게임 서버간 부하의 균일성에 기반한 부하 분산기 설계 및 구현

정회원 엄 남 경*, 문 형 진*, 이 상 호**

Design and Implementation of A Load Balancer Based on Load Equality between Game Servers

Nam-Kyoung Um*, Hyung-Jin Moon*, Sang-Ho Lee** ***Regular Members***

요 약

네트워크상에서 모바일 게임에 접속하는 사용자들은 서버 상에 발생하는 부하 없이 끊임없는 실시간 게임을 하기를 원한다. 그러나 기존의 서버부하분산 방식은 게임 사용자의 특성과 서버들 간의 공평한 분배율을 고려하지 않아, 효율적인 부하 분배 처리가 용이하지 않다. 즉, 평균 부하율이 같은 두가지 방식이 있다고 하더라도 각각의 서버간의 부하율이 균일하지 않다면 추후 오동작 오류를 일으킬 소지가 있다. 따라서 본 논문은 해당 접속 요청을 부하분산 알고리즘에 의해 적절한 게임 서버에 배분하되, 온라인 게임서버의 부하 분산을 위한 균일한 분산을 기반으로 하는 부하 분산기를 설계하고 개발하고자 한다. 이를 통해 일정 기간의 부하 관련 정보를 관리자에게 제공하여 능동적인 게임 서버의 증설 등에 대응할 수 있다.

Key Words : Online game, Game server, Load balancer

ABSTRACT

All of users who connect to mobile game want to play seamless real-time game without any loads on game servers. However, as for existing methods, it is not available to effectively distribute server loads. Therefore, in this paper, we design and implement an uniform variance-based load balancer for distributing loads of game servers for wireless online games by suitable load balancing methods. With this methods, we can deal with dynamically increasing game servers as the information about load balancing of specific period is applied to the administrator.

Ⅰ. 서 론

하드웨어의 눈부신 발전에 힘입어 고성능의 서버를 게임에 활용하면서, 게임 서버 운영에 이점이 되고 있지만, 접속하는 게임유저가 기하급수적으로 늘어나면서 서버 쪽에 발생되는 부하 또한 그 수치가 급격하게 증가하므로 게임유저의 서비스 만족도를 일정 수준으로 유지하기 위해서는 게임 서버의 효율적인 부하 분산 방안이 필요하다. 본 논문에서는 온라인 게임을 지원하기 위한 균일한 분산에 기반한 서버 부하 분산 기법을 제안하고, 그에 따라 설계 및 구현하고자 한다. 운영하는 전체 서버의 평균 부하율이 우수하더라도 각 서버간의 부하율의 균일한 분포가 이루어지지 않으면, 추후 각 서버가 가진 가

※ 본 연구는 유비쿼터스 바이오정보기술연구센터 과제(모바일 콘텐츠 서비스를 위한 통합 관리 시스템 개발)지원으로 수행되었습니다.
* 충북대학교 전기전자컴퓨터학부 네트워크보안연구실 (family@netsec.cbnu.ac.kr, nanhelper@hanmail.net)
** 충북대학교 전기전자컴퓨터학부 정교수 (shlee@cbnu.ac.kr)
논문번호 : KICS2006-10-464, 접수일자 : 2006년 10월 28일, 최종게재논문통보일자 : 2006년 3월 15일

용성에 대해 문제가 발생할 수 있기 때문이다. 그러나 기존의 방식들은 전체적인 평균부하율의 효율성을 기반으로 하므로, 각 서버가 가지는 균등한 분배 효율성은 제고하지 않는 경우가 있다. 특히, 일반적으로 처음부터 게임 서버에 맞는 서버부하 방식을 고려하지 않고, 게임 서버를 두어 하나의 지역에 많은 게임유저가 몰리게 된다면 시스템의 한계에 의해 서버의 처리 속도가 현저하게 느려지거나 심할 경우 운영체제가 다운될 수 있으므로 미리 대처해야한다.

논문의 구성은 다음과 같다. 2장에서는 기존 방식을 기술하고, 3장에서는 기존 방식들의 비교평가를 통한 문제점을 제기하며, 4장과 5장은 본 논문에서 제시하는 방식을 설계 및 실험하며, 6장과 7장에서는 각각 구현된 결과에 대해 결론을 맺도록 한다.

Ⅱ. 관련 연구

이 장에서는 온라인 게임 서버의 특징 및 관련된 부하 관리 기법에 대해 살펴본다.

2.1 온라인 게임 서버의 특징

온라인 게임 서버를 위한 특징은 다음과 같다[1]-[4].

1) 온라인 게임 서버는 온라인 게임을 위해 지원되어야 하는 로그인서버, 데이터베이스서버, 대기실 기능을 위한 로비서버, 실제 게임 진행을 위한 게임서버 등의 하나의 기능 서버의 역할을 하게 되며, 각 서버 간에는 멀티쓰레딩(Multi-threading)이 필요한 긴밀한 상호작용을 가진다. 그림 1은 온라인 게임을 위한 서버 구조를 보이며, 그림에서 보여지는 라인 위의 숫자는 메시지의 전달 순서를 보인다.

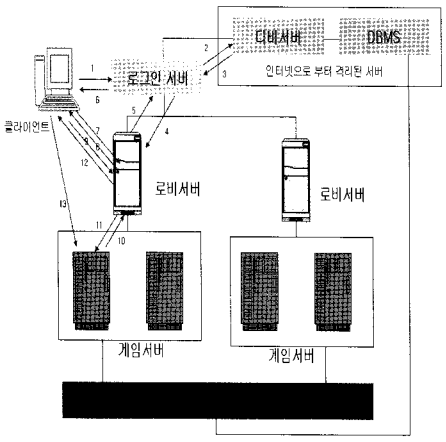


그림 1. 전체 시스템 구조

2) 게임서버에 접속 단계가 여러 단계로 부하가 자동적으로 분산되며, 그 예는 로그인 서버에서 로비 서버로 접속되고, 로비서버에서 게임서버로 가는 단계를 거친다.
3) 게임서버에 이미 접속되어 있는 접속자의 수가 게임을 이용하고자 하는 게임유저수와 큰 관련을 가진다. 즉, 네트워크 온라인 게임의 경우, 게임서버의 접속자 수가 적으면 사용자가 접속하지 않는다. 따라서 접속자 수에 대한 임계치를 최대뿐만 아니라 최소도 설정해야 한다.
4) 게임서버에서 동기화 매커니즘이 필요하다. 클라이언트들에 의한 명령이 처리되는 과정에서 동기화가 필요하다. 즉, 게임의 실시간성, 동기화 공간의 크기, 동기화 수준 정도를 파악해야 한다.
5) 게임서버는 프로세싱이 많은 실시간 시스템이기 때문에 DBMS에 덜 의존적이여야 한다.
6) 게임과 연계해야 하는 서비스를 지원해야 하며, 메신저, 웹, 커뮤니티가 지원되어야 한다.
7) 특히, 모바일 게임에서는 프로세싱이 주로 서버에서 집중되어야 하므로, 모바일 게임의 특성에 따르는 낮은 저장 공간, 낮은 계산속도, 낮은 전송속도 등도 고려되어야 한다.

2.2 일반적인 서버부하분산 방식

일반적으로 쓰이는 부하 분산을 위한 분배 알고리즘을 다음을 기본으로 한다[5].

1) 무작위 : 무작위로 서버에 요청을 할당한다
2) 최소부하 : 현재 부하가 가장 적은 서버에 요청을 할당한다. 세부적으로는 서버의 성능 용량에 기반해서 요청을 할당하는 성능 기반의 방식과, 서버들의 총 부하 수용량에 따라 요청을 할당하는 부하 기반의 방식이 있다.
3) 라운드로빈 : 서버 목록의 첫번째 서버로부터 목록의 끝까지 연속적으로 다음 서버를 선택한다. 또한 라운드로빈과 비슷하나 어떤 서버들은 목록에 여러번 나타나는 가중 라운드로빈 방식도 있다.
4) 동적 할당 : 어플리케이션의 알고리즘에 따라 동적으로 할당하는 방식이 있으며, 포트번호에 따라 요청을 할당하는 포트번호 기반의 방식과 URL이나 쿠키 등의 HTTP 헤더 안의 값에 따라 요청을 할당하는 HTTP 헤더 중심의 방식도 있다.

일반적으로 게임서버에서는 라운드로빈 기반 방식을 사용하거나, CPU 이용율 기반의 방식 또는 게임 서버에 이미 사용하고 있는 사용자수를 기반으로 하여 사용자의 입력이 들어오면, 미리 정렬된 값에 따라 가장 최소의 값을 가진 서버의 값을 클라이언트에게 전달하여 서버와 클라이언트간 연결을 하는 방식을 취한다.

2.3 서버 클러스터링 방식

서버 클러스터링 방식[5]이란 분산·병렬 컴퓨팅에서 시작된 개념으로 하나의 작업을 각각의 컴퓨터에서 분산 처리한후, 결과를 다시 모으는 방식으로 여러 개의 독립된 컴퓨터를 묶어서 하나의 시스템처럼 동작하게 하는 것이다. 서버의 가용성과 확장성에 아주 용이하다. 현재 국내 온라인 게임을 살펴보면 월드에는 내부에서 정하는 게임유저 제한치가 있으며, 이 제한치가 넘어가면 유저를 수용하기보다 다른 서버군을 오픈하여 유저에게 이동을 권하는 경우를 많이 사용한다. 서버군에 따라 각각 물리적으로 분리된 데이터베이스를 사용하는데, 이는 기존의 캐릭터를 그대로 옮겨놓거나 새로 만들어야 하는 문제가 생긴다. 대부분의 온라인 게임은 형평성을 위해 특별한 이유가 없으면 이전에 사용하던 캐릭터를 다른 서버군으로 옮기지 못하게 한다.

Ⅲ. 문제 제기

참고문헌[6]에서는, 표 1에서와 같이 기존에 활용되는 주요 서버 부하 분산 기법들인 라운드로빈 방식과 CPU 활용률 기반의 방식, 클러스터링 방식을 비교하였다. 항목은 일반적인 부하 분산과 관련해서 언급되는 부하분배시의 속도측면, 주어진 게임서버들이 균일한 분산도를 가지고 활용되고 있는지에 대한 서버가용성, 마지막으로 게임 서버를 더 확장해도 전체 시스템 상에 문제가 생길 가능성을 보는 확장성 등 세 가지 항목을 각각 "상", "중", "하"의 평가기준으로 비교하고 있다.

표 1. 기존 방식 비교

항목	라운드로빈	CPU활용율	클러스터링
부하분배계산 및 속도측면	상	중	중
서버가용성	중	상	상
확장성	중	중	중

라운드로빈 방식은 부하분배를 계산하지 않고, 사용자 접속이 들어오는 대로 이미 지정된 서버에 접속하므로 부하 분배시 계산되는 시간 및 속도는 효율적이나, 서버가용성과 확장성은 다른 방식에 비해 떨어진다. CPU 활용율은 기존의 서버 정보를 가지고 부하 분산을 하므로 서버가용성 측면에서는 우수성을 가지나, CPU 활용율만을 가지고 서버의 부하율을 측정하는 것은 정확치 않다는 지적이 있다. 클러스터링 방식은 서버가용성 측면에서 좋은 방법이나, 서버의 수가 매우 많지 않은 상황에서는 구현측면의 비용이 많이 드는 것이 단점이다[7].

본 논문에서는 구현측면의 비용을 줄이는 동시에 기존의 서버의 부하율 정보를 가지고 효율적인 방안을 실험하여 이를 대상으로 부하분산 도구를 설계하고자 한다.

Ⅳ. 부하 분산 도구 설계

이 장에서는 제안되는 기법을 가지고 온라인 게임 서버에서 사용되는 균일한 분산 분포를 가지는 부하 분산 도구를 설계하고자 한다. 본 논문에서는 설계 및 구현된 온라인 게임 서버를 위한 부하 분산 도구를 GSLB(Game Server Load Balancer)라 명명한다.

4.1 고려사항

시스템 설계 시에 고려되어야 할 사항은 다음과 같다.

1) 동시 사용자수의 충분한 고려 : 서버에 접속하려는 사용자가 동시에 부하 분산기로 입력될 때를 충분히 고려해야 한다.
2) 분산 처리 : 게임월드를 지역적인 혹은 사용자 수 밀도 등의 관점에서 분할하여 처리할 수 있어야 한다.
3) 확장성 : 게임의 크기 혹은 사용자의 수 등에 따라 게임 서버를 적절하게 유지하고, 필요에 따라 게임 서버의 숫자를 조절해야 한다.

4.2 시스템 설계

4.2.1 클라이언트

클라이언트는 게임유저를 의미하며, 전체 시스템의 구조는 그림 2와 같다. 또한 게임유저는 TCP (Transport Control Protocol) 프로토콜에 의해 GSLB(Game Server Load Balancer)에 접속된다

4.2.2 GSLB

(1) 큐 : 들어온 사용자는 큐(Queue)에 저장되어 처리할 수 있도록 보관하는 모듈이다. 처리되는 순서는 FIFO(First Input First Output) 형태로 처리된다.

(2) 처리부 : 항상 게임서버에 위치한 프로브(probe)로부터 값을 받아 처리하는 모듈이다. 이 모듈은 프로브로부터 받아 데이터베이스의 테이블에 저장된 값을 다음의 세가지 방식에 의해 처리하도록 하였으며, 기본적으로 연결자수에 의한 방식을 채택하도록 설계하였다.

- CPU활용율(cpu(a))에 따른 방법
- 연결자수(C(i))에 따른 방법

```
알고리즘: 연결자수에 따른 부하 분산 기법
Procedure CU            /* Connected User
  QUEUE(u) <- New User
  Check Number(Users of Every Server)
  for I=1 -> n do
    if Number(Users of Server(n)) <= min_user
    then
      Add Server(n) <- QUEUE(u)
      Count Sum(Server(n)) = Server(n) + 1
    end if
  end for
end Procedure CU
```

- 임계치계산에 의한 방법

```
알고리즘: 임계치계산에 의한 부하 분산 기법
Procedure TH            /* Threshold Capacity
  Y= 0 or 1 by Response time
  A= Weight(a) x Number(Connected Users)
  B= Weight(b) x CPU utilization
  Threshold = Y x A x B
end Procedure TH
```

(3) 서버등록부 : 게임서버의 값을 추가, 삭제, 수정 등을 해서 데이터베이스에 보관하는 모듈이다.

(4) 모니터링부 : 프로브에 의해 측정된 값을 데이터베이스로부터 가져와 그래프 형태와 표 형태의 값으로 출력하는 모듈이다. 추후 검색 기능 등을 보강할 예정이다.

(5) 관리부 : 관리자 정보 및 초기 셋업 정보를 관리하는 모듈이다. 추후 데이터베이스 정보의 이관 등을 관리할 수 있는 기능을 추가한다.

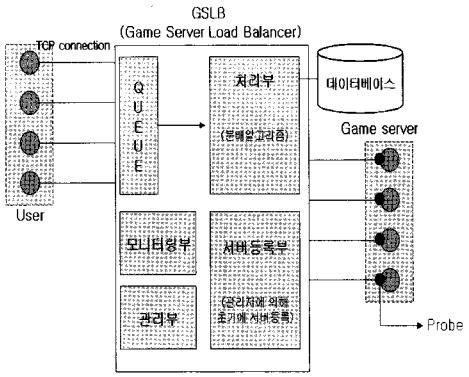


그림 2. 전체 시스템 구조

4.2.3 게임 서버

네트워크 게임을 할 수 있도록 제공되는 서버로써, 2.1에서 설명한 특징들을 처리할 수 있도록 제안되었다. 본 논문에서는 게임 서버에 대한 내용은 언급하지 않으며, 이에 대해서는 참고문헌[1]을 통해 살펴볼 수 있다.

4.3 메시지 처리 절차

메시지 처리 절차는 그림 3과 같다.

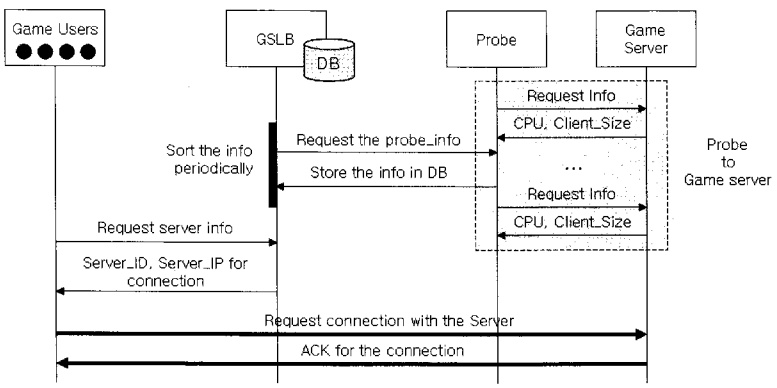


그림 3. 메시지 처리 절차

1) 프로브는 주기적으로 게임서버에서 CPU 사용율 및 게임서버에 있는 게임유저의 수를 측정한다.
2) 또한 그 정보를 GSLB 즉 게임서버 로드 밸런서에서 주기적으로 가지고 오는데 그때 사용하는 GSLB와 프로브간 패킷 구조는 다음과 같다. 길이(Length)는 2바이트로 헤더 및 페이로드의 크기를 의미한다. 코드(Code)는 GSLB에서 프로브로 요청 시와 응답 시로 나뉠 수 있는데, 십진수로 표기되는 그 값은 표 1과 같다.
   (1) 길이(Length): 2bytes
   (2) 코드(Code): 2bytes
   00: GSLB에서 프로브로 값을 요청시
   51: 프로브에서 GSLB로 CPU 값을 전달시
   52: 프로브에서 GSLB로 서버내 접속자수 전달시
   53: 프로브에서 GSLB로 CPU값과 접속자합 전달시
   (3) 페이로드(Payload): 이하 bytes
3) 프로브로부터 가져온 정보는 데이터베이스로 저장하여 분배 알고리즘 계산 시와 모니터링 부에서 사용한다.
4) 클라이언트의 요청이 GSLB 쪽으로 들어오면 분배 알고리즘에 의해서 가장 적절한 서버 값을 계산하여 클라이언트에서 전달한다.
5) 클라이언트 내에서는 해당되는 서버 값의 IP로 접속하여 게임을 한다.

4.4 데이터베이스 스키마

데이터베이스 스키마는 표 2, 표 3, 표 4와 같이 설계되었다. Game_Server_Info 테이블은 게임서버의 정보를 보관하고, Game_Server_Load 테이블은 CPU 측정결과를 보관한다. 또한 Probe_Info 테이블은 서버 측에 인식된 에이전트(agent)로 서버의 측정값을 가져오는 프로브에 대한 Up/Down 정보 등을 가지고 있다.

표 2. Game_Server_Info 테이블

속성	타입	설명
sid_num	int(2)	Primary key
sid	varchar(20)	Primary key
ip	varchar(20)	게임서버 IP
port	varchar(5)	게임서버 IP의 임의 포트 주소
max_client	char(6)	게임서버의 최대허용사용자수

표 3. Game_Server_Load 테이블

속성	타입	설명
sid_num	int(2)	Primary key
date_time	datetime	측정시간
cpu_use	int(3)	부하률 (%로 측정)
weight	int(3)	CPU율과 현재접속자의 계산값
client_n	int(3)	현재 접속자수

표 4. Probe_Info 테이블

속성	타입	설명
probe_method	char(6)	측정방식
probe_client_weight	char(3)	서버상의 연결자수 가중치
probe_cpu_weight	char(3)	서버상의 CPU이용율 가중치
probe_meas_time	char(4)	측정시간
stat	int(2)	Probe의 online, offline 체크

V. 실험 및 평가

이 장에서는 네트워크 게임 서버에서와 같은 가정사항을 두어, 기존의 라운드로빈 방식과 본 논문에서 제시한 방식을 균등 분산 관점에서 실험하고 그 결과를 비교하고자 한다.

5.1 실험 모델 환경

실험은 AWSIM 시뮬레이션툴을 이용하였으며 게임서버의 최적 서버이용율은 많은 실험을 통해 70%, 서버이용임계범위는 60~80%로 정했으며, 하루 접속자수가 50,000명으로 추정되는 5개의 가상의 서버를 일주일간 실험한 결과를 대상으로 한다. 위의 대상에 대한 실험 결과를 좀더 효율적으로 분석하기 위해 아래의 실험 방법과 식을 사용하였다. 실험 모델은 다음의 A형 로드분산기와 B형 로드분산기, 두가지를 실험한 후 비교 평가하였다.

1) A유형: 사용자를 각 서버에 순서대로 할당
2) B유형: 사용자를 각 서버의 CPU사용율을 계산하여 할당
2) C유형: 사용자를 각 서버의 사용자수에 기반하여 할당

5.2 실험 방법

실험은 다음의 식에 표기된 방법으로 제안된 기법을 표현하였으며, 기존의 방법은 사용자를 서버에 차례대로 할당하는 방법을 이용하여 실험하였다. 표 5는 실험을 위한 식의 표기법이다.

표 5. 실험 표기법

표기명	내용기술
OSU	최적 서버 이용율
SUT_v	서버 이용 임계치
SUT_a	서버 이용 임계범위
LBE	서버부하분산 성능률
$V(X)$	변량 X에 대한 분산

서버이용 임계범위는 다음과 같이 정의한다.

$$SUT_a = [OSU - SUT_v,\ OSU + SUT_v]$$

또는

$$OSU - SUT_v \leqq SUT_a \leqq OSU + SUT_v$$

서버부하분산 성능평가를 다음과 같이 정의한다.

$$LBE = \frac{V(SUT_v)_{OSU} - V(S_i)}{V(SUT_v)_{OSU}} \times 100$$

$$\text{where } V(SUT_v)_{OSU} = \frac{\sum (SUT_v)^2}{N}$$

$$V(S) = \frac{\sum (S_i - m)^2}{N}$$

- $V(SUT_v)_{OSU}$: 최적 서버이용율기반의 서버이용임계치에 대한 분산
- $V(S_i)$: 게임서버 S_i 이용율에 대한 분산
- S_i: i 번째 게임서버 이용율 $(1 \leqq i \leqq N)$
- m : 평균서버이용율

5.3 실험 평가

A, B, C유형에 대한 실험 결과는 그림 4와 같다. 이를 통해 분산결과는 다음과 같이 평가된다.

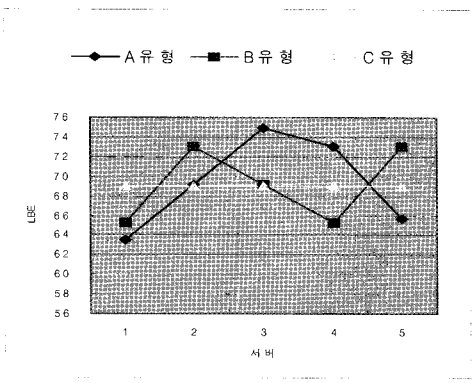


그림 4. 유형별 균일성 비교평가

그림 4에서와 같이 A유형에서는 서버1는 성능의 63%을 처리하고 있고, 서버3는 75%를 처리하고 있다. 이처럼 A형은 서버마다 성능에 상관없이 특정 서버에 로드가 편중되는 단점을 가지고 있다. B의 경우도 마찬가지이다. 반면 C알고리즘은 각 서버들이 70% 정도에서 균일한 로드를 처리하고 있음 알 수 있다. 즉 C형에서는 평균적으로 비슷한 양의 부하율로 수행이 되지만, 각 서버마다 균일한 성능을 유지하게 되므로, 본 논문에서 주장하는 "균일성"의 척도를 가지게 됨을 알 수 있다.

Ⅵ. 구 현

이 장에서는 설계방식에 따라 구현된 사용자 인터페이스를 간략하게 보이고자 한다. 구현된 네트워크 환경은 그림 5와 같다. 구현은 웹기반으로 C언어와 PHP를 이용하여 구현되어졌다.

구현된 시스템은 관리자 아이디와 비밀번호로 로그인 하여, 게임서버 등록 및 모니터링 그래프 출력, 부하 분배 관련 및 상세적인 메뉴로 이루어져 있다.

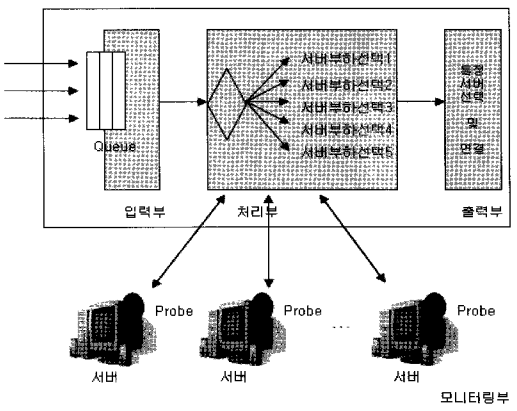


그림 5. 구현된 네트워크 환경

그림 6은 분배알고리즘을 위한 상세 선택 메뉴이다. 되어졌다. 상세 선택 메뉴는 본 논문에서 제시한 효율화된 사용자수 기반의 방법 이외에 서버 부하 분산 방식의 선택할 수 있도록 하였으며, 프로브의 기간 및 각 서버간 최대 유저수 제한 등을 선택적으로 조정할 수 있도록 하였다.

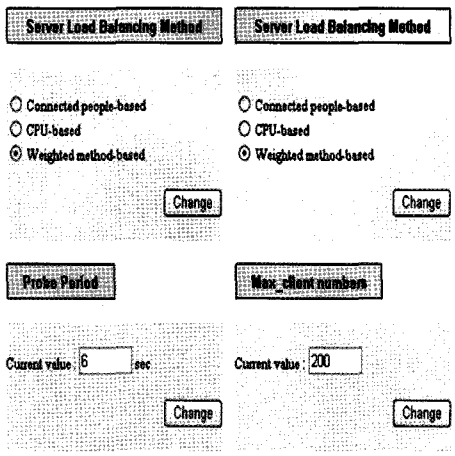


그림 6. 분배알고리즘 상세 선택메뉴

Ⅶ. 결 론

본 논문에서는 네트워크에서 게임을 하는 게임 접속 사용자수가 기하급수적으로 늘어나면서 서버 쪽에 발생되는 부하는 그 수치가 급격하게 증가하므로 게임유저의 서비스 만족도를 일정 수준으로 유지하기 위하여 게임 서버의 효율적인 부하 분산 방안을 제시하였다. 기존의 방식과의 분산도 평가를 통해, 제시하는 방법에 대한 부하 분배율에 대한 안정도를 평가하였다. 따라서 각 게임 서버들의 부하 분배율에 대한 안정성을 높여 사용자의 수와 활동 사항을 고려하지 않고 게임 서버를 두어 생길 수 있는 문제를 대처할 수 있게 되었다. 향후 연구에서는 서버 부하율을 더욱 줄이는 효율화된 방법을 제안하여 적용토록 할 것이다.

참 고 문 헌

[1] 임정열 외, "온라인 게임서버 기술의 분석 및 전망", *정보처리학회지 제12권 제6호*, pp.60-68, 2005년 11월.

[2] 이장섭 외, "무무선 게임 연동 논문 분석", *정보처리학회논문지D 제12권 제3호*, 2005년 6월

[3] 남재욱, 온라인 게임서버 프로그래밍, *한빛미디어*, 2004년 5월.

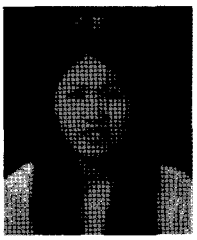
[4] Thor Alexander, Massively Multiplayer Game Development, *Charles River Media*, pp.213-227, 2003.

[5] Chandra Kopparapu, Load Balancing Servers, Firewalls, and Caches, pp.23-48, *WILEY*, 2002.

[6] Abdelkhalek et al, "Parallelization and performance of interactive multiplayer game servers", *The 18th International Parallel and Distributed Processing Symposium*, pp.72-75, 2004.

[7] Dugki Min. et al, "A load balancing algorithm for a distributed multimedia game server architecture", *IEEE International Conference on Multimedia Computing and Systems, Volume 2*, pp.882-886, 1999.

엄 남 경 (Nam-Kyoung Um) 정회원



1999년 2월 : 충북대학교 컴퓨터과학과 졸업

2002년 2월 : 충북대학교 전자계산학과 석사

2004년 2월 : 충북대학교 전자계산학과 박사수료

<관심분야> 유비쿼터스 네트워크, 네트워크 보안, 침입탐지 시스템

문 형 진 (Hyung-Jin Moon) 정회원

1996년 2월 : 충남대학교 수학과 졸업

2002년 2월 : 충남대학교 수학과 이학석사

2003년 3월~현재 : 충북대학교 전자계산학과 박사과정

<관심분야> 암호학, 정보보호, 프라이버시 보호

이 상 호 (Sang-Ho Lee) 정회원

1976년 2월 : 숭실대학교 전자 계산학과 졸업

1981년 2월 : 숭실대학교 전자계산학과 공학석사

1989년 2월 : 숭실대학교 전자 계산학과 공학박사

1981년~현재 : 충북대학교 전기전자컴퓨터공학부 교수

<관심분야> 프로토콜 공학, 네트워크 보안 및 관리