가상 환경에서 가변적인 네트웍 트래픽 상태를 고려하는 동기화 기법

홍의숙[0] 이동만 박은광 강경란
한 국 정 보 통 신 대 학 원 대 학 교
(silkrain[0], dlee, pek, krkang)@icu.ac.kr

An Efficient Synchronization Mechanism Adapting to Dynamic Network Traffic State in Networked Virtual Environments

Eui-Suk Hong[0], Dong-Man Lee, Eun-kwang Park, Kyung-ran Kang
Information and Communication University

요 약

실시간으로 상호 작용하는 어플리케이션에서 성능을 유지하면서 모든 참가자들에게 동일한 뷰를 보여주는 것이 중요하다. 동일한 뷰를 제공하기 위해 사용하는 기법 중 하나로, 참가자들은 받은 이벤트를 실행 시각까지 버퍼에 저장하고 모든 참가자들이 동시에 이벤트를 실행한다. 그러나 네트웍상에서 전송 지연 시간 때문에 실행 예정 시각이 지나서 수신된 이벤트 들은 손실된 이벤트로 처리되어 참가자들 간의 일치하지 않는 뷰를 초래하며 손실된 이벤트를 복구 하기 위한 오버헤드가 추가적으로 필요하게 된다. 본 논문에서 제안한 동기화 기법은 네트웍 트래픽 상태에 따라 재생 지연 시간을 동적으로 정합으로써 네트웍 트래픽이 감소하여 전송 지연 시간이 짧을 경우, 작은 재생 지연 시간을 적용하여 상호 작용 성능을 높여주고 네트웍상에 트래픽이 증가하는 경우에는 재생 지연 시간을 늘림으로 상호 작용 성능을 해치지 않는 범위에서 이벤트 손실율을 줄이고자 한다. 실험을 통해 이 기법이 효과적으로 작동하는지 알아본다.

1. 서론[1]

네트웍 가상 환경은 지리적으로 서로 떨어져 있는 참가자들이 동일한 뷰(view)를 가지고 실시간으로 상호 작용하는 시스템이다.[2] 지역적으로 참가자가 분산되어 있는 대규모 네트웍에서는 참가자 사이트에 참가자의 로컬 정보를 복제하여 둠으로써 상호 작용 성능을 높일 수 있다[1,8]. 하지만 각 참가자 사이트에서 뷰의 갱신이 일어날 경우 필요한 이벤트들이 전송되고 이때 모든 참가자들이 동일한 뷰를 갖도록 하기 위해 이벤트 실행에 대한 동기화가 필수적이고 동기화로 인한 상호 작용 성능의 저하를 초래해선 안 된다.

네트웍의 규모가 커지면서 전송 지연 시간이 다양해지고 서로 다른 참가자의 처리 능력으로 인하여 일치하지 않는 뷰를 가진 참가자들이 많아지게 되었다. 따라서 가변적으로 변하는 네트웍에서 이벤트 실행에 대한 동기화는 가상 환경에 있는 참가자들에게 동일한 뷰를 제공하기 위해 유지되어야 할 중요한 요소이다. [9]

MiMaze[3,4]와 긍정적 접근 방법(Optimistic approach)[6,7]에서는 뷰의 동기화를 위해 모든 참가자가 공통으로 사용하는 "재생 지연 시간"(Playout delay)을 설정한다. 수신자들은 "재생 시각"(playout time)보다 먼저 수신된 이벤트들을 자신의 시스템에 저장하여 모든 참가자가 동시에 주어진 이벤트를 처리할 수 있도록 한다. 하지만, "재생 지연 시간"(playout delay)이 고정적이기 때문에 가변적인 네트웍 트래픽 상태에서 전송 지연 시간이 짧아진 경우에는 수신자 시스템에서 저장되는 시간이 길어지게 된다. 반면, 네트웍 상에 트래픽이 많아져 전송 지연 시간이 길어지면 재생 시각 후에 수신되어 버려지는 이벤트의 수가 많아져 참가자들에게 서로 일치하지 않는 뷰를 제공하는 원인이 된다.

본 연구에서는 다양하게 변하는 네트웍 트래픽 상태를 고려하여 재생 지연 시간을 적용하는 기법을 제안한다. 네트웍 트래픽 상태는 여러 참가자의 이벤트 손실율(event loss rate)을 사용하여 측정된다. 가변적인 네트웍 트래픽 상태에 따라 재생 지연 시간을 조절함으로써 상호 작용 성능을 저하시키지 않으면서 참가자들에게 동일한 뷰를 제공해 줄 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 참가자에게 동일한 뷰를 제공하기 위한 관련 연구를 소개하고 3장에서는 다양하게 변하는 네트웍 트래픽 상태를 고려한 동기화 기법을 제안한다. 4장에서는 제안된 동기화 기법의 실험 결과를 보이고 5장에서 결론을 말하고자 한다.

2. 관련연구

동일한 뷰를 제공해 주기 위해서 MiMaze[3,4]는 버킷 동기화(*bucket synchronization*)와 추측항법(*dead-reckoning*)을 제안하였다. MiMaze는 송신자가 이벤트를 발생시킨 때부터 수신자에서 이벤트가 실행될 때까지 걸리는 시간을 재생 지연 시간이라 정의하고 전송되는 모든 이벤트에 대해서 고정적인 재생 지연 시간을 설정한다. 참가자들은 설정된 재생 지연 시간을 이용하여 계산된 재생 시각에 자신이 수신한 이벤트를 실행함으로써 동기화 된다. 수신자는 재생 시각 이후에 수신한 이벤트를 손실된 이벤트로 간주하고 이전에 수신한 이벤트는 재생 시각까지 버킷에 저장한다. 기본적으로 모든 참가자는 글로벌 클럭을 사용하기 때문에 같은 그룹에 속해 있는 모든 참가자들은 동일한 시각에 하나의 이벤트를 처리하게 되는 것이다.

[1] 본 논문은 한국과학재단 가상현실 연구센터의 지원으로 수행하였습니다.

MiMaze에서 네트웍 로드를 줄이기 위해 추측항법[5]을 사용하였다. 송신자는 아바타의 움직임과 같이 연속적으로 발생하는 이벤트에 대한 정보를 일정한 주기마다 보내며 수신자 측에서 아바타의 움직임 정보를 추측하여 처리하는 기법이다. 추측항법을 이용하여 네트웍 로드를 줄일 수 있지만 이 기법은 이벤트 수신자가 그 정보를 추측할 수 있을 때만 사용 가능하다는 단점이 있다.

긍정적 접근 방법[6,7]에서는 동일한 뷰를 제공하기 위해 MiMaze의 버킷 동기화 기법과 비슷한 개념인 *local lag* 기법을 사용한다. 각 참가자는 *local lag* 시각이 될 때까지 수신한 이벤트를 저장하고 *local lag* 시각에 이벤트를 처리하게 된다. 긍정적 접근 방법에서도 글로벌 클럭을 사용하여 참가자간의 같은 이벤트 처리를 보장한다. 긍정적 접근에서는 늦게 도착한 이벤트를 처리하기 위해 수신한 이벤트와 현재 뷰의 관련성을 조사하여 늦게 도착한 이벤트 중 선택적으로 처리하는 시간 왜곡(*timewarp*) 기법을 사용한다. 하지만 시간 왜곡 기법을 사용하기 위해 모든 참가자는 일정 시간 동안의 뷰의 변화된 사항을 저장해야 한다. 시간 왜곡 기법을 이용해서 처리할 수 없는 이벤트는 지정된 서버에 명시적으로 요청하여 다른 참가자와의 동일한 뷰를 갖도록 처리한다.

3. 가변적인 네트웍 트래픽 상태를 고려하는 동기화 기법

3.1 일관성 vs. 상호 작용 성능

네트웍 게임과 같이 상호 작용하는 시스템은 참가자 간 동일한 뷰를 가져야 하고 동시에 참가자가 만족할 만한 상호 작용 성능을 제공해야 한다. 하지만 이러한 두 가지 고려 사항은 서로 상충하는 관계를 가진다. 예를들어, 재생 지연 시간을 작게 잡는다면 상호 작용 성능은 항상 시킬 수 있지만 참가자들은 많은 이벤트를 재생 시각 후에 수신하여 손실되는 이벤트가 증가하게 된다. 그리고 재생 지연 시간을 길게 잡는다면 재생 시각 안에 많은 이벤트를 수신하게 되지만 재생 시각까지 수신자 시스템에 저장해야 하기 때문에 상대적으로 상호 작용 성능을 저하시키게 된다. 따라서 모든 참가자들에게 동일한 뷰를 제공해 줄 때 상호 작용 성능을 저하시키지 않으면서 네트웍 트래픽 상태에 맞도록 재생 지연 시간을 정하는 것이 중요하다.

3.2 네트웍 트래픽 상태 측정

모든 참가자는 네트웍 트래픽 상태를 고려하여 재생 지연 시간을 정하기 위해 동일한 시간 간격을 가지고 네트웍 트래픽 상태를 판단한다. 송신자는 수식 (1)을 이용하여 매 시간 간격마다 자신이 수신한 이벤트에 대한 손실율을 계산하고 이를 자신이 보내는 이벤트에 포함하여 전송한다. 이러한 이벤트들을 "피기백 이벤트"라 정의한다.

$$R_{k-1}(J)=\frac{N_{lost}+N_{late}}{N_{total}} \qquad (1)$$

수식 (1)에서 N_{lost}는 송신자 J가 (*k-2*)번째 시간 간격 동안 네트웍 상에서 손실된 이벤트의 수이고 N_{late}는 재생 지연 시간이 지난 후 수신된 이벤트의 수이다. N_{total}는 N_{lost}와 N_{late}의 합이다. 네트웍 상에서 발생할 수 있는 이벤트 손실을 고려하여 연속된 두 개의 이벤트에 포함시키고 만일, 모든 피기백 이벤트가 네트웍에서 손실되었다면 이전 이벤트 손실율을 사용한다.

참가자들은 다른 송신자들의 이벤트 손실율을 얻게 되고 그룹 내의 네트웍 트래픽 상태를 측정하기 위해 자신이 수신한 다른 송신자들의 이벤트 손실율의 평균을 사용한다. 따라서 수신자 A가 (*k-1*)번째 시간 간격에서 네트웍 트래픽 상태를 의미하는 $L_{k-1}(A)$는 수식 (2)를 통해 계산된다.

$$L_{k-1}(A)=\frac{\sum R_{k-1}(J)}{N(J)} \qquad (2)$$

수식 (2)에서 N(J)는 수신자 A가 받은 피기백 이벤트들의 송신자 수이다. 따라서 모든 참가자는 같은 시간 간격에 동일하게 네트웍 트래픽 상태를 판단한다.

3.3 재생 지연 시간 결정

각 참가자들은 (2)에서 계산된 평균 이벤트 손실율을 사용하여 재생 지연 시간을 결정하고 자신이 전송하는 이벤트의 재생 지연 시간을 설정한다. 재생 지연 시간은 MiMaze와 동일한 개념이다. 이벤트 손실율이 를 때에는 현재의 네트웍 트래픽이 증가하여 전송 지연 시간이 길어졌다는 것을 의미하므로 재생 지연 시간을 늘리고 이벤트 손실율이 작을 때에는 재생 지연 시간을 줄이도록 한다.

송신자 A가 *k* 번째 시간 간격에서 적용할 재생 지연 시간 $P_k(A)$의 계산은 다음과 같다.

$L_{K-1}(A) < L$일때,

$$P_k(A)=P_{k-1}(A)-\alpha_1\times(L-L_{k-1}(A)) \text{ 단, } k=2,3,4... \qquad (3)$$

$L_{K-1}(A) \geq L$일때,

$$P_k(A)=P_{k-1}(A)+\alpha_2\times(L_{k-1}(A)-L) \qquad (4)$$

L 값은 네트웍 트래픽 상태를 판단하는 값으로 이 값을 기준으로 재생 지연 시간의 증감을 결정한다. L값을 이용하여 서로 상충 관계인 일관성과 상호 작용 성능 중 다른 하나의 성질을 부각시킬 수 있기 때문에 L값은 어플리케이션의 특성에 따라 달라질 수 있다. α_1, α_2 는 상수 값으로 실험을 통해 보다 효과적인 값으로 정한다.

4. 실험 결과

4.1 실험 모델

우리는 NS-2를 이용한 실험을 통해 가변적인 재생 지연 시간을 사용하는 경우 네트웍 트래픽 상태를 조사하여 각 참가자가 정한 재생 지연 시간이 네트웍 트래픽 상태에 적응하는 것을 살펴 보았다. 그림 1은 NS-2를 사용하여 만든 네트웍 토폴로지를 나타내고 실험에서 사용된 변수는 표 1과 같다.

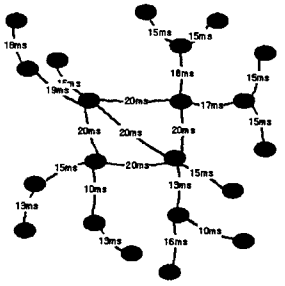

그림 1. 실험에서 사용된 네트웍 토폴로지

표1. 실험에서 사용된 환경 변수

항 목	데 이 터
전체 노드 수	21 개
멀티캐스트 그룹에 속한 노드	0, 1, 2, 3
실험 시간	200 초
실험 횟수	10번
재생 지연 시간 계산을 위한 간격	3초
α_1	20

α_2	50
L	0.05
P_1	150 ms
버킷 크기	20

네트웍 트래픽 상태를 가변적으로 만들기 위해서 노드 4, 5, 9, 11, 13, 15, 16, 19에서 간헐적으로 UDP 트래픽을 발생시키도록 하였다. 10번의 실험을 하여 평균값을 측정하였다.

4.2 전송 지연 시간

네트웍에 전송되는 트래픽의 수가 증가하면서 중간 노드에서 트래픽을 처리하기 위한 지연 시간이 필요하다. 실험에서 만들어지는 네트웍의 트래픽 상태를 판단하기 위해 모든 참가자가 수신한 이벤트의 평균 전송 지연 시간을 측정하였다. 그림 2에서 가로축은 실험 시간을 나타내고 세로축은 전송 지연 시간을 ms의 단위로 나타낸 결과이다. 그림 2가 나타내는 것과 같이 중간 노드들의 UDP 트래픽으로 인하여 50초부터 100초까지 모든 참가자가 수신한 이벤트의 전송 지연 시간이 증가하였다.

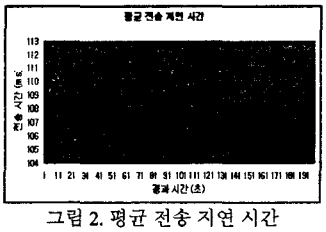

그림 2. 평균 전송 지연 시간

4.3 이벤트 손실율과 재생 지연 시간

그림 3는 고정적 재생 지연 시간을 적용하는 경우와 가변적 재생 지연 시간을 적용하는 경우, 모든 참가자들의 평균 이벤트 손실율을 나타낸다. 여기서의 이벤트 손실율은 전체 이벤트에 대해 네트웍에서 손실되는 이벤트, 재생 지연 시간이 지나서 수신된 이벤트, 그리고 제한된 버킷의 사용으로 인해 버려지는 이벤트들의 비율을 포함한다.

가변적 재생 지연 시간을 적용하는 경우, 약 50초 근처에서 전송 지연 시간이 크게 증가함에 따라 재생 지연 시간 이후 수신되는 이벤트가 증가하여 약 14%의 이벤트 손실율을 보인다. 하지만 다른 시간대에서는 3~4%정도의 비교적 낮은 이벤트 손실율을 보이고 있다.

고정적 재생 지연 시간을 적용하는 경우 가변적 재생 지연 시간을 적용하는 경우에 비해 재생 지연 시간 전에 받는 이벤트의 개수는 많지만 제한된 크기의 버킷 사용으로 인하여 버려지는 이벤트의 수가 많아서 평균 2.6%의 손실율을 보인다.

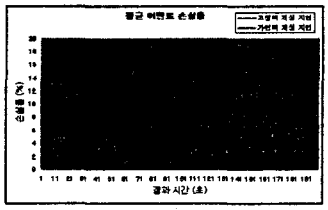

그림 3. 평균 이벤트 손실율

위에서 구한 평균 이벤트 손실율을 이용하여 참가자들이 정한 재생 지연 시간은 그림 4와 같다. 50초 근처에서 다른 시간대보다 높은 이벤트 손실율을 얻고 이에 따라 증가하는 네트웍 트래픽 상태를 반영하기 위해 재생 지연 시간을 늘린다.

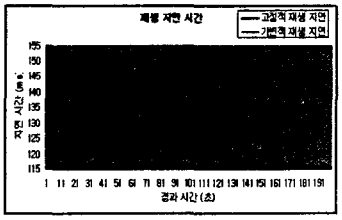

그림 4. 재생 지연 시간

150 ms의 고정적 재생 지연 시간을 적용하는 경우보다 가변적 재생 지연 시간을 적용하는 경우, 평균 2.6%에서 4%로 이벤트 손실율이 증가하였으나 평균 가변적 재생 지연 시간이 약 134 ms로 10% 이상의 감소를 보였다.

5. 결론

참가자들에게 동일한 뷰를 제공해 주기 위해 일정한 재생 시각이 될 때까지 수신자의 시스템에 저장한 후 참가자들끼리 동시에 이벤트를 실행시킨다. 그러나 이러한 고정적인 재생 지연 시간을 사용하는 것은 가변적 네트웍 트래픽 상태에서 전송 지연 시간이 짧은 경우 대부분의 참가자가 이벤트를 빨리 수신함에도 불구하고 일정 시간 동안 수신자 시스템에 저장해야 한다. 또한 네트웍 상에 트래픽이 증가하여 전송 지연 시간이 길 때에는 재생 지연 시간이 지난 후에 수신하는 이벤트의 수가 증가하여 손실된 이벤트를 처리하기 위한 오버헤드를 초래하기 때문에 가변적인 네트웍 트래픽 상태에는 비효율적이다.

본 연구에서 우리는 가변적인 네트웍 트래픽 상태를 판단하여 발생하는 이벤트에 효율적인 재생 지연 시간을 적용하는 기법을 제안하였다. 모든 참가자들의 이벤트 손실율을 수집하여 네트웍 트래픽 상태를 판단하였다. 실험을 통해 가변적인 네트웍 트래픽 상태에 따라 변하는 전송 지연 시간에 적응하는 재생 지연 시간을 적용함으로써 고정적 재생 지연 시간을 적용하는 경우와 비슷한 이벤트 손실율을 보이면서 약 10%이상의 재생 지연 시간을 줄였다.

6. 참고문헌

[1] M.Ahamad and R.Kordale, "Scalable Consistency Protocols for Distributed Services," IEEE Transations on Parallel and Distributed Systems, September 1999, pp.888-903.

[2] S.Bhola, G.Banavar and M.Ahamad, "Responsiveness and Consistency Tradeoffs in Interactive Groupware," ACM CSCW'98, November 1998, pp.79-88.

[3] L.Gautier and C.Diot, "A Distributed Architecture for Multiplayer Interactive Applications on the Internet," IEEE ICNP'99, July 1999, pp.6-15.

[4] L.Gautier and C.Diot, "End-to-end Transmission Control Mechanisms for Multiparty Interactive Applications on the Internet," IEEE INFOCOM'99, March 1999, pp.1470-1479.

[5] T.K.Capin, J.Esmerado and D.Thalmann, "A Dead-Reckoning Technique for Streaming Virtual Human Animation," IEEE Transactions on Circuits and Systems for Video Technology, April 1999, pp.411-414.

[6] M.Mauve, "Consistency in Replicated Continuous Interactive Media," ACM CSCW'00, December 2000, pp.181-190.

[7] J.Vogel and M.Mauve, "Consistency Control for Distributed Interactive Media," ACM MM'01, September 2001, pp.221-230.

[8] C.Sun and D.Chen, "A Multi-version Approach to Conflict Resolution in Distributed Groupware Systems," IEEE ICDCS'00, April 2000, pp. 316-325.

[9] L.Vaghi, C.Greenhalgh, and S.Benford, "Coping with Inconsistency due to Network Delays in Collaborative Virtual Environments," ACM VRST'99, December 1999, pp.42-49.