. Ratnasamy, P. Francis, M. Handley, R. Karp and S. Shenker: "A Scalable Content-Addressable Network", Proc. of ACM SIGCOMM 2001.

[5] Napster : http://www.napster.com/.

[6] Gnutella : http://gnutella.wego.com/

[7] Bamboo : http://www.bamboo-dht.org

[8] Kundan Singh, Henning Schulzrinne: "Peer-to-peer internet telephony using SIP". NOSSDAV 2005: 63-68

[9] Skype : http://www.skype.com

[10] Salman A. Baset and Henning Schulzrinne, "An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol", IEEE Infocom 2006.

[11] Kazaa : http://www.kazaa.com

[12] A. Mislove, A. Post, C. Reis, P. Willmann, P. Druschel, D. Wallach, X. Bonnaire, P. Sens, J. Busca, and L. Arantes-Bezerra. "Post: A secure, resilient, cooperative messaging system". In Proceedings of the Ninth Workshop on Hot Topics in Operating Systems (HotOS '03), Lihue, HI, 2003.

[13] M. Dischinger, "A flexible and scalable peer-to-peer multicast application using Bamboo", Report of the University of Cambridge Computer Laboratory, 2004

[14] Sean C. Rhea. "A Programmer's Tutorial on Event-Driven Programming, Asynchronous Input/Output, and the Bamboo DHT" December 15, 2005

[15] Marcel Dischinger. "Bamboo - A Tutorial" April 2004