제 18 회 한국정보처리학회 추계학술발표대회 논문집 제 9 권 제 2 호 (2002.11)

대규모 네트워크 게임 서버를 위한 소켓 I/O 모델의 비교 분석

최진성*, 박학봉**, 전재우**, 오삼권**
*호서대학교 벤처전문 대학원, **호서대학교 컴퓨터공학부
e-mail: bluemarble@empal.com, piaoxf@hanmail.net,
jean1225@nownuri.net, ohsk@office.hoseo.ac.kr

A Comparative Analysis of Socket I/O models for Massively Multi-player On-line Network Game Server

Jinseong Choi*, Xuefeng Piao**, Jaewoo Jeon**, Samkweon Oh**
*Graduate School of Venture, Hoseo University
**Dept. of Computer Engineering, Hoseo University

요 약

대규모 네트워크 게임은 최대한 많은 사용자를 수용할 수 있어야 하며 사용자들에게 안정적인 서비스를 제공할 수 있어야 한다. 그러나 많은 사용자들의 서버 동시 접속 및 제한된 네트워크 대역폭 등의 문제들로 인해 네트워크 게임 서버는 부하가 걸릴 수 있다. 이런 부하를 최대한 줄이기 위해 네트워크 게임 서버는 기능별로 분산되어 있는 것이 일반적이며, 분산된 서버들간의 통신 또는 서버와 클라이언트들간의 통신은 운영체제에서 제공하는 네트워크 통신 라이브러리를 사용한다. 본 논문에서는 MicroSoft 사의 윈도우즈 계열에서 제공하는 다섯 가지 소켓 I/O 모델들에 대해 조사하고 비교 분석한다. 비교 분석한 결과 대규모 네트워크 게임 환경에서는 IOCP 모델이 기타 소켓 모델에 비해 IO 처리가 여러 가지 장점을 가지고 있음을 알 수 있다.

1. 서론

게임에는 전용 게임기를 통해 즐길 수 있는 비디오 게임과 PC 로 즐길 수 있는 PC 게임이 있다. 최초의 비디오 게임은 1961 년 Steve Russell 에 의해 개발되었고 PC 게임은 1980 년대에 PC 가 일반 가정에 보급됨에 따라 선보이기 시작하였다. 게임의 역사는 1950 년대 말 Brook Haven 국립 연구소에서 발표된 테니스 게임에서부터 시작되었다. 초기의 게임은 연구 차원에서 진행 되었는데 1970 년대 말에 ATARI 사에서 비디오 게임기 "Atari 2600"이 출시되고 TAITO 사에서 개발된 "Space Invader"라는 게임이 상업화에 성공하자 게임에 붐을 이루기 시작하였다[1].

네트워크 게임은 1990 년대 초 모뎀을 이용한 일반 채팅으로부터 시작되었다. 채팅을 중심으로 하던 온라인 서비스가 채팅 방을 이용한 게임으로 변화되면서 1993 년에는 텍스트기반 MUD(Multi-User Dungeon)게임들이 서비스되기 시작하였다. 1997 년에 들어서면서 텍스트기반 MUD 게임은 그래픽기반 MUG(Multi-User Graphics)게임으로 발전하였고 최근에는 3D 그래픽 기술과 가상공간의 접목으로 MMORPG 장르의 네트워크 게임으로 발전하였다. MMORPG 는 텍스트를 기반으로 한 온라인 게임이었던 MUD(Multi User Dungeon)에 그래픽 요소를 가미한 일종의 RPG(Role Playing Game)게임이다[2]. 대표적으로 NC 소프트사의 "리니지", NEXON 의 "바람의 나라", Origin System 의 "Ultima Online" 등이 있다[3,4,5].

대규모 네트워크 게임은 최대한 많은 사용자들을

수용할 수 있어야 하며 많은 사용자들에게 안정적인 서비스를 제공할 수 있어야 한다. 이를 위해서는 게임 서버 시스템 구축시 많은 사용자들의 동시 접속문제나 제한된 네트워크 대역폭 문제를 고려해야 한다[6,7,8].

네트워크 게임 서버는 부하 문제를 해결하기 위해 일반적으로 분산 구조를 가지며 네트워크 대역폭 문제를 해결하기 위해 전송 메시지를 최대한 줄이는 방법을 사용하고 있다. 이런 분산 구조를 가진 서버들간 통신 또는 서버와 클라이언트들간의 통신은 운영체제에서 제공하는 통신 라이브러리를 사용하는데 I/O 처리 방식에 따라 여러 가지로 나눌 수 있다. 본 논문에서는 MicroSoft 사의 윈도우즈 계열에서 제공하는 다섯 가지 소켓 I/O 모델들을 비교 분석한다.

본 논문의 구성은 아래와 같다. 2 장에서는 네트워크 게임 서버 관련연구를 살펴보고 3 장 에서는 윈도우 소켓 모델별 I/O 처리방식을 설명하며, 4 장에서는 기능별 소켓 I/O 모델들에 대해 비교 분석한다. 그리고 5 장에서 결론을 맺는다.

2. 관련연구

세계적으로 네트워크 기술에 대한 활발한 연구와 초고속 통신망의 확보로 인터넷 사용자가 급증하면서 과거의 컴퓨터 게임은 세계 각국의 많은 사람들이 모여서 서로 대화하면서 즐길 수 있는 네트워크 게임으로 발전하였다. 대규모 네트워크 게임은 수천, 수만의 사용자들을 수용할 수 있어야 하며 안정적인 서비스를 제공하여야 한다. 그러나 수천, 수만 사용자들의 동시 접속, 게임을 즐기기 위한 사용자의 급증, 제한된 대역폭 등의 문제로 서버는 부하가 걸릴 수 있다. 따라서 이런 문제들을 해결하기 위한 많은 연구들이 최근까지 활발하게 진행되고 있다.

대규모 네트워크 게임과 같은 분산 가상환경에서의 서버 기술은 주어진 환경에서 최대한 많은 사용자들에게 안정적인 서비스를 제공하는 것을 궁극적인 목적으로 하고 있다. 이를 위해 분산 가상환경에서의 서버들은 일반적으로 분산 구조를 가지며 사용되고 있는 서버 기술들로는 실시간 처리를 위한 멀티스레드 프로그래밍기법, 서비스 종류를 세분화하여 각각을 담당하는 서버를 두고 서버들간의 연결구조를 적절히 구성하는 분산 서버기술, 사용자 기준으로 사용자를 여러 그룹으로 나누고 각각의 그룹을 위한 서버를 두거나 영역 기준으로 가상환경을 여러 영역으로 나누어 각각의 영역을 담당하는 서버를 두는 다중서버 기술들이 있다[8].

게임 서버 구조에 대한 연구를 보면 서버와 클라이언트 사이에 미들웨어 서버를 두어 접속하는 클라이언트 수에 따라 게임 서버의 수를 조절할 수 있고 이동되는 데이터를 분산시키고 모니터링 하는 부하 분산 미들웨어 구조나 분산기술로 각광 받고 있는 CORBA (Common Object Request Broker Architecture)기술을 3-tier 방식에 적용한 CORBA 기반 분산 서버구조 등이 제시되고 있다[9,10].

이밖에 게임 세계를 여러 영역으로 나누어 부하를 분산시키는 방법이 있는데 정적으로 영역을 분할할 경우 발생하는 부하문제를 보완하여 사용자의 수에 따라 게임 영역을 동적으로 분할하는 동적 분할 알고리즘이 제시되었다[10]. 서버 부하를 줄이는 다른 방법으로 서버에서 클라이언트들로 전송될 데이터를 최대한 줄이는 방법이 있는데 전송될 데이터의 양이 네트워크 대역폭의 한계를 넘을 경우를 보완하여 클라이언트들로 전송될 데이터를 스케줄링 하는 PRR (Priority Round Robin) 알고리즘이 제시되었다[11].

3. 소켓 I/O모델

MicroSoft 사의 윈도우즈 환경에서는 네트워크 프로그래밍을 보다 쉽게 할 수 있는 윈속(winsock)이라는 통신 라이브러리를 제공하는데 이는 BSD(Berkeley Software Distribution)유닉스 계열의 소켓 인터페이스를 윈도우즈 환경에 적용한 소켓 API(Application Program Interface)이다. 소켓 API 함수들은 프로세스 상태에 따라 블록킹(Blocking)모드와 논블록킹(Non-Blocking)모드로 구분된다. 블록킹모드는 사용자가 관련함수를 사용하여 I/O 요청을 하면 처리가 완료될 때까지 대기상태에 있게 되며 처리가 완료된 후에 다음 처리를 진행한다. 논블록킹모드는 I/O 처리가 완료되지 않은 상태에서 다음 처리를 진행할 수 있다. 논블록킹모드에서는 I/O 처리 함수를 호출할 때마다 WSAEWOULDBLOCK 에러를 발생하는데 이는 함수 호출이 정상적으로 완료되었지만 I/O 처리가 완료되지 않음을 의미한다[13,14].

본 절에서는 논블록킹모드에서의 다섯 가지 소켓 I/O 모델에 대해 살펴본다.

3.1 기능별 소켓 I/O모델

소켓 I/O 모델은 I/O 처리 방식에 따라 select, AsyncSelect, EventSelect, Overlapped I/O, IOCP 모델로 나눈다.

3.1.1 select모델

select 모델은 유닉스와 윈도우즈 계열에서 모두 사용 가능한 모델이다. 이 모델은 FD_SET 구조체를 사용하는 것이 특징인데 이는 네트워크 이벤트를 커널 영역에서 사용자영역으로 알리기 위해 사용되는 자료구조이다.

FD_SET 구조체는 관리 소켓 수와 각 소켓 핸들(handle)에 대한 정보들이 포함되어 있으며 새로운 네트워크 이벤트를 처리할 때마다 FD_SET 구조체는

NULL 로 초기화된다.

3.1.2 AsyncSelect모델

AsyncSelect 모델은 네트워크 이벤트를 커널영역에서 사용자영역으로 알리기 위해 윈도우 메시지를 사용하는 것이 특징이다. AsyncSelect 모델에서 I/O 완료 이벤트는 윈도우 메시지를 통해 해당 프로그램으로 통보된다.

3.1.3 EventSelect모델

EventSelect 모델은 AsyncSelect 모델과 동작방식은 유사하나 윈도우 메시지 대신 이벤트 객체를 통하여 네트워크 이벤트를 사용자영역으로 전달한다. 사용자 영역의 응용프로그램은 네트워크 이벤트를 전달 받게 되면 WSAWaitForMultipleEvents 함수가 수행되는데 이 함수를 호출하는 스레드마다 대기 가능한 이벤트의 수가 64 개로 제한되어 있기 때문에 한 스레드당 최대 64 개의 소켓만 관리할 수 있다. 따라서 64 개 이상의 소켓을 관리하기 위해서는 더 많은 스레드를 생성해야 한다.

3.1.4 Overlapped I/O모델

Overlapped I/O 모델은 I/O 함수들이 Overlapped 구조체를 사용하며 호출시 결과 값이 즉시 반환된다. 이 모델 또한 EventSelect 모델과 마찬가지로 스레드당 관리할 수 있는 소켓 수가 64 개로 제한되어 있어서 관리하는 소켓수의 증가에 따라 생성하는 스레드 수도 증가해야 한다. 이 모델은 Overlapped I/O 요청을 관리하기 위해 두 가지 방법이 있는데 하나는 이벤트 통보 (Event Notification)를 기다리는 방법과 완료 루틴 (Completion Routines)을 통해 완료된 요청을 수행하는 방법이 있다.

3.1.5 IOCP(I/O Completion Port)모델

IOCP 모델은 멀티프로세서 시스템을 지원하는 모델이다. 이 모델은 스레드에 따른 소켓 수의 제한을 받지 않으며 생성하는 스레드 수는 시스템의 프로세서 수에 비례하여 결정한다.

IOCP 모델은 Overlapped I/O 모델과 마찬가지로 Overlapped 구조체를 사용하여 Overlapped I/O 모델의 장점을 취하며 I/O 처리를 하기 위해서는 Completion Port 객체의 생성을 요구한다. I/O 처리가 완료되면 운영체제는 Completion Port 객체를 통해 커널영역에서 사용자영역으로 I/O 처리가 완료됨을 알린다.

4. 비교분석

4.1 비교분석 기준

본 논문에서는 각 소켓 I/O 모델을 비교 분석하기 위해 3 가지 항목에 대해 고려하였다. 이 항목 들은 각 소켓 I/O 모델을 구분하는 기준이 된다. <표 1>은 각 소켓 I/O 모델을 비교 분석하기 위한 각 항목의 내용을 기술하였다.

<표 1> 소켓 I/O모델 비교

소켓 I/O 모델	이벤트 통보방식	스레드 수의 제약 여부	Overlapped 구조체 사용여부
select	FD_SET	제한 없음	사용안함
AsyncSelect	윈도우즈 메시지	제한 없음	사용안함
EventSelect	이벤트 객체	스레드당 최대 64 개 이벤트 처리제한	사용안함
Overlapped I/O	이벤트 객체 /커널에서 호출	스레드당 최대 64 개 이벤트 처리제한	사용함
IOCP	Completion Port 객체	제한 없음	사용함

4.2 소켓 I/O모델 비교분석

소켓 I/O 모델은 네트워크 이벤트 통보방식과 스레드의 제약 여부 그리고 Overlappped 구조체 적용여부에 따라 프로그램 구현의 복잡성이나 수행능력이 달라진다.

select 모델, AsyncSelect 모델, EventSelect 모델의 경우 프로그램구현이 비교적 간단하다. 그러나 select 모델에서 사용되는 FD_SET 구조체는 네트워크 이벤트를 처리할 때마다 초기화와 재설정이 필요하게 된다. 이런 초기화와 재설정 작업은 소켓의 수가 증가함에 따라 많아짐으로 많은 I/O 처리를 위한 많은 작업들이 필요하게 된다. 그러므로 빠른 I/O 처리를 요구하는 대규모 네트워크 게임 서버에 적용하기에는 적합하지 않을 수 있다.

AsyncSelect 모델의 경우 네트워크 이벤트가 발생하면 윈도우 메시지를 사용하여 커널영역에서 사용자영역으로 전달한다. 전달될 윈도우 메시지는 메시지큐에 저장되는데 메시지큐에는 네트워크 이벤트 메시지뿐만 아니라 윈도우즈 자체에서 발생하는 모든 이벤트들에 대한 메시지가 저장되게 된다. 그러므로 네트워크 이벤트에 대한 처리에 있어서 지연이 생길 수 있기 때문에 빠른 처리가 요구되는 서버프로그램에서는 바람직한 방법이 아니다.

EventSelet 모델은 전반적인 동작방식은 AsyncSelect 모델 방식과 유사하지만 윈도우 메시지를 사용하지 않고 이벤트 객체를 사용하여 윈도우 메시지로 인한 처리지연 단점을 보완하였다. 하지만 이 모델을 사용하면 각 스레드마다 관리할 수 있는 소켓 수가 제한되어 있기 때문에 많은 소켓을 관리하기 위해서는 많

은 스레드를 생성해야 한다. 스레드 수가 많아지면 문맥교환으로 인한 서버의 부담이 생기게 되며 게임 서버 자체의 과부하가 생길 수 있다.

Overlapped I/O 모델은 Overlapped 구조체를 사용한다. Overlapped 구조체를 사용하게 되면 네트워크 이벤트 발생시 커널영역에서 사용자영역의 버퍼에 직접 데이터를 저장하고 저장이 완료되면 응용프로그램에게 I/O 처리가 완료되었음을 알린다. 그러므로 사용자영역 버퍼에서 커널영역 버퍼의 데이터를 읽어오는 작업이 필요 없기 때문에 응용프로그램 수행차원에서 부담이 줄어든다. 그러나 프로그램의 구현이 어렵고 EventSelect 모델과 마찬가지로 스레드당 관리할 수 있는 소켓 수가 최대 64 개로 제한되어 있어 수천, 수만 개의 소켓을 관리해야 하는 대규모 네트워크 게임 서버에 적용하기에는 부족한 면이 있다.

마지막으로 IOCP 모델은 구현이 어렵다는 단점은 있지만 Overlapped 구조체를 사용함으로써 Overlapped I/O 모델의 장점을 취하였고 스레드당 관리할 수 있는 소켓 수의 제한도 없고 멀티프로세서 시스템에 유연하게 동작하는 장점이 있다. 그러므로 이 모델은 윈도우 계열에서 제공하는 기타 소켓 I/O 모델에 비해 여러 가지 장점이 있으며 빠른 I/O 처리를 요구하는 대규모 네트워크 게임 서버에 적용이 적합할 수 있다.

5. 결론

본 논문에서는 네트워크 통신 라이브러리에서 제공되는 소켓 I/O 모델에 대해 조사하였으며 윈도우즈 환경에서 제공되는 다섯 가지 소켓 I/O 모델에 대해 비교 분석하였다.

향후 연구방향은 객관적이고 타당성 있는 네트워크 게임 서버환경을 반영할 수 있는 테스트 환경모델을 설계하고 윈도우에서 제공하는 다섯 가지 I/O 모델을 적용 하여 성능 평가를 수행하고자 한다. 나아가서 대규모 네트워크 게임 서버의 부하를 줄일 수 있는 최선의 방법에 대해 연구할 계획이다.

참고문헌

[1]. 민용식, 신형철, 서종한, "게임 매니아를 위 한 게임 제작하기", 정일 출판사, pp. 12-41, 1999.

[2]. 최석우 "리니지 개발 사례", 정보처리학회지, 제 9권, 제3호, pp. 85-99, 2002.

[3]. http://www.lineage.co.kr/

[4]. http://baram.nexon.co.kr/

[5]. http://uo.stratics.com/

[6]. 양광호, 심광현, 고동일, 박일규, 김종성, "온라인 게임 서버의 기술 동향", 전자통신동향분석, 제 16권, 제4호, pp. 14-22, 2001.

[7]. 이만재, "온라인 게임 엔진 기술 동향", 정보과학회지 제20권, 제1호, pp. 12-18, 2002.

[8]. 심광현, 고동일, 양광호, 박일규, 김종성, "분산 가상환경을 위한 네트워크 및 서버 기술", 정보과학회지, 제19권, 제5호, pp. 69 -76, 2001.

[9]. 신동일, 신동규, 김민수, 장재홍, 윤현숙, 이정훈, 한창완, "온라인 게임 서버를 위한 부하 분산 미들웨어의 구현에 대한 연구", 한국정보과학회 춘계 학술발표논문집, 제27권, 제2호, pp. 178-180, 2000.

[10]. 최재언, 이해원 하수철, "CORBA기반 분산 네트워크 게임 서버에 관한 연구", 한국정보처리학회 춘계 학술발표논문집, 제8권, 제1호, pp. 159-162, 2001.

[11]. Dugki Min, Eunmi Choi, Donghoon Lee, Byungseok Park, "A Load Balancing Algorithm For Distributed Multimedia Game Server Architecture", IEEE International Conference on Multimedia Computing and Systems, Vol. 2, pp. 882-886, 1999.

[12]. Chris Faisstnauer, Dieter Schmalstieg, Werner Purgathofer, "Scheduling for Very Larger Virtual Environments and Networked Games Using Visibility and Priorities", 4th IEEE International Workshop on Distributed Simulation and Real-time Applications, pp. 31-38, 2001.

[13]. Anthony Jones, Jim Ohlund, "Network Programming for Microsoft Windows, Microsoft Press, pp. 227-271, 1999.

[14]. W. Richard Stevens, "UNIX Networking Programming", Prentice Hall, pp. 349-373, 1998.