DOI: 10.3745/KIPSTC.2008.15-C.3.157

유비쿼터스 서비스 상태지속을 지원하는 안전한 Jini 서비스 구조

김 성 기[†] · 정 진 철[††] · 박 경 노[††] · 민 병 준[†††]

요 약

유비쿼터스 서비스 환경에서는 연결의 신뢰성이 낮고 서비스를 제공하는 시스템에 대한 침입이나 서비스 실패가 발생할 확률이 높다. 따라서 정당한 사용자가 보안상 신뢰할 수 있는 서비스를 중단이나 방해 없이 이용할 수 있게 하는 것이 중요하다. 본 논문에서는 표준 Jini 서비스 환경의 문제점을 지적하고 결함감내 Jini 서비스 개발을 돕는 Jgroup/ARM 프레임워크를 분석한다. 분석을 토대로 보안성과 가용성, 서비스 품질을 만족하는 안전한 Jini 서비스 구조를 제시한다. 본 논문에서 제시한 Jini 서비스 구조는 네트워크 분할이나 서버 붕괴와 같은 결함뿐만 아니라 취약점을 악용한 공격으로부터 시스템을 보호할 수 있으며 Jini 서비스 개체 간에 동적 신뢰를 확립할 수 있는 보안 메커니즘을 제공한다. 또한 사용자 세션별로 서비스 복제를 할당할 수 있어 사용자의 서비스 상태정보 일치를 위한 높은 연산비용을 유발하지 않는다. 테스트베드를 통해 실험한 결과, 서비스 품질 저하를 무시할 수 있는 수준에서 사용자의 서비스 상태지속을 보장하고 높은 보안성과 가용성을 제공할 수 있음을 확인하였다

키워드 : 서비스 생존성, Jini, 유비쿼터스 컴퓨팅, 보안

Secure Jini Service Architecture Providing Ubiquitous Services Having Persistent States

Sung-Ki Kim[†] · Jin-Chul Jung[††] · Kyung-No Park[††] · Byoung Joon Min[†††]

ABSTRACT

The ubiquitous service environment is poor in reliability of connection and also has a high probability that the intrusion against a system and the failure of the services may happen. Therefore, It is very important to guarantee that the legitimate users make use of trustable services from the viewpoint of security without discontinuance or obstacle of the services. In this paper, we point out the problems in the standard Jini service environment and analyze the Jgroup/ARM framework that has been developed in order to help fault tolerance of Jini services. In addition, we propose a secure Jini service architecture to satisfy the security, availability and quality of services on the basis of the analysis. The secure Jini service architecture we propose in this paper is able to protect a Jini system not only from faults such as network partition or server crash, but also from attacks exploiting flaws. It provides security mechanism for dynamic trust establishment among the service entities. Moreover, our secure Jini service architecture does not incur high computation costs to merge the user service states because of allocation of the replica based on each session of a user. Through the experiment on a test-bed, we have confirmed that proposed secure Jini service architecture is able to guarantee the persistence of the user service states at the level that the degradation of services quality is ignorable.

Key Words : Service Survivability, Jini, Ubiquitous Computing, Security

1. 서 론

Jini[1]는 장치의 종류, 통신 프로토콜과 같은 하부 이종성을 극복하면서 네트워크상에 편재된 장치와 소프트웨어 자원의 공유를 지원하는 Java 기반 미들웨어이다. Jini는 가용한 서비스를 Lookup 서비스를 통해 동적으로 발견하고 클라이언트가 요구하는 서비스를 연결시켜주는 메커니즘을 제공한다.

유비쿼터스 서비스 환경은 연결의 신뢰성이 낮아 네트워크 시스템이 분할되기 쉽고 서비스를 제공하는 시스템에 대한 침입이나 서비스 실패가 발생할 확률이 높다. 따라서 정

※ 본 연구는 인천대학교 자체연구비 지원에 의한 것임
† 정 회 원 : 인천대학교 정보기술교육원 초빙교수
†† 준 회 원 : 인천대학교 컴퓨터공학과 석사과정
††† 종신회원 : 인천대학교 컴퓨터공학과 교수
논문접수 : 2008년 2월 22일
수 정 일 : 2008년 3월 18일
심사완료 : 2008년 3월 18일

당한 사용자가 보안상 신뢰할 수 있는 서비스를 중단이나 방해 없이 이용할 수 있게 하는 것이 중요하다.

이러한 요구에 대해서 기존 Jini는 다음과 같은 문제점이 있다. 첫째 임차한 자원에 대한 서비스 실패에 대해서는 결함감내 서비스를 지원하지 않고 있다[2]. 둘째 Jini 연합(Jini federation)[3] 내에 동적 신뢰를 형성 할 수 있는 보안 메커니즘이 부족하다[4].

첫 번째 문제점은 통신상의 연결 실패로 Jini 연합이 분할되거나 서버(service provider)가 서비스 수행에 실패 했을 때 클라이언트가 상태 지속적인 서비스를 이용할 수 없다는 것이다. 이 문제를 해결하려면 서버의 서비스 실패 사실을 적시에 발견하고 제한된 시간 내에 이를 대신할 수 있는 서버가 상태 지속적인 서비스를 클라이언트에게 제공해야 한다. 그러나 기존의 Jini 구조에서는 네트워크가 분할되고 서버가 실패한 사실을 실시간으로 발견할 수 있는 메커니즘이 없다. 또 이러한 실패가 발생했을 때 결함감내를 위한 서비스의 중복수행을 지원하지 않는다.

두 번째 문제점은 분산 환경에서 동적인 Jini 서비스 추가와 변경, 서비스간의 연합이 자유롭기 때문에 야기되는 보안문제이다. 악의적인 서비스와 서비스 프락시가 개입되지 않도록 Jini 연합을 구성하는 개체(entity)간의 신뢰를 제공할 수 있어야 하며 취약점을 악용하는 침입에도 대응할 수 있는 보안 메커니즘이 필요하다.

Jgroup/ARM 프레임워크[5]는 Java 기반 객체 그룹 플랫폼이라는 개념을 도입하여 분산컴퓨팅 환경에서 의존 가능한 서비스(dependable service)를 구축하는 미들웨어 기술을 제시하였다. 분산된 복수의 객체들이 그룹이 되어 하나의 서비스를 책임지는 개념이다. 위에서 언급한 기존 Jini 시스템의 첫 번째 문제점을 해결해주는 프레임워크를 제공한다. 그러나 이 프레임워크를 현실에 응용하기에는 몇 가지 문제가 있다. 하나는 이 프레임워크가 네트워크 분할과 서버 붕괴 상황을 감내하면서 Jini 서비스 개체들 간의 서비스 수행 상태를 일치시키는 문제해결에 집중하다보니 침입이 발생할 경우를 간과하고 있다. 보안성이 없어 침입에 무방비할 뿐만 아니라 시스템 일부에서만 침입에 성공하더라도 서비스 상태 정보 공유를 위한 프로토콜 수행 때문에 Jini 분산시스템 전체에 침해를 끼칠 수 있다. 오히려 기존 Jini 시스템 구조보다 더 공격에 취약하다. 다른 하나는 클라이언트로의 응답을 지연시키는 불필요한 연산과 통신 오버헤드가 존재한다. 그리고 연결을 유지하고 있던 클라이언트의 수가 많을 경우 네트워크 분할이 복구되고 분할된 그룹이 통합 될 때 복제 서버 간에 서비스 상태 정보를 일치(merging)시키는 데 필요한 연산과 통신비용이 높다.

본 논문에서는 Jgroup/ARM 프레임워크의 문제점을 개선하기 위한 방안들을 제시한다. 그리고 Jini 연합 내에 동적 신뢰를 제공하고 결함과 침입을 감내 할 수 있는 안전한 Jini 서비스 구조를 제시한다. 본 논문에서 제시한 안전한 Jini 서비스 구조가 기존 Jini 및 Jgroup/ARM 구조에 비하여 서비스 실패에 대해 어느 정도 수준의 서비스 품질을 보이면서 서비스 지속이 가능 하였는지 구현 실험한 결과를 논한다.

본 논문의 2장에서는 관련연구들에 대하여 논하고 3장에서는 Jgroup/ARM 프레임워크를 소개하고 개선해야 할 점을 논한다. 4장에서는 서비스 상태 지속을 지원하는 안전한 Jini 서비스 구조를 제시한다. 5장에서는 구현 실험한 결과를 논한다. 그리고 6장에서 결론을 맺는다.

2. 관련 연구

2.1 Jini 서비스 환경

Jini 시스템은 Lookup 서버와 서비스를 구현한 서버, 클라이언트로 구성된다. 서버와 클라이언트는 디스커버리 프로토콜 수행을 통해 유비쿼터스 네트워크 환경에서 Lookup 서버의 존재를 발견한다. 발견 이후에 서버와 클라이언트는 Lookup 서버로부터 Lookup 서비스 이용에 필요한 Lookup 서비스프락시를 다운로드한다. 그 후 서버는 자신의 서비스 프락시를 Lookup 서버에 등록하고 클라이언트는 Lookup 서비스프락시를 통해 이용 가능한 서비스 구현들을 발견한다. 이때 원하는 서비스를 선택하면 해당 서버가 제공한 서비스 프락시를 Lookup 서버로부터 받게 된다. 클라이언트는 다운로드한 서비스프락시를 이용해 원격의 서비스 구현을 호출한다. 서비스 호출과 응답은 Java RMI(Remote Method Invocation) 통신을 이용하여 하부 통신 프로토콜에 투명하게 이루어진다[1].

네트워크상에서 Lookup 서버를 발견할 수 있는 범위는 멀티캐스트 디스커버리 메시지를 어디까지 전달할 수 있는가에 달려 있다. 통상적으로 이 메시지의 도달거리는 멀티캐스트 디스커버리 패킷의 TTL 파라미터(최대 15)에 의해 결정된다. 그러나 Jini 시스템을 구성하는 Lookup 서버와 서버, 클라이언트가 이 메시지를 수신할 수 있는 멀티캐스트 그룹에 속해야 하며 라우터의 구성이 이를 지원해야 한다. 따라서 네트워크상에 분산된 Lookup 서버들이 디스커버리 프로토콜 수행을 통해 상호 발견과 서비스 연합이 가능하다고 하더라도 Jini 서비스 환경은 관리 가능한 네트워크 경계까지로 그 범위가 국한될 수 밖에 없다.

멀티캐스트 디스커버리 메시지를 전달 할 수 있는 거리제한은 있지만 RMI 메시지의 거리제한은 없기 때문에 각 네트워크 도메인 간의 협력을 통해 Lookup 서버의 발견과 서비스 연합 문제를 해결한다면 Jini 서비스 환경은 더욱 확대될 수 있다.

2.2 Jini 서비스 환경에서의 문제점

Jini 서비스 환경에서의 문제점은 크게 Jini 서비스의 가용성과 서비스 상태에 대한 일치성 보장 문제, 그리고 보안성과 서비스 품질 제고 문제로 구분할 수 있다.

Jini 서비스의 가용성을 떨어트리는 요인으로는 네트워크가 분할되어 Jini 분산시스템 구성이 깨지는 상황이 발생하거나 서버붕괴와 같은 서비스 실패가 발생하는 경우를 들 수 있다.

네트워크 분할과 단일고장점 문제를 해결하여 가용성을 높이는 방법은 중복을 적용하여 서비스 복제를 구성하는 방법이다[6]. 서비스 복제를 구성하다보니 서비스 수행 상태에 대한 일치성 보장 문제가 뒤따른다.

Jini 서비스 환경은 JVM(Java Virtual Machine)을 기반으로 다양한 장치에서 이종성을 극복한다. 다양한 장치에서 이종성을 극복하고 Java 언어로 소프트웨어 개발을 통일한다는 것은 시스템 개발의 부담을 덜어주지만 공격자에게는 다양한 장치별로 취약점을 악용하고 시스템에 침입할 수 있는 보안 허점을 제공한다. 대표적인 예로 Java 런타임 코드가 JNI(Java Native Interface)를 통해 호출하는 시스템 원시코드(native code)에 공격자의 의도를 숨기는 방법은 Java 기반 시스템의 골치 아픈 취약점이다[7]. 이것은 JVM 상위 계층의 보안은 Java 언어 기반 기술이 책임질 수 있어도 하위 플랫폼의 보안성은 책임 질 수 없기 때문이다. 이러한 위험을 안고 Jini 서비스 환경은 서비스프락시와 같은 이동코드 기반으로 네트워크상에 동적 서비스 발견과 추가, 변경을 자유롭게 한다. 또한 서비스와 서비스를 연계하여 새로운 서비스를 만들 수 있고 서비스 환경의 연합이 가능하다. Jini 시스템에서 인증, 인가, 메시지의 기밀성과 무결성과 같은 보안성을 제공하는 것도 중요하지만 침입에 대한 Jini 시스템 차원의 보안 메커니즘이 필요한 부분이다.

위 4 가지 문제를 고려하여 Jini 서비스 환경을 구축하려면 trade-off 문제를 야기한다. 즉 가용성을 높이기 위해 중복을 적용하면 일치성 보장이 쉽지 않으며 보안상 보호해야 할 대상이 많아지게 된다. 또 중복과 보안성을 높이게 되면 서비스 응답까지 많은 지연이 발생하기 때문에 서비스 품질 저하를 초래하게 된다.

2.3 문제 해결을 위한 기존 연구

앞 절에서 논했던 Jini 서비스 환경에서의 4 가지 문제를 모두 해결하기 위한 연구는 아직까지 발견하지 못했다. [5]에서는 Jini 서비스의 가용성을 높이기 위해 중복을 적용하면서 복제 서비스 간에 서비스 수행 상태를 일치시키는 Java 기반 미들웨어 구조와 프로그래밍 모델을 제시하였다. 이 연구에 대해서는 3장에서 자세히 논한다.

클라이언트 요청에 응답하기 위해 서버가 또 다른 서버에게 요청이 필요할 경우 각 서버의 상태 일치를 위해 전통적으로 트랜잭션관리자를 사용한다[8]. 이 경우 각 서버는 물론 중간의 트랜잭션관리자에서도 실패가 발생할 수 있다. [8]에서는 [5]에서 제시한 미들웨어 구조에서 각 서버와 트랜잭션관리자를 중복했을 경우(*2 Server + 2 TM, 2*2 Server + 1 TM, 2*2 Server + 2 TM*) 클라이언트측 응답지연을 조사하였다. 그리고 장애극복(failover)을 수반했을 때의 응답지연이 클라이언트의 서비스 지속을 만족시킬 수 있는지도 조사하였다.

[9]에서는 [5]에서 제시한 환경에서 네트워크 분할이나 서버붕괴 시점부터 이들의 복구과정까지 그룹을 구성하는 서비스 복제의 뷰(view) 변경에 대해서 서버 측과 클라이언트의 그룹프락시, Lookup 서버의 뷰 일치를 다루는 방안을 제시하였다.

[10]에서는 서버의 단일고장점 문제를 해결하기 위해서 서버를 복수 배치하고 클라이언트에 스마트프락시(smart proxy)를 적용하여 여러 서버에 대한 동시 연결을 유지하는 방법을 제시하였다. 중복된 서버에 대한 일치성 보장이 서버와 스마트프락시간의 통신 프로토콜에 의존한다.

Jini 시스템의 보안성 향상을 위한 대표적인 연구들은 [11,12,13,14]의 연구가 있다. 이들 연구에서 공통적으로 제시했던 Jini 시스템 보안요구 사항들은 다음과 같다.

- 클라이언트, 서비스프락시, 서버 간의 신뢰확립
- 서비스 접근제어
- 이동 코드로부터 클라이언트 보호
- 통신채널의 기밀성과 무결성 보장

클라이언트, 서버, 서비스프락시 간의 신뢰 요구사항은 상호 접근에 대한 인증 메커니즘의 필요성을 의미한다. 또한 서비스 접근제어는 인증과 연결하여 권한에 따라 서비스 접근을 제어하는 인가 방법의 필요성을 의미한다. 그리고 이동코드로부터 클라이언트를 보호하는 것은 인증 받은 서비스프락시 코드라고 해도 그 코드의 수행이 클라이언트에게 해를 주지 않도록 제약하는 방안이 필요하다는 것을 의미한다. 마지막 보안요구는 필요한 모든 메시지 교환에 기밀성과 무결성을 제공해야함을 의미한다.

[11]에서는 Jini 연합의 중심에 인증과 권한부여를 담당하는 서비스 구현을 두고 이를 통해 Lookup 서버와 서비스프락시, 서버간의 신뢰확립과 접근제어 요구를 해결하는 Jini 서비스 구조를 제시하였다. [12]에서는 SPKI(Simple Public Key Infrastructure) 인증서를 이용하여 서버-서비스프락시-사용자를 연결하는 인증, 인가 체인을 형성할 수 있는 방안과 Jini 서비스 구조를 제시하였다. 인가 체인은 서비스를 구현한 서버에서 확인하고 인증 체인은 사용자가 확인할 수 있게 함으로써 상호 인증과 권한 인가 요구사항을 동시에 충족시키는 점이 특징이다. [13]에서는 [11]의 구조와 비슷하나 통신채널의 기밀성과 무결성을 제공하는 메카니즘이 다른 연구와 다르다. 또한 클라이언트 구현에 대한 보안투명성을 제공하기 위해서 보안 기능 수행의 상당부분을 서비스프락시가 담당하는 구조이다. [11,12]의 연구는 보안성 제공이 응용 구현에 의존하며 클라이언트에서 보안투명성이 없다.

서비스프락시 코드의 안전 문제는 [11,12,13]의 연구 모두가 인증된 서비스프락시 코드에 대해서는 안전을 신뢰한다는 가정에 기초하고 있다.

메시지 교환에서의 보안성 제공은 [11]에서는 Java 오픈소스로 구현한 SSL 기반 RMI 통신 구현에 의존하고 있고, [12]에서는 Java 표준 JSSE(Java Socket Security Extension)[15]를 이용한 TLS 기반의 RMI 통신 구현에 의존한다. [13]에서는 서버와 서비스프락시 간에 연결마다 동적인 DH(Diffie-Hallman) 키교환 알고리즘으로 공유세션키

를 생성한 다음 이를 통해 기밀성을 제공한다. 그리고 HMAC-MD5 알고리즘으로 메시지의 무결성을 보호한다.

Jini 2.0 기술명세서[14]에서는 기존 연구들이 해결하려 했던 모든 보안요구사항을 충족시키기 위한 지원들이 보강되었다. 특히 이동코드로부터 클라이언트를 보호하기 위한 제약(constraint)기반 원격호출 메커니즘이 추가되었다.

복제 서버를 구성하는 방법으로 침입을 감내하는 연구는 HACQIT(Hierarchical Adaptive Control of Quality of service for Intrusion Tolerance)[16]과 SITAR((Scalable Intrusion-Tolerance Architecture)[17]가 있다. 또한 복제서버들로부터 올바른 결과를 도출(commit)하기 위한 연구로는 [18]의 연구가 있다.

3. Jgroup/ARM 프레임워크

3.1 Jgroup/ARM 프레임워크의 설계목적

Jgroup/ARM 시스템의 구조적 특징은 이원화된 미들웨어와 통신 메커니즘을 제공한다는 것이다. 하나는 클라이언트를 위한 Jini 미들웨어 자체이고 다른 하나는 서비스 복제들이 하나의 개체(single entity)처럼 동작하도록 지원하는 미들웨어이다. 이를 지원하는 핵심 컴포넌트는 Jgroup 데몬(JD)과 GM(Group Manager, 즉, Server-side Proxy)이다. JD 가 신뢰성 있는 그룹멤버십 멀티캐스트 통신을 지원하고 서버의 실패와 네트워크 분할 사실을 실시간으로 탐지하여 관련 이벤트를 GM에게 제공한다. Jgroup/ARM 프레임워크는 네트워크 분할과 서버붕괴와 같은 상황을 대비하기 위한 서비스 개발자에게 복잡한 서비스 중복에 대한 개발 부담을 제거한다. JD 구현에 대한 인터페이스를 제공하고 그룹 멤버십 관리를 지원하는 GM(즉, Server-side Proxy)이 Jini 서비스 개발자가 본래의 서비스 구현에 집중하도록 다양한 인터페이스를 지원한다.

본 논문에서는 네트워크 분할과 서버붕괴 상황에서 Jgroup/ARM 시스템이 어떻게 이를 감내하고 대응하는지에 대해서는 논하지 않는다. Jgroup/ARM 프레임워크에서는 많은 서비스 자원이 중복되었지만 정적으로 서비스를 배치하는 것만을 의미하지 않는다. 서비스 복제의 동적 설치와 변경이 서비스 불능 상황을 극복하도록 관리된다는 의미를 담고 있다. 이러한 극복을 시스템 차원의 자동화된 메커니즘으로 구현할 수 있도록 하는 것이 Jgroup/ARM 프레임워크의 설계 목적이다[5].

굳이 Jgroup/ARM 프레임워크를 사용하지 않고 표준 Jini 시스템 환경 내에 서비스를 중복할 수 있다. 그러나 남아있는 임차시간동안 Lookup 서버에 연결할 수 없는 서비스의 프락시가 잔존하고 있어 문제의 서버로 연결시도가 반복되며 시스템 내 어떤 컴포넌트도 서버의 붕괴 사실이나 네트워크의 단절을 발견할 수 없어 임차시간이 지나야 사용자가 서비스의 부재사실을 알게 된다. 결국 사용자에 대한 서비스 품질은 기대할 수 없게 된다.

3.2 Jgroup/ARM 시스템의 문제점

3.2.1 사용자의 서비스 상태정보 일치를 위한 연산비용

사용자의 상태정보에는 두 가지가 있다. 하나는 세션 중에 사용자의 서비스 이용 상태를 나타내는 컨텍스트가 있다. 성능문제 등 서버에 지속적으로 저장할 필요가 없어 일시적으로 유지되는 컨텍스트이다. 예를 들어 VOD 서비스를 이용한다면 채널변경과 같은 조작관련 이벤트 정보 등이 될 수 있겠다. 본 논문에서는 이를 C_t (temporal context) 라고 부른다. 다른 하나는 세션 종결 이후에도 이용할 수 있도록 지속적인 저장이 필요한 컨텍스트이다. 예를 들면, 서비스 로그인 관련 정보나 과금 관련 정보 등이 될 수 있겠다. 본 논문에서는 이를 C_p(persistent context)라고 표현한다. 그리고 하나의 서버 그룹(G_s)을 다음과 같이 표현한다.

$$G_s \rightarrow \{R_1, R_2, R_3, \ldots R_n\}^u$$

여기서 R은 서비스 복제(replica)를 말하며 n 은 복제를 구별하는 번호이고 u는 서버 그룹이 서비스 중인 사용자의 수이다.

Jgoup/ARM 시스템 환경에서 문제가 없는 상황에서는 하나의 그룹 내 개별 복제는 동일한 사용자 수의 상태정보를 유지한다. 그러나 네트워크가 분할되면 다음과 같은 상황을 예로 들 수 있다.

$$G_s \rightarrow \left\{ \{R_1, R_2\}_a^{15}, \{R_3, R_4, R_5\}_b^{20}, \{R_6, R_7\}_c^{30} \right\}$$

즉, 분할된 서브그룹 별로 연결을 유지하고 있는 사용자의 수가 다를 수 있다. 문제는 다시 네트워크가 복구되어 그룹이 하나로 통합되고 각 복제의 상태를 일치시킬 때 발생한다. 서브그룹 a 는 15명의 사용자가 서비스를 이용하고 있고 서브그룹 b와 c 에는 각각 20명과 30명의 사용자가 자신의 상태정보를 유지하고 있는 서비스를 이용하고 있다고 가정하자.

이 경우 그룹이 하나로 통합하게 되면 서브그룹 a 에 속한 서버는 50명의 새로운 서비스 인스턴스를 추가해야하고 서브그룹 b에 속한 서버들은 45명에 대한 서비스 인스턴스를 새로 추가해야 한다. 또한 서브그룹 c 에 속한 서버들은 35명의 서비스 인스턴스를 추가해야 한다. 즉, 복제 $R_1 \sim R_7$ 까지는 그룹이 분할되기 전부터 연결 중이었던 사용자와 새로 연결한 사용자, 연결이 끊어진 사용자의 상태정보(특히, C_p)가 혼재하게 된다. 여기에 Jgroup/ARM 프레임워크의 중대한 문제점이 있다.

첫째, 상태정보 C_t 와 C_p 는 각 서브그룹의 리더(예를 들어, R_1, R_3, R_6)가 대표해서 리더 간에 교환하고 각자 자신의 멤버(R_1은 R_2, R_3는 R_4와 R_5, R_6는 R_7)에게 putState() 할 수 있다. 그러나 사용자의 수가 많을 경우 이 모두를 처리하는데 드는 비용이 지나치게 높다.

둘째, C_t와 C_p 의 주인을 어떻게 올바르게 연결(match)하

느냐의 복잡한 문제가 있다. 왜냐하면 각 서브 그룹에 연결된 사용자를 식별해야하고 어느 C_t 와 C_p 가 새로운 버전인지 확인되어야 정확한 상태일치가 완료될 수 있다.

3.2.2 장애극복 지연(failover latency)

참고문헌 [5]에 의하면 Jgroup/ARM 시스템에서 클라이언트는 분산된 각 복제서버와 통신하기 위해서 각 복제된 서비스의 프락시를 하나로 묶고 있는 그룹프락시(GP:Group Proxy)를 이용한다고 한다. [9]에서는 네트워크 분할이나 서버 붕괴로 실제로 연결 가능한 서비스 구현이 GP의 내용과 다를 때 발생할 수 있는 장애극복 지연(failover latency) 문제를 지적하고 있다. 이 지연은 예를 들어, $G_s \rightarrow \{ R_1, R_2, R_3, R_4 \}$ 로 이루어진 서버 그룹이 존재할 때 클라이언트의 실제 호출이 R_2의 서비스프락시에 있는 메서드를 호출하였지만 실제로 복제 R_2 는 이미 붕괴된(또는 연결 불가) 상태일 때 발생한다. 클라이언트에게 결함투명성을 제공하기 위해서 GP가 일정시간동안 응답이 없으면 다른 복제를 선택해 재호출을 수행하는데 이 때 소요되는 지연이다. 만약 재호출 시도에도 같은 상황이 반복되면 장애극복 지연은 더욱 길어지고 결국 사용자에게 결함투명성을 제공하지 못한다. 이 문제를 해결하기 위해서는 GP의 내용이 실제로 가용한 복제의 프락시를 유지 할 수 있도록 갱신하는 방안이 필요하다.

3.2.3 callback 객체에서의 개선 요구사항

참고문헌 [5]에 의하면 클라이언트 응용에서 서비스 구현에 대한 호출은 GP가 투명하게 임의로 선택하여 해당 서버의 GM에게 연결한다. 선택된 서버의 GM은 Jini 시스템 관리자가 설정한 시스템 구성정보(application.xml)에 따라 클라이언트의 요청 메시지를 'anycast' 하거나 'multicast' 내지 'leadercast' 한다. 'anycast' 프로토콜은 상태유지가 필요 없는 서비스, 즉 read() 오퍼레이션만 있는 서비스에 적합하며 중복된 각 서버 간에 부하분산 효과를 원하는 응용 개발에 적합하다. 'multicast' 와 'leadercast' 프로토콜은 서비스의 상태유지가 중요한 서비스, 즉 write() 오퍼레이션이 필요한 서비스에 적합하다. 단지 이 두 프로토콜의 차이는 하나의 클라이언트 요청에 대해 서비스 수행을 복제 서버 모두가 수행하느냐 아니면 특정 서버가 수행하고 그 결과를 멤버에게 알려주느냐의 차이다. 그런데 문제는 'multicast' 요청을 응답하는 각 복제의 결과를 최초의 요청을 수신한 복제가 취합하는 데 불필요한 지연이 발생한다는 것이다. 각 복제가 제시한 결과를 취합하는 callback 객체에서 지연을 최소화하는 방안이 요구된다. 즉, 클라이언트로의 응답을 보내기 위해서 처음 사용자의 서비스 요청을 받은 복제가 나머지 모든 멤버의 결과를 취합할 때까지 기다리는 것은 불필요한 지연요소이다. 즉, 어차피 기다려도 같은 결과라면 클라이언트에게 '올바르지 않은 정보' 만을 차폐(mask)하고 바로 결과를 돌려주는 것이 서비스 품질 향상을 위해 좋다. 또한 침입감내를 위한 개선 요구사항이기도 하다.

3.2.4 보안 문제

Jgroup/ARM 시스템에서 복제간의 그룹 통신은 JD에 의존하고 있다. 그러나 JD가 제공하는 멀티캐스트 통신은 메시지의 기밀성과 무결성을 제공하지 않아 네트워크상에서 쉽게 메시지의 열람과 수정이 가능하다. 또한 서비스 복제간에 상호 인증하는 메커니즘이 없다. 공격자가 특정 멀티캐스트 그룹 주소를 알고 있고 이 그룹에 해당하는 모든 노드의 구성정보(config.xml)를 수정한다면 해당 그룹의 복제들은 공격자 만든 서버를 새로운 멤버로 인정하게 된다.

4. 안전한 Jini 서비스 구조

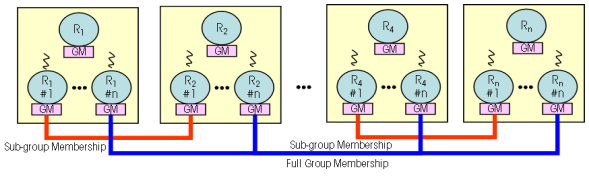
4.1 세션 기반 그룹 멤버십을 위한 확장

본 논문에서 제안하는 Jini 서비스 구조에서는 3.2절에서 지적한 서브그룹의 통합에 따른 사용자의 서비스 상태정보의 일치비용 문제를 해결하기 위해서 서비스 복제 그룹 멤버십을 세션단위로 세분할 수 있도록 Jgroup/ARM 프레임워크를 확장하였다.

(그림 1)은 하나의 클라이언트 연결에 대해 각 복제가 서비스 인스턴스를 쓰레드 단위로 할당하는 시스템 구조를 보이고 있다. 클라이언트의 요청을 'multicast' 내지 'leadercast' 로 각 복제에게 보내면 각 복제는 사용자의 서비스 연결에 대해 먼저 서비스 인스턴스를 쓰레드로 할당하고 서비스에 착수한다. 서버 프로세스 내에서 각 사용자의 서비스 인스턴스 단위로 원격 메서드 호출에 대해 문맥전환(context switching)이 이루어진다. 'multicast' 요청은 모든 복제의 서비스 인스턴스에서 호출한 원격 메서드가 수행되고 'leadercast' 요청은 최초의 요청을 받은 복제의 서비스 인스턴스에서 해당 메서드를 수행하고 다른 복제의 서비스 인스턴스에서는 상태정보만 갱신한다.

네트워크 분할과 복구과정을 거치면 분할된 서비스 객체 그룹의 멤버십도 변경하게 된다. Jgroup/ARM 프레임워크에서는 JD의 그룹멤버십 프로토콜로 네트워크 분할이나 복구(서버의 붕괴와 복구도 포함)를 탐지하게 되면 각 복제의 GM에서 viewchange() 이벤트를 수신하게 되고 그때의 상황에 맞게 현재 연결이 가능한 복제들로 새로운 멤버십을 유지하게 된다. 그러나 3.2절에서 지적한 바와 같이 복제 프로세스 단위의 그룹 멤버십은 서브그룹들이 원래의 서버그룹으로 통합될 때 사용자의 상태정보를 일치시키는데 많은 비용을 초래한다.

본 논문에서는 이 문제를 해결하기 위해서 서버 그룹 내



(그림 1) 세션 단위 그룹 멤버십 프로토콜

에서 세션 별로 서브그룹 멤버십을 유지할 수 있도록 하였다. 이를 설명하기 위해서 특정 시점 T 에서 시스템에 연결한 세션집합을 다음과 같이 표현한다.

$$Session_T \rightarrow \{r_1, r_2, r_3, \cdots r_n\}^u$$

r_1은 서비스 복제 R_1의 인스턴스를 의미하며 특정 시점 T에서 n 개의 서비스 복제에 연결중인 사용자의 수가 u 임을 뜻한다. 시점 T에서 네트워크 분할이 발생했다고 가정하면 서버그룹

$$G_s \rightarrow \left\{ \{R_1, R_2\}_a^{15}, \{R_3, R_4, R_5\}_b^{20}, \{R_6, R_7\}_c^{30} \right\}$$

에서 서브그룹 a 의 연결중인 세션 집합은

$$Session_T \rightarrow \{r_1, r_2\}^{15}$$

이다. 그러나 시점 $T+t$ 에서 복구가 이루어지고 동시에 새로운 사용자의 서비스 접속이 이루어졌다고 가정하면 그 시점에서 세션집합은

$$Session_{T+t} \rightarrow \left\{ \{r_1, r_2\}_a^{15}, \{r_3, r_4, r_5\}_b^{20}, \{r_6, r_7\}_c^{30}, \{r_1, r_2, r_3, r_4, r_5, r_6, r_7\}_d^{u} \right\}$$

이다. 즉 본 논문에서 제안하는 Jini 서비스 구조에서는 서버 그룹이 하나로 통합될 때 기존의 서버그룹이 유지하고 있는 사용자의 상태정보를 통합시키지 않는다. 대신 통합된 이후, 서버 그룹에 대해 새로운 사용자의 연결이 이루어지면 모든 복제에서 사용자의 상태정보를 가지고 있는 서비스 인스턴스만을 추가할 뿐이다. 따라서 서브그룹 d 에 할당된 서비스 인스턴스의 수가 20(즉, u=20)이라면 r_1 과 r_2 에 할당된 총 서비스 인스턴스의 수는 35 이다.

본 논문에서 Jgroup/ARM 프레임워크를 개선한 점은 세션별로 클라이언트의 요청 메시지에 대해 복제 간에는 그룹 멤버십을 유지시키면서 서버 측 엔드포인트를 식별하고 관리하는 Jgroup/ARM 미들웨어의 확장에 있다.

서브그룹의 복제의 수가 최소 2개 이상이 되도록 서비스 분산 및 중복정책을 적용한다면 네트워크 분할과 복구에 따른 고비용의 상태정보일치를 수행하지 않고도 사용자에게 상태지속적인 서비스를 제공할 수 있다.

4.2 장애극복 지연 문제의 해결

클라이언트측 GP 가 하나로 묶고 있는 서비스프락시들은 항상 네트워크상에 서비스 중인 현재 상황과 일치해야한다. 즉 특정 서버가 붕괴되거나 연결이 불가하면 그 상황을 반영하여 GP의 내용을 갱신해야한다.

본 논문에서 제안하는 문제해결 방법은 서버가 클라이언트에 응답할 때 현재 가용한 멤버의 정보를 추가하여 함께 반환하는 것이다. 연결될 수 없는 서버의 프락시는 제거하고 새로 연결할 수 있는 서버의 서비스프락시는 Lookup 서버로부터 얻는다.

4.3 침입감내를 위한 callback 객체의 확장

3.2절에서 지적한 바와 같이 유비쿼터스 환경에서는 다양한 장치와 소프트웨어가 편재되어 네트워크로 연결되다보니 취약점을 악용할 수 있는 소스도 편재되어 있기 마련이다. 이러한 문제점을 안고 네트워크상의 가용 자원을 묶어 하나의 Jini 시스템을 구축하려면 침입에 대비해야 한다.

모든 침입을 막는다는 것은 불가능하므로 다양성(diversity)에 기초를 둔 중복을 적용하여 침입을 감내한다. 다양한 하드웨어 및 운영체제별로 JVM이 존재한다는 것은 다양성을 추구하기 수월한 환경을 제공해 준다.

복제의 수를 정하는 문제는 동시에 몇 개의 복제에 문제가 발생하는 것을 감내할 것인가에 달려있다. Byzantine 일치 알고리즘[19]에 의하면 N이 중복의 개수이고 최대 t 개의 복제가 결함 및 공격에 의해 임의의 비정상적인 행위를 한다고 할 때, $N > 3t$ 이다.

N의 2/3 이상이 같은 결과를 가지고 자체 수용시험(acceptance test)을 통과하였다면 나머지 결과에 관계없이 먼저 도래한 결과를 클라이언트에게 전달할 수 있다. 만약 반복적으로 수용시험을 통과하지 못한 복제가 있다면 해당 복제를 시스템에서 제거한다.

본 논문에서는 이러한 기능을 수행하도록 3.2절에서 논한 callback 객체를 확장하였다. 수용시험은 서비스 구현의 예상되는 반환 값을 기초로 시험한다.

4.4 보안 메커니즘의 추가

4.4.1 그룹 멤버 간 통신

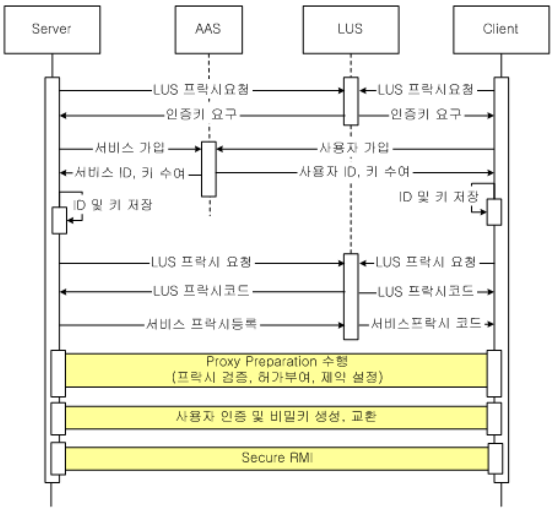
본 논문에서는 모든 복제 간의 통신이 안전하게 이루어지도록 하기 위해서 Jgroup/ARM 프레임워크를 확장하였다. 그룹의 공유 비밀키(shared secret key)로 메시지의 기밀성을 제공하고 HMAC 알고리즘으로 메시지의 무결성을 보호하도록 GM에서 멀티캐스트 서비스를 제공하는 베이스 계층[5]을 확장하였다.

본 논문에서는 그룹의 공유 비밀키 생성과 분배를 위해 'Secure Spread 라이브러리'에서 사용하고 있는 출자키(contributory key)에 의한 그룹키 일치 알고리즘[20]을 사용하였다. 그룹의 멤버정보가 바뀔 때 마다 멤버 간에 출자키를 얻어 그룹의 공유 비밀키를 생성하고 분배하는 방안이다. 빈번한 멤버십의 변화가 있는 응용에서는 키 분배비용이 높을 수 있으나 네트워크 분할이나 서버 붕괴와 같은 이벤트는 빈도가 높지 않으므로 키 일치와 분배비용을 무시할 수 있다.

4.4.2 Jini 시스템 구성요소 간 통신

본 논문에서 제안한 안전한 Jini 시스템 구조에서는 서비스와 클라이언트, Lookup 서버 간의 신뢰확립과 접근 제어를 위해서 아래 (그림 2)와 같은 통신을 수행한다.

클라이언트와 서버가 Lookup 서버에 접근하여 Lookup 서비스프락시를 얻기 위해서는 이에 접근할 수 있는 키를 얻어야 한다. 본 논문에서는 Jini 서비스 환경에서 Lookup



(그림 2) Jini 시스템 보안구조

서버에 대한 접근을 인증, 인가하고 하나의 Jini 시스템 내에서 CA 역할을 수행하는 Jini 서비스를 설계하였다. (그림 2)에서 '응용 레벨 인증 서버(AAS)' 가 이를 수행하는 Jini 기반 서버이다.

사용자가 가입을 하고 Lookup 서버에 대한 접근키를 얻어 서비스 발견의 자유를 얻는 반면, 서비스는 등록의 자유를 얻는다. 이때 Lookup 서버에 대한 접근키를 얻으면서 사용자와 서비스는 각각 자신의 전자서명에 대한 인증서도 얻는다. 이러한 접근키와 인증서는 서버와 클라이언트 노드에서 하나의 지역객체로 저장된다.

서비스 발견 후 클라이언트가 원격의 서버에 연결하면 클라이언트는 Jini2.0 기술부터 지원하는 프락시 준비 단계로 진입한다. 여기서 Lookup 서버로부터 받은 서비스프락시에 대한 신뢰를 검증하고, 보안허가(permission)의 부여와 제약을 설정한다.

Jini2.0 기술부터는 서버와 클라이언트 간의 안전한 통신을 위해서 보안성 있고 다양한 전송 프로토콜을 지원할 수 있는 서비스프락시를 'export' 할 수 있게 되었다. 그러나 서버와 서비스프락시를 연결하는 엔드포인트가 TLS/SSL 기반의 통신에 의존하고 있어 모든 메시지 통신의 암복호화에 과도한 연산 비용을 부담해야 한다.

본 논문에서는 이 문제를 해결하기 위해서 클라이언트가 서비스프락시의 신뢰를 검증하면서 얻은 서버의 공개키와 자신이 생성한 nonce 값 그리고 자신의 공개키 값을 서버의 공개키로 암호화해 보내고 서버 역시 클라이언트의 공개키로 자신이 확인한 nonce 값을 암호화해서 응답함으로서 양측이 비밀키를 공유하도록 한다. 이 과정에서 MITM(man in the middle) 공격에 대응하기 위해서 자신의 전자서명과 AAS로부터 받은 인증서를 첨부한다. 공유 비밀키에 의한 기밀성 제공은 향후 서버와 클라이언트를 잇는 엔드포인트에서 암복호화에 빠른 응답성을 보일 것이다. 메시지의 무결성 확인은 HMAC 알고리즘을 이용한다.

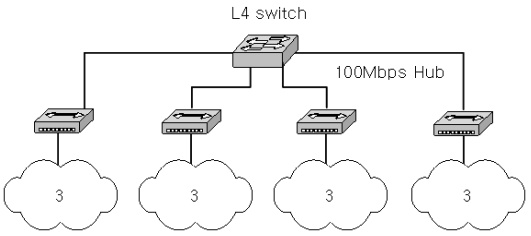
5. 실험 및 논의

5.1 실험 환경과 방법

본 논문에서는 아래 (그림 3)과 같이 총 12대의 PC(intel pentium4 CPU, 512MB 메모리)로 하나의 서브넷을 구성하였다. 네트워크 분할 상황을 모의실험하기 위하여 12대의 PC는 4대의 스위칭허브로 구분된다. 각 PC는 실험에 따라서 Jini 서비스 복제들을 호스팅하거나 클라이언트 소프트웨어를 호스팅 하는데 사용하였다. 소프트웨어 개발은 JDK1.6 버전과 Jini Starter kit 2.1, Jgroup 3.0 을 사용하였다.

두 가지 서비스를 구현하였다. 하나는 성능측정을 위한 실험이고 다른 하나는 서버붕괴에 대해 상태지속 여부를 확인하는 실험이다. 성능측정은 다시 세 가지 응용을 실험하였다. 하나는 클라이언트의 요청을 임의의 복제에게 'leadercast' 하는 실험이고 다른 하나는 클라이언트의 요청을 모든 복제에게 'multicast' 해서 단순히 그 결과를 취합 후 선택하여 반환하는 실험이다. 마지막 하나는 클라이언트 요청을 모든 복제에게 'multicast' 하되 그 결과에 대해 수용시험을 수행하고 비교 한 결과를 반환하는 'voting[18]' 실험이다.

첫 번째 서비스 구현은 클라이언트 요청에 예금과 출금, 잔금에 이자를 적용하여 갱신하는 구현이다. 사용자의 ID와 로그인 시간, 입력했던 데이터와 요청한 서비스 오퍼레이션을 기억한다. 두 번째 서비스 구현은 클라이언트 요청에 파일을 일정크기의 파일블록으로 나누어 전송하는 서비스를 구현하였다. 사용자의 ID와 로그인 시간, 요청했던 파일명을 저장하며, 전송한 파일과 파일블록번호를 기억한다.

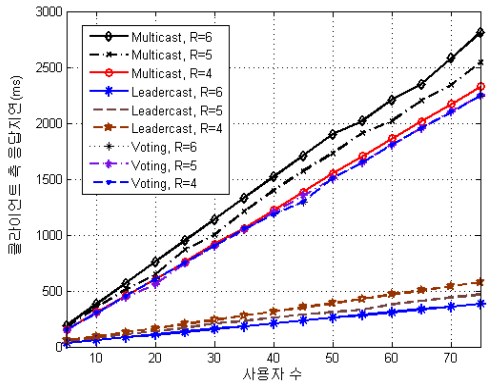


(그림 3) 실험 환경

5.2 서비스 응답지연 분석

(그림 4)의 클라이언트 요청에 대한 최종 응답지연은 (그림 2)에서 제시한 보안 메커니즘을 전혀 적용하지 않은 지연이다.

복제의 수(즉, R)가 증가 할수록 'leadercast' 응용은 사용자의 요청에 대한 자원 할당이 각 복제로 분산되는 효과를 얻어 복제의 수가 늘수록 클라이언트로의 응답지연이 감소하는 반면, 'multicast' 응용은 복제의 수와 사용자의 수가 증가할수록 클라이언트로의 응답지연이 큰 폭으로 증가한다. 'voting' 응용은 N개의 중복에서 최대 t 개의 복제가 결함 및 공격에 의해 임의의 비정상적인 행위를 한다고 할 때, $N > 3t$ 이므로 4 개 이상의 복제에서 3개의 복제로부터 올



(그림 4) 서비스 요청-응답지연

바른 결과를 수신하면 나머지 복제의 결과를 기다리지 않으므로 복제의 수 변화가 응답지연에 영향을 주지 않았다.

본 실험의 결과는 Callback 객체의 확장이 가져다주는 효과를 보여준다. 즉, 항상 안전한 결과의 반환이 요구되는 침입감내 서비스 제공을 위해 복제 간 'multicast' 프로토콜이 수행되더라도 복제의 수와는 관계없는 일정한 지연을 예측할 수 있게 한다.

5.3 기타 성능 비용 측정 결과

다음 <표 1>은 나머지 다른 성능비용을 측정한 결과를 나타내고 있다.

<표 1>에서 제시한 실험결과들은 하나의 서비스 접근에 대한 통신비용과 붕괴된 서버에 대한 복구 및 재구성 비용이다. 클라이언트와 서버가 AAS 로부터 각각 인증을 받았다고 가정하고 세션 생성에 소요되는 비용을 조사하였다. Proxy 준비(Proxy Preparation) 비용은 클라이언트가 Lookup 서버로부터 받은 서비스 프락시 코드를 신뢰할 수 있는지를 확인하기 위해 해당 서버와 인증서 확인절차를 포함한다. 또한 클라이언트 응용에서 프락시 코드에 대해 보안허가를 부여하는 지연이 포함된다. 비밀키 생성비용은 클라이언트와 서버 측 엔드포인트에서 Proxy 준비단계에서 얻은 상호 공개키를 토대로 세션 공유비밀키를 얻기 위해 소요된 비용이다. Proxy 준비 비용과 비밀키 생성 비용은 최초의 세션 수립을 위해 한번 요구되는 지연이지만 다소 그 값이 크다. 이것은 Proxy 준비단계와 비밀키 생성을 위해서 필수적으로 소요되는 공개키 생성과 상호 확인단계에서 드는 연산비용, 최적화되지 않은 구현에 기인한다고 여겨진다.

〈표 1〉 시스템 성능비용 실험결과

실험구분	단위	소요되는 지연
Proxy 준비비용	세션	4,320 ms
비밀키 생성비용	세션	3,450 ms
비밀키에 의한 통신채널 기밀성 및 무결성 보호 비용	세션	23 ms
서버 복구 및 재구성 비용	서버	3,665 ms

통신채널 기밀성 및 무결성 보호비용은 공유비밀키에 의한 메시지 암복호화와 무결성 확인에 드는 비용이다.

서버 복구와 재구성 비용은 붕괴된 서버가 서비스를 재개하기까지의 비용이다. 서버의 재시작과 초기화, 그룹 멤버십의 갱신까지의 지연이다.

5.4 서비스 상태지속 구현 실험

실험에서 파일을 전송하던 서버의 서비스 실패에도 불구하고 파일 전송이 그대로 지속되고 있는 모습을 확인할 수 있었다. 본 구현 실험에서 하나의 서버에 대한 Failover 지연은 평균 180 ms 소요되었으며 다른 복제로부터 상태지속적인 서비스를 받는 데 걸리는 시간은 160 ms 소요되었다. 따라서 340 ms 지연을 부담하면서 상태지속적인 서비스 이용이 가능하였다.

5.5 연구결과의 비교

다음 <표 2>는 Jini 서비스의 생존성 보장을 위한 구조적 요구사항 면에서 기존 연구와 비교한 결과이다.

〈표 2〉 연구 결과의 비교

요구사항	표준 Jini 서비스 구조	Jgroup/ARM 서비스 구조	제안한 Jini 서비스 구조
결함감내 서비스	미지원	지원	지원
침입감내 서비스	미지원	미지원	지원
보안 메커니즘	지원	미지원	지원
서비스 상태지속	미지원	지원	지원

6. 결 론

유비쿼터스 컴퓨팅 환경은 연결의 신뢰성이 낮고 서비스를 제공하는 시스템에 대한 침입이나 서비스 실패가 발생할 확률이 높다. 본 논문은 이러한 환경에서 정당한 사용자가 보안상 신뢰할 수 있는 서비스를 중단이나 방해 없이 이용할 수 있게 하는 안전한 Jini 서비스 구조를 제시하였다.

기존의 Jini 서비스 환경에서 네트워크 분할이나 서버 붕괴와 같은 상황을 감내하기 위해 서비스를 중복할 수 있으나 그러한 문제 상황을 발견할 수 있는 메커니즘이 없어 궁극적으로 사용자의 서비스 품질을 보장할 수 없다. 본 논문에서는 이러한 문제를 해결하기 위해서 제시된 Jgroup/ARM 프레임워크에 대해서 소개하였고, 이 프레임워크 문제점을 논하였다. 그리고 이 프레임워크의 문제점을 해결하기 위한 방안을 제시하였다.

본 논문에서 제시한 안전한 Jini 서비스 구조는 사용자의 세션별로 서비스 복제를 할당하고 결함과 침입을 감내한다. 따라서 Jgroup/ARM 프레임워크처럼 단순히 서비스 가용성

만 높이는 것이 아니라 침입을 감내하고 높은 보안성을 제공하며, 상태 지속적인 서비스 이용을 보장한다.

본 연구의 결과는 유비쿼터스 정보 서비스 환경을 구축할 때 좋은 참고 모델이 되리라 본다.

참 고 문 헌

[1] Sun Microsystems, "Jini™ Architecture Specification," Published Specification, http://java.sun.com/products/jini/2.0/doc/specs/html/jini-spec.html, 2003.

[2] D.Szentivanyi and S. Nadjm-Tehrani, "Middleware Support for Fault Tolerance," Chapter 28 in Middleware for Communications, Q. Mahmoud (Ed.), John Wiley & Sons, 2004.

[3] Sun Microsystems, "Jini Technology Core Platform Specification." Communication of the ACM, Vol.39, No. 4, pp.75-83, 1996.

[4] Frank Sommers, "Jini Starter Kit 2.0 tightens Jini's security framework," Los Alamitos, CA., IEEE Computer Society Press, 2003.

[5] Hein Meling, et al., "Jgroup/ARM: a distributed object group platform with autonomous replication managements," Software Practice and Experience, John Wiley & Sons, 2007.

[6] Johannes Osrael, et al.,"Using Replication to Build Highly Available .Net Applications," Proceedings of the 17th International Conference on Database and Expert Systems Applications, pp.385-398, 2006.

[7] Marc Schönefeld. "Hunting Flaws in JDK," In Blackhat Europe 2003. May 2003.

[8] Heine Kolltveit et al., "Preventing Orphan Requests by Integrating Replication and Transactions," LNCS 4690, Springer-Verlag Berlin, 2007.

[9] Hein Meling, et al., "Performance Consequences of Inconsistent Client-side Membership Information in the Open Group Model," Proceedings of the 23rd IEEE International Performance, Computing and Communications Conference. pp.777-782, 2004.

[10] M. Tichy, H. Giese. "An Architecture for Configurable Dependability of Application Services," Proc. of the ICSE 2003 Workshop on Software Architectures for Dependable Systems. pp.65-70, Portland, OR. April 2003.

[11] Peer Hasselmeyer, et al., "Trade-offs in a Secure Jini Service Architecture," LNCS 1890, Springer-Verlag Berlin, 2000.

[12] Pasi Eronen and Pekka Nikander. "Decentralized Jini security," In Proceedings of the Network and Distributed System Security Symposium (NDSS 2001), pages 161 - 172, San Diego, California, February 2001.

[13] Thomas Schoch, et al. "Making Jini Secure," Proc. 4th International Conference on Electronic Commerce Research, pages 276-286, Nov. 2001.

[14] Sun Microsystems, "Jini Technology Starter Kit Overview v2.0," Published Specification, http://java.sun.com/developer/products/jini/arch2_0.html, 2003.

[15] Sun Microsystems, "Java Secure Socket Extension(JSSE) Reference Guide for Java Platform Standard Edition 6," http://java.sun.com/javase/6/docs/tech-notes/guides/security/jsse/JSSERefGuide.html#Features.

[16] Reynolds, J. et al, "The Design and Implementation of an Intrusion Tolerant System," Proc. of Int'l Conference on Dependable Systems and Networks, 2002.

[17] Wang F., et al, "SITAR: A Scalable Intrusion-Tolerant Architecture for Distributed Services," DARPA Information Survivability Conference & EXposition, 2001.

[18] Byoung Joon Min, et al. "Committing Secure Results with Replicated Servers," LNCS 3043, Springer-Verlag Berlin, 2004.

[19] Marshall Pease, Robert Shostak, Leslie Lamport, "Reaching Agreement in the Presence of Faults," Journal of the ACM 27/2 228-234 1980.

[20] Amir, Y. et. al., "Secure Group Communication Using Robust Contributory Key Agreement," IEEE Transactions on Parallel and Distributed Systems, Vol.15, No.5, pp.468-480, May 2004.

김 성 기

e-mail : proteras@incheon.ac.kr
1996년 인천대학교 컴퓨터공학과(학사)
1998년 인천대학교 컴퓨터공학과(석사)
1998년~1999년 인천대학교 멀티미디어 연구센터 연구원
2006년 인천대학교 컴퓨터공학과(박사)
2006년~현 재 인천대학교 정보기술교육원 초빙교수
관심분야 : 컴퓨터 시스템 보안, 침입감내 시스템, 분산시스템

정 진 철

e-mail : smalljiny@incheon.ac.kr
2004년 인천대학교 컴퓨터공학과(학사)
2008년 인천대학교 컴퓨터공학과(석사과정)
관심분야 : 컴퓨터 시스템 보안, 침입감내 시스템, 분산시스템

박 경 노

e-mail : knpark@incheon.ac.kr
2007년 인천대학교 컴퓨터공학과(학사)
2008년~현 재 인천대학교 컴퓨터공학과 (석사과정)
관심분야 : 분산시스템, 컴퓨터 시스템 보안, 그리드 컴퓨팅

민 병 준

e-mail : bjmin@incheon.ac.kr
1983년 연세대학교 전자공학과(학사)
1985년 연세대학교 전자공학과(석사)
1991년 미국캘리포니아대학교(UCI) 전기및컴퓨터공학과(박사)
1884년~1986년 삼성전자 연구원
1992년~1994년 한국통신 선임연구원
1995년~현 재 인천대학교 컴퓨터공학과 교수
관심분야 : 분산시스템, 통신망관리, 보안