다수 사용자용 온라인 게임에서의 이중 시간간격을 이용한 상태 동기화

김상철

한국외국어대학교 컴퓨터공학과
kimsa@hufs.ac.kr

State Synchronization Using Double Time Intervals in Multi-user On-line Games

Hankuk University of Foreign Studies

Sangchul Kim

Dept. of Computer Science and Engineering, Hankuk University of Foreign Studies

요 약

다수 사용자용 인터넷 게임은 그 특성상 게임 클라이언트간 또는 게임 클라이언트와 게임 서버간에 네트워크 지연은 피할 수 없고 이로 인해서 사용자가 느끼는 실감은 상당히 줄어 든다. 본 논문에서는 클라이언트-서버 형태의 게임구조 하에서 이런 네트워크 지연에 따른 문제점을 해결하는 사용자들간의 상태 동기화 방법을 제안한다. 이 방법은 게임상 가상공간을 여러 영역으로 나눈 후, 각 클라이언트에 도착하는 메시지들의 발생 영역에 따라서 배치처리 시간간격 (time interval)을 달리하는 동기화 방법이다. 이 동기화 방법은 기존 방법에 비해서 롤백(roll back) 수의 큰 증가 없이 사용자 반응시간을 현저히 줄이는 효과를 보인다.

1. 서 론

다수 사용자용 온라인 인터넷 게임에서 게임 사용자들이 게임이 제공하는 가상공간내 자신의 행동이 사실적이라고 느끼려면 실시간 상호작용 의 지원이 중요하다. 특히 슈팅 게임, 전쟁 시뮬레이션 게임 및 격투 게임등과 같이 빠른 반응을 필요로 하는 게임에 있어서 실시간 상호작용의 지원은 필수적이라고 할 수 있다[1]. 실시간 상호작용을 가로막는 주된 이유는 한 게임 싸이트에서 발생한 이벤트 (또는 메시지)들이 인터넷을 통해서 다른 사용자 측에 전달되는 과정에서 발생하는 네트워크 지연이 주된 원인이라 할 수 있다.

본 논문에서는 현 인터넷 기술상 불가피하게 발생하는 네트워크 지연에 의한 실시간 상호 작용성을 높이기 위한 상태 동기화 방법을 제안한다. 한 서버와 통신하는 모든 클라이언트들이 일치되는 게임 상태를 공유하도록 하는 작업을 동기화라고 한다. 상호 작용성이 중요한 요소인 온라인 게임에서는 TSS(Trailing State Synchronization)같은 낙관적 방식의 동기화 알고리즘이 적합하다. 우리는 [1]에서 배치방식으로 메시지를 처리하도록 전형적인 TSS 알고리즘을 개선했다. 하지만, 이 방식은 모든 메시지가 즉각 처리되지 않고 일정 시간 (즉, 배치처리 시간간격)만큼 지연되어서 처리되는 단점이 있다. 본 논문에서는 제안하는 방법은 게임사용자 아바타(avatar)에 직접 영향을 미칠 수 있는 가까운 거리에서 발생한 메시지는 그렇지 않는 메시지보다 빨리 처리해주는 특징을 갖는다. 이 알고리즘은 이전 알고리즘에 비해서 거의 비슷한 롤백 수를 보이지만 메시지의 사용자 반응시간을 현저하게 줄여 준다.

본 논문에서의 게임 시스템 구조는 많은 상업적 게임에서 채택하고 있는 [2,3] 클라이언트-서버 방식을 가정한다.

2. 기존 연구

클라이언트 서버 방식의 구조에서는 각 클라이언트의 명령어가 서버에게 처리되므로, 사용자는 자신의 명령어가 즉시 실행되지 못하고 다소 지연되는 느낌을 가질 수 있다. 이런 단점을 극복하고자 클라이언트도 게임 엔진의 전부 또는 일부를 가지고, 사용자 명령어를 직접 실행하도록 하면서 동시에 그 명령어를 서버에게도 전달하는 방식을 택한다. 이런 경우에, 만약 나중에 서버로부터 받은 수정 메시지로부터 자신이 명령어를 수행할 시의 게임 상태가 최신 상태가 아님을 알게 되면, 그 명령어를 수행하기 전의 유효한 게임상태로 롤백하고 그 명령어를 다시 실행하게 된다 [4]. 이와 같이 클라이언트 들간의 동기화를 유지하는 방법을 낙관적(optimistic) 동기화라고 한다.

낙관적 동기화 방법에서는 주기적으로 게임 상태의 스냅샷 (snapshot)을 저장한다 [5]. 이것은 많은 메모리 공간을 차지하므로 [2]은 스냅샵을 이용하지 않고 게임 상태를 여러 카피(또는 상태)로 유지하는 방법을 제안했다. 각 카피는 모든 이벤트를 도착순으로 처리하고 카피들 간에는 현재 시뮬레이션 타임과 비교해서 서로 다른 지연을 갖는다. 하지만 메시지를 도착순으로 처리함으로써 메시지들의 도착순이 메시지의 발생순과 다를 경우에 롤백이 발생하는 단점이 있다.

위에서 언급한 문제를 해결하고자, 우리는 전체 시뮬레이션 타임을 일정한 간격으로 나누고 각 상태는 그 중 특정 구간내의 메시지들을 배치방식으로 처리하는 TSS 기반 동기화 알고리즘을 제안하였다. 하지만 이 방법은 메시지가 일정 시간동안 지연되어 처리되므로 사용자 반응시간이 늘어서 상호작용성의 중요한 요인이 실감인 떨어지는 단점이 있었다.

3. 가상 공간의 분할

게임상의 가상공간을 각 클라이언트의 캐릭터(즉 아바타)가 존재하는 위치를 기준으로 여러 영역으로 나눌 수 있다.우선 각 클라이언트의 입장에서 보여 지는 공간, 즉 가시 영역(visual domain)과 비가시 영역으로 나눌 수 있다. 서버는 비 가시영역에 존재하는 객체에 대한 메시지를 클라이언트들로 전달하지 않는다. 이런 방법을 메시지 필터링이라 부르고 메시지 양을 줄이는 방법 중 하나이다 [6].

본 논문에서는 기존 연구와 달리 가시영역 내를 상호동작 영역과 비 상호동작 영역으로 나눈다. 상호동작 영역이란 클라이언트 객체가 주변 객체들과의 활발한 상호작용 (예를 들면, 격투게임에서 상대방을 향해서 주먹을 날리거나, 슈팅게임에서 상대방에게 총을 발사함)이 일어나는 영역을 말한다. 게임사용자의 다음 동작은 상호동작 영역 내의 객체들의 최근 상태에 따라 결정되는 것이므로, 상호동작 영역 내 객체에 대한 메시지들은 그 외 영역의 메시지들 보다 우선적으로 처리되어야 할 것이다.

4. 복수 배치처리 시간간격을 지원하는 TSS

본 논문에서 제안하는 동기화 방법은 우리의 기존 연구 [1]를 개선한 것으로 기본적인 동작원리는 거의 비슷하다. 시뮬레이션 타임(또는 게임 싸이클 타임)은 일정한 크기의 여러 구간으로 나누어지고, 각 구간은 다시 서브구간들로 나누어진다. 본 논문에서는 각 구간은 2개 서브구간들로 나눈다. 상태(state)는 실행 중인 게임 로직을 나타내며, 한 구간, 한 서브구간 또는 여러 구간들 사이의 메시지들을 배치 방식으로 처리한다. 여러 상태들 사이에는 전통적인 TSS와 같이 일정한 지연이 존재한다. 제일 앞선 상태를 선도 상태라고 부르고, 이 상태는 매 서브구간 시작시점에서 메시지를 처리하고, 다른 모든 상태들은 매 구간 시작시점에서 메시지를 처리한다. 선도 상태 바로 뒤를 차기 선도 상태라고 부른다. 선도 상태와 차기 선도 상태 간에는 시뮬레이션 시점에 따라서 한 구간 또는 한 서브구간 만큼의 지연이 있다.

시뮬레이션 사이클이 수행되는 시점을 시뮬레이션 시점이라 부르는데, 시뮬레이션 시점은 구간 또는 서브구간의 시작이다. 시뮬레이션 시점 T에 대해서, p(T, N)는 T 보다 N개 서브구간만큼의 앞 시점을 나타내고, s(T, N)은 T 보다 N개 서브구간만큼의 뒷 시점을 나타내고, d(SI, SJ)은 상태 SJ가 상태 SI보다 몇 서브구간 만큼 지연된 지를 나타낸다. 현 시뮬레이션 사이클에서 각 상태는 다음 같이 동작한다: 상태의 시점을 t라고 가정한다:

```
if(선도상태) {
  [p(t,1), t] 내 처리되지 않은 메시지 처리.
  [t, s(t,1)] 내 상호작용 영역내의 메시지 처리.
}
else
if(시뮬레이션 시점이 구간의 시작시점)
  [t, s(t,2)] 내의 메시지 처리.
```

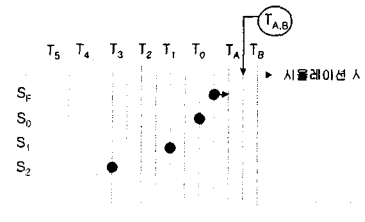

그림 1 동기화 알고리즘의 실행 예제 (시뮬레이션 시점= TA,B)

다음 시뮬레이션 시점을 기다리는 동안 지각 메시지가 도착할 수 있다. 이 경우에 다음 시뮬레이션을 시작하기 직전에 롤백이 수행된다. 롤백은 먼저 지각 메시지들 중에서 타임스탬프가 가장 빠른 메시지(예를 들면, S)를 찾은 후 다음 두 단계를 차례로 수행한다. 메시지 S의 타임스탬프 바로 뒤의 상태를 Sx,, 현 시뮬레이션 시점을 T라고 하자.

1단계: SX가 SX-1, SX-2, , S0, SF 로 각각 복사된다. 즉, SX보다 앞선 상태들은 SX로 돌아간다.

2단계: SF는 p(T,1) 시점까지 실행되고, S0은 p(T, 1+d(SF S0)) 시점까지 실행되고, S1은 p(T, 1+d(SF S0)+d(S0, S1)) 시점까지 실행되고, , SX는 p(T,1+d(S0,S1)+d(S1,S2)++d(SX-1,SX) 시점까지 실행된다. 여기서, T가 구간의 시작시점이면 d(SF, S0) = 1, 서브구간의 시작이면 d(SF, S0) = 2이다.

5. 실험 및 분석

시뮬레이션 구간의 크기는 각 게임별로 실험적으로 정해 질 것이다. 최대 사용자 반응 시간인 R은 다음과 같다: R = Ri + (1-) Rni, Ri = TI, Rni = 2TI + 2. 여기서, Ri는 상호작용 영역내 메시지의 최대 사용자 반응시간이고, Rni는 그 외 메시지의 최대 사용자 반응시간이다. TI는 한 구간의 길이이고, 는 한 서브구간내의 메시지를 실행하는데 걸리는 시간이다.

다음 표는 본 논문에서 제안한 동기화 방법을 [4]와 롤백 비율 및 평균사용자 반응시간 면에서 비교한 결과이다. VA 는 롤백 비율을, VB 는 평균 사용자 반응시간 비율을 나타낸다. 메시지들의 발생 시간은 기존 트래픽 모델 [7]을 따랐다.

NO 비고	조건	VA		VB
		제안된 방법	기존연구	
1	TI = 50msec, d(S0,S1)=2	1.95%	1.90%	85%
2	TI = 50msec, d(S0,S1)=4,	1.91%	1.90%	77%
3	TI = 50msec, d(S0,S1)=2, d(S1,S2)=2	1.50%	1.45%	89%
4	TI = 50msec, d(S0,S1)=2, d(S1,S2)=2 d(S2,S3)=8	1.34%	1.28%	92%
비고	초기 d(SF, S0)=1			

6. 결론 및 추후연구

빠른 사용자 반응을 요구하는 인터넷 게임의 사용자 실감을 높이려면 네트워크 지연에 따른 문제점을 최소화하여야 한다. 우리는 클라이언트-서버 형태의 게임구조하에서 이런 네트워크 지연을 줄이는 방법을 제안했다. 이것은 각 클라이언트에 도착하는 메시지들의 발생 영역에 따라서 배치처리 시간간격을 달리하는 새로운 동기화 방법이다. 실험에 따르면, 우리의 동기화 방법은 기존 방법에 비해서 롤백 수의 큰 증가 없이 사용자 반응시간을 현저히 줄이는 결과를 보였다.

참고문헌

[1] 김상철, 네트워크형 다수 사용자용 게임의 구조 및 동기화 방법, 정보산업공학과 논문집 제 8집, 2003.

[2] Eric, Burton, et. Al, "An Efficient Synchronization Mechanism for Mirrored Game Architecture," ACM NetGames 2002, p67-73.

[3] Matin Mauve, Stefan Fisher, Jorg Widmer, A Generic Proxy System for Networked Computer Games, ACM NetGames 2002.

[4] Johnathan Blow, "A Look at Latency in Networked Games," Game Developer, July 1998.

[5] M. Mauve, "How to keep a dead man from shooting," the 7th International Workshop on Interactive Distributed Multimedia Systems, October 2000.

[6] L. Zou, et. al,"An Evaluation of Grouping Techniques for State Dissemination in Networked Multi-User Games," Proc. Of MASCOT 2001.

[7] Johannes Farber, "Network Game Traffic Modeling," ACM NetGames 2002.