Journal of Korea Multimedia Society Vol. 17, No. 4, April 2014(pp. 526-534)
http://dx.doi.org/10.9717/kmms.2014.17.4.526

모바일 멀티플레이어 게임을 위한 하이브리드 클라이언트-서버 구조의 대역폭 요건과 우선순위 기반 동기화 기법

김 진 환†

Bandwidth Requirement and Priority-based Synchronization Methods in Hybrid Client-Server Architecture for Mobile Multiplayer Games

Jinhwan Kim†

ABSTRACT

Most of the multiplayer games available online are based on a client-server architecture because this architecture gives better administration control to the game providers than peer-to-peer architecture. In this architecture, the server is responsible for all the communication between the connected clients. The weakness of this architecture is its bandwidth requirement and scalability. Peer-to-peer architectures have then been proposed to solve these issues. In this paper, we propose a hybrid client-server architecture in which the game state is partially shared by the mobile terminal to achieve consistency among different players. Like a peer-to-peer architecture, this architecture uses client-side capacities to reduce bandwidth requirements for the server and improves consistency in wireless networks. Client events have different timeliness and consistency requirements according to their nature in the game world. These requirements lead to tasks with different priorities on CPU processing. In the proposed architecture, either the server or the client applies consistency mechanism according to the priority level. Simulation experiments show that the bandwidth of the server in this architecture is smaller than that of the client-server architecture. As a result, the server in the proposed architecture can accommodate more clients with enhancing the scalability.

Key words: Mobile Multiplayer Game, Bandwidth, Priority, Synchronization, Client-server

1. 서 론

클라이언트-서버(CS; client-server) 구조에서 서버는 연결된 모든 클라이언트들 간의 통신을 책임지는 역할을 수행한다. 게임 서버의 경우 모든 게임의 실행 상태를 서버가 주관하며 클라이언트는 단지 게임의 가상 세계를 단말기 화면에 표시하는 기능만 책임지게 된다[1]. CS 구조의 약점인 서버의 통신 대역폭 요건과 동시 접속 플레이어들에 대한 규모조정 문제들을 해결하기 위하여 P2P(Peer-to-Peer) 구조가 제시되었다. 그러나 P2P 구조에서는 각 플레이어가 보고 있는 게임의 상태가 이미 다른 플레이어의 해킹 등 부당한 행위로 작성된 게임 상태일 수도 있는 단점을 가지고 있다. 또한 모바일 단말기의 경우

※ Corresponding Author : Jinhwan Kim, Address: (136-792) 116 Samsungyo-ro 16 gil, Seongbuk-gu, Seoul, Korea, TEL : +82-2-760-4340, FAX : +82-2-760-4488, E-mail : kimjh@hansung.ac.kr
Receipt date : Sep. 3, 2013 , Revision date : Jan. 22. 2014
Approval date : Feb. 20, 2014
† Dept. of Multimedia Engineering, Hansung University
※ This research was financially supported by Hansung University.

상이한 처리 속도, 기억장치 용량, 스크린 해상도, 운영체제 등을 가진 다양한 종류의 장치들로 구성되기 때문에 게임 개발업체는 수많은 단말기에서 실행될 수 있는 게임 코드의 작성이 실제로 어렵다. 따라서 게임 회사는 아직도 CS 구조를 선호하고 있으며 CS 구조의 집중식 제어 방식은 영리를 추구하는 게임 회사에서 중요한 역할을 수행한다. 수익 창출 외에도 집중화된 게임 방식은 P2P 구조보다 구현하기가 더 수월하며 P2P 구조에서 나타나는 분산 이벤트 순서화, 부당 행위 방지, 분산 저장장치 관리 방법 등을 고려할 필요가 없다[2].

모바일 클라이언트들이 무선 통신망을 통하여 서버에 연결될 때 통신망의 지연시간은 상황에 따라 변동성이 크고 jitter 현상[3]도 발생할 수 있다. 게임 상태의 정확성을 의미하는 일관성 유지와 이벤트 순서화를 정상적으로 유지하고 있는 서버에서 게임 논리가 실행되기 때문에 서버가 송신하는 갱신 이벤트들은 통신망의 지연시간의 변동이 클 경우 서로 다른 플레이어들에게 상이한 시점에 도달할 수 있다. 상이한 클라이언트들이 시점에 따라 다른 결과를 볼 경우 게임 상태의 비일관성이 발생할 수 있다. 서버는 일관성 있는 가상 세계가 클라이언트의 모바일 단말기에 표시될 수 있도록 갱신 메시지를 충분히 자주 보내야만 한다. 그러나 메시지를 많이 보낼 경우 서버의 통신 대역폭 요건이 상당히 증가하게 되고 결과적으로 동시 접속하는 플레이어들의 수를 의미하는 규모조정도 어렵게 된다. 예를 들면 게임 서버의 통신 대역폭 요건은 플레이어 수의 제곱에 비례하여 증가하기 때문에 참여하는 플레이어의 수가 증가하면 서버에 대한 통신 대역폭 병목 현상이 쉽게 발생할 수 있다[4].

멀티플레이어 게임에 관한 다른 중요한 사항인 플레이어 간의 상태 일관성도 정확히 유지될 필요가 있다[5]. 예를 들어 자동차 경주 게임에서 모든 플레이어는 각 자동차의 위치에 관한 동일한 화면을 볼 수 있어야 하며 FPS(First Person Shooting) 게임에서는 모든 플레이어들의 생사 여부를 동시에 동일하게 파악할 수 있어야 한다. CS 구조는 전체 게임의 일관성을 단순하게 유지하며 보장할 수 있는 장점이 있는 반면 P2P 구조에서는 서버의 대역폭 요건에 대한 병목 현상은 없는 대신 모든 클라이언트들이 항상 정확한 상태를 일관성있게 유지할 수 있도록 복잡한 distributed agreement 프로토콜이 반드시 필요하다는 단점이 있다. 즉 각 플레이어는 자신의 지역적 상태와 다른 모든 플레이어들의 상태 간의 일관성을 항상 점검해야 하기 때문에 상당한 통신 오버헤드가 수반된다. 본 논문에서는 멀티플레이어 게임을 위한 두 구조의 장점을 모두 사용하는 하이브리드 CS 구조를 제시하며 전체 게임의 상태 일관성을 항상 정확히 유지하면서 서버의 대역폭 요건을 감소시키는 이벤트 우선순위 동기화 기법을 제시한다. 멀티플레이어 게임 시스템에서 클라이언트가 발생하는 이벤트는 게임 시스템의 실시간적 요건과 중요도에 따라 우선순위가 설정될 수 있다[6]. 본 논문에서는 우선순위가 낮은 이벤트는 플레이어들이 P2P 방식으로 직접 상호 교환하며 우선순위가 높은 이벤트는 플레이어와 서버가 CS 방식으로 통신하는 동기화 과정이 수행된다.

본 논문은 2장에서 CS 구조와 P2P 구조의 대역폭 요건과 일관성 유지 방법이 기술되며 3장에서 하이브리드 CS 구조의 서버 대역폭 감소 요건과 우선순위 기반 동기화 기법이 기술된다. 그리고 4장에서 분석과 시뮬레이션을 통하여 다른 구조와 비교된 성능을 분석하며 5장에서는 결론을 기술한다.

2. CS 구조와 PP 구조

2.1 모델 특성

대부분의 상업적 멀티플레이어 게임들은 게임 상태의 유일한 사본이 서버에서 실행되는 CS 구조를 사용한다. 실제로 Quake[7], Starcraft[8] 게임들도 CS 구조에서 수행된다. 클라이언트인 플레이어가 키를 누르면 이 키가 서버로 전송된다. 클라이언트로부터 서버로 전송되는 메시지를 명령어 또는 이벤트로 정의할 수 있다[9]. 서버는 모든 클라이언트로부터 명령어를 수집한 후 새로운 게임 상태를 작성하며 갱신된 상태를 클라이언트들에게 다시 전송하게 된다. 클라이언트들이 최종적으로 새로운 상태를 렌더링하며 자신들의 단말기 화면에 출력하게 되는 것이다. 본 논문에서 오직 명령어 입력과 게임 상태의 렌더링만 책임지는 클라이언트들을 thin 클라이언트들로 간주한다.

그러나 P2P 구조에서는 게임 상태를 한 곳으로 집중하여 보관하는 장소가 없다. 대신 각 클라이언트

가 다른 모든 클라이언트들로부터 수신한 메시지에 기반하여 게임 상태의 사본을 각자 유지하게 된다. 즉 P2P 모델에서는 각 플레이어(peer를 의미함)가 자신의 게임 시뮬레이션을 직접 실행할 책임을 가지게 된다[9]. 그러나 플레이어들 간에 비일관성을 탐지하고 이를 해결하는 서버가 없기 때문에 임의의 비일관성은 distributed agreement 프로토콜을 사용하는 플레이어들에 의해 직접 탐지되어야만 한다. 예를 들면 분산 게임에서 비일관성을 해결하기 위하여 trailing state[10] 또는 bucket 동기화[11] 기법들이 제시된 바 있다. P2P 구조는 Xpilot[12], MiMaze[13] 등 단순한 게임에서 사용되었으며 STOW-E[14]같은 복잡하며 규모조정이 가능한 전쟁 시뮬레이터에서도 사용된 바 있다.

2.2 대역폭 요건

CS 구조와 P2P 구조의 상대적인 전체 통신 대역폭 크기는 거의 동일하다. P2P 구조에서는 각 플레이어가 매주기마다 입력 메시지를 다른 플레이어들에게 멀티캐스트(multicast)하므로 플레이어의 수를 N이라고 할 경우 주기당 N^2 개의 메시지를 수신하게 된다. CS 구조에서는 각 플레이어가 매주기마다 중앙 서버에게 한 개의 입력 메시지를 유니캐스트(unicast)하며 서버는 주기마다 해당 플레이어 즉 클라이언트마다 게임 상태 메시지를 전송하게 된다. 따라서 CS 구조 전체에서 주기당 N 개의 입력 메시지와 N 개의 게임 상태 메시지가 발생한다. 전형적으로 게임 상태 메시지는 N 개의 입력 메시지의 결과로써 발생한 변동 상황을 반영해야 하기 때문에 N에 선형비례하여 증가하게 된다. 결과적으로 두 구조의 전체적인 대역폭은 공통적으로 N^2에 비례하여 증가하게 된다[9]. 그러나 CS 구조에서 통신량은 중앙 서버에 집중되는 현상이 발생하며 항상 N^2 개의 메시지를 처리해야만 하는 문제가 있다.

interest management 기법이 두 구조의 대역폭 요건을 완화하기 위해 사용될 수 있다[15]. 플레이어들은 게임 상태의 일정 영역에만 관심을 가지게 된다. 게임에 참여하는 플레이어가 추가되더라도 게임 상태의 영역이 동일하다면 interest management 기법이 두 구조의 규모조정 가능성을 향상시킬 수 있게 된다. 집중화된 서버는 클라이언트가 관심을 두고 있는 상태의 부분을 결정할 수 있으며 이들에게만 여과된 상태 정보를 전송하게 된다. 결과적으로 고정된 크기의 상태 갱신은 전체 대역폭을 일정 크기만큼만 증가시키게 된다. 예를 들면 고정된 크기를 c라 할 때 전체 통신 대역폭은 cN에 비례하여 증가된다. P2P 구조에서도 각 플레이어가 관심 영역 내에 있는 플레이어들과 멀티캐스트 그룹을 구성하여 통신하게 되므로 전체적으로 cN 대역폭을 요구하게 된다.

2.3 동기화 기법

동기화는 멀티플레이어 온라인 게임의 기본이며 게임에 참여하는 모든 플레이어들이 실질적으로 일관성 있는 상태에서 게임을 할 수 있도록 보장하게 된다. CS 구조와 P2P 구조에서 효율적인 동기화 기법을 구현하기 위한 알고리즘들은 보수적인 알고리즘과 낙관적인 알고리즘들이 있다. 먼저 보수적 알고리즘들은 잘못된 순서를 직접 방지함으로써 동기화 문제를 해결한다. 전형적인 보수적 알고리즘은 lockstep 동기화[16]이며 서버는 모든 플레이어들의 명령어를 수신한 후에만 다음 조치를 취하게 된다. 인터넷 기반의 소규모 3D 멀티플레이어 게임인 MiMaze에서는 bucket 동기화 기법이[17] 사용되었으며 이는 서버가 이벤트들의 잘못된 순서를 방지하기 위하여 게임 상태를 갱신하기 전에 짧지만 고정된 시간만큼 기다리게 된다. 많은 FPS 온라인 게임들은 이 기법을 여전히 사용하며 frame-based 동기화 기법으로도 기술되고 있다. 공평성을 유지하기 위해 가장 속도가 느린 플레이어를 기다려야만 하는 실시간 전략(RTS; Real-time Strategy) 게임에서는 lock-step 동기화 기법의 변형인 turn-based 동기화 기법이 활용된다. 이 기법에서는 각 플레이어가 다른 모든 플레이어들의 통신 지연 시간을 항상 탐지하여 다음번 주기를 조정하게 된다. 보수적 알고리즘들은 단순한 게임에 적절하게 사용될 수 있으나 이벤트의 잘못된 순서를 탐지하고 복구하는 능력이 없기 때문에 빠른 속도로 실행되는 게임에서는 적용되기 어렵다.

반면 낙관적 알고리즘들은 비일관성을 탐지하며 교정하는 절차가 있고 일정한 비율의 시뮬레이션을 유지할 수 있어 복잡한 멀티플레이어 게임에 적합하다. 낙관적 알고리즘들은 먼저 발생된 이벤트들이 늦게 도착하지 않을 것이라고 낙관적으로 이벤트들을 처리하며 예상이 틀렸을 경우에는 비일관성을 해결

하는 방법을 사용하게 된다. 실제로 게임 상태가 이전 상태로 복구(rollback)되는 것이다. 낙관적 알고리즘에서는 복구 전략과 이벤트들의 잘못된 순서를 탐지하는 방법이 매우 중요한 역할을 수행하게 된다. TW(Time Warp) 동기화 기법[18]은 매 실행 상태마다 스냅샷(snapshot)을 유지하고 가장 최근에 실행된 이벤트보다 먼저 발생된 이벤트가 나중에 도착하면 즉시 이전 상태로 복구된다. 복구시 상태는 스냅샷으로 먼저 복원되며 스냅샷과 실행 시점 사이에 있던 모든 이벤트들은 다시 실행된다. TW 기법은 이벤트들이 새로운 이벤트들을 직접 생성하는 것을 가정한다. 이전에 생성된 이벤트들 중 이미 무효화된 이벤트들을 취소하기 위한 anti-message들이 복구 과정의 일부로써 전송된다. 이러한 anti-message들이 이미 실행이 완료된 경우 다른 복구 과정을 유발하게 되어 더 많은 anti-message들이 생성되며 결과적으로 통신망의 통신량을 급격히 증가시킬 수 있게 된다. 최근 개발된 알고리즘들은 이러한 anti-message 문제를 완화시키고 있다.

TS(Trailing State) 동기화 기법[19]은 Quake 게임을 위해 개발되었으며 비일관성이 탐지될 때 복구 과정이 수행된다. TW 동기화 기법과는 달리 모든 명령어마다 스냅샷을 유지하는 대신 게임 상태를 동기화 지연시간만큼 분리하여 구성된 두가지 상태로 유지한다. 시간 관점에서 가장 최신 상태를 Leading 상태라 하며 지연시간만큼 차이나는 이전 상태를 Trailing 상태라 한다. Leading 상태에서 비일관성이 탐지되면 복구 과정이 수행되며 Trailing 상태에 있던 게임 상태가 Leading 상태로 복사된다. 현재 시점과 비일관성이 탐지된 시점 사이에 있는 모든 명령어들은 다시 수행된다. TS 동기화 기법은 TW 동기화 기법과는 실질적으로 다른 복구 과정이 수행되며 더 적은 양의 기억장치와 오버헤드가 수반된다. Quake 게임처럼 대부분의 명령어들이 독립적인 경우는 anti-message들에 대한 문제의 심각성이 완화될 수 있다.

P2P 구조에서 임의의 플레이어 j가 자신의 상태와 일치하지 않는 갱신 메시지를 다른 플레이어 i로부터 수신할 때 비일관성을 탐지할 수 있다[20]. 이 경우 플레이어 j는 플레이어 i를 포함한 모든 플레이어들에게 갱신을 거절하는 anti-message를 전송하게 된다. 이 메시지의 목적은 플레이어 i의 최신 갱신을 무효화하기 위한 것이며 플레이어 i는 이 메시지를 수신하는 즉시 이전 상태로 복구(rollback)된다.

2.4 문제점

CS 구조는 구현이 비교적 단순하며 집중화된 서버를 통하여 상태 일관성을 유지하기가 수월하고 플레이어의 부정행위가 어렵다. 또한 게임 제공업체는 영리를 추구할 수 있는 플레이어 가입 구조를 만들고 관리할 수 있으며 지속적인 게임 감시도 가능하다. 이러한 이유 때문에 CS 구조가 멀티플레이어 게임에 많이 활용되고 있다. 그러나 CS 구조는 여러 문제점들이 존재한다. 첫째, 서버의 결함이 발생할 경우 게임 수행 자체가 불가능하게 된다[10]. 둘째, 플레이어 측면에서의 대역폭은 크지 않지만 서버 측면에서의 대역폭은 매우 커질 수 있다[10]. 결과적으로 서버는 대용량의 통신 대역폭이 필요하며 서버를 통해 플레이어들에게 전달되는 메시지들은 통신 지연시간이 다소 증가될 수 있다.

집중화된 서버의 부재는 P2P 구조에서 클라이언트 간 메시지 지연 시간의 감소[11], 서버의 결함으로 인한 시스템 중지 사유 소멸 등 여러 가지 이점을 제공한다. 가장 중요한 것은 P2P 구조에서 서버의 병목 현상이 없다는 것이다. 그러나 P2P 구조는 게임 산업에서 CS 구조처럼 보편적으로 사용되는 상황은 아직 아니다. 중요한 이유는 P2P 구조가 CS 구조보다 구현이나 구성이 더 어렵기 때문이다. 또한 P2P 구조는 게임에 대한 제어를 사용자들에게 많이 양도하고 있고 해커의 공격에도 매우 취약하다. 서버의 부재는 게임을 수행하는 사람과 게임 실행 시간 등에 대한 제어가 불가능함을 의미하며 가입자를 통해 이윤을 추구하려는 게임 업체를 어렵게 만드는 상황을 조성하게 된다.

본 논문에서는 CS 구조에서 서버의 통신 대역폭을 감소시키면서 P2P 구조의 일관성 문제를 해결할 수 있도록 하이브리드 CS 구조를 제시한다.

3. 하이브리드 CS 구조

3.1 이벤트의 우선순위

게임 세계에서 발생하는 이벤트의 본질에 따라 이벤트들은 상이한 일관성 요건을 가지게 된다[6]. 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[6]. 실제 대규모 멀티플레이어 게임인 Quake의 명령어인 이벤트들 중 Fire, Impact 이벤트들은 일관성 실시간 요건을 가지며, Move 이벤트는 실시간 요건만 가지고 Damage/die, Spawn 이벤트들은 일관성 요건만 가지는 것으로 분류된다.

상이한 이벤트들의 요건들은 게임 서버의 CPU 처리를 위한 스케쥴링과 전송 스케쥴링 과정에서 상이한 우선순위로 설정될 수 있다. 본 논문에서는 일관성 실시간 이벤트인 Quake의 Fire, Impact 이벤트들의 우선순위를 가장 높은 1 그룹으로 설정하며 실시간 이벤트인 Move 이벤트를 다음 우선순위인 2 그룹으로 설정한다. 그리고 실시간 요건이 없는 일관성 이벤트인 Damage/die와 Spawn 이벤트를 가장 낮은 우선순위인 3 그룹으로 설정한다.

3.2 우선순위 기반 대역폭 요건

멀티플레이어 게임에서 각 플레이어는 CS 구조에서 클라이언트 역할을 수행하며 서버에게 이벤트를 전송하게 된다. 본 논문에서는 클라이언트가 1 초 동안 서버에게 전송하는 평균 이벤트 또는 메시지의 수를 M이라 하며 1, 2, 3 그룹의 평균 이벤트 수를 각각 M_1, M_2, M_3로 정의한다. 즉 수식 (1)과 같이 M은 M_1, M_2, M_3의 합이 된다.

$$M=M_1+M_2+M_3 \quad (1)$$

CS 구조에서 서버의 대역폭 병목 현상을 감소시키고자 우선순위가 높은 1 그룹과 2 그룹의 이벤트들은 클라이언트가 서버에게 전송하고 우선순위가 낮은 3 그룹의 이벤트들은 P2P 구조처럼 클라이언트들이 서버를 통하지 않고 직접 상호 교환하는 방식을 본 논문에서 하이브리드 CS 구조로 가정한다(Fig. 1 참조).

Fig. 1에서 클라이언트는 C로 서버는 S로 표현된다. 이 하이브리드 CS 구조에서 클라이언트의 수를

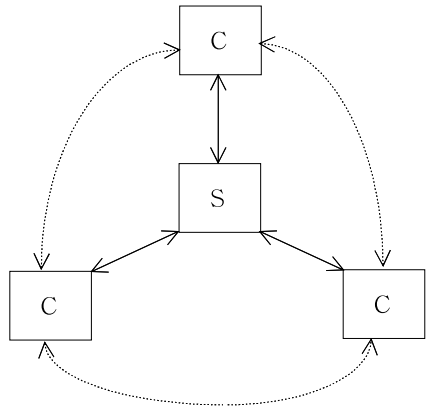

Fig. 1 Hybrid CS architecture.

N이라하고 메시지의 평균 크기를 m(단위 바이트)이라 할 때 클라이언트가 1 그룹과 2 그룹의 이벤트들을 서버에게 전송할 경우 매초 $(M_1+M_2)\cdot m$ 대역폭이 필요하며 3 그룹의 이벤트를 다른 모든 클라이언트들에게 전송할 경우 매초 $M_3\cdot(N-1)\cdot m$ 대역폭이 필요하다. 따라서 클라이언트의 상향(upstream) 대역폭 C_{ups}는 수식 (2)와 같다.

$$C_{ups}=(M_1+M_2)\cdot m+M_3\cdot(N-1)\cdot m \quad (2)$$

이 구조에서 임의의 클라이언트는 매초 서버로부터 $N\cdot(M_1+M_2)\cdot m$ 대역폭에 해당하는 전역 상태 메시지를 수신하게 되며 다른 모든 클라이언트들로부터 $M_3\cdot(N-1)\cdot m$ 대역폭의 메시지를 수신하게 되므로 하향(downstream) 대역폭 C_{dns}는 수식 (3)과 같이 설정된다.

$$C_{dns}=N\cdot(M_1+M_2)\cdot m+M_3\cdot(N-1)\cdot m \quad (3)$$

따라서 클라이언트의 하향과 상향 대역폭을 합한 결과는 수식 (4)와 같다.

$$C_{ups}+C_{dns}=(M_1+M_2)\cdot(N+1)\cdot m+2M_3\cdot(N-1)\cdot m \quad (4)$$

이 구조에서 서버는 매초 1 그룹과 2 그룹의 이벤트들을 N 플레이어로부터 수신하기 때문에 하향 대역폭 S_{dns}는 $N\cdot(M_1+M_2)\cdot m$이며 게임 세계의 전역 상태 메시지를 N 플레이어들에게 다시 전송하는 상향 대역폭 S_{ups}는 $N^2\cdot(M_1+M_2)\cdot m$이 된다. 따라서 서버의 상향과 하향 대역폭을 모두 포함한 대역폭은 다음과 같다.

$$\begin{aligned} S_{dns}+S_{ups} &= N\cdot(M_1+M_2)\cdot m+N^2\cdot(M_1+M_2)\cdot m \\ &= N\cdot(N+1)\cdot(M_1+M_2)\cdot m \end{aligned} \quad (5)$$

제시된 하이브리드 CS 구조에서도 서버의 대역폭은 플레이어들의 수 N^2에 비례하여 증가하지만 CS 구조의 서버 대역폭보다 M_3/M만큼 감소되기 때문에 서버의 병목 현상이 완화될 수 있는 것이다. 단 1 그룹과 2 그룹의 이벤트들을 서버에게 전송할 경우 서버의 대역폭이 감소되는 만큼 클라이언트들의 대역폭은 $M_3 \cdot (N-1) \cdot m$만큼 증가하게 된다. 결과적으로 CS 구조, P2P 구조, 하이브리드 CS 구조 모두 필요한 전체 통신 대역폭은 거의 동일하기 때문에 서버의 대역폭이 감소되는 대신 클라이언트의 대역폭이 증가하게 된다. Table 1은 세 가지 구조의 클라이언트와 서버의 대역폭을 수식으로 비교한 결과이며 하이브리드 CS 구조는 1 그룹과 2 그룹의 이벤트들만 서버가 수신하는 상황을 가정한 것이다.

Table 1에서 전체 대역폭은 각 클라이언트의 대역폭에 플레이어의 수 N을 곱한 대역폭과 서버의 대역폭을 합한 결과이다. M이 3이고 $M_1=M_2=M_3=1$일 경우 CS 구조, 하이브리드 CS 구조, P2P 구조의 전체 대역폭은 각각 $6Nm(N+1)$, $2Nm(3N+1)$, $6Nm(N-1)$로 기술되며 근소한 차이가 있는 것으로 분석된다.

3.3 이벤트 우선순위 기반 동기화 기법

하이브리드 CS 구조에서는 우선순위가 높은 1 그룹과 2 그룹의 메시지를 서버가 클라이언트로부터 수신하며 우선순위가 낮은 3 그룹의 메시지들은 클라이언트들이 직접 상호 교환하는 방식이 적용되기 때문에 CS 구조와 P2P 구조의 동기화 방법이 모두 적용된다. 즉 1 그룹과 2 그룹의 메시지들이 클라이언트로부터 서버로 전송될 경우 서버는 전형적인 CS 구조의 비일관성 해결 방법을 사용하며 3 그룹의 메시지들에 대해서는 클라이언트가 직접 비일관성을 해결하는 P2P 구조의 방법을 사용한다. 실제로 전형적인 CS 구조에서 게임 상태는 유일한 서버에 의해 사본이 오직 한 개만 유지되기 때문에 게임 상태의 일관성은 문제가 없다. 그러나 P2P 구조에서는 게임 상태가 여러 개의 사본으로 존재하기 때문에 통신망에서 메시지가 손실되거나 순서가 변경되면 비일관성이 발생하게 된다. 본 논문의 하이브리드 CS 구조에서도 임의의 플레이어가 자신의 상태와 일치하지 않는 갱신 메시지를 다른 플레이어로부터 수신할 때 비일관성을 탐지할 수 있는 것을 가정한다. 이 경우 비일관성을 탐지한 임의의 플레이어는 다른 모든 플레이어들에게 갱신을 거절하는 anti-message를 전송하며 갱신이 거절된 플레이어는 이 메시지를 수신하는 즉시 이전 상태로 복구(rollback)되는 상황을 가정한다. 제시된 하이브리드 CS 구조에서 Quake 게임을 위해 개발된 TS 동기화 기법[19]이 적용되는 것을 가정하므로 Leading 상태에서 비일관성이 탐지될 때 Trailing 상태에 있던 게임 상태가 Leading 상태로 복사된다. 비일관성이 탐지되면 현재 시점과 비일관성이 탐지된 시점 사이에 있는 모든 명령어들은 다시 수행된다.

4. 성능 분석

본 논문에서는 대규모 멀티플레이어 온라인게임 Quake의 각 플레이어가 1초 마다 평균 3 개의 이벤트를 발생하며 각 우선순위 그룹의 이벤트가 한 개씩 동일한 비율로 구성된 것을 가정한다. 즉 $M_1=M_2=M_3=1$이며 $M=M_1+M_2+M_3=3$이다.

플레이어가 발생한 이벤트는 메시지로 서버에게 전송되며 평균 크기(m)은 20 바이트이고 최소값 10 바이트와 최대값 30 바이트 분포 범위에서 실제 메시지의 크기가 설정된다. 모바일 통신망 환경을 가정하여 평균 통신 지연시간은 10 msec이며 최소 시간 5 msec와 최대 시간 15 msec 분포에서 실제 통신 지연시간이 설정된다. 통신망에서 패킷들이 손실되거나 순서가 변경되는 경우는 없는 것으로 가정한다. 플레이어의 수는 500명부터 1000명까지 100 명씩 증가시킨 상태에서 10분간 서버의 평균 대역폭을 Mbyte

Table 1. Communication bandwidth of CS and P2P architectures

Architecture	Client Bandwidth	Server Bandwidth	Total Bandwidth
CS	$(N+1)Mm$	$N(N+1)Mm$	$2N(N+1)Mm$
Hybrid CS	$(M_1+M_2)(N+1)m+2M_3(N-1)m$	$N(N+1)(M_1+M_2)m$	$N((M_1+M_2)(N+1)+2M_3(N-1)+(N+1)(M_1+M_2))m$
P2P	$2(N-1)Mm$	0	$2N(N-1)Mm$

(=2^{20} 바이트) 단위로 측정하였다.

본 논문의 하이브리드 CS 구조에서 1 그룹과 2 그룹의 메시지들만 서버가 처리한 대역폭은 HCS (1,2)로 표기되며 1 그룹의 메시지들만 서버가 처리한 대역폭은 HCS(1)로 표기된다. HCS(1,2)와 HCS (1)은 그룹별 구분이 없는 CS 구조의 서버 대역폭과 Fig. 2에서 비교 분석되었다. P2P 구조는 서버가 존재하지 않기 때문에 Fig. 2에서 P2P 구조의 결과는 기술되지 않는다. CS 구조의 서버는 우선순위 그룹에 상관없이 모든 플레이어의 이벤트를 처리해야 하는 반면 하이브리드 CS 구조의 서버는 우선순위가 높은 그룹의 이벤트만을 처리하기 때문에 서버의 대역폭이 감소된다. HCS(1,2)와 HCS(1)은 전형적인 CS 구조의 서버 대역폭보다 각각 약 33.33%와 66.67%가 감소된 결과가 Fig. 2에서 나타났다. 따라서 하이브리드 CS 구조에서는 감소된 서버의 대역폭만큼 게임에 참여하는 플레이어의 수를 더 증가시킬 수 있거나 각 플레이어의 초당 평균 이벤트 수가 동일한 비율로 증가되는 상황을 수용할 수 있다. 예를 들어 CS 구조의 서버가 1000명을 수용할 수 있는 대역폭을 하이브리드 CS 구조의 서버가 사용할 경우 1 그룹과 2 그룹의 이벤트만을 처리할 때는 1000명의 1/3인 약 333명이 증가된 1333명의 플레이어들을 서버가 수용할 수 있게 되며 1 그룹의 이벤트만을 처리할 때는 1000명의 2/3가 증가된 약 667명을 추가로 수용할 수 있게 된다.

Fig. 3에서는 P2P 구조를 포함하여 각 플레이어의 평균 대역폭이 Kbyte(=2^{10} 바이트) 단위로 비교되고 있다. 하이브리드 CS 구조에서 서버가 1 그룹과 2 그룹의 이벤트들만 수신하고 3 그룹의 이벤트들은 플레이어들이 직접 상호 교환하는 것을 가정하였다. Fig. 3에서 플레이어의 대역폭은 CS 구조가 가장 작은 것으로 나타났으며 이는 다른 구조에 비하여 서버의 대역폭이 훨씬 더 크기 때문인 것으로 분석된다. 하이브리드 CS 구조에서 서버가 1 그룹과 2 그룹의 이벤트들을 수신하는 경우 플레이어는 3 그룹의 이벤트들만 다른 플레이어들과 상호 교환하기 때문에 플레이어의 대역폭이 CS 구조보다는 크게 나타났다. 즉 하이브리드 CS 구조의 경우 서버의 대역폭이 감소되는 반면 각 플레이어의 대역폭이 증가되는 것으로 분석된다. 서버없이 플레이어들만 메시지를 상호 교환하는 P2P 구조에서는 플레이어의 대역폭이 가장 크며 이는 CS 구조보다 거의 두 배 정도인 것으로 측정되었다. Fig. 3에 나타난 바와 같이 각 구조에서 공통적으로 플레이어의 대역폭은 플레이어의 수에 비례하여 증가하는 것으로 나타났다.

Table 2에서는 세 구조의 전체 대역폭이 Mbyte 단위로 플레이어의 수 N에 따라 비교되고 있다. 각 구조의 전체 대역폭은 Table 1에서 분석된 바 있다. 실제 세 구조의 전체 대역폭은 CS 구조가 가장 큰 것으로 나타났으며 P2P 구조가 가장 작은 것으로 나타났다. 서버가 1 그룹과 2 그룹의 이벤트들을 수신하는 하이브리드 CS 구조에서 전체 대역폭은 CS 구조보다 작고 P2P 구조보다 큰 것으로 나타났다.

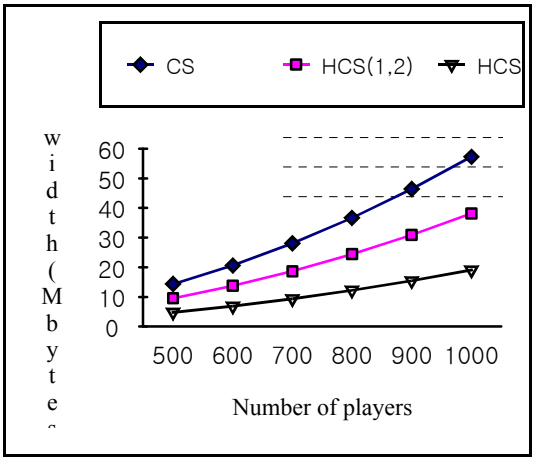

Fig. 2. Server bandwidth of CS and Hybrid CS architectures.

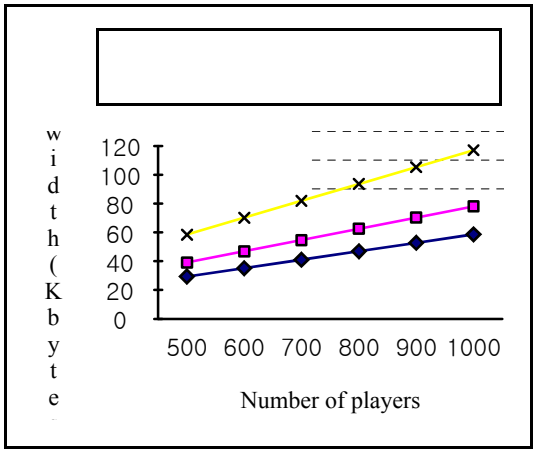

Fig. 3 Player bandwidth of each architecture.

Table 2. Total bandwidth of CS, Hybrid CS and P2P architectures (Unit: Mbytes)

Architecture	Players					
	500	600	700	800	900	1000
CS	28.67	41.27	56.16	73.33	92.80	114.56
HCS(1,2)	28.63	41.22	56.10	73.27	92.73	114.48
P2P	28.55	41.13	55.99	73.15	92.59	114.33

5. 결 론

대규모의 플레이어들이 참여하는 멀티플레이어 온라인게임에서 서버의 통신 대역폭을 감소시키고 이벤트 우선순위 기반의 동기화 기법을 구현하기 위하여 하이브리드 CS 구조를 본 논문에서 제시하였다. 전형적인 CS 구조의 서버는 플레이어 수의 제곱에 비례하는 통신 대역폭 특성상 병목 현상이 쉽게 발생할 수 있다. 서버가 없는 P2P 구조에서는 플레이어의 대역폭이 CS 구조의 플레이어보다 두 배 정도로 증가되며 상태 일관성을 유지하기 위한 복잡한 distributed agreement 프로토콜이 필요하다. 본 논문에서는 플레이어의 이벤트 특성에 연관된 우선순위를 활용하여 우선순위가 높은 이벤트들을 CS 구조의 서버가 처리하고 우선순위가 낮은 이벤트들은 플레이어들이 직접 상호 교환하는 P2P 구조의 기법을 활용하여 서버의 대역폭을 감소시킴으로써 병목 현상이 완화되도록 하였다.

제시된 하이브리드 CS 구조의 서버에서 감소된 대역폭만큼 더 많은 수의 플레이어들을 수용할 수 있다. 더 많은 플레이어들을 수용하는 대신 감소된 통신 대역폭을 여유 대역폭으로 활용할 경우 서버의 대역폭 용량 한계로 발생할 수 있는 일시적 과부하 문제도 해결할 수 있다. 또한 TS 동기화 기법을 제시된 하이브리드 CS 구조에 적용함으로써 게임 시스템 전체의 상태 일관성이 항상 정확히 유지될 수 있다. 본 논문에서 제시된 하이브리드 CS 구조와 동기화 기법은 향후 모바일 환경의 대규모 멀티플레이어 게임시스템의 대역폭과 규모조정 요건 충족 및 성능 향상을 위하여 적용될 것으로 기대된다.

REFERENCES

[1] A.M. Khan, I. Arsov, M. Preda, S. Chabridon, and A. Beugnard, "Adaptable Client-Server Architecture for Mobile Multiplayer Games," *Proceeding of International Conference on Simulation Tools and Techniques*, pp. 11-11, 2010.

[2] G. Coulouris, J. Dolimore, and T. Kindberg, *Distributed Systems Concepts and Design*, Addison-Wesley, Boston, 2001.

[3] F. Lamberti and A. Sanna, "A Streaming-Based Solution for Remote Visualization of 3D Graphics on Mobile Devices," *IEEE Transactions on Visualization and Computer Graphics*, Vol. 13, No. 2, pp. 247-260, 2007.

[4] P. Koutsakis, M. Vafiadis, and A. Lazaris, "A New Bandwidth Allocation Mechanism for Next Generation Wireless Cellular Networks," *Wireless Network*, Vol. 16, Issue 2, pp. 331-353, 2010.

[5] 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,* Vol. 55, No. 4, pp. 1883-1889, 2009.

[6] A. Hsu, J. Ling, Q. Li, and C.C. Jay Kuo, "On the Design of Multiplayer On-line Video Game Systems," *Proceeding of International Conference on Internet, Performance & Control of Networks*, pp. 180-191, 2003.

[7] Doom, Quake, ID Software, Inc., http://www.idsoftware.com03, 2003.

[8] Diablo II, Starcraft. Blizzard entertainment, http://www.blizzard.com, 2001.

[9] J.D. Pellegrino and C. Dovrolis, "Bandwidth Requirement and State Consistency in Three Multiplayer Game Architectures," *Proceeding of*

Workshop on Network and System Support for Games, pp. 52-59, 2003.

[10] E. Cronin, B. Filstrup, A. Kurc and S. Jamin, "An Efficient Synchronization Mechanism for Mirrored Game Architectures," *Proceeding of Workshop on Network and System Support for Games*, pp. 67-73, 2002.

[11] C. Diot and L. Gautier, "A Distributed Architecture for MultiPlayer Interactive Applications on the Internet," *IEEE Network*, Vol. 13, No. 4, pp. 6-15, 1999.

[12] B. Stabell and K.R. Schouten, "The Story of XPilot," *ACM Crossroads*, Vol. 3, Issue 2, pp. 3-6, 2001.

[13] L. Gautier and D. Diot, "A Distributed Architecture for Multiplayer Interactive Applications on the Internet," *IEEE Networks Magazine*, Vol. 13, Issue 4, pp. 6-15, 1999.

[14] S. Brunett, D. Davis, T. Gottschalk, P. Messina, C. Kesselman, "Implementing Distributed Synthetic Forces Simulations in Metacomputing Environments," *Proceeding of the Heterogeneous Computing Workshop*, pp. 29-42, 1998.

[15] L. Zou, M.H. Ammar, and C. Diot, "An Evaluation of Grouping Techniques for State Dissemination in Networked Multi-User Games," *Proceeding of International Symposium Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pp. 33-40, 2001.

[16] E. Cronin, B. Filstrup, and S. Jamin, "Cheat-Proofing Dead Reckoned Multiplayer Games," *Proceeding of International Conference on Application and Development of Computer Games*, 2003.

[17] L. Gautier, C. Diot and J. Kurose, "End-to-End Transmission Control Mechanisms for Multiparty Interactive Applications on the Internet," *Proceeding of Annual Joint Conference of the IEEE Computer and Communications Societies*, pp. 1470-1479, 1999.

[18] J. Smed, T. Kaukoranta, and H. Hakonen, "Aspects of Networking in Multiplayer Computer Games," *The Electronic Library*, Vol. 20, No. 2, pp. 87-97, 2002.

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

[20] M. Jung and K. Park, "A P2P Real-time Game System for Multiplayer on Overlay Network," *Journal of Korea Multimedia Society*, Vol. 13, No. 1, pp. 38-46, 2010.

김 진 환

1982년 3월~1986년 2월 서울대학교 컴퓨터공학과 학사
1986년 3월~1988년 2월 서울대학교 컴퓨터공학과 석사
1988년 3월~1994년 2월 서울대학교 컴퓨터공학과 박사
1994년 3월~1996년 2월 서울대학교 컴퓨터신기술공동연구소 특별연구원
1995년 3월~현재 한성대학교 멀티미디어공학과 교수
관심분야 : 멀티미디어시스템, 분산 실시간 시스템