Journal of Korea Multimedia Society Vol. 9, No. 7, July 2006(pp. 827-834)

라운드 로빈 스케줄링을 이용한 가상환경 이벤트의 병행 처리 알고리즘

유 석 종†

요 약

참여자의 수가 대규모화되고 있는 분산가상환경에서 사용자의 행위에 대한 응답성은 시스템의 만족도를 결정하는 중요한 요소 중의 하나이다. 시스템의 응답성은 서버에서 이벤트 처리 알고리즘의 효율성에 의하여 영향을 받으며 대부분의 가상환경은 이벤트 메시지를 발생순서에 따라 순차적으로 처리하는 FCFS알고리즘을 사용한다. FCFS방식은 이벤트의 우선순위는 고려하지 않고 발생시간에만 전적으로 의존함으로써 일반 이벤트 메시지의 처리로 인해 사용자의 상호작용에 의해 생성된 긴급 이벤트의 처리가 지연되어 시스템 응답성을 저하시키는 문제점을 가지고 있다. 본 연구에서는 기존의 FCFS 이벤트 처리방식의 문제점을 개선하기 위하여 이벤트에 우선순위를 부여하고 이에 기반하여 서버에서 다중 메시지의 병행 처리가 가능한 이벤트 스케줄링 알고리즘을 제안한다. 본 알고리즘은 서버에서 처리충돌을 일으키는 이벤트에 대하여 발생시간과 우선순위 정보를 상호절충하여 처리함으로써 시스템의 일관성과 응답성의 목표를 동시에 만족시키는 효과가 있다.

Concurrent Processing Algorithm on Event Messages of Virtual Environment Using Round-Robin Scheduling

Seokjong Yu†

ABSTRACT

In distributed virtual environment, system response time to users' interactions is an importance factor to determine the degree of contentment with the system. Generally, response time is affected by the efficiency of event message processing algorithm, and most of previous systems use FCFS algorithm, which processes message traffic sequentially based only on the event occurrence time. Since this method totally depends on the order of occurrence without considering the priorities of events, it has a problem that might drop the degree of system response time by causing to delay processing message traffic when a bottleneck phenomenon happens in the server side. To overcome this limitation of FCFS algorithm, this paper proposes a concurrent event scheduling algorithm, which is able to process event messages concurrently by assigning the priorities to the events. It is also able to satisfy the two goals of system together, consistency and responsiveness through the combination of occurrence time with priority concept of events.

Key words: Distributed Virtual Environment(분산가상환경), Server Scalability(서버 확장성), Concurrent Event Processing(병행 이벤트처리), Message Scheduling(메시지 스케줄링)

※ 교신저자(Corresponding Author) : 유석종, 주소 : 서울시 용산구 효창원길 52(140-742), 전화 : 02)710-9831, FAX : 02)710-9296, E-mail : yusjong@sookmyung.ac.kr
접수일 : 2005년 10월 4일, 완료일 : 2006년 3월 9일

† 정회원, 숙명여자대학교 컴퓨터과학과 조교수
※ 본 연구는 숙명여자대학교 2006년도 교내연구비 지원에 의해 수행되었음

1. 서 론

분산가상환경(distributed virtual environment : DVE)은 분산된 참여자들 간에 상태 정보를 주고 받으며 온라인게임, 원격회의 등의 공동작업을 수행하는 시스템을 말한다[1,2,3,4]. 인터넷과 같은 전송에 대한 신뢰도가 낮은 시스템에서 사용자의 만족도에 영향을 미치는 요소는 일관된 성능의 서비스라고 할 수 있다. 초기 시스템에서 10명 정도 수준에서 출발한 네트워크 게임은 현재 수천명 ~ 수만명 수준의 MMORPG(Massively Multiplayer Online Role-Playing Game)로 발전하였으며 최근에는 백만명 규모의 가상공간의 설계를 목표로 하고 있다[5,6]. DVE에서 참여자가 발생시킨 사건을 이벤트(event)라 부르며 발생 즉시 메시지 패킷의 형식으로 서버를 경유하여 관련된 참여자에게 통지되어야 하며 이러한 특성을 시스템의 일관성(consistency)이라고 한다. DVE의 참여자의 수가 대규모화될수록 서버에서 데이터 처리량이 급격히 증가하므로 참여자 및 데이터 트래픽에 대한 효율적인 관리기술은 더욱 중요해지고 있다[2,4].

서버는 데이터 트래픽을 조절하기 위하여 AOI모델[2,4,7,8], 데드레코닝(dead reckoning) 알고리즘[3,4]등의 메시지 필터링 기법을 적용하는 것이 일반적이다. AOI모델은 참여자가 관심을 두고 있는 주변의 영역으로 지속적으로 일관된 상태 유지가 필요한 공간이다. 즉, AOI내의 참여자에게만 이벤트를 통지함으로써 메시지 트래픽을 효과적으로 줄일 수 있다. 기존의 서버에서 이벤트의 처리 방식은 메시지 패킷이 서버에 도착하는 순서에 따라 메시지 큐에 저장한 후 순차적으로 처리하는 FCFS(First Come First Served) 알고리즘을 사용하였다[1,2,3,4]. 서버에서 특정 메시지가 처리되고 있는 시점에 새로운 이벤트 메시지가 도착하게 되면 이전 메시지의 처리가 완료되기까지의 시간은 그대로 대기시간으로 이어진다. FCFS 알고리즘은 이벤트의 발생량이 많은 경우 메시지 트래픽이 증가하여 서버의 병목현상을 유발할 수 있으며 이벤트의 통지에 상당한 시간지연이 발생할 수 있다는 문제점을 앉고 있다. 참여자와 참여자간, 또는 참여자와 가상 객체간의 상호작용과 관련된 이벤트에 대한 처리지연은 시스템의 응답성을 떨어뜨리는 상황으로 이어질 수 있다.

본 연구에서는 서버에서의 메시지 처리방식은 FCFS방식의 문제점을 개선하기 위하여 두 개 이상의 이벤트 메시지가 서버에서 충돌되는 경우(메시지 처리중에 새로운 메시지가 도착하는 경우)에 다수의 이벤트를 병행 처리하는 스케쥴링 알고리즘을 제안하고자 한다. 병행 처리 스케쥴링은 특정 메시지가 서버를 독점하지 않고 이벤트의 우선순위를 고려하여 분배함으로써 이벤트의 처리지연시간을 최소화하여 시스템의 일관성을 유지하면서 응답성을 높인다는 장점을 가질 수 있다. 2장에서는 관련연구에 대하여 소개하고, 3장에서는 제안된 이벤트 병행처리 모델에 대하여 기술하였으며, 4장에서는 기존의 FCFS방식과 제안된 병행 처리 방식의 성능평가 실험 및 분석 결과를 제시하였다.

2. 관련연구

2.1 분산가상환경

분산가상환경은 다수의 참여자에 의해 공유되는 통신망 상의 가상세계(cyberspace)를 말한다[1,2,3,4]. 기존의 가상공간이 2차원의 평면적인 것이라면 DVE은 3차원 공간을 지향한다. 참여자는 자신의 분산인 3차원 아바타(avatar)로 표현되어 자신의 행동과 존재를 외부에 알릴 수 있다[4,7]. 다수의 참여자들간에 실세계의 사회생활과 유사한 협동, 경쟁, 토론 등의 공동행위(collaboration)가 이루어진다. DVE의 개념과 기술은 대규모 온라인 게임, 가상 커뮤니티, 원격회의, 가상도시 등 다양한 분야에 응용되고 있다[1,2,3,5,6].

DVE에서 원활한 공동작업을 위해서는 참여자에 의해 발생한 이벤트 사실을 타 참여자에게 즉시 통지해 주어야 하므로 참여자의 수가 증가할수록 메시지 트래픽도 비례하여 증가된다. 시스템에 접속한 참여자의 수에 대해 시스템에서 제공되는 서비스의 성능에 저하가 발생하지 않는 성질을 DVE의 확장성(scalability)라고 부른다[4,8,10]. 이벤트란 DVE에서 참여자에 의해 일어나는 사건을 말하며 이벤트 발생 사실을 가상공간에 참여한 타 참여자에게 통지하는데 이벤트 메시지가 사용된다. 이벤트를 발생시킨 대상을 주시하는 참여자의 수에 비례하여 메시지 트래픽도 증가하게 된다. 메시지 트래픽을 여러가지 기준에 의해 적절하게 조절하는 기법을 메시지 필터링(message filtering)이라 부르며 공간분할기법(spatial

partitioning)[2,4,7], 데드레코닝 알고리즘[3,4]이 여기에 속한다.

2.2 공간분할과 이벤트 우선순위

공간분할기법(spatial partitioning)은 가상환경에서 시스템의 확장성을 높이기 위하여 폭넓게 사용되는 필터링 방법 중의 하나로 가상공간을 여러 개의 지역(zone)으로 분할하여 각각의 담당서버에 의하여 관리하도록 한다[1,2,4]. 하나의 지역은 또다시 더 작은 셀(cell)로 세분화되며 여러 개의 셀을 묶어 AOI로 정의한다. AOI는 이동성 기능에 따라 고정형(fixed type)과 이동형(movable type)으로 나뉘고, AOI을 구성하는 영역의 형태에 따라 육각형(hexagon), 사각형(rectangle), 원형(circle), 불규칙형(irregular)으로 나뉜다[1,2,4,8]. 육각형 셀을 사용한 AOI모델은 군사훈련용 분산시뮬레이션 시스템인 NPSNET에서는 육각형 셀의 AOI모델이 사용되었다[4]. AOI는 참여자가 관심을 갖는 주변의 영역으로 이벤트를 통지할 대상을 제한함으로써 메시지 트래픽을 효과적으로 필터링할 수 있다. AOI의 기능을 확장하여 영역을 거리에 따라 분할하여 이벤트의 우선순위를 부여하는 것이 가능하다. 즉 참여자간의 상호작용, 참여자의 가상객체와의 상호작용에 대한 이벤트에 대해서는 높은 우선순위를 부여하고 AOI 내지만 참여자의 상호작용과는 직접관련 없는 이벤트는 낮은 우선순위를 부여하는 개념이다.

2.3 참여자의 밀집도와 서버 병목 현상

이벤트 메시지는 일반적으로 중앙의 서버의 대기큐에 저장되어 순차적으로 처리되어 타 참여자에게 전달된다. 서버의 처리용량보다 초과하여 메시지 트래픽이 과다하게 발생하는 경우 병목현상(bottleneck)은 불가피하며 일부 이벤트의 통지지연시간은 증가하게 된다. 메시지 트래픽의 증가로 인한 서버병목현상은 국지적인 참여자의 밀집도(density)와 관련성이 높다. 특히 이벤트 발생자(event producer)의 AOI 내 참여자가 밀집되어 있는 경우 급격한 메시지 트래픽 증가로 이어진다. 참여자의 밀집도는 균등형 분포(uniform distribution), 편중형 분포(skewed distribution), 클러스터형 분포(clustered distribution)으로 나눌 수 있다[8]. 이중 특정 지역에서 밀집도가 높은 편중분포나 클러스트 분포의 경우에 메시지 트래픽 폭주와 서버병목현상을 일으킬 가능성이 매우 높다고 할 수 있다. 참여자 밀집도와 관련된 연구로는 AOI 축소방법[2]과 분산서버간의 부하균등법(load balancing)[8,10,12]이 있다. AOI 축소방법은 밀집도가 증가하는 경우 AOI의 크기를 줄이는 방법이다. 부하균등법은 다중 서버에 의해 분할 관리되는 되는 특정 서버에 부하가 편중되는 것을 여러 서버로 분산시키는 방법이다.

이벤트 메시지는 수신자의 수에 따라 서버의 처리량(throughput)이 달라진다. 밀집도가 높은 지역에서 발생한 이벤트는 상대적으로 처리량이 높으며 서버에 대한 점유율도 상승한다. 서버가 특정 이벤트 메시지를 처리하는 동안 새로운 이벤트 메시지가 도착하는 경우를 이벤트 충돌(collision)라 정의할 수 있으며 서버에 도착 시간에서부터 처리되는 시간간의 차이를 처리지연시간(processing delay time)이라 정의한다.

3. 이벤트 병행 처리 모델

3.1 FCFS 기반 이벤트 스케줄링

이벤트는 발생되는 즉시 메시지의 형식으로 중앙서버에 전달되며 이벤트 메시지 큐에 저장된다. 대기큐에 저장된 이벤트 메시지는 FCFS방식[1,2,3]에 의해 필터링 알고리즘을 적용 받은 후 이벤트 수신자에게 전달된다. 이벤트 메시지 큐가 길어질 수록 대기시간도 함께 증가된다. 이벤트 메시지는 이벤트 발생자의 AOI 내에 많은 참여자가 존재하는 경우(이벤트 수신자가 많은 경우) 처리량도 커지게 된다. FCFS방식의 단점은 이벤트가 서버에 도착하는 순서에만 의존하여 처리함에 따라 처리량이 큰 메시지가 대기중인 경우 전체적인 메시지의 처리 대기시간도 증가하게 된다. 또한 참여자와 참여자간, 참여자의 객체간의 상호작용과 같이 이벤트의 중요도(priority)가 전혀 반영되지 못함으로써 시스템의 응답성이 떨어질 가능성이 높다.

그림 1은 서버에 도착한 이벤트 메시지간의 충돌과 처리지연시간을 나타낸 것이다. 메시지가 서버에서 처리중에 새로운 메시지가 도착하는 것을 이벤트 충돌(event collision)으로 표현하였다. 5개의 이벤트 메시지의 처리량은 각각 20, 15, 18, 20, 10이며 이들 메시지를 처리하는 데 지연된 총 시간은 d1, d2, d3,

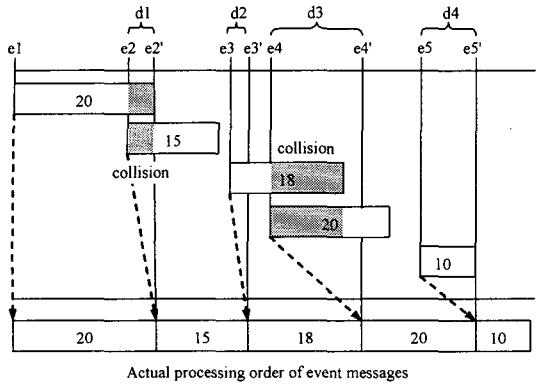


그림 1. 이벤트 발생시간과 처리시간

d4의 합으로 표현할 수 있다. 메시지가 충돌되는 순간 메시지 큐에 남아있는 처리량에 비례하여 지연시간도 증가하는 것을 알 수 있다.

3.2 라운드-로빈 기반 이벤트 스케줄링

본 연구에서는 서버에서의 효율적인 메시지 처리를 위하여 우선순위 대 지연시간 비율(Priority-Delay Ratio)이 최소화될 수 있는 라운드-로빈(Round-Robin: RR) 알고리즘에 기반한 이벤트 메시지 스케줄링 기법을 제안한다. 서버에서 충돌되는 이벤트 메시지는 시간적으로 볼 때 거의 동시에 발생되는 사건이라고 할 수 있다. FCFS방법은 단일 대기큐를 사용하여 순차적으로 메시지를 처리함으로써 거의 동시적으로 발생한 이벤트 메시지의 처리시간을 불공정하게 증가시키는 단점이 있다. 본 연구에서 이러한 문제점을 개선하고 전체적인 메시지 처리시간을 최소화하기 위하여 서버에서 충돌되는 이벤트 메시지를 라운드-로빈 방식으로 병행처리(concurrent processing)하는 스케줄링 알고리즘을 도입하였다. 제안된 방법은 메시지의 충돌이 없을 때에는 기존 방법과 같이 FCFS방식으로 수행하다가 메시지 충돌이 발생하는 시점부터는 라운드-로빈 방식으로 스케줄링 알고리즘을 전환하여 충돌을 일으킨 이벤트 메시지 간에 발생시간과 우선순위를 고려하여 병행처리를 수행한다.

이벤트의 중요도는 상호작용 관련성을 나타내는 것으로 이벤트 수신자와의 관심도(DOI: degree of interest)[2]를 통하여 결정된다. DOI는 발생자와 수신자간의 거리(distance)와 방향(orientation)으로부터 계산되며 0.1 ~ 1.0 범위의 값을 갖고, DOI 값이 높을수록 대상에 대하여 높은 주의를 기울이고 있다는 것을 의미한다. 본 연구에서 이벤트 중요도는 특정 이벤트의 모든 수신자의 DOI 평균이 어느 범위에 있는가에 따라 다음과 같이 정의한다. 평균이 상위 30%(1.0 ~ 0.67)일 경우 중요도 높음(우선순위 1), 중간의 30%(0.66 ~ 0.34)일 경우 중요도 보통(우선순위 2), 하위 30%(0.33 ~ 0)경우 중요도 낮음(우선순위 3)에 속한다. 이벤트의 우선순위는 라운드-로빈 알고리즘 적용시 처리단위로 환산되어 사용된다. 라운드-로빈 알고리즘은 기본적으로 1:1의 비율로 순환 처리하는데 비하여 제안된 알고리즘은 이벤트 메시지의 우선순위에 기반한 가중치를 적용하기 위하여 1:d 비율로 순환처리하는 방식을 사용하였다. 즉 우선순위가 동일한 이벤트에 대해서는 1:1비율이 적용되지만 우선순위가 상이한 이벤트에 대해서는 식(1)과 같이 우선순위 차이(d)를 계산하여 비균등비율로 처리된다.

$$d = \frac{1}{|P(A) - P(B)| + 1} \tag{1}$$

P(A)와 P(B)는 각각 이벤트 A와 B의 우선순위를 의미한다. 표 1는 서로 다른 우선순위를 갖는 이벤트 간의 라운드-로빈 비율을 정리한 것이다. 우선순위 1인 이벤트와 3인 이벤트 메시지가 충돌되는 경우 1:1/3 (d=1/3) 비율의 라운드-로빈 알고리즘이 적용되어 처리된다. 3:1의 라운드-로빈 비율은 A메시지가 3단위 처리되고 난 후 B메시지가 1 단위 처리된다는 의미이다. 표에서 1:2, 1:3의 숫자는 각 이벤트 처리량(workload)에서 서버가 한 주기에 처리할 메시지의 수를 의미한다.

수식 (2)는 n개의 이벤트 메시지가 $r_1 : r_2 : \cdots : r_n$의 비율로 서버에 의해 s 사이클 처리된다는 의미이다.

$$RR[e_1(r_1), e_2(r_2), \cdots, e_n(r_n)]^s \tag{2}$$

n은 서버에서 충돌한 메시지 수이고, $r_1, r_2, \ldots, r_n$는

표 1. Round-Robin Ratio

우선순위(A) \ 우선순위(B)	1 (높음)	2 (보통)	3 (낮음)
1 (높음)	1:1	2:1	3:1
2 (중간)	1:2	1:1	2:1
3 (낮음)	1:3	1:2	1:1

RR의 한 주기에서 각 이벤트 메시지 당 처리될 처리량(workload)이며, S는 RR에 의한 처리 횟수를 각각 의미한다.

RR 스케줄링 알고리즘은 이벤트 메시지의 수(n)이 2 이상일 때 적용되며 이벤트 메시지가 1이 되면 정상적인 FCFS 알고리즘으로 전환된다. 그림 2는 두개 이상의 이벤트 메시지가 서버에서 충돌되었을 때 RR 스케줄링 알고리즘으로 처리되는 비율을 도시한 것이다.

- 그림 2(a)는 이벤트 메시지 e_1이 모두 처리된 이후에 e_2가 도착되어 충돌이 일어나지 않는 경우이다. 따라서 RR 스케줄링도 적용되지 않는다.
- 그림 2(b)는 처리량 20, 우선순위 3의 e_1의 처리량(workload)이 15단위 처리되고 난 후 처리량 10, 우선순위 1의 e_2가 도착한 경우로 1:3의 비율로 RR 알고리즘이 적용된다. RR 알고리즘이 4 사이클 적용되고 나면 e_2은 모두 처리되고 남은 e_2의 처리량 1이 정상 알고리즘으로 처리된다.
- 그림 2(c)는 우선순위 1, 처리량 20의 e_1이 처리중에 우선순위 2, 처리량 10의 e_2가 도착하였다. 2:1의 비율로 3 cycle 동안 RR 방식으로 처리되고 e_2의 처리량 7이 남는다.
- 그림 2(d)의 경우는 우선순위가 같은 이벤트 메시지가 1:1의 비율의 RR 방식으로 처리되는 것을 보여준다.
- 그림 2(e)는 e_1이 처리도중에 e_2가 도착하여 3:1의 비율로 처리된다. (f)는 세 개의 이벤트 메시지가 충돌되는 경우이다. 처음에는 e_1과 e_2가 3:1의 비율로 RR 처리되다가 e_3가 도착하여 e_1, e_2, e_3가 3:1:2의 비율로 RR 처리된다.

4. 실험 및 평가

본 실험의 목적은 기존의 FCFS 스케줄링 알고리즘과 제안된 라운드-로빈 스케줄링 알고리즘의 성능을 비교하여 이벤트 메시지 처리지연시간을 측정하는데 있다. 실험은 가상의 데이터를 이용한 시뮬레이션을 통하여 수행하였으며 시뮬레이션 시스템을 Microsoft Visual Studio를 이용하여 C코드로 구현하였다. 이벤트 메시지의 우선순위는 DOI의 값에 의하여 높음, 보통, 낮음의 3구간으로 분할하였으며, 30 ~ 50 단위의 처리량을 갖는 이벤트 메시지를 발생시

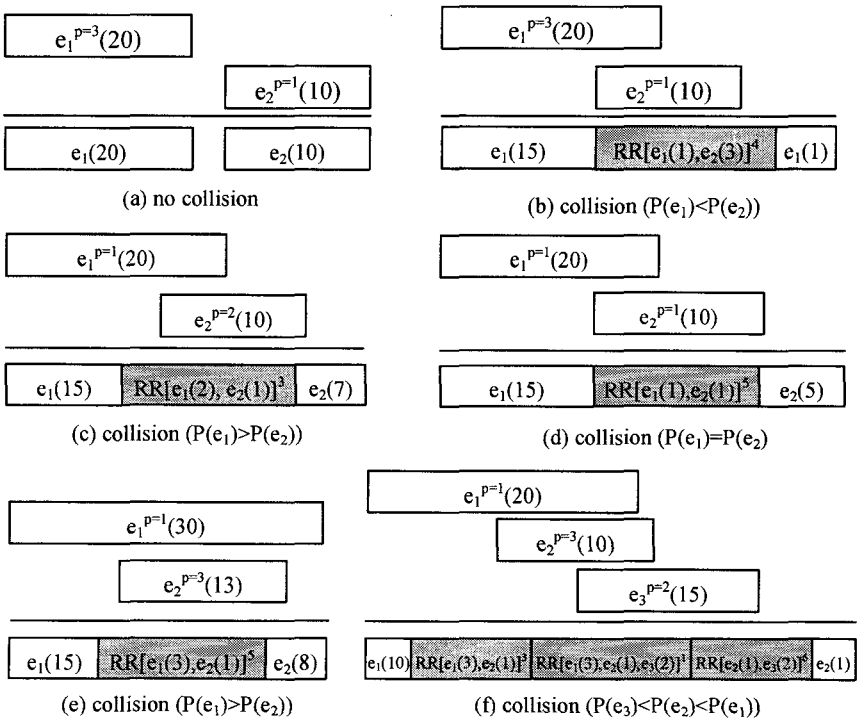


그림 2. 라운드-로빈 스케줄링 알고리즘

켜 실험을 진행하였다. 제안된 알고리즘은 n개 이상의 메시지 충돌에 대하여 적용될 수 있도록 제안되었으나 실험을 단순화하기 위하여 충돌 가능한 최대 메시지의 수는 2로 제한하였다. 메시지 충돌 확률을 10 ~ 100% 범위에서 10%씩 증가시키면서 서버의 큐에서의 평균 대기시간을 측정하였다. 메시지 충돌 확률이란 하나의 메시지가 서버에서 처리되고 있는 도중에 새로운 메시지가 도착할 확률이다. 충돌확률 50%의 의미는 10개의 메시지가 서버에 도착하였을 때 이중 5개의 메시지에서 처리충돌이 발생한다는 의미이다. 두 메시지 간의 충돌시점은 먼저 처리중인 메시지 A의 크기(예, 30) 내에서 랜덤하게 한 시점(1 ~ 30)을 선택하여 결정하도록 하였다.

제안된 라운드-로빈 스케줄링 알고리즘이 가상환경 시스템에서 일관성과 응답성의 목표를 함께 달성하게 하기 위하여 이벤트의 발생시간과 우선순위를 동시에 고려하고자 하였다. 우선순위가 높을수록 처리하는데 할당되는 시간이 증가되고 우선순위가 낮은 이벤트 메시지는 적은 시간이 할당하였다. 우선순위가 낮을수록 이벤트 큐에서 대기하는 시간이 증가될 수 밖에 없다. 실험에서 이벤트의 우선순위는 이벤트에 대한 수신자의 관심도를 사용하여 적용하였다. S, P가 각각 이벤트 처리량(event workload: S), 이벤트 우선순위(event priority : P) 일때, 큐 대기시간(queue waiting time: D)은 이벤트 처리량에 비례하고, 이벤트 우선순위에 반비례하므로 식(3)과 같이 표현될 수 있다. 즉, 이벤트의 우선순위가 높을수록 큐 대기시간은 감소한다.

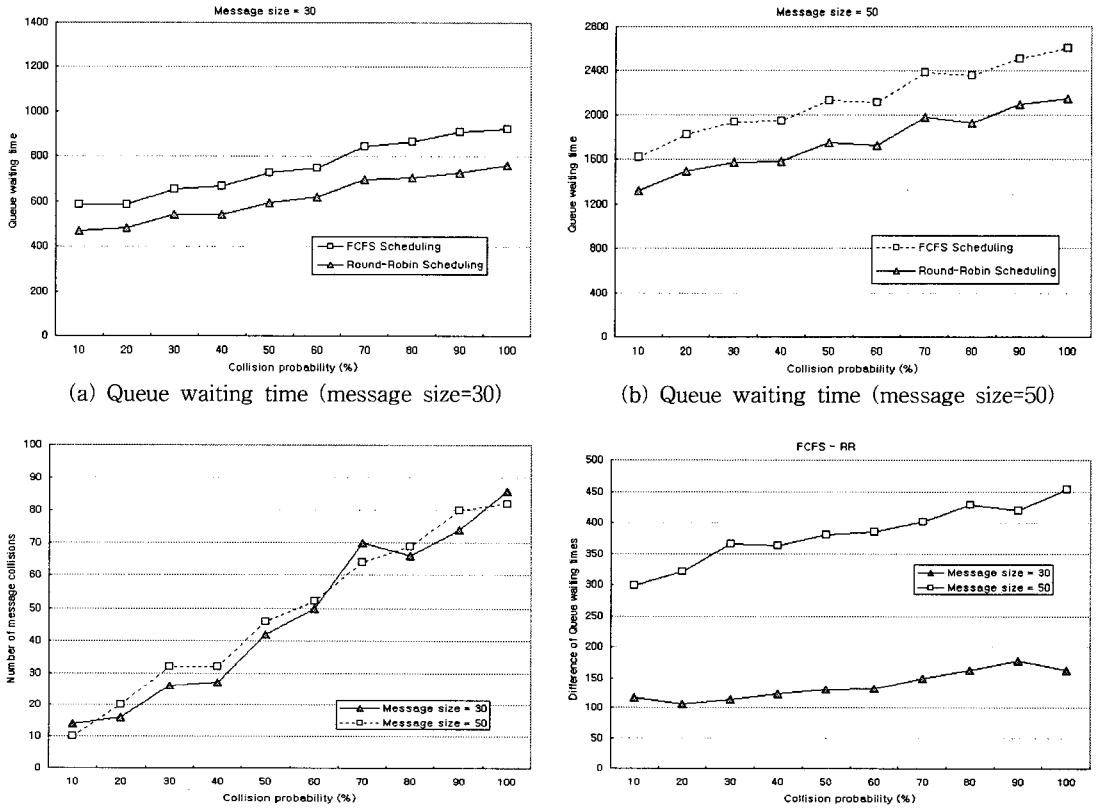
$$D = \frac{S}{P} \tag{3}$$

표 2는 큐에서의 각각 메시지 크기가 30과 50일때 이벤트 큐에서의 메시지 대기시간을 측정한 결과이다. 실험에서 메시지 충돌확률을 10 ~ 100%로 변화시키면 FCFS 스케줄링 알고리즘과 라운드-로빈 스케줄링 알고리즘을 각각 적용한 후 큐 대기시간을 측정하였다. 측정결과에 의하면 메시지 충돌확률이 높을수록 메시지 크기가 클수록 큐대기시간이 증가되는 것을 알 수 있다. 충돌횟수는 평균적으로 47.1회와 48.7회로 나타났다. 메시지 처리량이 각각 30과 50일때 FCFS 알고리즘이 적용된 이벤트 메시지의 평균 큐대기시간은 751.54와 2142.91로 측정되었다. 같은 조건에서 라운드-로빈 알고리즘을 적용하였을 때 평균 큐대기시간은 각각 614.42와 1761.44로 나타났다. 메시지 크기가 30과 50의 경우에 두 알고리즘 간의 큐대기시간 차이는 137.14와 381.57이며, 이 데이터를 통하여 FCFS방식에 비하여 RR방식이 각각 18.24%(=137.14/751.56)와 17.8%(=381.57/2142.91)의 큐대기시간 감소효과가 있음을 확인할 수 있다.

그림 3(a)과 3(b)는 메시지 크기가 각각 30과 50일 때 FCFS알고리즘과 RR알고리즘을 이벤트 메시지에 적용하여 메시지 충돌확률의 증가에 대하여 측정된 큐대기시간을 그래프로 나타내었다. 그래프에 의

표 2. Queue waiting times (message size = 30)

collision probability	Message size = 30				Message size = 50			
	FCFS scheduling (F)	RR scheduling (R)	number of event collisions	F - R	FCFS scheduling (F)	RR scheduling (R)	number of event collisions	F - R
10%	587.87	470.93	14	116.94	1623.94	1325.36	10	298.58
20%	587.18	481.51	16	105.67	1820.23	1499.43	20	320.80
30%	655.43	542	26	113.43	1937.93	1572.79	32	365.14
40%	665.71	543.46	27	122.25	1950.56	1588.68	32	361.88
50%	726.3	595.84	42	130.46	2130.53	1749.59	46	380.94
60%	749.96	618.83	50	131.13	2111.74	1726.84	52	384.90
70%	845.46	695.74	70	149.72	2385.88	1984.51	64	401.37
80%	867.5	705.03	66	162.47	2356.24	1927.91	69	428.33
90%	907.03	729.82	74	177.21	2512.40	2093.39	80	419.01
100%	923.16	761.07	86	162.09	2605.96	2151.84	82	454.12
Mean	751.56	614.42	47.1	137.14	2142.91	1761.44	48.7	381.57



(a) Queue waiting time (message size=30)

(b) Queue waiting time (message size=50)

(c) Number of message collisions

(d) Difference of queue waiting times

그림 3. 실험 측정 결과

하면 메시지가 서버에서 충돌확률이 증가될수록 두 가지 알고리즘 모두 큐대기시간이 점진적으로 증가되는 것을 볼 수 있으며 RR알고리즘이 FCFS 알고리즘보다 평균 큐대기시간이 적게 나타나고 있다. 메시지 크기가 50인 경우의 평균 큐대기시간이 메시지 크기가 30일 때에 비하여 급격히 증가되는 것을 알 수 있다. 그림 3(c)는 메시지 충돌확률을 증가시키면서 측정된 실제 충돌횟수를 도식한 것이다. 충돌확률에 대하여 선형적으로 횟수가 증가되었다. 그림 3(d)는 두 알고리즘은 큐대기시간의 차이를 나타낸 것이며, 충돌확률이 증가되는 것에 대하여 메시지 크기가 30인 경우의 그래프는 큐대기시간의 차이는 비교적 일정하게 유지되었고 50인 경우에는 충돌확률이 높아짐에 따라 다소 대기시간 차이도 벌어지는 지는 것을 확인하였다. 위의 실험 측정결과를 미루어 볼 때 제안된 RR 알고리즘은 기존의 FCFS알고리즘에 비하여 충돌확률이 높을수록 이벤트 메시지의 서버의 대기큐에서의 대기시간을 감소시키는 효과가 있으며, 그 효과는 메시지 크기가 클수록(이벤트 수신자의 수가 클수록) 비례하여 증가된다는 것을 의미한다고 볼 수 있다.

5. 결 론

컴퓨팅 성능의 향상과 네트워크 발달로 인해 사이버공간을 이용한 행위에 대한 필요성과 중요성이 커져가고 있다. 분산가상환경은 가상 커뮤니티, 온라인 게임, 원격회의와 같은 수요가 높아지는 차세대 가상공간을 구현하는 핵심기술로, 서버의 확장성은 단순한 시스템의 성능이 아니라 사용자가 시스템을 선택하는 중요기준이 되 고 있다. 본 논문에서 제안하는 라운드-로빈 스케줄링 알고리즘은 참여자 수의 증가와 함께 대두되고 있는 서버에서의 효과적인 메시지 트래픽 처리 문제에 초점을 맞추고 있다. 제시된 실험결과에 의하면 제안된 스케줄링 알고리즘은 기존의 FCFS 알고리즘의 단점이었던 서버 병목현상 발

생시 메시지 처리시간 증가와 이벤트의 우선순위 미반영의 문제를 개선하는데 효과가 있는 것으로 확인되었다. 인터넷 환경에서 대규모 참여자간에 분산 협업은 매우 많은 데이터의 처리를 필요로 하는 작업으로 서버의 하드웨어적인 성능의 발전과 함께 이 자원을 효율적으로 활용하는 소프트웨어 부분에 대한 연구가 병행적으로 이루어져야 한다고 할 수 있다.

참 고 문 헌

[1] J.W. Barrus, R.C. Waters, and D.B. Anderson, "Locales and Beacons: Efficient and Precise Support for Large Multi-User Virtual Environments," *Proceedings of the IEEE Virtual Reality Annual International Symposium*, pp. 204-213, 1996.

[2] C. M. Greenhalgh and S. D. Benford, "MASSIVE: A Distributed Virtual Reality System Incorporating Spatial Trading," *Proceedings of 15th International Conference on Distributed Computing Systems*, Los Alamitos CA, ACM Press, pp. 27-34, 1995.

[3] D. Lee, M. Lim, and S. Han, "ATLAS A Scalable Network Framework for Distributed Virtual Environments," *ACM Collaborative Virtual Environment (CVE2002)*, Bonn Germany, pp. 47-54, 2002.

[4] S. Singhal and M. Zyda, "Networked Virtual Environments," *ACM Press*, New York, 1999.

[5] Lineage. http://www.lineage.com/.

[6] Ultima online. http://www.uo.com/.

[7] T. K. Capin, I. S. Pandzic, N. Magnenat-Thalmann, and D. Thalmann, "Networking Data for Virtual Humans," *Avatars in Networked Virtual Environments*, Wiley, 1999.

[8] C.S. John, M. Lui, and F. Chan, "An Efficient Partitioning Algorithm for Distributed Virtual Environment Systems," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 13, No. 3, pp. 193- 211, 2002.

[9] J. Huang, Y. Du, and C. Wang, "Design of the Server Cluster to Support Avatar Migration," *IEEE VR2003*, Los Angeles, USA, pp.7-14, 2003.

[10] E. F. Churchill and D. Snowdon, "Collaborative Virtual Environments: An Introductory Review of Issues and Systems," *Virtual Reality*, Vol. 3, No. 1, Springer-Verlag, pp. 3-15, 1998.

[11] B. Ng, A. Si, R. W.H. Lau, and F. W.B. Li, "A Multi-Server Architecture for Distributed Virtual Walkthrough," *Proceedings of the ACM symposium on Virtual Reality Software and Technology*, pp. 163-170, 2002.

유 석 종

1994년 연세대학교 전산과학과 학사

1996년 연세대학교 대학원 컴퓨터과학과 석사

2001년 연세대학교 대학원 컴퓨터과학과 공학박사

2001년 6월~2002년 12월 University of Ottawa, Postdoctoral fellow

2003년 1월~2005년 2월 국립한밭대학교 정보통신컴퓨터공학부 전임강사

2005년 3월~현재 숙명여자대학교 컴퓨터과학과 조교수

관심분야 : 컴퓨터그래픽스, 가상현실, 온라인게임