# 분산 가상 환경을 위한 버킷 기반 적응형 동기화 제어 계층 구현

김재윤° 김종원
광주과학기술원 정보통신공학과 네트워크미디어 연구실
{jykim°, jongwon}@netmedia.gist.ac.kr

# Implementation of Adaptive Bucket-based Synchronization Layer for Distributed Virtual Environment

JaeYoun Kim° and JongWon Kim
Networked Media Lab., Dept. Information and Communications,
GIST(Gwangju Institute of Science and Technology)

## 요 약

연속적이고 동기화된 미디어 스트림을 요구하는 분산 가상 환경(DVE: Distributed Virtual Environment) 시스템은 크게 3개 계층, 즉 가상환경, 동기화, 그리고 네트워크 계층 등으로 나누어 질 수 있다. 그 중 동기화 계층의 목적은 그룹 멤버들 간에 일관된 전역적인 상태와 멤버쉽을 유지하게 하는 것이다. 분산 가상 환경에서 이질적인 종단 시스템들 간에 효율적인 미디어 재생을 위해 적응형 동기화 기법들이 연구되어 왔으나, 기존의 적응형 동기화 기법은 가변적인 재생지연의 조절로 인하여 skipping과 pausing 현상을 야기한다. 본 논문에서는 분산 가상환경 시스템에서 다자간 미디어 재생을 지원하기 위한 동기화 계층을 설계하고, 구현하는 방법에 대해서 기술한다. 특히, 가변적인 네트워크 상태로 인해 발생되는 이벤트 패킷 지연(delay)과 순서 오류(out of sequence)를 수신 측에서 효과적으로 제어하기 위한 적응형 동기화 재생 기법을 제안하고 skipping과 pausing 현상을 감소시키기 위한 재생 버퍼 구조를 소개한다. 실험을 통하여 제안된 기법의 효율성을 검증하였다.

## 1. 서 론

분산 가상 환경(DVE: Distributed Virtual Environment) 시스템은 일반적으로 서로 다른 환경으로 구성된 시스템과 네트워크들로 구성된다. 이러한 이질적인 통신 환경으로 인해 DVE 시스템의 참여자들의 상태는 짧은 기간부터 오랜 기간에 걸친 불일치 현상에 영향을 받는다. 이러한 문제를 해결하기 위하여 이벤트 기반의 적응형 동기화 기법(event-based adaptive synchronization control)이 요구된다. DVE 시스템은 3개 계층(가상환경 시스템, 동기화, 네트워크)으로 나누어 질 수 있는데 동기화 계층의 목적은 그룹 멤버들 간에 일관된 상태와 멤버쉽을 유지하는 것이다.

현재까지 많은 이벤트 기반의 동기화 기법들이 제안되어 왔으며, DVE 시스템을 위한 대표적인 동기화 기법[1,2,3]에는 버킷 동기화(bucket synchronization), 시간 왜곡(time wrap), 추적 상태 동기화(*trailing state synchronization*)등이 있다. 위의 동기화 기법들에서는 불일치 현상이 발생했을 때 이를 회복하기 위해서 추측항법(dead-reckoning), 되돌림(roll-back), 추적 상태(trailing state)등과 같은 회복 기법들을 제안하고 있다. 하지만 이러한 회복 기법들을 위해서는 많은 비용이 들게 되므로 불일치 현상이 발생하는 원인들을 사전에 파악하고 이에 대처하는 방법이 필요하다. 그 중 가장 큰 원인은 가변적인 네트워크 상태에서 패킷 순서오류로 인한 재생 버퍼에서의 이벤트 손실이다. 이를 해결하기 위해서 적응형 동기화 기법이 요구된다.

기존에 연구된 적응형 동기화 기법들[4,5]에서는 동적인 네트워크 상태에 따라 재생지연을 조절하는 방법을 제안하였다. 하지만 동적으로 재생시간을 조절할 경우 이벤트 버퍼에서는 두 가지의 문제점이 발생한다. 첫 번째로 재생시간을 줄일 때, 새롭게 도착한 이벤트가 이전에 도착한 이벤트와 겹치게 되어 이전에 도착한 이벤트는 버려지는 스키핑(skipping) 현상이 발생 한다 [7]. 그리고 재생시간을 갑자기 늘이게 될 경우, 이전과 새로운 재생시간의 차이만큼 이벤트가 재생되지 않으므로 멈추어 있는 것처럼 보이게 되는 포우징(pausing) 현상 이 발생 한다 [7].

본 논문에서는 동기화 계층을 분산 모델로 설계하였으며, 전송지연 때문에 발생하는 이벤트 손실을 줄이기 위해 단 방향 전송지연(one-way delay)에 기반 하는 네트워크 적응형 동기화 제어 기법을 제안하고, 적응형 기법 중 이벤트 버퍼에서 발생할 수 있는 스키핑(skipping), 포우징(pausing) 현상을 줄이기 위한 재생 버퍼 구조를 소개한다.

본 논문의 구성은 다음과 같다. 2절에서는 전체적인 시스템 구조와 기본적인 접근 방법을 소개하고, 3절에서는 적응형 이벤트 기반 동기화 계층 구조를 제안한다. 그리고 4절에서는 실험 결과를 설명하고 마지막으로 5절에서 결론 및 향후 연구 방향에 대해서 말하고자 한다.

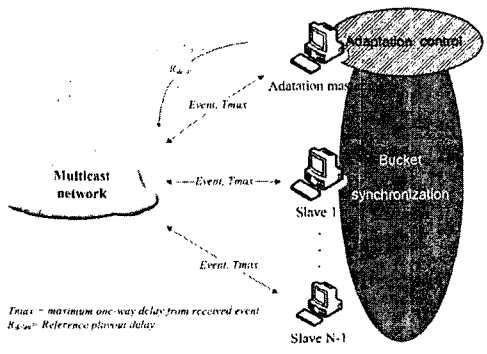


그림 1. 시스템 모델.

## 2. 시스템 모델 및 접근 방향

DVE 시스템에서 다자간 동일한 재생 상태를 가지기 위해서는 가변적인 네트워크 상에서 발생되는 패킷 전송지연과 순서 오류 제어를 우선적으로 고려해야 한다. 수신자 측에서는 어떠한 형태로든 이 문제를 해결해야 하는데 이것을 intra-media 동기화 기법[6] 관점에서 보면 재생되는 이벤트들 간의 일시적인 간격을 조절 하는 것이다. Inter-client 동기화 기법을 위한 가장 직관적인 방법은 버킷 동기화 기법에서처럼 모든 참가자

들이 이벤트를 발생시간(generation time)으로부터 일정한 시간동안 지연시킨 후 시간 간격으로 나누어져 있는 이벤트 버퍼(버킷)의 같은 위치에 저장하는 것이다 [1,6]. 그 일정 시간 동안의 지연을 재생지연(playout delay)이라 하고, 재생시간(presentation time)은 이벤트에 재생지연이 적용된 후 실제 처리되는 시간을 말한다. 이 재생지연은 참가자들 중에서 누군가 경험할 수 있는 전송지연보다 커야하며, 이 전송지연을 그룹 최대 지연(group maximum delay: *Gtmax*)이라 한다. 이러한 관점에서 적응형 동기화 기법이란 적절한 재생시간을 설정하기 위하여 조절 가능한 재생지연을 결정하는 것으로 정의된다.

## 2. 1 시스템 구조

DVE 시스템은 그림 1과 같이 *N* 명의 송, 수신 기능을 하는 참가자들로 구성되고 그룹 통신을 위해서 하나의 멀티캐스트 주소가 사용된다. 하나의 그룹 관점에서 볼 때, 적응형 동기화 제어를 하기 위해 참가자들 중 임의의 한 노드가 adaptation master로 선출되고, 나머지 노드들은 slave로 구성된다. Slave 노드들은 가변적인 네트워크 상황으로 인해 서로 다른 패킷 전송지연을 경험하게 되므로, 모든 참가들 간 재생시간 동기화를 위해 master 노드는 참조 재생지연(reference playout delay)을 결정하여 이벤트와 함께 전송하게 된다. 여기서 우리는 모든 참가자들의 개별 시스템 클럭(system clock)은 하나의 전역 시간(global clock)으로 동기화 되어 있다고 가정하고 이것은 NTP(Network Time Protocol)를 이용하여 설정할 수 있다.

그림 2에서는 동기화 계층 구성도를 나타낸다. 각 수신자들은 동기화 계층 모듈에서 이벤트들을 수신하면서 전송지연($T_i$)를 (1)과같이 측정하고 그림 2에서의 report attacher를 이용하여 일정 시간동안 받은 $T_i$ 중 최대 전송지연(*Tmax*)을 주기적으로 adaptation master에게 피드백한다.

$$T^i max = max(Ti),(\ i=1,\ 2,\ \ldots,\ N),$$
$$where\ Ti = local\_clock - generation\_time\ of\ event. \quad (1)$$

Adaptation master는 모든 참가자에서부터 전송된 *Tmax*를 이용하여 그림 2의 adaptation controller에서 참조 재생지연을 결정하고 이를 역시 report attacher를 이용하여 자신의 이벤트와 함께 송신한다.

## 2. 2 버킷 기반 동기화 기법

분산 모델에서는 각 참가자 들이 같은 시간에 동일한 상태를 유지하기 위해서 동기화 기법이 반드시 필요하다. 버킷 기반 동기화 기법에서 재생버퍼는 일정 시간 간격으로 나누어지고 버킷은 단위시간 간격에 연관된다 [1]. 수신된 이벤트들은 발생시간에 따라 동기화를 위한 재생지연을 참조하여 동일 시간대를 갖는 해당 버킷으로 할당되며, 각 버킷이 종료 될 때 마다 재생된다. 만약 어떤 이벤트가 해당 버킷이 종료된 후에 도착하게 된다면 재생 될 수 없다. 반면에, 이 이벤트가 다음 이벤트의 상태를 추측 할 수 있는 용도로 사용될 수 있도록 추측 항법(dead-reckoning)을 사용한다. 하지만 추측 항법은 불일치 현상을 해결하기엔 부족하고, 오히려 더 정확하지 않은 상태로 판단할 소지가 있다. 이에 따라, 재생지연을 충분히 주는 방식으로 불일치 현상을 해결할 수 있으나, 실시간 멀티미디어 전송을 요구하는 DVE시스템 환경에서는 합리적이지 못하다. 즉, 재생지연의 결정은 즉시성 (responsiveness) 과 일관성 (consistency)을 동시에 고려해야 하며, 서로 상충하는 관계를 가지므로, 주어진 상황에 따라 효율적 재생지연을 선택하는 기법이 무엇보다 중요하다.

## 3. 적응형 이벤트 기반 동기화 모델

적응형 버킷 기반 동기화 기법에서는 재생지연을 네트워크 상태에 따라 적절하게 대응하기 위해 동적으로 줄이거나 늘이면서 조절한다. 하지만 끊임없이 전송되어 오는 이벤트 스트림에 대해서는 스키핑 현상 없이 참조 재생지연을 줄일 수 없다. 이를 위해 참조 재생지연을 적용 시킬 시점에 대해 충분히 고려해야 한다. 우리는 이 문제를 해결하기 위해서 이산적인 이벤트 통신을 하는 DVE 시스템의 특성을 이용한다. 연속적인 미디어 스트림과는 다르게 DVE 시스템에서는 매번 이벤트를

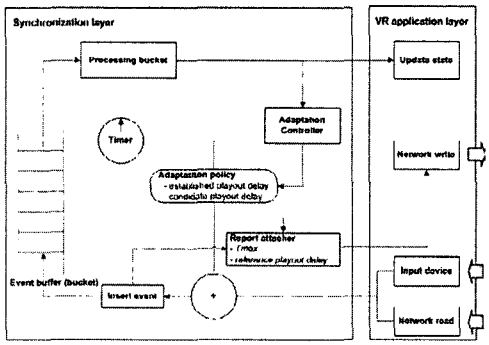


그림 2. 동기화 계층 구성도.

$$Gtmax_{(m)} = max(Tma_{x(i)}), \quad (1)$$
$$C_{delay(m)} = (1\text{-}d)C_{delay(m-1)} + dGtmax_{(m)}, \quad (2)$$
*if* $C_{delay(m)} > UTT$ *then* //risk state (3)
$$UTT_{(m)} = C_{delay(m)} + offset_{(u)}$$
$$LTT_{(m)} = UTT_{(m)} - a$$
$$D_m = UTT_{(m)} + ß$$
*end if*
*else if* $C_{delay(m)} < LTT$ *then* //inefficient state (4)
$$LTT_{(m)} = C_{delay(m)} - offset_{(l)}$$
$$UTT_{(m)} = LTT_{(m)} + a$$
$$D_m = UTT_{(m)} + ß$$
*end if,*
*where a denotes the interval between UTT and LTT, and B denotes interval between reference plaout delay and UTT, and $D_m$ is reference playout delay.*

그림 3. 참조 재생지연 결정 과정.

네트워크로 전송할 필요가 없다. 만약 가상환경에 변화가 없다면 수신자 측에서는 이전 상태를 그대로 이용하면 되고 추가적으로 데이터 집합(data aggregation) 기법이나 추측항법(dead reckoning)을 통하여 전송 횟수를 줄일 수 있다. 이처럼 일정 기간동안 이벤트가 발생하지 않는 기간을 전송 휴지시간(sending idle period)이라 하고 이 기간을 이용하여 스키핑 없이 재생지연을 줄인다. 이 개념은 voice traffic의 휴지시간(silence period)을 가지는 특성을 이용하는 적응형 재생 기법 [7]과 유사하다. 그리고 버킷 동기화에서는 각 이벤트에 적용될 재생지연은 버킷 간격의 배수가 되어야 한다. 왜냐하면 같은 버킷에서 발생한 이벤트가 재생지연이 적용된 후 서로 다른 버킷에 저장이 될 수 있으며 각 이벤트는 버킷의 끝에서 재생이 되므로 버킷 주기보다 작은 재생시간의 지연은 의미가 없을 수 있기 때문이다.

## 3. 1 이벤트의 재생지연 결정 방법

제안된 기법에서는 참조 재생지연은 그룹 최대 전송지연(*Gtmax*)에 의하여 결정된다. 그림 3은 재생지연을 결정하는 과정을 기술하였다. Adaptation master는 버킷의 끝마다 그림 3의 (1)을 이용하여 *Gtmax*을 구할 수 있게 된다. 지터에 의한 짧은 기간동안의 심한 변동을 완화 하기 위하여 스무딩 함수(geometric weighting smoothing function)의 결과 값 Cdelay (2)을 사용한다. $\delta$ 값은 현재 *Gtmax* 값의 반영 정도를 의미한다. Adaptation master 는 (3)과 (4) 조건식에서 Cdelay 값과 두개의 임계값을 이용하여 참조 재생지연을 결정 한다. 두 개의 임계값은 상위 목표 한계값(UTT: upper target threshold)과 하위 목표 임계값(LTT : lower target threshold)이고 Cdelay 값이 UTT 와 LTT 사이에서 유지되도록 두 임계값과 참조 재생지연을 조절한다.

## 3. 2 적응형 재생 기법을 위한 버퍼구조

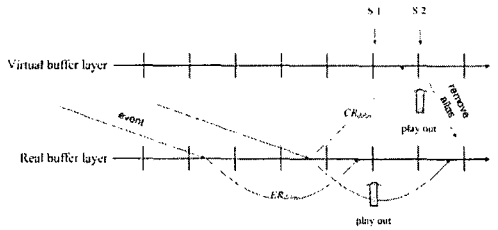


그림 4. 이중 이벤트 버퍼 구조.

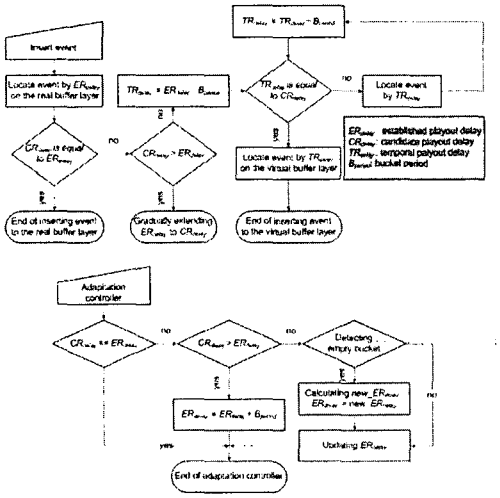


그림 5. 이벤트 처리 순서도.

본 절에서는 재생지연을 늘이거나 줄일 때 발생하는 포우징, 스키핑 현상을 줄이기 위한 이중 버퍼 구조를 소개한다. 그림 4와 같이 이벤트 버퍼는 2개의 계층(virtual buffer layer, real buffer layer)으로 나누어진다. 재생지연을 늘일 경우에는 점차적으로 적용시키고, 줄일 경우에는 전송 휴지 시간을 감지하고 그 기간을 이용하여 스키핑에 의한 손실을 최대한 줄이면서 재생지연을 점차적으로 단축시킨다. 그림 5와 같이 제안된 기법에서는 재생지연은 이벤트가 수신되는 즉시 적용되지 않으므로 기존에 설정된 재생 지연($ER_{delay}$: established playout delay)과 후보로 등록된 재생 지연($CR_{delay}$: candidate playout delay)에 의해서 이벤트가 저장될 버퍼의 계층이 결정된다. 참가자 들은 이벤트를 수신하였을 때 이것에 $ER_{delay}$을 적용 시킨 후 real buffer layer에 위치시키고 그 이벤트의 복사본(alias)을 $CR_{delay}$로 적용 시킨 후 virtual buffer layer의 버킷에 위치시키게 된다. 마지막으로 스케쥴러가 버킷을 시간 순서대로 재생 시킬 때 만약 real buffer layer에 이벤트가 없으면 virtual buffer layer에 있는 이벤트들을 재생 하고 그것들의 복사본들을 모두 제거 한 후 $ER_{delay}$ 값을 점차 적으로 감소시킨다.

## 4. 실험 결과

본 논문에서 제안된 기법의 효율성을 입증하기 위해서 스키핑을 고려하지 않은 적응형 동기화 기법과 제안된 기법을 그림 6과 같은 환경에서 비교 실험하였다. 스키핑을 고려하지 않은 기법은 그림 7-(a)와 같이 네트워크 지연의 변화가 심할 경우 잦은 재생지연의 변화 인해 그림 7-(c)에서 처럼 많은 스키핑 현상이 발생한다. 하지만 제안된 기법 에서는 그림 7-(b)와 같이 잦은 재생지연의 변화 중에서도 그림 7-(d)에서 볼 수 있는 것처럼 스키핑이 거의 발생하지 않음을 알 수 있다. 왜냐하면 제안된 이중 버퍼구조에서는 전송 휴지 시간 동안에 재생지연을 단축시킴으로써 스키핑에 의한 버퍼에서의 이벤트 손실을 최대한 줄였기 때문이다.

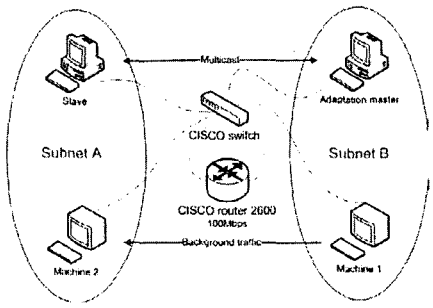


그림 6. 실험 환경.

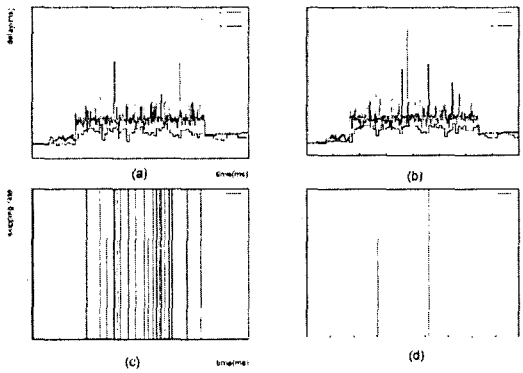


그림 7. 제안된 동기화 계층의 성능 비교.

## 5. 결론 및 향후 연구 방향

본 연구에서는 분산 가상 환경을 위해서 네트워크 상태에 따른 적응형 동기화 제어 기법과 재생지연 조절 때문에 발생하는 스키핑, 포우징 현상을 줄이기 위한 이중 버퍼 구조를 제안 하였다. 향후, 데이터 모음(data aggregation)기법과 추측 항법을 함께 적용하여 송신자 측에서 네트워크 혼잡 발생시 송신률을 조절하여 혼잡을 제어하는 송신자 측 적응형 동기화 기법을 추가 연구할 계획이다.

## 참고 문헌

[1] L. Gautier and C. Diot, "End-to-end transmission control mechanisms for multiparty interactive Application on the Internet," in *Proc. IEEE INFOCOM'99*, March 1999, pp.1470-1479.

[2] D. Jefferson, "Virtual time," *ACM Transactions on Programming Languages and System*, vol. 7, no. 3, pp. 404-425, 1985.

[3] E. Cronin, B. Filstrup, and A. Kurc, "An efficient synchronization mechanism for mirrored game architectures," in *Proc. Network and System Support for Games*, April 2002.

[4] E. Hong, D. Lee, E. Park, and K. Kang, "An efficient synchronization mechanism adapting to heterogeneous transmission delay in networked virtual environments," in *Proc. SPIE/ACM MMCN 2003*, pp. 24-33, January 2003.

[5] Y. Ishibashi, "Group synchronization control for haptic mecia in networked virtual environments," in *Proc. International Symposium on Haptic Interface for Virtual Environment and Teleoperator Systems*, 2004.

[6] G. Blakowski and R. Steinmetz, "A media synchronization survey: reference model, specification, and case studies," *IEEE J. Sel. Areas in Communication*, vol. 14, no. 1, pp. 5-35, January 1996.

[7] Y. Ishibashi and S. Tasaka, "A comparative survey of synchronization algorithms for continuous media in network environments," in *Proc. LCN 2000*, pp. 337-348, November 2000.