이동 에이전트 시스템을 위한 효율적인 중복 프로토콜

(Efficient Replication Protocols for Mobile Agent Systems)

안 진 호 †

(JinHo Ahn)

요 약 본 논문에서는 각 중복서비스의 수행형태가 결정적이냐 비결정적이냐에 따라 알맞은 수동형 중복 프로토콜을 적용함으로써 이동 에이전트 시스템에서 중복 서비스의 결함포용성과 확장성을 향상시키는 새로운 전략을 제안한다. 이러한 목적을 위해 두 개의 수동형 중복 프로토콜들인 PRPNS과 PRPDS를 각각 비결정적 중복 서비스와 결정적 중복 서비스를 위해 설계한다. 이 두 프로토콜들은 모두 방문 이동 에이전트들이 반드시 주 서비스 에이전트만이 아니라 보조 서비스 에이전트를 수행하는 임의의 노드로 전달되어 수행될 수 있도록 한다. 특히, 프로토콜 PRPDS는 임의의 보조 서비스 에이전트가 이동 에이전트 요구 메시지를 수신하고, 주 서비스 에이전트로부터 그 요구 메시지의 전달 일련번호를 얻은 후에, 그 보조 서비스 에이전트가 해당 요구 메시지를 처리하고 다른 중복 서비스 에이전트들과의 조정역할을 할 수 있도록 한다. 따라서, 이 두 프로토콜들을 사용하는 본 논문의 전략은 이동 에이전트 시스템에서 매우 많은 수의 이동 에이전트들이 동시에 접근하고자 하는 중복서비스의 높은 확장성을 보장할 수 있다. 본 시뮬레이션 결과는 제안된 전략이 기존의 수동형 중복프로토콜만을 사용하는 전략에 비해 매우 향상된 성능을 발휘한다는 것을 보여준다.

키워드 : 이동 에이전트 시스템, 중복 서비스, 확장성, 결함포용성, 수동형 중복, 결정성

Abstract In this paper, we propose a strategy to improve fault-tolerance and scalability of replicated services in mobile agent systems by applying an appropriate passive replication protocol for each replicated service according to whether the service is deterministic or non-deterministic. For this purpose, two passive replication protocols, PRPNS and PRPDS, are designed for non-deterministic and deterministic services respectively. They both allow visiting mobile agents to be forwarded to and execute their tasks on any node performing a service agent, not necessarily the primary agent. Especially, in the protocol PRPDS, after a backup service agent has received each mobile agent request and obtained its delivery sequence number from the primary service agent, the backup is responsible for processing the request and coordinating with the other replica service agents. Therefore, our strategy using the two proposed protocols can promise high scalability of replicated services a large number of mobile agents attempt to access in mobile agent systems. Our simulation results show that the proposed strategy performs much better than the one using only the traditional passive replication protocol.

Key words : mobile agent system, replicated service, scalability, fault-tolerance, passive replication, determinism

1. 서 론

이동 에이전트 시스템은 다수의 이동 에이전트(mobile agent), 서비스 에이전트(service agent)와 장소(place)로 구성된다[1,2]. 이동 에이전트는 한 네트워크에서 노드와 노드간을 이동하면서 그 노드들에서 제공하는 서비스를 이용하여 사용자 대신 특정된 일을 수행하는 능동적이고 자율적인 프로그램이다. 서비스 에이전트란 특정한 노드에 위치하면서 시스템 혹은 애플리케이션 수준의 서비스를 다른 에이전트에게 제공하는 고정 에이전트(stationary agent)이다. 장소란 에이전트가 노드

· 본 연구는 2005학년도 경기대학교 학술연구비(일반연구과제) 지원에 의하여 수행되었음(과제번호: 2005-004).

† 정 회 원 : 경기대학교 정보과학부 교수
jhahn@kyonggi.ac.kr

논문접수 : 2004년 2월 5일
심사완료 : 2006년 9월 1일

상에서 수행하기 위한 환경을 말한다. 한 에이전트가 다른 노드로 이주할 때, 그 에이전트의 코드와 상태정보가 캡쳐되고, 이주할 노드로 전송된다. 이 에이전트가 해당 노드에 전달되었을 때, 에이전트 수행이 초기화된 후, 그 노드의 지역 서비스 에이전트들과 상호작용 함으로써 자신의 작업을 수행한다. 이러한 이동 에이전트 개념은 기존의 클라이언트-서버 패러다임(client-server paradigm)에 비해 통신량의 감소(reduction of network traffic) 및 비동기적 상호작용(asynchronous interaction)이라는 중요한 장점을 가짐으로써 많은 연구 분야의 초점이 되고 있다[3]. 현재 자율 컴퓨팅(autonomic computing)[4], 무선 센서 네트워크[5], 그리드 컴퓨팅[6] 등과 같은 응용분야에서 이동 에이전트 패러다임을 적용하고자 하는 연구가 활발히 진행되고 있다.

계속적으로 이러한 이동 에이전트 시스템이 대중화되어 이동 에이전트 사용자 수가 급속도로 증가한다면, 많은 수의 이동 에이전트들이 특정한 서비스를 동시에 요구함으로써 그 서비스 노드에서 과부하가 발생할 수 있다. 이 경우 이동 에이전트 시스템의 전체적인 성능이 매우 저하된다. 또한, 두 번째 중요한 문제점으로써 만약, 그 서비스 노드가 고장난다면, 해당 서비스를 요구하는 모든 이동 에이전트들의 수행이 정지될 수 있다. 이 경우, 이동 에이전트 시스템의 전체적인 가용성(*availability*)이 매우 낮아진다. 이러한 문제점들을 해결하기 위해서는 서비스 에이전트들을 다수 노드들에 중복시키고, 동일한 서비스를 수행하는 다수의 에이전트들에게 이동 에이전트들을 균형적으로 분산시킬 수 있어야 한다. 또한, 몇몇 서비스 에이전트들이 고장난다 하더라도 해당 서비스 상태의 일관성을 유지시키며 계속적으로 서비스를 제공할 수 있어야 한다.

각 이동 에이전트는 서비스 에이전트 집합 내에 하나 이상의 서비스 에이전트들에게 일련의 요구 메시지들을 전달한다. 이렇게 다수의 이동 에이전트들로부터 송신된 인터리빙된 요구 메시지들은 모든 서비스 에이전트에서 일관적으로 처리되어야 한다. 이러한 일관성 조건을 만족시키는 분산시스템에서의 중복 서비스 프로토콜들을 다음과 같은 특성을 갖는 두 가지 접근방법 즉, 능동형 중복(active replication)과 수동형 중복기법(passive replication)으로 나눌 수 있다[7]. 먼저, 능동적 중복기법[8, 9]이 이동 에이전트 시스템에 적용된다면, 모든 살아있는 중복 서비스 에이전트가 이동 에이전트로부터 일련의 오퍼레이션들을 동일한 순서대로 요구받고, 독립적으로 처리하고 응답한다. 이 기법의 장점으로는 몇몇의 서비스 에이전트들이 고장나더라도 다른 살아있는 에이전트들이 요구된 오퍼레이션들을 처리하고 응답할 수 있기 때문에, 이동 에이전트들에게 고장에 대한 투명성(failure transparency)을 제공한다. 그러나, 이 기법은 모든 중복 서비스 에이전트가 이동 에이전트가 요구한 동일한 오퍼레이션들을 각각 순서대로 수신하고 처리하기 때문에, 정상수행시 높은 자원의 소비(high resource consumption)를 발생시킨다. 또한, 이 기법은 서비스 에이전트가 수행하는 모든 오퍼레이션들이 결정적(deterministic)이어야 한다는 제한사항을 가지고 있다.

두 번째로 수동적 중복기법[10]에서는 중복 서비스 에이전트 집합에서 하나의 에이전트 즉, 주 서비스 에이전트(primary service agent)만 이동 에이전트로부터 일련의 오퍼레이션들을 요구받고, 처리한 후 자신의 상태를 포함한 메시지를 다른 에이전트들, 즉 보조 서비스 에이전트(backup service agent)들에게 전달한다. 각 보조 에이전트는 수신된 메시지를 이용하여 자신의 상태를 갱신하고, 주 서비스 에이전트에게 확인 메시지를 보낸다. 주 서비스 에이전트는 모든 보조 에이전트들로부터 확인 메시지를 수신한 후, 해당 이동 에이전트에게 응답 메시지를 전달한다. 따라서, 주 서비스 에이전트가 고장나는 경우, 이동 에이전트에게 고장에 대한 투명성을 제공하지 못하고, 높은 재구성 비용(high reconfiguration cost)을 발생시킨다. 그러나, 이 기법은 능동적 중복기법에 비해 정상수행시 낮은 자원의 소비를 발생시키며, 서비스 에이전트에서 수행되는 모든 오퍼레이션들이 비결정적(non-deterministic)이라도 몇몇의 중복 서비스 에이전트들이 고장나더라도 해당 서비스 일관성을 보장할 수 있다. 또한, 모든 이동 에이전트가 주 서비스 에이전트에게만 서비스 요구 메시지를 송신하기 때문에 각 이동 에이전트는 멀티캐스트(multicast) 기능이 아닌 유니캐스트(unicast) 기능을 사용하기만 하면 된다[11]. 따라서, 이러한 장점들 때문에, 본 논문에서는 이동 에이전트 시스템에서 중복 서비스의 결함포용성을 제공하기 위해 수동형 중복기법을 사용하고자 한다.

그러나, 기존의 수동형 중복기법은 이동 에이전트 시스템에 중복 서비스를 위한 결함포용기법으로 적용될 때 다음과 같은 이유로 확장성 및 성능 문제를 발생시킬 수 있다. 즉, 이전 연구[10,11]에서는 각 중복 서비스 수행형태가 비결정적인지 결정적인지에 관계없이 그 서비스에 기존 수동형 중복기법을 일률적으로 적용해왔다. 그러나, 이러한 방법에서는 모든 이동 에이전트의 요구 메시지가 반드시 주 서비스 에이전트로만 전송되어, 그 요구 메시지를 처리하고, 다른 살아있는 중복 서비스 에이전트들과 일관성 유지를 위한 동기화를 수행한 후 그 요구 메시지에 대한 응답 메시지를 해당 에이전트에게 전달한다. 그러나, 이러한 주 서비스 에이전트의 특정된 역할은 비결정적 서비스의 일관성 보장을 위해서만 반드시 필요하다. 또한, 기존 수동형 중복기법은 모든 방

문 이동 에이전트들이 각 서비스 도메인의 주 서비스 에이전트가 수행하는 노드로만 순서대로 전송되어 그들의 일을 수행하도록 한다. 이러한 본질적인 특징들에 의해 매우 많은 수의 이동 에이전트들이 특정 서비스 도메인으로 전달되는 경우, 해당 주 서비스 에이전트 노드에서 과부하가 발생될 수 있다. 따라서, 이러한 방법은 높은 확장성과 성능을 발휘할 수 없다.

본 논문에서는 결정적 서비스가 자신의 일관성 보장을 위해 비결정적 서비스보다 약한 제한사항들을 요구한다는 점을 고려하여, 각 서비스의 수행형태에 따라 알맞은 수동형 중복 프로토콜을 해당 서비스에 적용하는 확장적인 전략을 제안한다. 이러한 목적을 위해, 두 개의 수동형 중복 프로토콜들인 PRPNS과 PRPDS를 각각 비결정적 중복 서비스와 결정적 중복 서비스를 위해 설계한다. 이 두 프로토콜들은 모두 방문 이동 에이전트들이 반드시 주 서비스 에이전트만이 아니라 보조 서비스 에이전트를 수행하는 임의의 노드로 전달되어 수행될 수 있도록 한다. 특히, 프로토콜 PRPDS는 임의의 보조 서비스 에이전트가 이동 에이전트 요구 메시지를 수신한 경우, 주 서비스 에이전트로부터 그 요구 메시지의 전달 일련번호를 얻은 후, 그 보조 서비스 에이전트가 직접 해당 요구 메시지를 처리하고 다른 중복 서비스 에이전트들과의 조정역할을 할 수 있도록 한다. 이러한 특징 때문에, PRPDS는 멀티 쓰레드 서버와 같은 비결정적 서버의 고장을 포용하는 PRPNS보다 높은 성능을 발휘할 수 있다.

본 논문의 이후 구성은 다음과 같다. 2절에서는 이동 에이전트 시스템 모델에 대해 알아본다. 3절에서는 본 연구진이 설계한 두 중복 프로토콜들을 설명하고, 4절에서는 기존의 프로토콜과 제안된 프로토콜들간의 성능평가를 시뮬레이션을 통해 살펴본다. 5절과 6절에서는 관련연구와 본 논문의 결론에 대해 각각 기술한다.

2. 이동 에이전트 시스템 모델

본 논문에서 가정하는 이동 에이전트 시스템은 어떠한 전역 메모리(global memory)와 전역 클락(global clock)을 가지고 있지 않다. 또한, 이 시스템은 비동기적이다: 모든 프로세스는 각자의 속도로 수행하고, 메시지 전송지연(message transmission delay) 시간은 유한하지만 임의적(arbitrary)이다. 또한, 각 에이전트는 메시지를 통해서만 다른 에이전트들과 통신을 한다. 그리고, 이 시스템은 분산 합의(distributed consensus)에 대한 불가능성 문제점[12]을 해결하기 위해 비신뢰적인 고장 감지기(unreliable failure detector)를 포함한다[13]. 시스템은 이동 에이전트들에게 n(n>0)개의 서비스 집합 $S=\{s_0,s_1,...,s_{n-1}\}$을 제공한다. 하나의 서비스 $s_i(0\le i\le n-1)$는 이동 에이전트들에 의해 접근되고 하나의 그룹인 서비스 에이전트 집합 $u_i=\{u_i^0,u_i^1,...,u_i^{k-1}\}(1\le k)$에 의해 구현된다. 여기서, $u_i^w(0\le w\le k-1)$는 노드 $node_i^w$에서 수행된다. 모든 서비스 에이전트는 자신의 지역 상태를 유지하고 있으며 에이전트 상호작용을 통해 원자적으로 자신의 상태를 바꿀 수 있다. 사용자를 대신하여 특정한 작업을 수행하기 위해, 한 이동 에이전트 a_j (j>0)는 자신의 여정에 따라 l(l>0)개의 장소(place)들 $p_j=[p_{j0},p_{j1},...,p_{j(l-1)}]$에서 인덱스 순서대로 이주하면서 수행한다. 여기서 에이전트의 여정은 해당 에이전트가 자신이 생성된 노드에서 작업을 시작하기 전에 정적으로 결정될 수도 있고, 이주 혹은 수행 중 동적으로 결정될 수 있다. 한 장소 $p_{jm}(0\le m\le l-1)$에서 에이전트 a_j를 수행하는 것을 그 에이전트 수행의 단계 $Stage_j^m$라 한다. 에이전트 a_j가 장소들의 시퀀스 $[p_{j0},p_{j1},...,p_{jm}]$에서 모든 단계 오퍼레이션들을 수행한 후의 에이전트 상태를 a_j^{m+1}로 표시하고, 이 상태의 에이전트는 p_{jm}로부터 $p_{j(m+1)}$로 전달되고 장소 $p_{j(m+1)}$에서 수행될 것이다. 예를 들면, 그림 1은 세 개의 이동 에이전트들 a_t^l, a_s^m, a_r^n이 노드 $node_i^0$에서 수행하는 서비스 에이전트 u_i^0를 통해 서비스 u_i를 사용하고자하는 예를 보여준다. 이 그림에서 세 개의 이동 에이전트들은 먼저 u_i^0의 메시지 큐인 wq_i^0에서 대기한 후, 요구-실행-응답 형태로 서비스 에이전트 u_i^0와 상호작용 함으로써 그들의 작업들을 순서대로 수행한다.

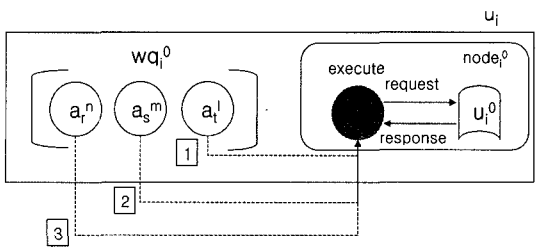

그림 1 세 이동 에이전트들이 서비스 u_i를 접근하고자 하는 예

에이전트들은 고장이 발생하면 자신의 휘발성 저장소에 있는 자신의 상태를 잃어버리고, 수행을 멈추는 파손-고장(crash-failure) 모델을 따른다고 가정한다[14]. 또한, 이동 에이전트들은 선입선출(First-In First Out) 순서로 메시지를 전달하고, 신뢰성 있고 분할되지 않는 네트워크에 연결되어있다.

이동 에이전트를 위한 중복 서비스 에이전트들의 집합은 이동 에이전트에게 동일한 서비스를 제공해야 한다. 따라서, 중복 서비스 에이전트들이 다수의 이동 에이전트들로부터 공유객체(shared object)에 대한 다수의

읽기와 갱신으로 구성된 일련의 오퍼레이션들을 요구받은 경우, 이 오퍼레이션들은 일렬화가능(linearizable)해야 한다[7]. 이러한 중복 서비스를 위한 결함포용프로토콜이 지켜야할 일관성 기준인 일렬화가능성(linearizability)은 다음과 같이 속성들을 만족해야 한다:

- 인터리빙된 오퍼레이션들은 해당 객체의 올바른 복사본의 명세를 만족해야 함
- 요구된 모든 오퍼레이션들의 수행순서는 발생한 실시간과 일관적이어야 함

그룹 u_i에서 서비스 에이전트 q는 유니캐스트와 멀티캐스트 통신 프리미티브를 사용하여 그룹 u_i에 속한 다른 서비스 에이전트들과 통신할 수 있다. 시스템에서 사용되는 멀티캐스트 통신 프리미티브는 일반적으로 수동형 중복기법의 정당성을 보장하기 위해 사용되는 뷰 동기적 멀티캐스트(View Synchronous Multicast-VSCAST)[15]이다. VSCAST는 그룹 u_i의 컨텍스트내에서 정의되고, 그룹 u_i의 일련의 뷰들 $v_0(u_i),v_1(u_i),...,v_f(u_i),...$의 개념에 기반한다. 뷰 $v_f(u_i)(f\geq 0)$는 어떤 시간 t에 정상적이라고 인식되는 그룹 u_i의 멤버들의 집합이다. 뷰 $v_f(u_i)$에 속한 서비스 에이전트 q가 고장났다고 여겨지거나 새로운 서비스 에이전트 z가 그룹 u_i에 가입하려고 한다면, 새로운 뷰 $v_{f+1}(u_i)$가 설정된다. u_i에 속한 어떤 서비스 에이전트 q가 메시지 m를 $v_f(u_i)$에게 송신한다면, VSCAST는 다음과 같은 속성을 보장한다: 뷰 $v_f(u_i)$에 속한 서비스 에이전트 q가 메시지 m을 $v_f(u_i)$에서 전달하고 뷰 $v_{f+1}(u_i)$를 설정했다면, 뷰 $v_{f+1}(u_i)$가 설정한 $v_f(u_i)$에 속한 모든 r은 m을 전달한 후 $v_{f+1}(u_i)$를 설정한다.

3. 확장적 수동형 중복 프로토콜 설계

본 절에서는 결정적 서비스가 비결정적 서비스보다 일관성 보장을 위해 보다 약한 제한조건만을 만족시키면 된다는 관점에서 앞에서 언급한 기존 수동형 중복 프로토콜 연구들의 문제점을 해결하기 위해 각 서비스의 수행형태에 따라 그 서비스에 알맞은 수동형 중복 프로토콜을 적용하는 확장적인 전략을 제안한다. 이러한 목적을 달성하기 위해 비결정적 중복 서비스 프로토콜로 PRPNS(Passive Replication Protocol for Non-deterministic Services)를, 결정적 중복 서비스 프로토콜로 PRPDS(Passive Replication Protocol for Deterministic Services)를 설계한다.

3.1 비결정적 서비스의 수동형 중복 프로토콜(PRPNS)

3.1.1 정상수행 알고리즘

이동 에이전트들이 동시에 비결정적 중복 서비스 도메인으로 전송될 때, 기존의 수동형 중복 프로토콜 CPRP(Centralized Passive Replication Protocol)을 적용함으로써 일관성이 보장된다. 이동 에이전트 a_j가 비결정적 중복 서비스 도메인 u_i의 서비스를 사용하고자 할 때, a_j는 현재의 노드로부터 u_i의 주 서비스 에이전트 u_i^{prim}이 수행하는 노드로 전송되어야 한다. 이후에 다음과 같은 단계들이 순서대로 수행된다:

단계 1. 주 서비스 에이전트 u_i^{prim}이 이동 에이전트 a_j로부터 요구 메시지를 수신한 경우, 주 서비스 에이전트는 그 메시지를 처리한다. 이 때, (resp, psn, update-state)를 생성한다. 여기서 resp는 그 요구 메시지 처리 결과이고, psn은 메시지 처리일련번호이고, update-state는 그 메시지를 처리함으로써 갱신된 주 서비스 에이전트의 상태이다.

단계 2. 주 서비스 에이전트 u_i^{prim}는 모든 보조 서비스 에이전트들에게 갱신 메시지update_msg(resp, psn, update-state, aid, reqid)를 VSCAST 프리미티브를 사용하여 전달한다. 여기서, aid는 그 요구 메시지를 송신한 이동 에이전트의 식별자이고, reqid는 그 메시지의 송신 일련번호이다. 각 보조 서비스 에이전트가 그 메시지를 수신하였을 때, update-state를 사용하여 자신의 상태를 갱신하고, (resp, aid, reqid)를 자신의 버퍼에 저장한 후, 확인(acknowledgement) 메시지를 주 서비스 에이전트에게 전달한다. 여기서, (resp, aid, reqid)는 이후에 u_i^{prim}이 고장나더라도 해당 메시지의 처리가 정확하게 한번만 수행되는 시멘틱(exactly-once semantics)[2]을 보장하기 위해 유지되어야 한다.

단계 3. 모든 살아있는 보조 서비스 에이전트로부터 확인 메시지를 수신한 후, 주 서비스 에이전트는 resp를 이동 에이전트 a_j에게 전달한다.

그러므로, 기존 프로토콜 CPRP은 모든 방문 이동 에이전트들이 각 서비스 도메인의 주 서비스 에이전트를 수행하는 노드로만 전송되어 자신의 일을 수행하도록 한다. 이러한 수행형태는 매우 많은 수의 이동 에이전트들이 임의의 비결정적 서비스를 사용하기 위해 해당 서비스 도메인으로 전송되는 경우, 극심한 과부하를 발생시킨다. 예를 들어, 그림 2에서 세 이동 에이전트 a_t^l, a_s^m, a_r^n이 동시에 u_i의 주 서비스 에이전트 u_i^2가 수행되는 노드 $node_i^2$로 전송된다. 그 이동 에이전트들이 앞에서 언급한 프로토콜 CPRP를 사용하여 u_i^2만을 통해 u_i의 서비스를 순서대로 수행한다. 따라서, 본 논문에서는 이러한 문제점을 해결하기 위해 이동 에이전트들이 반드시 주 서비스 에이전트만이 아니라 임의의 보조 서비스 에이전트를 수행하고 있는 노드로 전송되어 수행될 수 있도록 하는 PRPNS를 설계하고자 한다.

PRPNS에서는 CPRP와 달리 이동 에이전트들이 반드시 주 서비스 에이전트만이 아닌 임의의 서비스 에이전트 p를 수행하는 노드로 각각 전송될 수 있도록 한다.

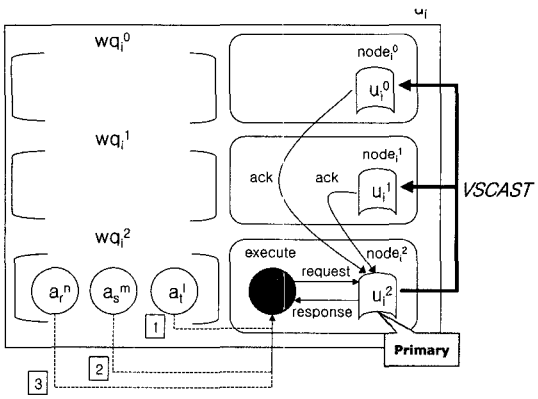

그림 2 기존 수동형 중복 프로토콜 CPRP에서 세 이동 에이전트들이 중복 서비스 u_i를 접근하는 수행 예

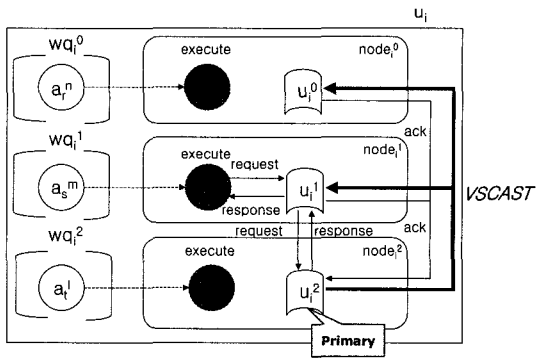

그림 3 제안된 프로토콜 PRPNS에서 세 이동 에이전트들이 중복 서비스 ui를 접근하는 수행 예

이후에, 그 서비스 에이전트 p는 자신의 노드로 이주한 이동 에이전트들로부터 요구 메시지를 수신한다면, 그 요구 메시지를 주 서비스 에이전트에게 전달한다. 다음 순서로 비결정적 서비스를 위한 일관성 조건을 만족시키기 위해, 주 에이전트는 앞에서 언급한 CPRP 단계들 중 단계 1부터 2까지 수행한다. 주 서비스 에이전트가 모든 살아있는 보조 서비스 에이전트로부터 확인 메시지를 수신한 후에, 그 요구 메시지 처리에 대한 응답 메시지를 서비스 에이전트 p에게 송신한다. 이러한 응답 메시지를 수신한 p는 그 메시지를 해당 이동 에이전트에게 전달한다. 그림 3은 제안한 프로토콜 PRPNS가 수행하는 과정의 예를 보여준다. 이 그림에서 세 이동 에이전트 a_t^l, a_s^m, a_r^n는 서비스 u_i를 접근하기 위해 각각 노드 $node_i^2$, $node_i^1$, $node_i^0$로 전송된다. 이후에 그 이동 에이전트들은 각각 서비스 에이전트 u_i^2, u_i^1, u_i^0를 통해 자신의 작업을 수행한다. 여기서, u_i^1가 이동 에이전트 a_s^m로부터 요구 메시지를 수신한 경우, 이 메시지를 주 서비스 에이전트 u_i^2에게 전달한다. 이후에 주 서비스 에이전트는 그 메시지를 처리하고, VSCAST 프리미티브를 사용하여 다른 서비스 에이전트 u_i^1, u_i^0과 동기화 한다. u_i^2는 보조 서비스 에이전트 u_i^1, u_i^0로부터 확인 메시지를 수신한 후, 그 요구 메시지 처리에 대한 응답 메시지를 서비스 에이전트 u_i^1에게 전달한다. 또한, u_i^1는 이 응답 메시지를 이동 에이전트 a_s^m로 송신한다. 이 예로부터 매우 많은 수의 이동 에이전트들이 임의의 비결정적 서비스를 사용하기 위해 해당 서비스 도메인으로 전송되는 경우, PRPNS는 기존의 수동형 중복 프로토콜 CPRP에 비해 중복 서비스의 확장성을 향상시킬 수 있다는 것을 알 수 있다. PRPNS에서 한 이동 에이전트가 보조 서비스 에이전트를 통해 해당 서비스를 접근하고자 할 때, CPRP보다 요구 메시지당 두 개의 부가적인 메시지를 요구한다. 그러나, 각 서비스 도메인은 일반적으로 이더넷과 같은 근거리 네트워크상에서 구성되기 때문에 이러한 부가적인 메시지 비용은 매우 낮다.

3.1.2 회복 알고리즘

3.1.1절은 프로토콜 PRPNS에서 모든 서비스 에이전트들이 정상적으로 수행되는 경우에 대해서만 설명하였다. 그러나, 서비스 에이전트가 고장나는 경우, 고장 시에도 일관성 조건을 보장하기 위해서는 이를 처리하는 회복 알고리즘이 필요하다. 여기서, $request_i$를 이동 에이전트 i로부터 전달받은 요구 메시지라 하고, $resp_i$를 $request_i$의 처리 후 응답 메시지라 한다. 또한, $update_i$를 주 서비스 에이전트가 $request_i$를 처리함으로써 갱신된 자신의 상태를 포함하는 갱신 메시지라 하고, ack_i를 $update_i$의 확인 메시지라 한다. 만약, 임의의 보조 서비스 에이전트 u_i^{backup}가 고장난 경우, PRPNS는 중복 서비스 에이전트 그룹 u_i로부터 u_i^{backup}를 단지 제거시키기만 하면 된다. 그러나, 주 서비스 에이전트 u_i^{prim}이 고장나는 경우에는 다음과 같이 수행해야 한다.

경우 1. 주 서비스 에이전트 u_i^{prim}이 단계 1을 수행하기 전에 고장난 경우

이 경우에는 모든 보조 서비스 에이전트 중 새로운 주 서비스 에이전트 $u_i^{prim-new}$가 선출된다. 또한, 이전 주 서비스 에이전트 u_i^{prim}로부터 $request_i$에 대한 어떠한 응답 메시지도 전달되지 못하기 때문에, 요구 메시지 $request_i$는 $u_i^{prim-new}$에게 재전송된다. 이 메시지를 수신한 $u_i^{prim-new}$는 이후에 단계 1부터 3까지 수행하기만 하면 된다.

경우 2. 주 서비스 에이전트 u_i^{prim}이 $update_i$를 모든 서비스 에이전트들에게 송신하였지만, 이동 에이전트 i에게 아직 $resp_i$를 전달하지 못한 상태에서 고장난 경우

경우 1과 같이 모든 살아있는 서비스 에이전트들에 의해 새로운 주 서비스 에이전트 $u_i^{prim-new}$가 선출된다. 이 때, 일렬화가능성(linearizability)을 보장하기 위해,

모든 보조 서비스 에이전트들이 $update_i$를 수신하던지, 아니면 어떠한 서비스 에이전트도 수신하지 못하던지 해야 한다. 이러한 조건은 시스템 모델에서 언급한 VSCAST의 특성에 의해 반드시 보장된다. 만약, 모두 $update_i$를 수신하지 못한 경우는 경우 1과 동일하다. 그렇지 않다면, 단계 2에서 주 서비스 에이전트 $u_i^{prim-new}$의 버퍼에 저장된 ($resp_i$, i, reqid)를 사용하여 정확하게 한번만 수행되는 시멘틱을 보장할 수 있다. 즉, $u_i^{prim-new}$에게 그 메시지 $request_i$가 재 전송된다면 $u_i^{prim-new}$는 그 메시지를 재처리하지 않고, 단지 자신의 버퍼에 있는 $resp_i$를 해당 이동 에이전트 i에게 전달한다.

경우 3. 주 서비스 에이전트 u_i^{prim}이 이동 에이전트 i에게 $resp_i$를 전달한 후 고장난 경우

경우 1과 같이 새로운 주 서비스 에이전트 $u_i^{prim-new}$가 선출된다. 따라서, 이후부터 이동 에이전트 i가 자신의 지역 서비스 에이전트로 송신하는 모든 요구 메시지는 서비스 에이전트 $u_i^{prim-new}$로 전달되어 처리된다.

3.2 결정적 서비스의 수동형 중복 프로토콜(PRPDS)

3.2.1 정상수행 알고리즘

결정적 중복 서비스를 수행할 경우에는 주 서비스 에이전트만 모든 이동 에이전트 요구 메시지를 처리할 필요가 없기 때문에 반드시 주 서비스 에이전트만이 아니라, 임의의 중복 서비스 에이전트가 그 메시지를 처리할 수 있도록 함으로써, 동시에 매우 많은 이동 에이전트로부터 요구 메시지들이 전달되더라도 중복 서비스 에이전트들의 부하를 고려하여 분배함으로써 전체적인 시스템 성능이 급격히 떨어지지 않으면서 서비스 수행이 가능하다. 따라서, 이러한 목적을 달성하기 위해서는 먼저 결정적 중복 서비스가 일관성 조건을 만족하기 위한 속성들을 재 정의하면 표 1과 같다.

이러한 바람직한 속성을 만족시키기 위해, 프로토콜 PRPDS는 각 방문 이동 에이전트가 접근하고자 하는 서비스 도메인에서 중복 서비스 에이전트들 중 임의의 서비스 에이전트가 수행하는 노드로 전달되고 수행할 수 있도록 한다. PRPDS는 이동 에이전트 i가 주 서비스 에이전트 u_i^{prim}이 수행하는 노드로 전송된다면, 앞에서 언급한 PRPNS의 3 단계들을 동일하게 수행한다. 만약, 이동 에이전트 i가 한 보조 서비스 에이전트 u_i^{backup}이 수행하는 노드로 전송된다면, 다음과 같은 단계들이 순서대로 수행된다:

단계 1. 보조 서비스 에이전트 u_i^{backup}가 이동 에이전트 i로부터 요구 메시지를 수신한다. 이후에 u_i^{backup}는 그 메시지의 처리일련번호 psn을 주 서비스 에이전트 u_i^{prim}에게 요구한다. 이 경우, 주 서비스 에이전트는 그 메시지의 psn을 결정한 후 u_i^{backup}에게 알려준다. 이후에 주 서비스 에이전트는 그 메시지를 처리하고, 그 메시지의 (resp, psn, update-state, aid, reqid)를 자신의 버퍼에 저장한다. 한편, u_i^{backup}는 그 메시지의 psn을 수신한 후, 그 메시지를 처리하고, (resp, psn, update-state)를 생성시킨다.

단계 2. 보조 서비스 에이전트 u_i^{backup}는 모든 다른 서비스 에이전트들에게 (resp, psn, update-state, aid, reqid)를 포함하는 갱신 메시지 update_msg를 VSCAST 프리미티브를 사용하여 전달한다. 주 서비스 에이전트를 제외한 모든 보조 서비스 에이전트가 그 갱신 메시지를 수신한 경우에는, update-state를 사용하여 자신의 상태를 갱신하고, (resp, aid, reqid)를 자신의 버퍼에 저장한 후, 확인 메시지를 u_i^{backup}에게 전달한다. 주 서비스 에이전트가 갱신 메시지를 수신한 경우에는 단지 해당 (resp, psn, update-state, aid, reqid)를 삭제하고, (resp, aid, reqid)를 자신의 버퍼에 저장한 후, 확인 메시지를 보조 서비스 에이전트 u_i^{backup}에게 전달한다.

단계 3. u_i^{backup}는 모든 살아있는 서비스 에이전트로부터 확인 메시지를 수신한 후, 해당 응답 메시지 resp를 이동 에이전트 i에게 전달한다.

그림 4는 PRPDS에서 세 이동 에이전트 a_t^l, a_s^m, a_r^n가 각각 서비스 에이전트 u_i^2, u_i^1, u_i^0를 통해 서비스 u_i를 접근하고자 하는 수행 예를 보여준다.

이 그림에서 단지 이동 에이전트 a_s^m가 u_i^1에게 요구 메시지를 송신한 경우, PRPDS의 수행과정만을 표현하였다. 이 경우 u_i^1는 주 서비스 에이전트 u_i^2에게 요구 메시지의 psn을 요청한다. 그러면, u_i^2는 그 메시지의 psn을 결정하여 u_i^1에게 전달하고, 그 메시지를 처리한 후, 그 메시지의 (response, psn, update-state, aid, reqid)를 자신의 버퍼에 저장한다. 한편, u_i^1는 그 메시지의 psn을 수신한 후, 그 메시지를 처리하고, 결정적

표 1 결정적 중복 서비스가 일관성 조건을 만족하기 위해 유지해야 할 속성

원자성 속성(Atomicity property)
• 임의의 중복 서비스 에이전트 p가 이동 에이전트로부터 요구 메시지를 수신하고, 그 요구 메시지의 처리일련번호를 주 서비스 에이전트로부터 수신한 후에는 그 서비스 에이전트 p가 해당 요구 메시지를 직접 처리하고, 다른 살아있는 중복 서비스 에이전트들과 일관성을 유지하기 위한 과정을 수행한다.
순서 속성(Ordering property)
• 주 서비스 에이전트만이 모든 요구 메시지의 처리일련번호를 결정한다.

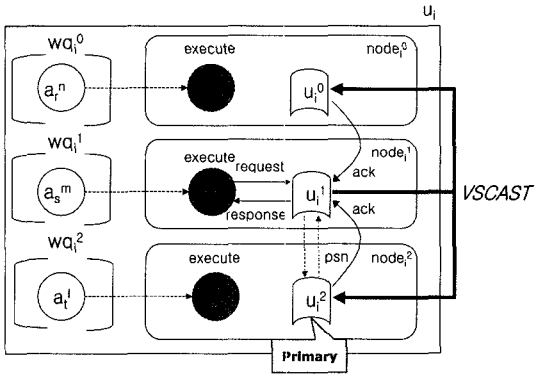

그림 4 제안된 프로토콜 PRPDS에서 세 이동 에이전트들이 중복 서비스 ui를 접근하는 수행 예

서비스를 위한 일관성 조건을 만족시키기 위해 VSCAST 프리미티브를 사용하여 다른 서비스 에이전트 u_i^2, u_i^0과 동기화 한다. u_i^1는 모든 살아있는 서비스 에이전트로부터 확인 메시지를 수신한 후에, 그 요구 메시지의 응답 메시지 response를 이동 에이전트 a_s^m로 송신한다. 따라서, 이 그림의 예로부터 프로토콜 PRPDS는 각 방문 이동 에이전트가 각 서비스 도메인에서 중복 서비스 에이전트들 중 임의의 서비스 에이전트를 통해 해당 서비스를 사용하게 함으로써 중복 서비스의 확장성을 매우 향상시키고, 서비스 에이전트들 간에 요구 메시지 처리 및 동기화에 의한 부하를 균등하게 분산시킬 수 있다는 것을 알 수 있다.

3.2.2 회복 알고리즘

PRPDS는 주 서비스 에이전트 u_i^{prim}이 고장나는 경우에 3.1.2에서 설명한 PRPNS에서의 회복 알고리즘을 동일하게 수행한다. 그러나, 임의의 보조 서비스 에이전트 u_i^{backup}가 고장난 경우, PRPDS는 PRPNS와 달리 다음과 같은 부가적인 경우들을 고려해야 한다:

경우 1. 보조 서비스 에이전트 u_i^{backup}가 단계 1에서 주 서비스 에이전트 u_i^{prim}에게 요구 메시지의 psn을 요청하기 전에 u_i^{backup}이 고장난 경우

이 경우에는 단지 보조 서비스 에이전트 u_i^{backup}만 중복 서비스 에이전트 그룹에서 제외시키면 된다.

경우 2. 보조 서비스 에이전트 u_i^{backup}가 단계 1을 완료하기 전에 고장난 경우

경우 2.1. 주 서비스 에이전트 u_i^{prim}이 고장난 경우

u_i^{backup}와 u_i^{prim}를 제외한 모든 서비스 에이전트들에 의해 새로운 주 서비스 에이전트 $u_i^{prim-new}$가 선출된다.

경우 2.2. 주 서비스 에이전트 u_i^{prim}이 살아있는 경우

주 서비스 에이전트 u_i^{prim}이 u_i^{backup}의 고장을 감지했을 때, 해당 요구 메시지의 (response, psn, update-state, aid, reqid)를 자신의 버퍼로부터 찾아서 VSCAST를 사용하여 다른 모든 보조 서비스 에이전트들에게 전달한다. 만약, 보조 서비스 에이전트 r에게 그 요구 메시지 $request_i$이 재전송된다면 r는 그 메시지를 처리하지 않고, 단지 자신의 버퍼에 있는 response를 해당 이동 에이전트에게 전달한다.

경우 3. 보조 서비스 에이전트 u_i^{backup}가 단계 2에서 갱신 메시지 $update_i$를 모든 다른 서비스 에이전트들에게 송신하였지만, 이동 에이전트 i에게 아직 response를 전달하지 못한 상태에서 고장난 경우

이 경우에 u_i^{backup}가 $update_i$를 VSCAST를 사용하여 전송하였기 때문에, 시스템의 일관성 조건은 만족된다. 따라서, 단지 보조 서비스 에이전트 u_i^{backup}만 그 중복 서비스 에이전트 그룹에서 제외시키면 된다.

경우 4. 보조 서비스 에이전트 u_i^{backup}가 단계 3을 완료 후 고장난 경우

이 경우 단지 보조 서비스 에이전트 u_i^{backup}만 그 그룹에서 제외시키면 된다.

4. 성능평가

본 절에서는 본 연구에서 PARSEC 시뮬레이션 언어 [16]를 사용하여 제안된 두 중복 서비스 프로토콜 PRPNS, PRPDS와 기존의 수동형 중복 프로토콜 CPRP과 성능을 비교하기 위해 시뮬레이션을 수행한다. 이 시뮬레이션에서는 이동 에이전트의 요구 메시지 당 평균 처리 응답시간 ARTMARM(Average Response Time per Mobile Agent Request Message)을 성능평가 기준으로 한다. 또한, 한 네트워크에 하나의 특정한 서비스만을 제공한다고 가정하고, 해당 서비스 에이전트 그룹을 형성시킨다.

시뮬레이션되는 시스템의 구성은 다음과 같다. 모델링된 네트워크는 이더넷과 같은 다중접근 근거리통신망(multi-access LAN)이다. 시스템을 구성하는 노드들은 그 네트워크 상에서 동일하고 균일하게 분산된다. 모든 에이전트는 UDP/IP 프로토콜을 사용하여 상호적으로 통신한다. 임의의 두 노드간 메시지전송 지연시간은 표 2과 같이 구성된다.

한 에이전트가 메시지를 송신 혹은 수신하고자 할 때, CPU의 부하가 높은 경우 그 CPU의 부하가 낮아질 때까지 그 메시지의 처리는 지연된다. 또한, 네트워크 자원은 선입선출(First In First Out)형으로 노드들간에 임의적으로 할당된다. 그리고, 시뮬레이션 환경에서 고정적으로 설정되어지는 파라미터 값들은 표 3과 같고, 성능에 영향을 미치는 변화 가능한 파라미터들은 표 4와 같다. 특히, 중복 서비스 에이전트들 중 생성된 각 이동 에이전트의 목적 서비스 에이전트 선택은 균일한 분산분포(uniform distribution)를 따르고, 단위시간당

표 2 메시지전송 지연시간의 구성 요소

시간	설명
T_s	· 송신노드에서 송신 오퍼레이션을 수행하는데 걸리는 시간
T_t	· 송신노드의 네트워크 인터페이스로부터 수신노드의 네트워크 인터페이스까지 메시지를 전송하는데 걸리는 시간
T_r	· 수신노드에서 수신 오퍼레이션을 수행하는데 걸리는 시간

표 3 설정된 시뮬레이션 파라미터

파라미터	설정된 값
T_s	0.269 ms
T_t	0.107 ms
T_r	0.292 ms
이동 에이전트가 요구한 하나의 오퍼레이션 처리시간	1 ms
각 이동 에이전트가 발생하는 요구 메시지 수	1~20 개
서비스 에이전트의 상태를 갱신하는데 걸리는 시간	80 μs

표 4 성능 파라미터

파라미터	설명
NORSA	한 서비스 도메인에 할당된 중복 서비스 에이전트 수
NOMAT	단위시간당 서비스 도메인에 전송되는 평균 이동 에이전트의 수

서비스 도메인에 전송되는 이동 에이전트의 수는 지수 분포를 따른다. 따라서, 본 연구진은 앞에서 언급한 시뮬레이션 환경에 기반하여 표 4에서의 성능 파라미터에 따라 응답시간을 측정하여 설계된 중복서비스 프로토콜들의 성능을 분석하고자 한다.

그림 5부터 7까지 하나의 결정적 서비스 도메인에 중복된 서비스 에이전트들의 수 NORSA를 일정한 범위로 변화시키면서 세 개의 중복 프로토콜 CPRP와 PRPNS, PRPDS의 이동 에이전트의 요구 메시지 당 평균 처리 응답시간 ARTMARM을 보여준다. 이 그림들에서 단위 시간당 서비스 도메인에 전송되는 평균 이동 에이전트의 수 NOMAT를 각각 30개/s, 40개/s, 50개/s로 각각 설정하였다. 그림 5에서 세 프로토콜의 NORSA가 증가함에 따라, 그들의 응답시간이 모두 증가한다. 이는 한 서비스 도메인에 중복된 서비스 에이전트 수 NORSA가 증가함에 따라, 그 프로토콜들은 해당 서비스 에이전트 그룹의 더 높은 동기화 비용을 발생시키기 때문이다. 그러나, PRPNS와 PRPDS의 응답시간은 CPRP보다 많이 낮다는 것을 알 수 있다. 또한, PRPDS는 PRPNS보다 매우 좋은 성능을 발휘한다. 특히, NORSA가 높아질수록, 그들의 증가율간의 차이가 더 큰 것을 알 수 있다. 이 경우, PRPDS와 PRPNS의 응답시간은 CPRP의 응답시간에 비해 각각 22.4%~45.1%과 10.3%~20.3%만큼 줄인다. 이와 유사하게 그림 6과 7에서는 제안한 두 프로토콜들의 응답시간 감소율은 PRPDS에서 (24.4%~47.8%, 26.3%~52.3%)와 PRPNS에서 (11.5%~23.5%, 15.2%~26.4%)정도이다. 따라서, 이 세 그림들을 비교함으로써 단위시간당 서비스 도메인에 전송되는 평균

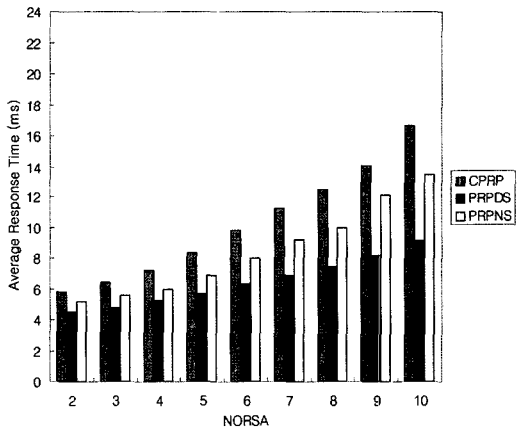

그림 5 ARTMARM vs. NORSA (NOMAT = 30개/s)

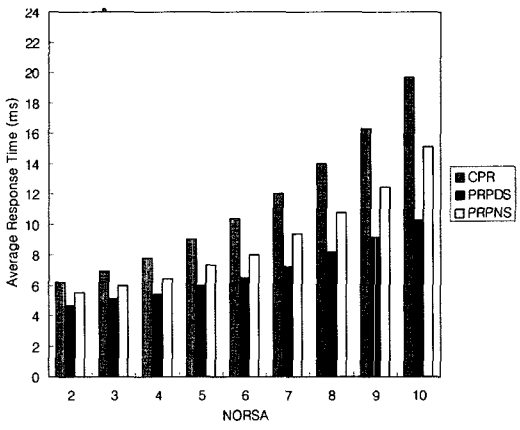

그림 6 ARTMARM vs. NORSA (NOMAT = 40개/s)

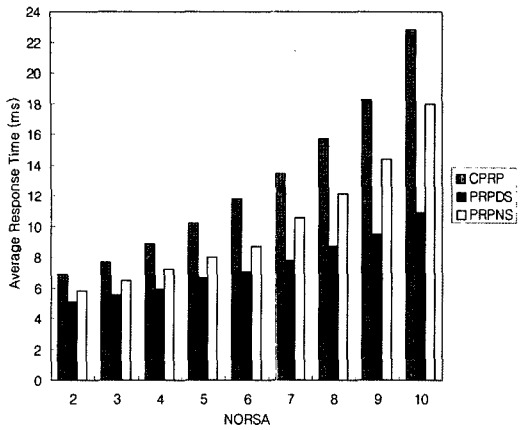

그림 7 ARTMARM vs. NORSA (NOMAT = 50개/s)

이동 에이전트의 수 NOMAT를 증가함에 따라, 세 프로토콜들의 응답시간 증가율간의 차이가 더 커짐을 알 수 있다. 따라서, 이러한 시뮬레이션 결과로부터 중복된 서비스 도메인에서 제안된 두 프로토콜들, 특히 PRPDS는 CPRP에 비해 많은 수의 이동 에이전트들에 의해 발생하는 높은 부하를 중복 서비스 에이전트들에게 균등하게 분배함으로써 시스템의 확장성을 매우 향상시킬 수 있다는 결론을 내릴 수 있다.

5. 관련연구 및 논의

준 능동적 중복(semi-active replication) 프로토콜[17]은 능동적 중복기법의 단점 중에 하나인 서비스 에이전트에서 수행되는 모든 오퍼레이션들이 결정적(deterministic)이어야 한다는 제한사항을 극복한 기법이다. 이러한 목적은 모든 서비스 에이전트가 이동 에이전트로부터 요구받은 오퍼레이션들의 순서를 정하고 독립적으로 처리한 후, 수동적 중복기법과 유사하게 서비스 에이전트들 중 주 서비스 에이전트가 자신의 상태를 다른 서비스 에이전트들에게 전달하게 함으로써 달성된다. 그러나, 이 기법은 능동적 중복기법과 같이 정상수행시 높은 자원의 소비를 발생시키고, 수동적 중복기법과 같이 주 서비스 에이전트 고장 시 높은 재구성 비용을 발생시킨다.

준 수동적 중복(semi-passive replication) 프로토콜[11]은 주 서비스 에이전트의 고장에 대한 합의를 하기 위해 어떠한 그룹 관리 서비스도 필요로 하지 않으면서 수동적 중복기법의 모든 장점을 가지는 기법이다. 즉, 이 기법에서는 합의문제(consensus problem)를 해결하기 위해 일반적으로 사용되는 회전식 조정자 패러다임(rotating coordinator paradigm)에 기반함으로써 에이전트의 고장을 짐작하기(suspecting) 위해 빠른 타임아웃 값을 사용하면서, 만약 고장에 대한 짐작이 잘못되었다 하더라도 높지 않은 재구성 비용을 발생시킨다. 그러나, 이 기법은 각 이동 에이전트가 서비스 요구 시 능동적 중복기법과 같이 모든 서비스 에이전트들에게 브로트캐스트(broadcast) 혹은 멀티캐스트(multicast)를 해야 한다는 단점이 있다.

조정자-코홀트(coordinator-cohort) 중복 프로토콜[18]은 ISIS 시스템의 컨텍스트에서 구현된 하나의 변형된 수동형 중복 프로토콜이다. 통신 패턴의 관점에서 수동형 중복과 매우 유사하나 이 프로토콜에서는 능동형 중복 기법에서와 같이 모든 중복자가 클라이언트 요구를 수신할 수 있다. 이러한 특성은 주 중복자 즉, 조정자의 고장 또한 마스크할 수 있도록 한다. 그러나, 조정자만이 클라이언트 요구를 처리하고, 검사점에 의해 코홀트들의 상태를 갱신한다. 따라서, 요구처리 결과는 비결정적일 수 있는 조정자에서의 수행에 의해 결정된다. 만약, 조정자가 고장나는 경우, 코홀트 중 새로운 조정자를 선출하여 마지막 검사점으로부터 그 수행을 진행시킨다. 그러므로, 검사점들은 출력에 따라 조정되어야 한다.

6. 결 론

본 논문에서는 결정적 서비스가 비결정적 서비스보다 일관성 보장을 위해 보다 약한 제한조건만을 만족시키면 된다는 관점에서 각 서비스의 수행형태가 비결정적인지 결정적인지에 따라 그 서비스에 효율적인 수동형 중복 프로토콜을 적용하였다. 이를 위해 본 논문에서는 비결정적 중복 서비스 프로토콜로 PRPNS를, 결정적 중복 서비스 프로토콜로 PRPDS를 설계하였다. 이 두 프로토콜들은 모두 방문 이동 에이전트들이 반드시 주 서비스 에이전트가 아니라 보조 서비스 에이전트를 수행하는 임의의 노드로 전달되어 수행할 수 있도록 한다. 특히, 프로토콜 PRPDS는 임의의 보조 서비스 에이전트가 이동 에이전트 요구 메시지를 수신한 경우, 주 서비스 에이전트로부터 그 요구 메시지의 전달 일련번호를 얻은 후에, 그 보조 서비스 에이전트가 직접 해당 요구 메시지를 처리하고 다른 중복 서비스 에이전트들과의 조정하는 역할을 할 수 있도록 한다. 따라서, PRPDS는 알맞은 부하균등 기법[19]과 결합된다면, 동시에 매우 많은 이동 에이전트들이 특정된 서비스 도메인으로 전달되더라도 각 서비스 에이전트의 작업부하를 고려하여 알맞게 분배함으로써 전체 시스템 성능이 급격히 떨어지지 않으면서 계속적으로 서비스 수행이 가능하다. 시뮬레이션 결과로부터 제안된 두 프로토콜들은 이동 에이전트 시스템에서 매우 많은 수의 이동 에이전트들이 동시에 서비스를 접근하고자 하는 경우에 기존 프로토

콜에 비해 해당 중복서비스에 대한 확장성을 매우 향상시킬 수 있다는 것을 알 수 있다.

본 논문에서 설계된 프로토콜들의 성능을 최적으로 발휘하기 위해서는 이동 에이전트에 의한 부하를 동적으로 균등하게 분배하는 기법의 역할이 매우 중요하다. 따라서, 현재 본 연구진은 제안된 두 프로토콜에 가장 적합한 동적부하균등기법을 설계하여 결합하는 연구를 수행하고 있다. 또한, 제안한 프로토콜들은 이동 에이전트에 대한 인증(authentication), 권한(authorization), 자원접근제어(resource access control) 및 부당한 코드변경(code tampering) 등의 보안문제[20] 해결에 초점을 맞추고 있지 않기 때문에, 이에 대한 구체적인 보안 메커니즘들이 제시되지 않고 있다. 그러나, 이러한 문제점들은 이동 에이전트 기반 애플리케이션이 널리 사용되는데 매우 치명적인 장애물이 될 수 있다[21]. 그러므로, 본 논문에서 설계된 프로토콜들이 다양한 시스템 환경의 특성에 따라 알맞은 기존의 보안 메커니즘들[20,22]과 통합되기 위한 향후 연구 과제를 수행하고자 한다.

참 고 문 헌

[1] Pleisch, S. and Schiper, A., "Fault-Tolerant Mobile Agent Execution," IEEE Transactions on Computers, Vol.52, No.2, pp. 209-222, 2003.

[2] Rothermel, K. and Strasser, M., "A Fault-Tolerant Protocol for Providing the Exactly-Once Property of Mobile Agents," Proc. of the 17th IEEE Symposium on Reliable Distributed Systems, pp. 100-108, 1998.

[3] Picco, G., "Mobile Agents: An Introduction," Journal of Microprocessors and Microsystems, Vol.25, No.2, pp. 65-74, April 2001.

[4] Li, Z. and Parashar, M., "A Decentralized Agent Framework for Dynamic Composition and Coordination for Autonomic Applications," Proc. of the 3rd International Workshop on Self-Adaptive and Autonomic Computing Systems, Copenhagen, Denmark, pp. 165-169, August 2005.

[5] Curino, C., Giani, M., Giorgetta, M., Giusti, A., Murphy, A. and Picco, G., "Mobile Data Collection in Sensor Networks: The TinyLime Middleware," Journal of Pervasive and Mobile Computing, Vol.4, No.1, pp. 446-469, December 2005.

[6] Fukuda, M., Kashiwagi, K., Kobayashi, S., "AgentTeamwork: Coordinating Grid-Computing Jobs with Mobile Agents," In Special Issue on Agent-Based Grid Computing, International Journal of Applied Intelligence, to appear in 2006.

[7] Guerraoui, R. and Schiper, A., "Software-Based Replication for Fault Tolerance," IEEE Computer, Vol.30, pp. 68-74, 1997.

[8] Marchetti, C., Piergiovanni, S. and Baldoni, R., "A Three-tier Active Replication Protocol for Large Scale Distributed Systems," IEICE TRANSACTIONS on Information and Systems Vol.E86-D, No.12, pp.2544-2552, Dec. 2003.

[9] Schneider, F., "Implementing fault-tolerant services using the state machine approach: A tutorial," ACM Computing Surveys, Vol.22, pp. 299-319, 1990.

[10] Budhiraja, N., Marzullo, K., Schneider, F. and Toueg, S., Distributed Systems, 2nd Ed., pp. 199-216, Addison-Wesley, 1993.

[11] Defago, X. and Schiper, A., "Semi-passive Replication and Lazy Consensus," Journal of Parallel and Distributed Computing Systems, Vol.64, No.12, pp.1380-1398, 2004.

[12] Fischer, M. J., Lynch, N. A. and Paterson, M. S., "Impossibility of distributed consensus with one faulty process," Journal of ACM, Vol.32, pp. 374-382, 1985.

[13] Chandra, T. D. and Toueg, S., "Unreliable failure detectors for reliable distributed systems," Journal of ACM, Vol.43, No.2, pp. 225-267, 1996.

[14] Schlichting, R. D. and Schneider, F. B., "Fail-stop processors: an approach to designing fault-tolerant distributed computing systems," ACM Transactions on Computer Systems, Vol.1, pp. 222-238, 1985.

[15] Schiper, A. and Sandoz, A., "Uniform reliable multicast in a virtually synchronous environment," In Proc. of the 13rd International Conference on Distributed Computing Systems, pp. 561-568, 1993.

[16] Bagrodia, R., Meyer, R., Takai, M., Chen, Y., Zeng, X., Martin, J. and Song, H. Y., "Parsec: A Parallel Simulation Environment for Complex Systems," IEEE Computer, pp. 77-85, 1998.

[17] Powell, D., Chereque, M. and Drackley, D., "Fault-tolerance in Delta-4," ACM Operating Systems Review, Vol.25, pp. 122-125, 1991.

[18] Birman, K. P., Joseph, T. A., Raeuchle, T. and Abbadi, A. E., "Implementing fault-tolerant distributed objects," IEEE Transactions on Software Engineering, Vol.11, No.6, pp. 502-508, 1985.

[19] Bryhni, H., Klovning, E. and Kure, O., "A Comparison of Load Balancing Techniques for Scalable Web Servers," IEEE Network, Vol.14, pp. 58-64, 2000.

[20] Yueming, L., Yuefeng, J., Wenjie, W., Aibo, L., "Design and Implementation of a Secure Execution Environment for the Mobile Agent," In Proc. of the International e-Conference on Computer Science, appear in July 2006.

[21] Jansen, W., "Countermeasures for mobile agent security," Computer Communications, Vol.23, No.17, pp. 1667-1676, Nov. 2000.

[22] Claessens, J., Preneel, B., Vandewalle, J., "(How) can mobile agents do secure electronic trans-

actions on untrusted hosts? A survey of the security issues and the current solutions," ACM Transactions on Internet Technology, Vol.3, No.1, pp. 28-48, Feb. 2003.

안 진 호

1997년 고려대학교 컴퓨터학과 졸업(이학학사). 1999년 고려대학교 컴퓨터학과 졸업(이학석사). 2003년 고려대학교 컴퓨터학과 졸업(이학박사). 2003년~현재 경기대학교 이과대학 전자계산학과 조교수 관심분야는 결함포용 분산시스템, 이동에이전트 시스템, 그룹통신, p2p 컴퓨팅