# 부분적 분산형 수동적 중복 알고리즘

안 진 호†


## 요 약

본 논문에서는 메시지 전달 분산시스템에서 결정적 서버를 위한 부분적 분산형 수동적 중복 알고리즘을 제안한다. 이 알고리즘은 반드시 주 서버가 아니더라도 임의의 보조서버가 자신이 수신한 클라이언트 요구에 대해 주 서버로부터 그 요구의 전달일련번호를 얻은 후, 그 보조서버가 직접 해당 요구를 처리하고, 이에 대한 다른 중복 서버들과의 조정에 대한 책임자 역할을 수행할 수 있도록 한다. 이러한 바람직한 특성때문에, 제안된 알고리즘이 기존의 부하균등 기법과 결합된다면, 주 서버에의 급격한 부하 발생을 효율적으로 피할 수 있다. 따라서, 이 알고리즘은 기존의 수동적 중복 알고리즘보다 결정적 중복 서버 시스템에 대한 높은 확장성을 제공할 수 있다. 본 논문에서 수행한 시뮬레이션에서 기존 알고리즘에 비해 제안한 알고리즘이 클라이언트 요구당 평균응답시간을 16.5%~52.3%까지 줄일 수 있다는 것을 보여준다.


**키워드 : 메시지 전달 분산시스템, 결함포용성, 수동적 중복, 결정적 서버, 확장성**

# Partially Decentralized Passive Replication Algorithm

Jinho Ahn†


## ABSTRACT

This paper presents a partially decentralized passive replication algorithm for deterministic servers in message-passing distributed systems. The algorithm allows any backup server, not necessarily the primary server, to take responsibility for processing its received client request and coordinating with the other replica servers after obtaining the delivery sequence number of the request from the primary. Thanks to this desirable feature, the algorithm with conventional load balancing techniques can efficiently avoid extreme load conditions on the primary. Therefore, it can provide better scalability of deterministic and replicated server systems than traditional passive replication algorithms. Simulation results indicate that the proposed algorithm can reduce 16.5%~52.3% of the average response time of a client request compared with the traditional ones.


**Key Words : Message-passing Distributed System, Fault-tolerance, Passive Replication, Deterministic Servers, Scalability**

## 1. 서 론

다수의 컴퓨터들로 구성된 분산 시스템은 비행교통제어, 주식거래 및 전자상거래와 같이 장시간 수행하는 애플리케이션(long-running application)을 위한 경제적인 고성능 컴퓨팅 환경을 제공해준다. 그러나 임의의 애플리케이션이 분산 시스템에서 병렬적으로 수행할 때, 시스템을 구성하는 하나의 컴퓨터가 고장나더라도, 해당 애플리케이션의 전체적 수행이 중단 혹은 실패할 수 있다. 특히, 분산시스템의 규모가 점점 더 커짐에 따라, 그 시스템의 고장률이 보다 높아질 수 있다. 따라서, 이러한 시스템은 효율적인 결함포용(fault-tolerance) 기술을 필요로 한다. 이러한 기술들 중, 프로세스 중복(process replication) 기법은 다수의 물리적으로 독립적인 컴퓨터에서 각각 동일한 서비스를 수행하는 서버 그룹을 사용함으로써 해당 서비스의 고 가용성(high availability)을 보장하는 잘 알려진 결함포용 기술이다. 프로세스 중복기법에서, 한 서비스의 각 중복 서버는 일부 다른 서버들의 고장이 발생하더라도 전역적으로 일관적인 상태(globally consistant state)를 유지해야 한다. 이러한 기법의 적용실례로 한 대형 항공사 A의 비행기 좌석예약관리시스템을 들 수 있는데, 전 세계의 수많은 일반 사용자나 여행사들이 언제, 어디서나, 자신의 컴퓨터를 사용하여 이 시스템을 통해 비행기 좌석을 경쟁적으로 예약하고자 한다. 그런데, 이러한 시스템이 단순히 사용자 동시접속에 의한 서버 과부하만을 해결하기 위해 네트워크로 연결된 다수의 물리적 컴퓨터들로 구성하여 시스템의 성능을 향상시킬 수 있다. 그러나 몇몇 서버 컴퓨터들이 고장나는 경우, 해당 항공사 A의 좌석예약이 부분적으로 혹은 전혀 이루어질 수 없게되어 항공사에게 매우 큰 경제적 손실 및 신뢰성 하락을 가져올 수 있다. 그러므로 사용자들에게 서버 고장에 대한 불편없이 좌석예약서비스를 계속적으로 제공하기 위해서는, 이 시스템에 프로세스 중복기법을 적용하여 고장난 서버 컴

† 정 회 원 : 경기대학교 정보과학부 전자계산학과 조교수
논문접수 : 2005년 3월 30일, 심사완료 : 2005년 9월 30일

퓨터의 기능을 일관성 있게 살아있는 다른 컴퓨터가 대신 수행할 수 있도록 해야한다.

이러한 일관성을 보장하기 위해 사용되는 대표적인 두 가지 중복 기법들로는 능동적 중복(active replication)과 수동적 중복(passive replication)이 있다[10]. 능동적 중복기법[13, 15]에서는, 각 정상수행 중복 서버가 클라이언트로부터 수신한 서비스 요구들을 같은 순서로 전달받고, 처리후 클라이언트에게 응답한다. 따라서, 이 기법의 장점으로는 몇몇의 중복 서버들이 고장나더라도 다른 살아있는 서버들이 전달된 서비스 요구들을 처리하고 응답할 수 있기 때문에, 클라이언트들에게 고장에 대한 투명성을 제공하고, 수동적 중복기법에 비해 중복 서버 고장 시 클라이언트에게 낮은 응답지연시간을 요구한다. 그러나 이 기법은 모든 중복된 서버가 클라이언트로부터 수신한 서비스 요구들을 동일한 순서대로 처리해야하기 때문에, 높은 정상수행시 비용(high failiure-free overhead)을 발생시키고, 서버가 수행하는 모든 오퍼레이션의 수행형태가 결정적(deterministic)이어야 한다는 제한사항을 가지고 있다.

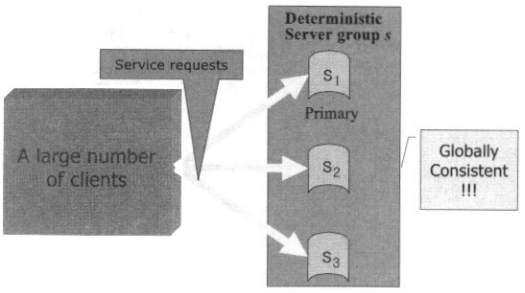
이에 비해, 수동적 중복기법[2, 5, 8]에서는 중복된 서버 집합에서 하나의 서버 즉, 주 서버(primary server)만 클라이언트로부터 일련의 서비스 요구들을 전달 받고, 처리한 후 자신의 갱신된 상태를 포함한 메시지를 다른 서버들, 즉 보조 서버(backup server)들에게 전달한다. 각 보조 서버는 수신된 메시지를 이용하여 자신의 상태를 갱신하고, 주 서버에게 응답메시지를 보낸다. 주 서버는 모든 보조 서버들로부터 응답메시지를 수신한 후, 해당 클라이언트에게 응답메시지를 전달한다. 이러한 특성은 주 서버가 고장나는 경우, 클라이언트에게 고장에 대한 투명성을 제공하지 못하고, 높은 재구성 비용을 발생시킬 수 있다. 그러나 수동적 중복기법은 다음과 같은 세 가지 중요한 장점들을 가지고 있다. 첫 번째로 이 기법은 중복 서버가 비결정적인 형태로 수행하더라도 일관성을 보장할 수 있기 때문에, 서버의 수행형태에 관계없이 적용 가능하다. 두 번째로는 능동형 중복 기법에 비해 정상수행시 낮은 자원의 소비를 발생시킨다. 세 번째로 클라이언트가 멀티캐스트가 아닌 유니캐스트 프리미티브(unicast primitive)만을 사용할 수 있다는 것이다. 따라서, 이러한 장점들 때문에, 수동적 중복 기법은 범용적이고 저비용의 결함포용 기술로써 메시지 전달 분산시스템에 적용될 수 있다.

그러나 기존의 수동적 중복 알고리즘에서는 모든 클라이언트가 서비스 요구 메시지를 주 서버에게만 전송하여 처리하도록 한다. 이후에 갱신된 상태를 모든 보조 서버들에게 전달하면, 각 보조 서버는 그 상태를 자신에게 반영하고, 주 서버에게 확인 메시지(acknowledgment message)를 송신한다. 주 서버는 모든 보조 서버로부터 확인 메시지를 수신한 후에, 해당 클라이언트에게 응답 메시지를 전달한다. 이러한 중앙집중형 제약성은 클라이언트의 서비스 요구 수가 매우 많은 경우, 주 서버가 오버로드되도록 하여, 전체 시스템 성능이 매우 저하될 수 있지만, 비결정적 서비스를 위한 일관성 보장 조건을 만족시키기 위해 반드시 필요하다. 그러나 중복 서버의 수행형태가 결정적인 경우에는 비결정적 서버보다 일관성 보장 조건이 보다 약화될 수 있기 때문에 이러한 중앙집중형 제약성을 피할 수 있다. 따라서, 본 논문에서는 결정적 서버에 적합한 일관성 보장조건을 만족하는 부분적 분산형 수동적 중복 알고리즘을 제안한다. 이 알고리즘은 임의의 보조 서버가 클라이언트 요구 메시지를 수신한 경우, 주 서버로부터 그 요구 메시지의 전달 일련번호를 얻은 후, 그 보조 서버가 직접 해당 요구 메시지를 처리하고 다른 중복 서버들과의 조정역할을 할 수 있도록 한다. 이러한 특성은 매우 많은 클라이언트로부터 서비스 요구 메시지들이 동시에 전달되더라도, 제안된 알고리즘이 기존의 부하균등 기법[4, 6]과 결합된다면, 주 서버에게만이 아니라 전체적인 중복 서버들에게 부하를 고르게 분배시킬 수 있게 함으로써 전체 시스템 성능을 급격히 떨어뜨리지 않으면서 서비스 수행을 가능하도록 한다. 따라서, 제안 알고리즘은 기존 알고리즘들에 비해 중복 서버 시스템에 대한 보다 높은 확장성을 제공할 수 있다.

본 논문은 다음과 같이 구성되어 있다. 먼저, 2절에서는 본 논문에서의 시스템 모델을 기술하고, 3절에서는 기존 수동적 중복 알고리즘의 문제점을 설명하고 이를 해결하는 새로운 알고리즘을 제안한다. 4절과 5절에서는 제안한 알고리즘의 성능평가와 관련연구에 대해 기술하고, 마지막으로 6절에서 결론을 맺는다.

## 2. 시스템 모델

본 논문에서는 어떠한 전역 메모리와 전역 시계를 가지고 있지 않고, 메시지 전송지연시간이 유한하지만 임의적인 비동기적인 분산시스템을 가정한다. 이 시스템은 분산 합의에 대한 불가능성 문제점[9]을 해결하기 위해 비신뢰적인 고장감지기를 포함한다[7]. 또한, (그림 1)과 같이 시스템은 클라이언트 프로세스들에 의해 호출되고, 서버 프로세스들에 의해 구현된 서비스 집합으로 구성된다. 각 서비스는 서버 프로세스 집합이 하나의 그룹형태로 구현된다. 그리고, 각 서버 프로세스는 자신의 로컬 상태를 가지고 있으며, 클라이언트 호출을 통해 그 상태를 갱신한다. 프로세스들은 고장이 발생하면 자신의 휘발성 저장소에 있는 로컬 상태를 잃어버리고, 수행을 멈추는 파손-고장 모델을 따른다고 가정한다[14]. 또한, 프로세스들은 선입선출 순서로 메시지를 전달하고, 신뢰성 있고 분할되지 않는 네트워크에 연결되어있다. 각 프로세스의 수행은 부분 결정적이다[17]. 즉, 이 모델에서는 한 프로세스 수행의 어떠한 시점에서도 그 프로세스의 상태구간(state interval)은 하나의 비결정적 이벤트에 의해 결정되는데, 본 연구에서 비결정적 이벤트는 하나의 프로세스가 메시지를 수신하는 경우 그 메시지를 해당 애플리케이션에게 전달해주는 이벤트이다. 프로세스 p의 k번째 상태구간 $si_p^k(k>0)$는 p의 k번째 수신한 메시지 m의 전달 이벤트 $dev_p^k(m)$에 의해 시작된다. 따라서, p의 초기 상태




(그림 1) 가정하는 시스템 모델의 예

(initial state) $s_p^0$와 그 프로세스의 수행시 발생해왔던 i개의 비결정적 이벤트들이 주어진다면, 그 프로세스의 상태 $s_p^i$는 $s_p^0$로부터 전달된 순서대로 해당 이벤트들을 처리함으로써 유일하게 결정된다.

일렬화가능성(linearizability)은 대부분의 수동적 중복 기법들이 보장해야 하는 중복 서비스를 위한 일관성 조건이다[10, 18]. 즉, 이 조건은 인터리빙된 오퍼레이션들은 해당 객체의 올바른 복사본의 명세를 만족해야하고, 요구된 모든 오퍼레이션들의 수행순서는 발생한 실시간과 일관적이어야 한다는 것이다[11].

그룹 s에서 서버 프로세스 p는 유니캐스트와 멀티캐스트 통신 프리미티브를 사용하여 그룹 s에 속한 다른 서버 프로세스들과 통신할 수 있다. 시스템에서 사용되는 멀티캐스트 통신 프리미티브는 일반적으로 수동적 중복기법의 정당성을 보장하기 위해 사용되는 뷰 동기적 멀티캐스트(View Synchronous Multicast-VSCAST)[18]이다. VSCAST는 그룹 s의 컨텍스트내에서 정의되고, 그룹 s의 일련의 뷰(view)들 $v_0(s),v_1(s),...,v_i(s),...$의 개념에 기반한다. 뷰 $v_i(s)(i\geq 0)$는 어떤 시간 t에 정상적이라고 인식되는 그룹 s의 i번째 멤버들의 집합이다. 뷰 $v_i(s)$에 속한 서버 프로세스 q가 고장났다고 여겨지거나 새로운 서버 프로세스 q가 그룹 s에 가입하려고 한다면, 새로운 뷰 $v_{i+1}(s)$가 설정된다. s에 속한 어떤 프로세스 q가 메시지 m를 $v_i(s)$에게 송신한다면, VSCAST는 다음과 같은 속성을 보장한다: 뷰 $v_i(s)$에 속한 프로세스 q가 메시지 m을 $v_i(s)$에서 전달하고 뷰 $v_{i+1}(s)$를 설정했다면, $v_i(s)$의 멤버이면서 뷰 $v_{i+1}(s)$를 설정한 모든 프로세스 r은 m을 전달한 후에 $v_{i+1}(s)$를 설정한다.

## 3. 제안하는 수동적 중복 알고리즘

### 3.1 기존 수동적 중복 알고리즘의 문제점

기존의 수동적 중복 알고리즘(Traditional Passive Replication Algorithm-TPRA)[2, 5, 8]에서는 한 클라이언트가 요구 메시지를 주 서버에게 송신하는 경우에, 정상수행시 일렬화가능성을 보장하기 위해 다음의 4단계의 오퍼레이션을 수행한다:

**〈단계-1〉** 주 서버가 클라이언트로부터 요구 메시지를 수신한다.

**〈단계-2〉** 주 서버는 그 메시지를 처리한 후, (resp, rsn, update-state)를 생성시킨다. 여기서 resp는 그 메시지의 응답이고, rsn은 메시지의 처리일련번호이고, update-state는 그 메시지를 처리함으로써 갱신된 주 서버의 상태이다.

**〈단계-3〉** 주 서버는 모든 보조 서버들에게 메시지 (resp, rsn, update-state, cid, reqid)를 VSCAST(View-Synchronous multiCAST) 프리미티브를 사용하여 전달한다. 여기서, cid는 요구 메시지를 전달한 클라이언트의 식별자이고, reqid는 그 메시지의 송신 일련번호이다. 각 보조 서버가 그 메시지를 수신하였을 때, update-state를 사용하여 자신의 상태를 갱신하고, (resp, cid, reqid)를 자신의 버퍼에 저장한 후, 확인(acknowledgment) 메시지를 주 서버에게 전달한다.

**〈단계-4〉** 주 서버는 모든 살아있는 보조 서버로부터 확인 메시지를 수신한 후, resp를 해당 클라이언트에게 송신한다.

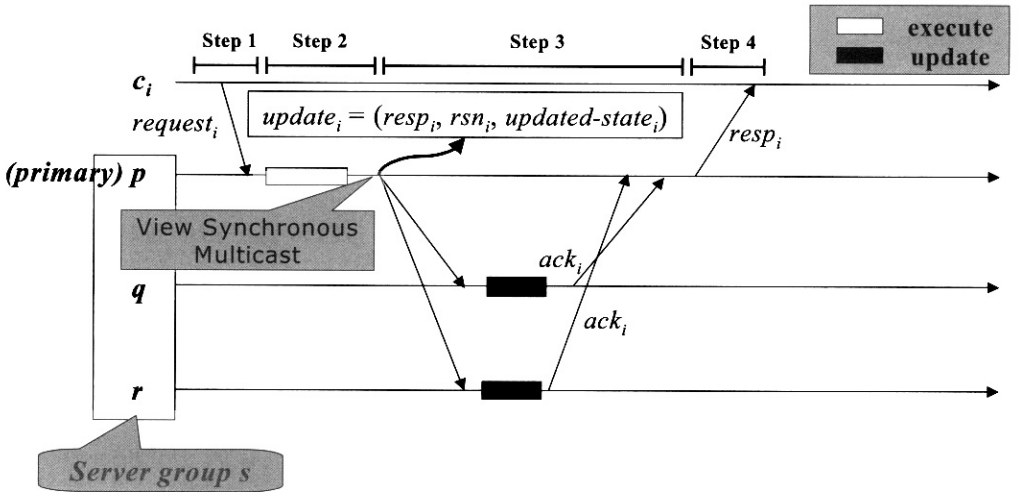
그러나 주 서버의 고장 시에도 일관성 조건을 보장하기 위해서는 이를 처리하는 부가적인 단계들이 필요하다. 따라서, 앞에서 언급한 TPRA의 수행단계들이 실행되는 과정을 보여주는 (그림 2)의 예를 통해 주 서버가 고장나는 각 경우에 대해 처리과정을 기술하고자 한다. 여기서, $request_i$를 클라이언트 $c_i$로부터 전달 받은 요구 메시지라 하고, $resp_i$를 $request_i$의 응답 메시지라 하자. 또한, $update_i$를 주 서버가 $request_i$를 처리함으로써 갱신된 자신의 상태를 포함하는 갱신 메시지라 하고, $ack_i$를 $update_i$의 확인 메시지라 한다.

**〈경우-1〉 주 서버 p가 단계 3을 수행하기 전에 고장난 경우**

이 경우에는 모든 보조 서버들 중 새로운 주 서버 q가 선출되고, 클라이언트 $c_i$는 p의 고장을 감지하게 된다. 따라서, 새로운 주 서버 q의 식별자를 알고 난 후, q에게 그 요구 메시지 $request_i$를 재전송한다. q는 그 요구 메시지를 수신한 후, 단계 1부터 4까지 수행하기만 하면 된다.

**〈경우-2〉 주 서버 p가 updatei를 모든 보조 서버들에게 송신하였지만, 클라이언트 ci에게 아직 respi를 전달하지 못한 상태에서 고장난 경우**

모든 보조 서버들에 의해 새로운 주 서버 q가 선출되고, 클라이언트 $c_i$는 p의 고장을 감지하고, 새로운 주 서버 q의 식별자를 알게 된다. 일관성 조건을 보장하기 위해, 모든 보조 서버들이 $update_i$를 수신하던지, 모두 수신하지 못하던지 해야 한다. 이는 VSCAST의 특성에 의해 반드시 보장된다. 만약, 모두 $update_i$를 수신하지 못한 경우는 <경우-1>과 동일하게 수행한다. 그렇지 않다면, 단계 3에서 각 서버에서의 $(resp_i, i, reqid)$를 사용하여 exactly-once semantics[16]를 보장할 수 있다. 즉, q에게 그 요구 메시지 $request_i$가 재전송된다면 q는 그 요구 메시지를 다시 처리하지 않고, 단지 자신의 버퍼에 있는 $resp_i$를 해당 클라이언트 $c_i$에게 전달한다.




(그림 2) 기존 수동적 중복 알고리즘의 수행 예

**〈경우-3〉 주 서버 p가 클라이언트 ci에게 respi에게 전달한 후 고장난 경우**

<경우-1>과 같이 새로운 주 서버 q가 선출되고, 클라이언트 $c_i$는 p의 고장을 감지하고, 새로운 주 서버 q의 식별자를 알게 된다.

(그림 2)에서 본 바와 같이, TPRA에서는 일렬화가능성의 순서화 조건을 만족시키기 위해 클라이언트들로부터의 모든 요구 메시지들은 주 서버 p로 전송된다. 따라서, 요구 메시지들에 의해 발생되는 오퍼레이션들은 메시지의 처리일련번호에 기반하여 전체적으로 순서화된다. 그러나, 매우 많은 클라이언트들이 요구 메시지들을 동시적으로 주 서버 p에게 전달한다면, TPRA에서는 p가 그 모든 메시지들을 순서대로 처리하고 다른 보조 서버들에 대한 모든 조정역할을 수행해야한다. 따라서, 이러한 수행형태는 주 서버 p에 대한 과부하를 발생시키고, 클라이언트 평균 응답시간을 매우 증가시킬 수 있다.

### 3.2 부분적 분산형 수동적 중복 알고리즘

본 절에서는 일렬화가능성을 보장하면서 기존의 중앙집중형 수동적 중복 알고리즘의 문제점을 해결하는 결정적 서버를 위한 부분적 분산형 중복 알고리즘(Partially Decentralized Passive Replication Algorithm-PDPRA)을 제안한다. 알고리즘 PDPRA는 주 서버와 보조 서버들이 능동적 중복 기법에서와 같이 동일한 순서로 요구 메시지들을 동시에 처리한다 하더라도, 결정적 서버의 경우에는 일관성을 유지할 수 있다라는 기본적인 아이디어를 기반하여 제안되었다. 따라서, 주 서버가 모든 클라이언트 요구의 처리 순서를 결정기만하면, 이후에 그러한 요구들에 대한 처리와 이에 따른 다른 서버와의 조정역할을 기존 수동적 중복 알고리즘처럼 주 서버만이 모두 수행할 필요가 없다는 것이다. 이러한 기본 개념에 기반하여, 제안하는 알고리즘 PDPRA는 다음과 같은 특성을 가지도록 설계되었다:

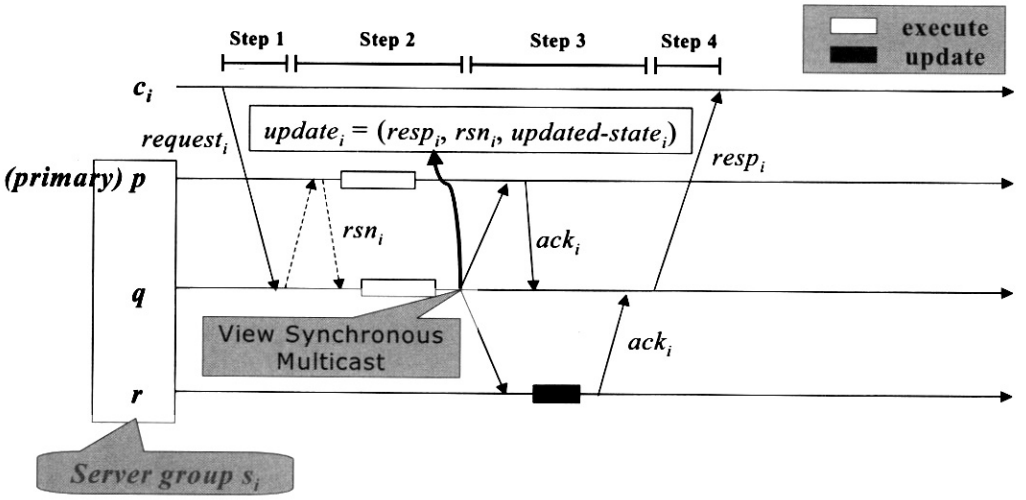
- 모든 클라이언트는 각 서버스 요구 메시지를 주 서버 혹은 보조서버에게 전달할 수 있다.
- 주 서버만이 모든 클라이언트 요구 메시지의 처리순서를 결정한다.
- 임의의 보조 서버가 클라이언트 요구 메시지를 수신한 경우, 주 서버로부터 그 요구 메시지의 처리일련번호를 얻은 후, 그 보조 서버가 직접 해당 요구 메시지를 처리하고 다른 중복 서버들과의 조정역할을 수행한다.

이러한 요구조건들을 만족시키기 위해 제안된 알고리즘은 다음과 같이 4단계 과정들을 수행하게 된다.

**〈단계-1〉** 보조 서버 q가 클라이언트로부터 요구 메시지를 수신한다.

**〈단계-2〉** q는 수신한 요구 메시지의 처리일련번호 rsn을 주 서버에게 요구한다. 이 경우, 주 서버는 그 요구 메시지의 rsn을 결정한 후 q에게 알려준다. 이후에 주 서버는 해당 요구(request)를 처리한 후 생성된 (resp, rsn, update-state, cid, reqid)를 자신의 버퍼에 저장한다. 한편, q는 수신한 요구의 rsn을 주 서버로부터 얻은 후, 그 요구를 처리한 후, (resp, rsn, update-state)를 발생시킨다.

**〈단계-3〉** 보조 서버 q는 모든 다른 서버들에게(resp, rsn, update-state, cid, reqid)를 포함하는 갱신 메시지를 VSCAST(View-Synchronous multiCAST) 프리미티브를 사용하여 전달한다. 주 서버를 제외한 모든 보조 서버가 그 갱신 메시지를 수신한 경우에는, update-state를 사용하여 자신의 상태를 갱신하고, (resp, cid, reqid)를 자신의 버퍼에 저장한 후, 확인 메시지를 보조 서버 q에게 전달한다. 주 서버가 갱신 메시지를 수신한 경우에는 단지 해당 (resp, rsn, update-state, cid, reqid)를 삭제하고, (resp, cid, reqid)를 자신




(그림 3) 보조 서버 q에게 클라이언트 요구가 전달될 때 제안된 알고리즘 PDPRA의 수행 예

의 버퍼에 저장한 후, 확인 메시지를 보조 서버 q에게 전달한다.

〈단계-4〉 모든 살아있는 보조 서버로부터 확인 메시지를 수신한 후, 보조 서버 q는 해당 resp를 사용자에게 전달한다.

(그림 3)은 한 보조서버 q가 클라이언트 $c_i$로부터 요구 메시지를 수신하는 경우에 제안한 알고리즘 PDPRA이 수행하는 과정을 보여준다. 첫 번째 단계에서 클라이언트 $c_i$는 한 요구 메시지 $request_i$를 보조서버 q에게 송신한다. 두 번째 단계에서 q는 주 서버 p에게 $request_i$의 처리순서를 결정하여 알려주도록 요청한다. 이때, p는 처리순서를 결정하고 즉시 이를 q에게 알려준다. 그리고, p는 $request_i$를 처리하고, 요구에 대한 응답, 요구의 처리일련번호와 자신의 갱신된 상태를 ($resp_i$, $rsn_i$, update-$state_i$) 형태로 자신의 버퍼에 저장한다. 주 서버 p로부터 $request_i$의 처리일련번호를 전달받은 q는 $request_i$를 처리한다. 세 번째 단계에서 q는 ($resp_i$, $rsn_i$, update-$state_i$)를 자신의 버퍼에 저장하고, 이를 포함한 갱신 메시지 $update_i$를 다른 모든 중복 서버에게 송신한다. 이 경우, r이 갱신 메시지 $update_i$에 포함된 update-$state_i$를 이용하여 자신의 $rsn_i$번째 상태 갱신하고, 자신의 버퍼에 $resp_i$를 저장한 후, q에게 확인 메시지 $ack_i$를 송신한다. 그러나, 갱신 메시지 $update_i$를 수신한 주 서버 p는 단지 자신의 버퍼로부터 ($resp_i$, $rsn_i$, update-$state_i$)를 삭제하고, 다시 버퍼에 $resp_i$를 저장한 후, q에게 확인 메시지 $ack_i$를 송신한다. q는 다른 중복 서버 p와 r로부터 모든 확인 메시지를 수신한 후, 응답 메시지 $resp_i$를 4단계에서 클라이언트 $c_i$에게 송신한다.

(그림 3)의 예에서 본 바와 같이, 제안한 알고리즘 PDPRA는 각 보조서버가 주 서버로부터 요구 처리일련번호를 얻은 후에 해당 클라이언트 요구에 대한 처리과정의 책임자 역할을 할 수 있도록 함으로써 주 서버와 네트워크에 발생하는 부하를 줄여줄 수 있다. 따라서, PDPRA는 효과적인 부하균등 기법과 결합된다면 기존 알고리즘에 비해 평균응답시간을 매우 줄이면서 보다 높은 확장성을 제공할 수 있다.

주 서버가 고장나는 경우에는 제안한 알고리즘 PDPRA이 3.1절에서 언급했던 기존 수동적 중복 알고리즘 TPRA의 주 서버가 고장나는 각 경우에 대한 처리과정을 동일하게 수행하면 된다. 그러나 보조 서버 q의 고장 시에도 일관성 조건을 보장하기 위해서는 기존 알고리즘과 달리 다음과 같은 네 가지 경우들에 대한 처리과정을 수행해야 한다.

**〈경우-1〉 보조 서버 q가 단계 2를 수행하기 전에 고장난 경우**

이 경우에는 단지 보조 서버 q만 해당 중복 서버그룹에서 제외시키면 된다. 이후에 클라이언트 $c_i$는 q의 고장을 감지하게 된다.

**〈경우-2〉 보조 서버 q가 주 서버에게 requesti를 송신했지만, 단계 3을 수행하기 전에 고장난 경우**

〈경우-2.1〉 주 서버가 고장난 경우

q와 주 서버를 제외한 모든 보조 서버들에 의해 새로운 주 서버 r이 선출되고, 클라이언트 $c_i$는 q의 고장을 감지한다.

〈경우-2.2〉 주 서버가 살아있는 경우

주 서버가 q의 고장을 감지했을 때, 자신의 버퍼로부터 현재 처리완료되지 않은 클라이언트 요구에 대한 (resp, rsn, update-state, cid, reqid)를 찾아서 VSCAST를 사용하여 다른 모든 보조 서버들에게 전달한다. 만약, 보조 서버 r에게 클라이언트 요구 $request_i$이 재전송된다면 r는 그 요구를 처리하지 않고, 단지 자신의 버퍼에 있는 $resp_i$를 해당 사용자에게 전달한다.

**〈경우-3〉 보조 서버 q가 updatei를 모든 다른 서버들에게 송신하였지만, 클라이언트 ci에게 아직 respi를 전달하지 못한 상태에서 고장난 경우**

이 경우에는 $update_i$가 VSCAST를 사용하여 전송하였기 때문에, 시스템의 일관성 조건은 만족된다. 따라서, 단지 보조 서버 q만 해당 서버그룹에서 제외시키면 된다.

**〈경우-4〉 보조 서버 q가 클라이언트 ci에게 respi에게 전달한 후 고장난 경우**

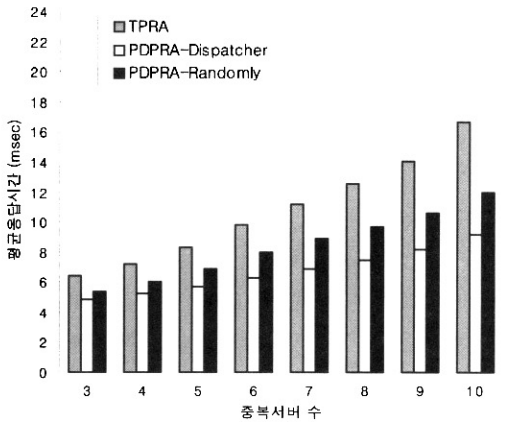
이 경우 단지 보조 서버 q만 해당 서버그룹에서 제외시키면 되고, 클라이언트 $c_i$는 q의 고장을 감지한다.

## 4. 성능 평가

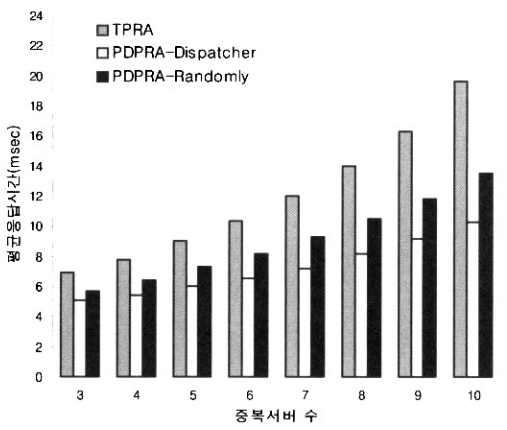
이 절에서는 PARSEC 이산-이벤트 시뮬레이션 언어[3]를 사용하여 본 논문에서 제안한 수동적 중복 알고리즘(PDPRA)과 기존의 수동적 중복 알고리즘(TPRA)의 성능을 비교하기 위한 시뮬레이션을 수행하고자 한다. 이 시뮬레이션에서의 성능평가 기준은 한 클라이언트 요구에 대한 평균 응답시간이다. 시뮬레이션되는 시스템의 구성은 다음과 같다. 모델링된 네트워크는 이더넷과 같은 다중접근 근거리통신망(multi-access LAN)이다. 시스템을 구성하는 노드들은 그 네트워크 상에서 동일하고 균일하게 분산된다. 모든 프로세스는 UDP/IP 프로토콜을 사용하여 상호적으로 통신한다. 임의의 두 노드간 메시지전송 지연시간은 다음과 같이 세 가지로 구성된다: 1) $S_{time}$, 송신노드에서 송신 오퍼레이션을 수행하는데 걸리는 시간, 2) $T_{time}$, 송신노드의 네트워크 인터페이스로부터 수신노드의 네트워크 인터페이스까지 메시지를 전송하는데 걸리는 시간, 3) $R_{time}$, 수신노드에서 수신 오퍼레이션을 수행하는데 걸리는 시간. 이 시뮬레이션에서는 이러한 세 가지 시간들을 각각 0.269 ms, 0.120 ms와 0.292 ms로 설정한다. 한 프로세스가 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[12]. 그리고, 시뮬레이션 환경에서 각 클라이언트 요구를 처리하는데 걸리는 시간은 1ms이고 서버 상태를 갱신하는데 필요한 시간은 80μs로 설정한다. 특히, 단위시간당 중복 서비스로 전송되는 클라이언트 요구간 평균간격 $MSI_{time}$은 지수 분포를 따른다.

본 시뮬레이션에서 제안한 알고리즘 PDPRA가 기존의 부하균등 기법[4,6]과 결합한 경우, 그 효율성이 매우 높아짐을 실험을 통해 보여주기 위해, 부하균등 기법과 결합되지 않은 PDPRA(PDPRA-Randomly)와 디스패처 기반 부하균등 기법과 결합된 PDPRA(PDPRA-Dispatcher)로 나누어 각각 그 성능을 살펴보고자 한다.

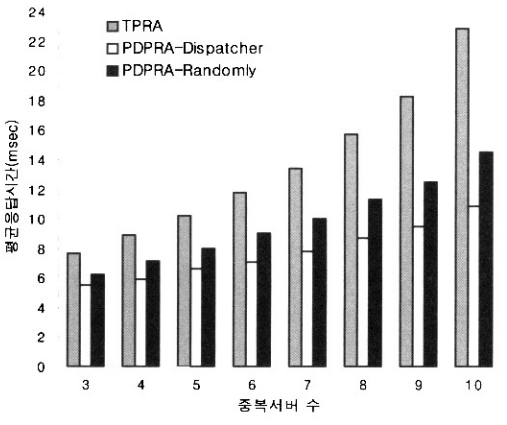
(그림 4)부터 (그림 6)까지 한 네트워크 내에서 중복 서버들의 수를 일정한 범위로 변화시켰을 때 세 개의 수동적 중복 알고리즘들 PDPRA-Randomly, PDPRA-Dispatcher와 TPRA의 클라이언트 요구 당 평균 응답시간을 보여준다. 세 그림들에서 $MSI_{time}$을 각각 30ms, 20ms, 10ms로 각각 설정하였다. (그림 4)에서 세 알고리즘의 중복서버 수가 증가함에 따라, 그들의 응답시간 또한 증가함을 알 수 있다. 이는 한 서비스의 중복정도가 커짐에 따라, 세 알고리즘들은 해당 중복서버 그룹의 더 높은 동기화 비용을 발생시키기 때문이다. 그러나 PDPRA-Randomly의 응답시간은 TPRA보다 낮고, PDPRA-Dispatcher의 응답시간은 PDPRA-Randomly보다 낮은 것을 알 수 있다. 특히, 서비스의 중복서버 수가 커질수록, 세 알고리즘들의 응답시간 증가율간의 차이가 점점 더 커지는 것을 알 수 있다. (그림 4)에서는 PDPRA-Randomly와 PDPRA-Dispatcher는 TPRA에 비해 평균응답시간을 각각 16.5%~28%와 24.9%~45.1%만큼 줄인다. 또한, (그림 5)와




(그림 4) $MSI_{time}$ = 30msec인 경우




(그림 5) $MSI_{time}$ = 20msec인 경우




(그림 6) $MSI_{time}$ = 10msec인 경우

(그림 6)에서도 PDPRA-Randomly와 PDPRA-Dispatcher의 응답시간 감소율이 각각 17.5%~36.4%와 26.1%~52.3%이다. 따라서, 이 세 그림들에서 $MSI_{time}$이 커짐에 따라, 세 알고리즘들의 응답시간 증가율간의 차이가 보다 더 커짐을 알 수 있다. 이러한 PDPRA의 응답시간감소는 PDPRA가 중복서버 그룹의 주 서버에 대한 중앙집중형 제약성을 매우 약화시킴으로써 얻어지는 것이다. 특히, PDPRA-Dispatcher와 같이 PDPRA가 기존의 부하균등 기법과 결합하는 경우 PDPRA-Randomly에 비해 클라이언트 요구처리에 대한 부하가 모든 서버노드들에게 균등하게 분산됨으로써 보다 높은 평균응답시간을 감소시킬 수 있다는 것을 알 수 있다.

## 5. 관련 연구

조정자-코홀트(coordinator-cohort) 중복 알고리즘[2]은 ISIS 시스템 환경 [1]에서 구현된 하나의 변형된 수동적 중복 알고리즘이다. 통신 패턴의 관점에서 수동적 중복과 매우 유사하나 이 알고리즘에서는 능동적 중복 기법에서와 같이 모든 중복자가 클라이언트 요구를 수신할 수 있다. 이러한 특성은 클라이언트에게 주 중복자 즉, 조정자의 고장에 대한 투명성을 제공한다. 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.

준 능동적 중복(semi-active replication) 알고리즘[13]은 능동적 중복기법의 단점 중에 하나인 중복 서버에서 수행되는 모든 오퍼레이션들이 결정적(deterministic)이어야 한다는 제한사항을 극복한 기법이다. 이러한 목적은 모든 서버가 클라이언트로부터 요구받은 오퍼레이션들의 순서를 정하고 독립적으로 처리한 후, 수동적 중복기법과 유사하게 중복서버들 중 주 서버가 자신의 상태를 다른 서버들에게 전달하게 함으로써 달성된다. 그러나 이 알고리즘은 능동적 중복기법과 같이 정상수행시 높은 자원의 소비를 발생시키고, 수동적 중복기법과 같이 주 서버 고장 시 높은 재구성 비용을 발생시킨다.

준 수동적 중복(semi-passive replication) 알고리즘[8]은 주 서버의 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 중복서버의 고장을 짐작하기(suspecting) 위해 빠른 타임아웃 값을 사용하면서, 만약 고장에 대한 짐작이 잘못되었다 하더라도 높지 않은 재구성 비용을 발생시킨다. 그러나, 이 기법은 각 클라이언트가 서비스 요구 시 능동적 중복기법과 같이 모든 중복 서버들에게 브로트캐스트(broadcast) 혹은 멀티캐스트(multicast)를 해야 한다는 단점이 있다.

앞에서 언급한 모든 알고리즘들은 주 서버만이 모든 서비스 요구들을 처리하고 다른 서버들과의 조정역할을 수행해야하는 반면, 본 논문에서 제안된 알고리즘은 클라이언트 요구를 수신한 임의의 서버가 주 서버로부터 전달 일련번호만을 얻은 후 직접 처리하게 함으로써 결정적 중복 서버의 확장성을 향상시킨다.

## 6. 결 론

본 논문에서는 결정적 서비스가 비결정적 서비스보다 시스템 일관성을 보장하기 위한 제약성이 약하다는 관점에 기반하여 기존의 수동적 중복 알고리즘에 비해 확장적인 결정적 중복 서버를 위한 수동적 중복 알고리즘 PDPRA을 제안한다. 이 알고리즘은 반드시 주 서버가 아니라 서비스 요구 메시지를 수신한 임의의 서버가 그 메시지의 전달 일련번호를 주 서버로부터 얻은 후, 자신이 직접 그 요구를 처리하고, 다른 서버들과의 조정역할을 할 수 있도록 한다. 이러한 특성 때문에 제안된 알고리즘은 기존 알고리즘의 주 서버에 대한 중앙집중형 제약성을 피할 수 있다. 본 논문의 시뮬레이션 결과로부터 한 서비스의 중복서버 수와 클라이언트 요구 수가 증가하더라도 제안된 알고리즘 PDPRA이 기존 알고리즘에 비해 상당히 낮은 클라이언트 요구 응답시간을 발생시키고, PDPRA이 부하균등 기법과 결합된다면 보다 높은 확장성을 제공할 수 있다는 것을 알 수 있다.

## 참 고 문 헌

[1] K. P. Birman and T. A. Joseph, "Reliable communication in the presence of failures," ACM Transactions on Computer Systems, Vol.5, No.1, pp.47-76, 1987.

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

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

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

[5] N. Budhiraja, K. Marzullo, F. Schneider and S. Toueg, "The primary-backup approach, Distributed Systems," ch.8, pp.199-216, 2nd Ed., Addison-Wesley, 1993.

[6] V. Cardellini, M. Colajanni and P. Yu, "Dynamic load balancing on Web- server systems," IEEE Internet Computing, Vol.3, pp.28-39, 1999.

[7] T. D. Chandra and S. Toueg, "Unreliable failure detectors for reliable distributed systems," Journal of ACM, Vol.43,

No.2, pp.225-267, 1996.

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

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

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

[11] M. Herlihy and J. Wing, "Linearizability: a correctness condition for concurrent objects," ACM Transactions on Progr. Languages and Syst., Vol.12, No.3, pp.463-492, 1990.

[12] M. Malcom and W. Zhao, "Hard real time communication in multiple-access networks," Real-Time Systems, Vol.8, pp.35-77, 1995.

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

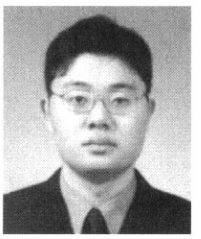
[14] R. D. Schlichting and F. B. Schneider, "Fail-stop processors: an approach to designing fault-tolerant distributed computing systems," ACM Transactions on Computer Systems, Vol.1, pp.222-238, 1985.

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

[16] A. Spector, "Performing remote operations efficiently on local computer network," Communications of the ACM, Vol.25, No.4, pp.246-260, 1982.

[17] R. B. Strom and S. Yemeni, "Optimistic recovery in distributed systems," ACM Transactions on Computer Systems, Vol.3, pp.204-226, 1985.

[18] M. F. Wiesmann, A. Schiper, B. Kemme and G. Alonso, "Understanding Replication in Databases and Distributed Systems," In Proc. of the 21st International Conference on Distributed Computing Systems, pp.464-474, 2000.



**안 진 호**

e-mail : jhahn@kyonggi.ac.kr

1997년 고려대학교 컴퓨터학과(이학학사)

1999년 고려대학교 컴퓨터학과(이학석사)

2003년 고려대학교 컴퓨터학과(이학박사)

2003년~현재 경기대학교 정보과학부 전자계산학과 조교수

관심분야 : 결함포용 분산시스템, 이동에이전트 시스템, 그룹통신, p2p 컴퓨팅