http://dx.doi.org/10.7236/JIWIT.2012.12.4.57
JIWIT 2012-4-9

다중사용자용 실시간 게임 서버를 위한 우선순위 기반 그룹 태스크 스케쥴링 정책

Priority-based Group Task Scheduling Policy for a Multiplayer Real-time Game Server

김진환*

Jinhwan Kim

요 약 게임 서버는 명시된 시간 내에 많은 클라이언트들의 요청을 처리해야 하기 때문에 다중 사용자용 실시간 게임들은 일종의 연성 실시간 시스템이다. 클라이언트 이벤트들은 게임 세계의 본질에 따라 상이한 시간 요건과 일관성 요건을 가지고 있다. 이러한 요건들은 CPU 처리시 상이한 우선순위를 유발하게 되며 이벤트들은 일관성과 우선순위 정도에 따라 여러 그룹으로 분류될 수 있다. 우선순위가 상이한 이벤트들의 시간적 요건을 충족하기 위하여 본 논문에서는 우선순위 기반 그룹 태스크 스케쥴링 정책이 제시된다. 클라이언트의 수나 클라이언트가 발생시키는 이벤트들의 수는 일시적으로 증가할 수 있다. 일시적인 과부하가 발생한 경우에 게임 서버는 우선순위가 높은 이벤트들을 우선적으로 처리하기 위하여 더 많은 CPU 대역폭을 할당할 필요가 있다. 제시된 스케쥴링 정책은 우선순위가 높은 이벤트일수록 종료시한내에 성공적으로 종료되는 수를 최대화함으로써 전체 시스템의 실시간적 성능을 향상시킬 수 있다. 이 정책의 성능은 다양한 실험을 통하여 평가되었다.

Abstract Multiplayer, real-time games are a kind of soft real-time systems because a game server has to respond to requests from many clients within specified time constraints. Client events have different timeliness and consistency requirements according to their nature in the game world. These requirements lead to different priorities on CPU processing. Events can be divided into different groups, depending on their consistency degree and priority. To handle these events with different priority and meet their timing constraints, we propose a priority-based group task scheduling policy in this paper. The number of clients or events requested by each client may be increased temporarily. In the presence of transient overloading, the game server needs to allocate more CPU bandwidth to serve an event with the higher priority level preferentially. The proposed scheduling policy is capable of enhancing real-time performance of the entire system by maximizing the number of events with higher priority completed successfully within their deadlines. The performance of this policy is evaluated through extensive simulation experiments.

Key Words : priority, scheduling policy, multiplayer, real-time, CPU bandwidth

*정회원, 한성대학교 멀티미디어공학과
접수일자 : 2012년 2월 27일, 수정완료 : 2012년 7월 1일
게재확정일자 : 2012년 8월 10일

Received: 27 February 2012 / Revised: 1 July 2012
Accepted: 10 August 2012
**Corresponding Author: kimjh@hansung.ac.kr
Dept. of Multimedia Engineering, Hansung University, Korea

Ⅰ. 서 론

온라인과 연관된 다중사용자 게임은 게임 수행에 필요한 시간적 제약 사항 내에 행위가 전달되어야 하는 실시간적 요건을 가지게 된다[1]. 즉 클라이언트가 요청한 이벤트를 서버가 늦게 처리하여 결과가 클라이언트에게 지연되어 전달될 경우 게임 자체의 수행이 의미가 없어질 수도 있는 것이다. 다중사용자용 실시간 게임들은 약속한 서비스 품질을 보장하기 위하여 기본 통신망에 대한 엄격한 요건과 낮은 지연 시간을 요구하게 된다. 다중사용자용 실시간 게임의 구조는 동영상 스트림을 제공하는 VOD 구조처럼 타임스탬프를 사용하는 UDP[2] 통신망 기반의 클라이언트/서버 구조로 구성된다[3].

게임 클라이언트의 응용은 개발자가 사전에 정의한 행동, 게임 수행시 상호작용하는 가상 게임 세계, 클라이언트들의 행위에 의해 발생된 이벤트 등으로 구성된다. 게임 서버는 모든 클라이언트들의 이벤트들을 수집하고 게임 환경을 다시 설정한 후 가상 세계의 갱신된 정보를 게임 참여자인 클라이언트들에게 전송하게 된다. 따라서 게임 서버 개발시 연결가능한 최대 클라이언트들의 수와 게임 수행에 필요한 최소 통신망 대역폭에 대한 결정이 필요하다[4, 5].

게임 클라이언트는 임의의 입력 행위를 할 수 있으며 대부분 종료시한이 설정된 비주기적인 태스크로 간주될 수 있다[1]. NPC(Non Player Character)를 생성하거나 게임 세계를 갱신하는 일부 이벤트들은 주기적인 것으로 간주될 수 있다. 게임 서버는 다수의 비주기적 태스크와 주기적인 태스크들을 관리하고 실시간으로 스케쥴링할 수 있는 능력을 구비해야 한다.

클라이언트의 이벤트가 종료시한 내에 처리되지 않을 경우 해당 이벤트의 의미가 급격히 상실되며 게임의 지속적인 수행이 불가능하게 될 수 있다. 특히 클라이언트들의 수가 증가하거나 클라이언트가 발생시키는 이벤트들의 수가 증가할 경우 게임 서버에 일시적인 과부하가 발생할 수 있다. 기존의 EDF(Earliest Deadline First) 실시간 스케쥴링 정책[6]을 사용할 경우 과부하 상태에서는 시스템의 실시간적 성능을 향상시키기 매우 어렵다. 본 논문에서는 게임 서버가 과부하 상태에서 종료시한 내에 처리되는 이벤트들의 수가 최대화될 수 있도록 우선순위 기반 그룹 태스크 스케쥴링 정책이 제시된다. 데이터 일관성과 실시간적 특성을 모두 고려하여 클라이언트 이벤트들은 서버에서 우선순위가 설정되며 우선순위별로 그룹화된다. 서버는 한정된 CPU 대역폭을 우선순위가 높은 그룹일수록 더 많이 할당함으로써 종료시한 내에 처리되는 이벤트들의 수를 최대화하며 전체 시스템의 실시간적 성능을 향상시킬 수 있다.

본 논문은 2 장에서 다중사용자용 실시간 게임 클라이언트/서버 구조와 이벤트의 특성이 기술되며 3 장에서 우선순위별로 그룹화된 이벤트들을 실시간으로 스케쥴링하는 정책이 기술된다. 그리고 4 장에서 시뮬레이션을 통하여 다른 기법과 비교된 성능을 분석하며 5 장에서는 결론을 기술한다.

Ⅱ. 게임 클라이언트/서버 구조와 이벤트

1. 다중사용자용 실시간 게임 클라이언트/서버

그림 1은 다수의 클라이언트들과 게임 서버가 고속의 유선 또는 무선 통신망을 통하여 연결된 구조를 나타내고 있다[7,8]. 게임 서버는 클라이언트의 이벤트를 실시간으로 스케쥴링하며 자원 관리 서버와도 상호 작용을 한다. 자원 관리 서버는 게임 서버가 요청한 작업들을 데이터베이스 서버에 즉시 저장할 것인지 또는 나중에 저장할 것인지를 결정하게 되며 나중에 저장할 경우를 대비해서 지역 저장 장치를 보유하게 된다. 데이터베이스 서버는 게임을 수행하는 클라이언트의 정보와 아이템 재고, 게임 세계의 각종 객체, 게임 기록 등 게임 세계의 현황을 저장한다.

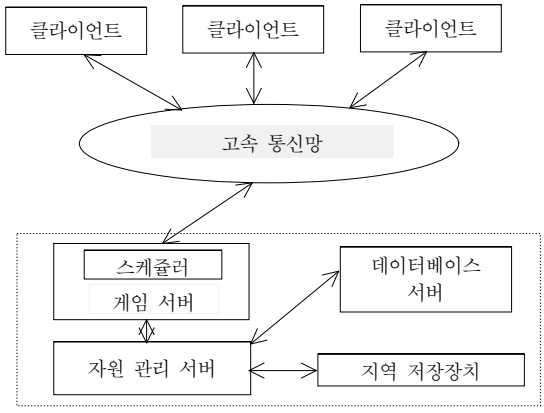

그림 1. 다중사용자용 실시간 게임 클라이언트/서버 구조
Fig. 1. Architecture for a Multiplayer real-time game client-server

2. 게임 클라이언트 이벤트

게임 세계에서 발생하는 이벤트의 본질에 따라 이벤트들은 상이한 일관성 요건을 가지게 된다[9]. 실시간 이벤트는 엄격한 시간 요건을 갖는 반면 다소 느슨한 일관성 요건을 갖게 되며 반대로 일관성 이벤트는 느슨한 시간 요건과 엄격한 일관성 요건을 갖게 된다. 엄격한 시간 요건과 일관성 요건을 모두 가지는 일관성 실시간 이벤트는 가장 중요한 이벤트로 간주된다. 사람이 이벤트에 대한 결과를 감지할 수 있는 시간인 100ms보다 큰 이벤트들은 일관성 이벤트로 분류되며 이보다 작은 시간을 갖으면 실시간 이벤트 또는 일관성 실시간 이벤트로 분류된다[9]. 실제 Quake 게임의[10] 명령어인 이벤트들은 다섯 가지 유형으로 분류된다(표 1 참조).

표 1. Quake 이벤트 분류
Table 1. Event classifications for the Quake

이벤트 유형	이벤트 기술	이벤트 특성
Move	새로운 위치로 아바타 이동	실시간
Fire	아바타가 로켓을 발사	일관성 실시간
Impact	로켓이 충돌후 폭발	일관성 실시간
Damage/die	아바타가 손상되거나 사망	일관성
Spawn	아바타가 다시 생성	일관성

상이한 이벤트들의 요건들은 게임 서버의 CPU 처리를 위한 스케쥴링 과정에서 상이한 우선순위로 설정될 수 있다. 본 논문에서는 일관성 실시간 이벤트인 Quake의 Fire, Impact 이벤트들의 우선순위를 가장 높게 설정하며 실시간 이벤트인 Move 이벤트를 다음 우선순위로 설정한다. 실시간 요건이 없는 일관성 이벤트인 Damage/die와 Spawn 이벤트는 가장 낮은 우선순위로 설정된다.

Ⅲ. 우선순위기반 그룹 태스크 스케쥴링 정책

1. CPU 활용율과 스케쥴링 가능성

게임 서버가 게임 세계를 갱신하는 주기를 T_s라 하며 이 주기는 클라이언트 화면의 재생율에 의하여 결정된다[11]. 예를 들어 1 초 동안 화면이 갱신되는 수를 의미하는 재생율이 25인 경우 T_s는 40ms가 된다. 게임 서버는 T_s 동안 처리되는 평균 태스크의 수 N과 각 태스크의 평균 실행 시간 m을 고려하여 스케쥴링 가능성을 검사한다. 예를 들어 N이 40이고 m이 1 ms인 경우 N×m이 T_s 이하이므로 40개의 태스크들이 처리될 수 있고 이 경우 CPU 활용도는 이론상 최대값인 100%가 된다. 본 논문에서는 태스크 간 CPU switching context에 대한 시간적인 오버헤드는 없는 것으로 가정한다. 그리고 클라이언트가 서버에 요청한 이벤트는 서버에서 독립적인 태스크로 간주되며 각 태스크마다 종료시한이 설정된다.

CPU 활용율이 100% 이하인 상태에서 EDF(Earliest Deadline First) 스케쥴링 정책을[6] 사용할 경우 실시간 태스크들의 종료시한을 거의 대부분 충족할 수 있다. 그러나 게임 서버의 주기 T_s 동안 클라이언트들이 요청한 이벤트 즉 태스크들의 수가 평균 태스크의 수 N을 초과하거나 평균 실행시간 m을 초과하여 게임 서버에 일시적으로 과부하가 발생할 경우 EDF 스케쥴링 정책은 실시간적 성능을 향상시킬 수 없다. 따라서 본 논문에서는 게임 서버의 CPU 활용율이 100%인 상태에서 과부하가 발생할 경우 이를 효과적으로 처리하며 실시간적 성능을 향상시킬 수 있는 우선순위 기반 그룹별 태스크 스케쥴링 정책을 제시한다.

2. 우선순위 기반 그룹 태스크 스케쥴링

표 1에서 기술된 일관성 실시간 이벤트들은 이벤트들 중 우선순위가 가장 높지만 게임 서버가 특정 주기마다 게임 세계를 갱신해야 하는 태스크를 수행해야 하기 때문에 본 논문에서는 Quake 게임 서버가 수행하는 태스크에 가장 높은 우선순위를 설정한다. 즉 서버의 게임 세계를 주기마다 갱신하는 태스크는 우선순위가 가장 높은 1 그룹이며 Fire와 Impact 이벤트들은 두 번째로 우선순위가 높은 2 그룹이 된다. 실시간 이벤트인 Move 이벤트는 다음 우선순위를 갖는 3 그룹으로 설정하고 일관성 이벤트인 Damage/die와 Spawn 이벤트는 우선순위가 가장 낮은 4 그룹으로 설정된다.

본 논문에서는 게임 서버의 주기 T_s와 동일한 CPU 대역폭을 우선순위가 높은 그룹의 태스크부터 CPU를 먼저 사용할 수 있게 하는 PBG(Priority-based Group) 태스크 스케쥴링 알고리즘을 제시하며 이는 그림 2에서 기술된다.

```
1: Bsum=Ts;
2 while ( Bsum > 0)
3: begin
4:  Select_Priority(i);
5:  Select(Taski);
6:  if ( Exec_time(Taski) ≤ Bsum )
7:   begin
8:     Bsum=Bsum-Exec_time(Taski);
9:     Exec_time(Taski)=0;
10:  end
11:  else
12:   begin
13:    Exec_time(Taski)=Exec_time(Taski)-Bsum;
14:     Bsum=0;
15:  end
16: end
```

그림 2. 우선순위 기반 그룹 태스크 스케쥴링 알고리즘
Fig. 2. Task Scheduling Alogorithm for Priority-based Group

그림 2에서 태스크들이 게임 서버의 주기 내에 사용할 수 있는 CPU 대역폭 B_{sum}은 서버의 주기 T_s와 같기 때문에 1행에서 B_{sum}은 T_s로 설정된다. B_{sum}이 0보다 크면(2행) 우선순위가 가장 높은 그룹 번호 i를 결정하며 이 과정은 Select_Priority(i)로 4행에서 기술된다. 이때 그룹의 태스크가 1 개 이상 존재하는 경우에만 해당 그룹의 우선순위가 선택될 수 있다. 가장 높은 우선순위로 선택된 그룹의 태스크들은 EDF(Earliest Deadline First) 원칙에 따라 종료시한이 가장 이른 태스크 $Task_i$가 선택되며 이 과정은 5행에서 Select($Task_i$)로 기술된다. 선택된 $Task_i$의 실제 실행 시간 Exec_time($Task_i$)은 평균 실행 시간 m보다 크거나 작을 수 있다. Exec_time($Task_i$)가 B_{sum}보다 작거나 같을 경우(6행) $Task_i$는 Exec_time ($Task_i$)만큼 CPU 대역폭을 이용하여 실행되고 실행이 종료되면 B_{sum}은 Exec_time($Task_i$)만큼 차감된다(8행). 그리고 $Task_i$는 실행이 종료될 때 Exec_time($Task_i$)은 0이 된다(9행). $Task_i$가 종료되는 시점에서 종료시한 준수 여부가 확인되며 해당 그룹의 큐에서 삭제된다. 한편 Exec_time($Task_i$)가 B_{sum}보다 클 경우(11행) $Task_i$는 B_{sum} 시간만큼 CPU를 사용할 수 있다. $Task_i$는 B_{sum} 동안 CPU를 사용한 이후 Exec_time($Task_i$)는 B_{sum}을 차감한 상태로 다시 설정되며(13행) 다음 주기에 새로 설정된 Exec_time($Task_i$)만큼 CPU를 사용할 수 있게 된다. 0이 된 B_{sum}은(14행) 게임 서버의 다음 주기에 T_s로 다시 설정되며 2행부터 16행까지의 동일한 스케쥴링 과정이 반복 수행된다.

PBG 태스크 스케쥴링 알고리즘에서는 우선순위가 높은 그룹의 태스크들이 CPU 대역폭을 먼저 사용하게 되며 우선순위가 낮을수록 우선순위가 높은 그룹의 태스크들이 사용하고 남은 대역폭을 사용하게 된다.

3. 그룹별 대역폭이 할당된 태스크 스케쥴링

2절에서 제시된 알고리즘과 달리 3절에서는 우선순위 기반 그룹별로 CPU 대역폭이 설정된 상태에서 태스크들이 스케쥴링되는 GB(Group Bandwidth) 태스크 스케쥴링 알고리즘이 제시된다. PBG 태스크 스케쥴링 알고리즘에서는 항상 우선순위가 높은 그룹의 태스크들이 주기 내에 CPU 대역폭을 최대한 사용할 수 있는 반면 GB 알고리즘에서는 그룹별로 대역폭이 설정되어 있기 때문에 우선순위가 낮은 그룹의 태스크들도 할당된 CPU 대역폭을 주기 내에 사용할 수 있다.

주기 T_s 중 게임 세계를 갱신하는 서버의 태스크 즉 1 그룹이 사용하는 CPU 대역폭은 B_1으로 기술된다. 그룹의 수는 n(= 4)이며 우선순위에 따라 2 그룹, 3 그룹, 4 그룹이 사용하는 CPU 대역폭은 각각 B_2, B_3, B_4로 기술되며 각 그룹이 사용하는 대역폭의 합은 T_s가 된다.

$$T_s = \sum_{i=1}^{n} B_i \quad (1)$$

T_s동안 생성되는 전체 태스크의 수를 N이라 할 때 그룹 i의 태스크 수는 N_i로 표기되며 각 그룹의 CPU 대역폭은 수식 (2)와 같이 산정된다.

$$B_i = \frac{N_i}{N} \times T_s \quad (2)$$

본 논문에서 그룹 간 공평한 CPU 대역폭을 할당하기 위하여 각 그룹의 대역폭은 전체 태스크들 중 해당 그룹의 태스크들이 차지하는 비율에 의하여 결정되는 것을 가정한다.

GB 태스크 스케쥴링 알고리즘이 기술된 그림 3에서 태스크들이 존재하는 그룹 중 우선순위가 가장 높은 i를 결정하는 과정은 Select_Priority(i)로 기술되며(1행) 그림 2의 4행과 동일하다. 해당 그룹 i에 할당된 CPU 대역폭 B_i가 0보다 클 경우(2행) 그룹에서 EDF 정책에 따라 종료시한이 가장 이른 태스크 $Task_i$를 결정하게 되며(4행) 이 과정은 그림 2의 5행과 동일하다. B_i가 0일 경우 해당 그룹의 대역폭이 모두 사용된 것을 의미하기 때문에 이 그룹 내의 태스크들은 게임 서버의 다음 주기에 실행될 수 있다.

```
1: Select_Priority(i);
2 while ( Bi > 0)
3: begin
4:   Select(Taski);
5:   if ( Exec_time(Taski) ≤ Bi )
6:    begin
7:      Bi=Bi-Exec_time(Taski);
8:      Exec_time(Taski)=0;
9:    end
10:  else
11:   begin
12:     Exec_time(Taski)=Exec_time(Taski)-Bi;
13:     Bi=0;
14:   end
15: end
```

그림 3. 그룹별 대역폭이 있는 태스크 스케줄링 알고리즘
Fig. 3. Scheduling Alogorithm for Group Bandwidth

$Exec_time(Task_i)$가 B_i보다 작거나 같을 경우(5행) $Task_i$는 $Exec\_time(Task_i)$ 시간만큼 CPU를 이용하여 실행될 수 있다. 실행이 종료되면 B_i는 $Exec_time(Task_i)$을 차감한 값으로 다시 설정되며(7행) $Exec_time(Task_i)$는 0이 된다(8행). 이때 $Exec_time(Task_i)$가 0이 되는 $Task_i$는 종료시한 준수 여부가 확인된 후 해당 그룹의 큐에서 삭제된다.

$Exec_time(Task_i)$가 B_i보다 클 경우(10행) $Task_i$는 B_i만큼 CPU를 이용하게 되며 $Exec_time(Task_i)$는 B_i를 차감한 값으로 다시 설정된다(12행). 이때 $Task_i$는 서버의 다음 주기에 해당 그룹의 대역폭을 이용하여 남은 실행시간 $Exec_time(Task_i)$만큼 CPU를 사용하게 된다. 해당 그룹의 대역폭 B_i는 0이 되므로(13행) 그룹 내 다른 태스크들도 서버의 다음 주기에 그룹 대역폭을 할당받은 후 실행될 수 있다. 이와 같이 그림 3의 GB 태스크 스케쥴링 알고리즘에서는 우선순위가 가장 높은 그룹의 태스크들이 우선순위가 낮은 그룹의 태스크들보다 먼저 실행되지만 그룹에 할당된 대역폭만 사용할 수 있는 특성을 가지고 있다.

IV. 성능 분석

1. 실험 환경과 변수

본 논문에서는 Quake 게임 서버가 초당 게임 세계를 25회씩 갱신하는 것을 가정하므로 재생율은 25로 설정되며 이에 대한 갱신 주기 T_s는 40ms가 된다. 클라이언트가 발생한 이벤트는 게임 서버에서 태스크로 전환되며 각 태스크의 평균 실행 시간은 1ms이며 최소값 0.5ms와 최대값 1.5ms 분포 범위에서 실제 실행 시간이 설정된다. 서버의 갱신 주기 40ms마다 실행될 수 있는 평균 태스크의 수 N은 각 태스크의 평균 실행시간 1ms를 고려할 때 40이 되며 이때 CPU 활용율은 최대값인 100%가 된다. 본 논문에서는 CPU 활용율을 100%로 유지하기 위하여 325명의 클라이언트들이 매초 평균 3 개의 이벤트를 발생하는 것으로 가정하였다. 게임 서버는 1초 즉 1000ms마다 325명의 클라이언트들이 발생시키는 975(=325×3)개의 태스크와 게임 세계를 갱신하는 서버 태스크 25개를 합한 1000개의 태스크를 스케쥴링하게 된다. 이는 서버가 주기 40ms마다 평균 실행시간이 1ms인 40개의 태스크들을 처리하는 것을 의미하며 이때 CPU 활용율은 100%가 된다.

40개의 태스크들 N_1의 비율은 2.5%이며 N_2, N_3, N_4는 각각 19.5%, 39%, 39%로 가정함에 따라 각 그룹의 평균 태스크 수는 다음과 같이 설정된다.

$$N_1=1,\ N_2=7.8,\ N_3=15.6,\ N_4=15.6$$

이에 따라 GB 알고리즘을 위한 그룹별 CPU 대역폭은 다음과 같이 설정된다.

$$B_1=1.0ms,\ B_2=7.8ms,\ B_3=15.6ms,\ B_4=15.6ms$$

2. 종료시한 성공율

CPU 활용율이 100% 이하인 상태에서는 기존의 실시간 스케줄링 기법인 EDF 정책을 적용할 경우 태스크들의 종료시한 성공율(전체 태스크들 중 종료시한 내에 성공적으로 종료된 태스크들의 백분율)은 100%에 근접한 결과가 나타나며 실제 실험에서도 이러한 결과를 확인하였다. 본 논문에서는 종료시한 성공율을 성능 평가의 주요 지표로 설정하였다.

게임 서버의 CPU 활용율이 100%인 상태에서 클라이언트의 수가 증가하거나 각 태스크의 실행 시간이 평균 시간보다 증가할 수 있으며 또한 이벤트 발생 시간 간격이 감소하여 일시적인 과부하가 발생할 수 있다. 이 경우 기존의 EDF 정책만으로는 실시간적 성능 향상을 기대할 수 없다. 본 논문에서는 클라이언트의 이벤트 평균 발생 시간 간격을 감소시켜 초당 발생되는 이벤트 수를 증가시켰으며 게임 서버가 주기마다 처리해야 하는 태스크의 수를 증가시킨 상황에서 종료시한 성공율을 측정하였다.

그림 4에서 각 클라이언트가 발생하는 평균 이벤트 수는 초당 3 개에서 3.3 개로 10% 증가되었으며 게임 서버의 CPU 활용율이 이미 100%인 상태에서 10% 정도의 과부하가 발생한 5초 동안 스케줄링한 결과를 1초 간격으로 나타내고 있다. 실험에서 그룹 1의 태스크는 모든 스케줄링 알고리즘에서 항상 종료시한 내에 종료되기 때문에 그룹 1을 제외한 그룹 2, 3, 4의 태스크들을 대상으로 종료시한 성공율을 비교하고 있다. EDF의 경우 우선순위를 고려하지 않기 때문에 처음 1초 동안은 75.70%의 높은 종료시한 성공율이 나타나지만 시간이 지날수록 종료시한이 초과되어 종료되는 태스크들의 수가 급증하기 때문에 종료시한 성공율이 급격히 감소하는 것으로 분석되었다. 5초가 되었을 때 EDF의 종료시한 성공률은 약 15.15%가 나타났다. 그룹별로 대역폭을 할당하지 않은 PBG 알고리즘은 우선순위가 높은 그룹의 태스크들이 대부분 종료시한 내에 종료되기 때문에 항상 종료시한 성공율이 60%를 초과하는 것으로 나타났다. 실험에서 실제 그룹 2와 그룹 3의 태스크들은 그룹 1을 제외한 전체 태스크들 중 평균 60%로 구성되며 그룹 4의 태스크들이 나머지 40%로 구성된다.

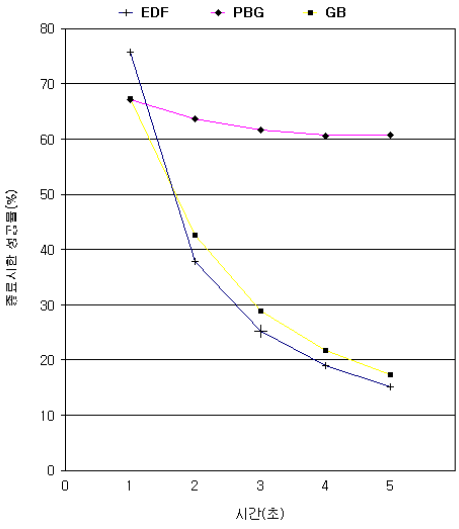

그림 4. 태스크들의 종료시한 성공율(10% 과부하)
Fig. 4. Ratio of tasks completed successfully within their deadlines(10% overloading)

그룹에 대역폭이 할당된 GB 알고리즘은 처음 1초 동안은 EDF 알고리즘에 비하여 종료시한 성공율이 낮게 나타났지만 이후에는 약 2-3% 정도 더 높은 결과가 나타났다. 그림 5는 5초가 되었을 때 그룹 1을 제외한 각 그룹에 대한 종료시한 성공율을 비교하고 있다. 그룹 1의 태스크는 종료시한 성공율이 모든 스케줄링 알고리즘에서 100%이다. EDF 알고리즘의 경우 우선순위를 고려하지 않았기 때문에 그룹 2, 3, 4의 종료시한 성공율은 14.65%, 15.05%, 15.50%로 낮게 나타났으며 그룹간 차이가 거의 없다. PBG 알고리즘은 우선순위가 높은 그룹 2와 3은 100.0%와 99.95%로 매우 높은 반면 우선순위가 낮은 그룹 4는 5.82%로 매우 낮게 나타났다. 우선순위가 높은 그룹 2와 3은 전체 태스크들 중 60%의 비율로 구성되며 항상 CPU 대역폭을 충분히 사용할 수 있는 반면 우선순위가 낮은 4 그룹은 그룹 2와 3이 사용하고 남는 대역폭을 사용하기 때문에 종료시한 성공율이 낮은 것으로 분석되었다. 그룹별 대역폭이 할당된 GB 알고리즘에서는 그룹 2의 종료시한 성공율이 38.31%로 EDF보다는 높고 PBG보다는 낮게 나타났으며 그룹 3의 종료시한 성공율은 14.72%로 EDF보다 0.33%정도 낮은 것으로 나타났다. 그룹 4의 종료시한 성공율은 10.07%로 EDF보다는 4.43% 낮지만 그룹에 할당된 대역폭을 사용할 수 있기 때문에 PGB보다는 4.25% 높은 것으로 나타났다.

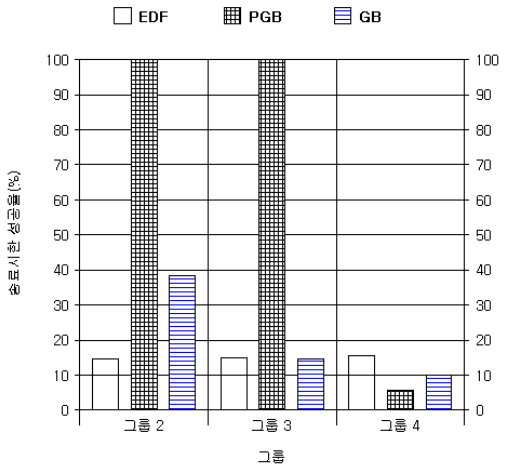

그림 5. 우선순위 기반 그룹의 종료시한 성공율
Fig. 5. Ratio of priority-based group tasks completed successfully within their deadlines(10% overloading)

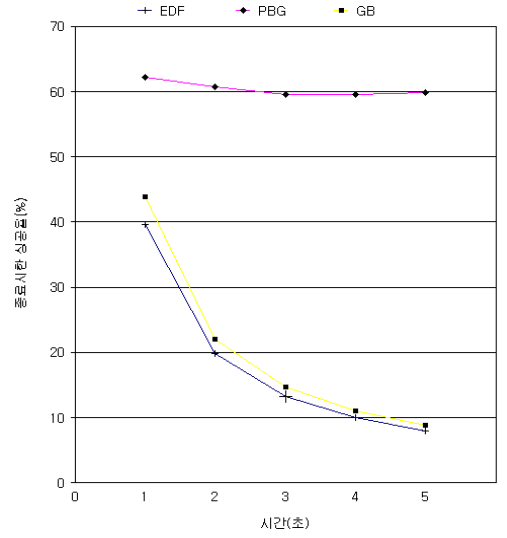

그림 6. 태스크들의 종료시한 성공률(20% 과부하)
Fig. 6. Ratio of tasks completed successfully within their deadlines(20% overloading)

그림 6은 클라이언트의 평균 이벤트 수가 초당 3.6 개로 초당 3 개에서 20% 증가된 상태에서 게임 서버가 5초 동안 스케쥴링한 결과를 1초 간격으로 나타내고 있다. EDF 알고리즘은 평균 20% 과부하가 발생할 경우 종료시한 성공율이 1초간 40% 미만이며 5초가 되면 7.95%로 급격히 감소되는 결과가 나타났다. GB 알고리즘은 EDF 알고리즘보다 처음 1초간 2% 정도 높게 나타났고 5초일 경우 0.44% 높게 나타나서 시간이 지남에 따라 종료시한 성공율의 차이가 감소된 결과가 나타났다. PBG 알고리즘은 우선순위가 높은 그룹 2와 3의 태스크들이 전체 태스크들 중 60%로 구성되며 항상 CPU 대역폭을 충분히 사용하기 때문에 종료시한 성공율이 60%를 초과하거나 근접하는 것으로 분석되었다. 게임 서버에 20% 과부하가 발생한 경우에 대한 각 그룹의 종료시한 성공율은 그림 5와 거의 유사한 것으로 나타나 본 논문에서 생략하기로 한다.

V. 결론

클라이언트들이 발생시키는 이벤트의 실시간성과 일관성을 고려하여 우선순위가 설정되었으며 우선순위에 기반한 그룹 태스크 스케쥴링 정책이 본 논문에서 제시되었다. 게임 서버의 CPU 활용율이 100% 이하일때는 기존의 EDF 스케쥴링 정책을 적용하여도 시스템의 실시간적 성능을 향상시킬 수 있다. 그러나 클라이언트 수의 증가, 이벤트 처리를 위한 평균 실행 시간의 증가, 이벤트 발생 간격 시간의 감소 등으로 인하여 게임 서버에 일시적인 과부하가 발생할 경우 EDF 스케쥴링 정책만으로는 시스템의 실시간적 성능을 향상시키기 매우 어렵다.

본 논문에서는 게임 서버의 CPU 활용율이 100%인 상태에서 과부하가 발생할 경우 우선순위 기반 그룹 태스크들을 효율적으로 스케쥴링할 수 있는 정책을 제시하였으며 시스템의 실시간적 성능을 향상시킬 수 있는 실험결과를 확인하였다. 게임 서버에 일시적인 과부하가 발생할 경우에도 기존의 EDF 알고리즘보다 전체 태스크들 중 종료시한 내에 성공적으로 종료되는 태스크들의 비율을 증가시킬 수 있고 우선순위가 높은 그룹일수록 이 비율이 높게 나타나는 스케쥴링 정책이 본 논문에서 제시되었다. 온라인으로 수행되는 다중사용자 게임에서 일시적인 과부하가 발생할 경우 본 논문의 스케쥴링 정책이 적용되면 클라이언트의 게임 수행에 필요한 이벤트의 실시간적 요건을 더욱 충족시킬 수 있을 것으로 기대된다.

참 고 문 헌

[1] Y. W. Ahn, A. M. K. Cheng, J. Baek, and P. S.

Fisher, "A Multiplayer Real-Time Game Protocol Architecture for Reducing Network Latency," IEEE Transactions on Consumer Electronics 55(4), pp. 1883 - 1889, Nov. 2009.

[2] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, RTP: A Transport Protocol for Real-time Applications. Internet Engineering Task Force, RFC 1889, 1996.

[3] Y. E. Liu, J. Wang, M. Kwok, J. Diamond, and M. Toulouse, "Capability of IEEE 802.11g Networks in Supporting Multi-player Online Games," Consumer Communications and Networking Conference, pp. 1193-1198, Jan. 2006.

[4] S. Harcsik, A. Petlund, C. Griwods, and P. Halvorsen, "Latency Evaluation of Networking Mechanisms for Game Traffic," 6th Workshop on Networks and System Support for Games, pp. 129-134, Sep. 2007.

[5] P. Koutsakis, M. Vafiadis, and A. Lazaris, "A New Bandwidth Allocation Mechanism for Next Generation Wireless Cellular Networks," Wireless Network 16, pp. 331-353, 2010.

[6] C. L. Liu and J. Layland, "Scheduling algorithms for multiprogramming in a hard real-time environment", Journal of the ACM 2(4), 1973.

[7] A. Shaikh, S. Sahu, M. Rosu, M. Shea, and D. Saha, "Implementation of a Service Platform for Online Games," ACM Special Interest Group on Data Communication, Aug. 2004.

[8] E. Cronin, B. Filstrup, and A. Kurc, "A Distributed Multiplayer Game Server System," UM EECS589 Course Project Report, 2001.

[9] J. Yuen, K. Y. Lam, and E. Chan, "A Fair and Adaptive Scheduling Protocol for Video Stream Transmission in Mobile Environment,"IEEE International Conference on Multimedia and Expo, Aug. 2002.

[10] A. Hsu, J. Ling, and Q. Li, and C. C. Jay Kuo, "On the design of Multiplayer On-line Video Game Systems," SPIE ITCOM, Sep. 2003.

[11] Doom, Quake, ID Software, Inc. http://www.idsoftware. com

[12] A. Abdelkhalek, A. Bilas, and A. Moshovos, "Behavior and Performance of Interactive Multi-player Game Servers," IEEE Int'l Symp. on Performance Analysis of Systems and Software, Nov. 2001.

저자 소개

김 진 환(정회원)

- 1986년 : 서울대학교 컴퓨터공학과 졸업(학사)
- 1988년 : 서울대학교 컴퓨터공학과 졸업(석사)
- 1994년 : 서울대학교 컴퓨터공학과 졸업(박사)
- 1994년 ~ 1995년 : 서울대학교 컴퓨터신기술공동연구소 특별연구원
- 1995년 ~ 현재 : 한성대학교 멀티미디어공학과 교수

<주관심분야 : 멀티미디어 시스템, 실시간 게임 시스템>

※ 본 연구는 한성대학교 교내연구비 지원과제임.