|  |
| --- |
| Kookmin University |
| Network and Data Communication |
| HW #3 - MMORPG |

|  |
| --- |
| 20093267  Kim Sung Geun  Due Date. 14.06.02 |

**1. Briefly describe the MMORPG service and the underlying technologies focusing on the network perspectives.**

MMOPRG란 Massively Multiplayer Online Role Playing Game의 약어로 대규모로 한번에 접속하여 게임관리자가 게임의 규칙을 결정해두고 유저는 그 규칙 아래에서 각각에 맞는 역할을 수행하여 성장해 나아가는 게임을 말합니다.

따라서 MMORPG는 서버로부터 많은 양의 데이터가 다수의 클라이언트로 전송되고, 다수의 클라이언트는 해당 데이터를 응답 받고, 수행한 역할에 대한 데이터를 서버로 전송합니다. 예를 들어 사용자가 어떤 오브젝트를 공격한다고 할 때, 그 오브젝트는 서버로부터 온 데이터이며, 공격을 한다는 내용은 클라이언트가 서버로 전송하는 내용입니다. 이러한 네트워크 전송과 응답은 실시간으로 다수의 클라이언트와 서버 사이에서 수행되며, 이를 수행하기 위해서는 서버에서 동시에 많은 데이터 처리를 할 수 있도록 구성되어야 합니다.

이런 서버의 예로 “한게임”에서 제작한 MMORPG인 TERA를 예시로 들도록 하겠습니다. TERA는 보통 이전에 구현되었던 MMORPG보다 더 많은 양의 데이터 전송을 요구하고 있습니다. 그 이유로는 TERA에서 적용된 기술 중하나인 오브젝트 공격 시에 Targeting 방식이 아닌 Non-Targeting 방식으로 공격한다는 점과, 한 세계 안에서의 Loading과 Teleport가 없도록 구현하였기 때문입니다. 일반 콘솔 게임의 경우에는 해당방식 게임이 이미 존재 하였지만 네트워크를 통한 MMORPG의 경우에는 대부분 이러한 것을 유지 하여 개발하였습니다. 하지만 TERA의 경우 해당 조건들을 사용하였고 서버를 Arbiter와 World서버로 구분하여 설계하였습니다.

Arbiter 서버의 경우 World서버의 Network I/O를 감소시키기 위하여 커뮤니티(채팅, 파티 시스템 등)을 처리하고 DB접근(경매 시스템, 창고 시스템 등)과 관련된 것들을 제공하는 역할을 하도록 구성되어 있습니다. World 서버의 경우 전투 시스템이나, 배경에 대한 정보를 처리하도록 구성되어 있습니다. 이러한 구조로 Arbiter가 클라이언트가 요청한 정보를 받아서 역할에 따라서 커뮤니티나 DB접근을 제공하거나, 다른 시스템의 경우 World 서버로 전송하여 많은 수의 요청을 처리하도록 구성이 되어 있습니다.

여기까지 말해 왔던 것은 MMORPG가 어떻게 작동 되고, 그것이 한 예시로 TERA를 들어 게임 시스템이 어떻게 서버에 구현 되어 있는지를 설명하였습니다. 그러면 이것을 구현하기 위하여 사용되고 있는 네트워크의 기술은 무엇인지에 대해 파악해 보았습니다.

그것들 중 한 예로 Nagle Algorithm의 제거 입니다. Nagel Algorithm은 ACK오기를 기다렸다가 도착하게 되면, 그 동안 쌓여있던 데이터를 한꺼번에 보내도록 구성되는 것을 말합니다. 이러한 방식은 Bandwidth가 낮은 곳에서 빈번한 전송을 줄이게 됨으로써 효과를 가지는 Algorithm입니다. 하지만 MMOPRG의 경우에는 이런 방식을 사용하게 되면 문제가 생기게 됩니다. 예를 들어 사용자가 반복적으로 어떠한 오브젝트를 공격하고 있다고 할 때에 공격에 대한 데이터가 서버로 전송되어야 하는데, 반복에 대한 데이터가 서버로 전송이 되지 않고 모아두는 경우가 생기게 됩니다. 이렇게 되면 실시간으로 반영되어야 할 정보가 반영되지 않기 때문에 다른 쪽에 있던 유저에게 이 정보에 대한 것을 받지 않게 되어 게임을 진행하는 상황에 알맞지 않게 됩니다. 따라서 Nagle Algorithm을 제거하여 ACK 수신 없이 자신의 전송 버퍼 데이터가 있을 시 바로 바로 상대방 호스트에게 전달하여 방금 예시를 든 상황에 대하여 문제가 발생되지 않게 하고 있습니다.

두 번째로 Window에서의 IOCP와 Linux에서의 epoll 기술입니다. IOCP란 Input Output Completion Port의 약자로 해당 포트로 파일 입출력을 하는 것을 의미합니다. 중첩된 파일 입출력을 지원해주어 파일 입출력 동안에 다른 작업이 원활하게 돌아갈 수 있도록 지원해줍니다. 해당 작업은 비동기적으로 I/O가 연결됨을 뜻합니다. 비동기적 이라는 것은 상대방의 상태와 관계없이 일반적으로 동작한다는 것인데, 일반적인 MMORPG 시스템에서 클라이언트가 받을 준비가 되어있다고 신호를 보낸 이후에 서버는 계속해서 요청을 기다리고 있는 것이 아니고 모든 상황에 대한 업데이트를 클라이언트의 상태와 무관하게 보내야만 합니다. 때문에 IOCP가 사용됩니다. 또한 IOCP기술은 무수히 많은 클라이언트가 접근 되었을 때 무수히 많은 thread가 생성이 되는 것을 방지해주기도 합니다. 따라서 서버의 CPU가 과부하 되는 것을 방지하고 이를 관리하게 됩니다.

세 번째로 MMORPG의 경우 일반적으로 Client/Server Model을 이용합니다. MMORPG가 아닌 MORPG의 경우에는 클라이언트가 방장이 되어 P2P Model로 구현되는 경우도 있지만, MMORPG의 경우에는 모든 과정을 서버에서 처리하기 위하여 Client/Server Model을 이용합니다. MMORPG가 Client/Server Model을 이용하는 이유는 보안에 매우 강력하기 때문입니다. 만약 클라이언트가 비정상적으로 게임을 플레이 할 때에 서버는 MMORPG를 운영하는 회사에만 존재하기 때문에 비정상적인 유저를 찾아내기에 매우 쉽습니다. 하지만 단점도 존재하는데 대용량 서버를 구현할 때에 많은 돈이 필요하다는 점입니다. 서버가 모든 데이터를 받아서 처리하기 때문에 서버 운영에 있어서 비용이 많이 들게 됩니다. 이것에 반대로 MORPG의 경우 P2P로 구성한다면, 대용량 서비스에 대해 비용을 많이 절감시킬 수 있습니다.

네 번째로 P2P Model에서의 홀펀칭 기술입니다. MMORPG의 경우에는 위에서 설명하였듯이 대부분의 경우에는 Client/Server Model로 구현되지만 특이한 경우로 P2P Model이 사용되는 경우도 있습니다. 예를 들어 사용자간 이동을 조금 더 원활하게 보이도록 하는 것이나, MMORPG 내부에서 또 다른 파티를 이용한 던전 탐험시스템의 경우P2P Model이 필요하게 됩니다. 그럼 다시 본론으로 돌아와서 P2P Model의 경우에는 클라이언트가 호스트가 되어야 하는데 방화의 NAT라는 장애물 때문에 호스트가 되지 못하는 상황이 발생되게 됩니다. 이 상황을 해결하기 위한 것이 홀펀칭 기술입니다.

홀펀칭을 하는 과정은 다음과 같습니다. 가장 먼저 클라이언트가 NAT 외부의 서버와 진행하는 과정을 거칩니다. 사설 IP를 사용하는 클라이언트는 자신이 NAT 외부의 Peer와 어떤 공인 IP/PORT로 통신하고 있는지 알 수 없는데, 이 과정은 이것에 대한 정보를 획득하는 과정입니다. NAT 하위에 있는 클라이언트는 NAT 밖에 존재하는 게임서버에 빈 패킷을 보내게 되는데 이때 이 빈 패킷에 TCP 헤더에 사설 IP/PORT가 아닌 NAT가 할당 받은 공인IP/PORT를 부여하여 전송시켜 서버로 전달하고 서버는 해당 정보를 페이로드에 기록하여 클라이언트에게 다시 전달해주게 된다. 이 과정의 결과로 클라이언트는 자신이 어떤 공인 IP/PORT로 외부와 통신하고 있는지 알게 됩니다.

다음 과정으로 자신이 알고 있는 공인 IP/PORT에 대해서 P2P를 해야 되는 유저들에게 이 정보에 대해서 전달해주게 된다. 이 과정을 통하여 서로 서로 통신이 가능한 모든 상태를 알게 된다.

마지막 과정으로 클라이언트 Peer는 다른 클라이언트에 접속하기 위하여 3가지 방법으로 시도하게 됩니다. 사설 IP/PORT로 보내거나 공인 IP/PORT로 전송하여 연결이 성공한 경우에 홀펀칭이 성공하였다고 말하게 됩니다. 이 두 과정으로 성공하지 못한다면 릴레이 서버에게 패킷 전송을 위임하여 연결하게 되며, 이는 Client Server Model 구조임으로 항상 성공하게 됩니다. 홀펀칭 기술은 MMORPG에서의 P2P 연결뿐 아니라 대부분의 많은 P2P 에서 방화벽을 해결하기 위해서 사용되고 있습니다.

끝으로 MMORPG 내부에는 여기에 언급되지 않은 많은 네트워크 기술이 사용되고 있습니다. 정말 너무 사소할지 모르는 TCP를 사용할 것인지 UDP를 사용할 것인지에 대해서도 속도, 신뢰성 등 몇 가지 조건하에 선택하는 프로토콜이 변경될 수 있습니다. 결론적으로 위에 언급한 4가지의 기술은 어떤 MMORPG에서는 쓰일 수도 있고 쓰지 않을 수도 있다고 생각합니다.

이것으로 MMORPG에 대한 간단한 설명과 기술에 대해서 설명을 마치도록 하겠습니다.

참고자료

- How to Support an Action-Heavy MMORPG from the Angle of Server Architecture PPT

(<http://www.slideshare.net/ujentus/cgdc2010>)

**2. Describe your opinion about the future of the MMORPG service in that how the service in the future.**

MMORPG는 대용량의 데이터를 얼마나 빨리 전송을 하고 전송을 받는 것에 집중에 될 것 같습니다. 가장 큰 이유는 MMORPG가 대부분 3D 시스템으로 구성되고 있고 사람들은 이 3D 적인 요소가 더 완벽하게 실행되기를 원하기 때문입니다. 아직도 콘솔게임이 인기 있는 이유는 게임의 품질이라고 생각합니다. 콘솔게임은 보통 인터넷게임과 다르게 엄청나게 높은 그래픽 품질을 제공하고 있습니다. 이렇게 높은 품질을 제공할 수 있는 이유는 로컬 저장소만을 이용하기 때문에 네트워크에 대한 접근이 필요가 없기 때문이라고 생각합니다. 이와 반대로MMORPG의 경우에는 네트워크에 대한 접근이 반드시 필요합니다. 게임의 품질이 점점 높아짐에 따라 많은 데이터가 컴퓨터의 로컬 저장소에 저장되고 있지만 수많은 연산과정에 대해서 클라이언트는 반드시 서버에 전송이 되어야 하고 또한 이 정보는 다른 모든 클라이언트에게 전달이 되어야 합니다. 이러한 점 때문에 대용량 데이터가 많은 손실 없이 빠르게 전파가 되는 것이 가장 중요한 발전 방향일 것 같습니다.

두 번째로 많은 사람들의 접속에 따른 데이터 처리에 대한 분산도 계속해서 발전해 나갈 것 같습니다. 현재도 많은 데이터 접속을 처리하기 위해 데이터에 대한 분산처리가 많이 발전한 상태이긴 하지만 이점에 대한 것도 더 발달하여 적은 비용으로 최대한 처리가능 하도록 변화될 것 같습니다.

세 번째로 대규모 서비스에서의 여러 세계가 아닌 단일 월드로 변화될 것 같습니다. 많은 MMORPG 사용자들은 최대한 많은 사람들과 함께 서로 즐기고, 이야기 하고 싶어합니다. 현재 많은 MMORPG에는 A 채널, B 채널 식으로 많은 World를 분산해두고 각각의 월드는 독립적으로 존재하고 있는 형태가 많이 있습니다. 하지만 대부분의 사용자의 경우에 월드간 전쟁이나 월드간 이동을 원하는 사람이 많이 있습니다. 따라서 앞으로 대규모의 서비스는 이런 월드를 구현하는 쪽으로 발전 해 나아갈 것 같습니다.

마지막으로 네트워크에 대한 관점이 아닌 다른 관점으로 보면MMORPG 서비스는 점점 현실화가 될 것 같습니다. MMORPG가 대부분의 경우에 전투 시스템에 초점이 맞추어져 있고 판타지 세계를 표현하고 있습니다. 이는 사람들의 전투에 흥미를 가지고 그것에 대해 재미를 느끼는 점 때문인데 만약 현실세계에 대한 모든 것이 실제로 보는 것과 같이 된다면 이는 변화가 될 것 같습니다. 예를 들어 내가 나무로만 구성된 통나무 집을 만들고 싶은데 이를 현실에서 할 수 없다면, 현실과 똑같은 가상의 세계에서 자신의 역할을 목수로 발달 시켜서 해당 통나무집을 만들고 이에 대한 성취감을 느끼는 방향으로 바뀔 것 같습니다. 이렇게 발전되기 위해서는 앞에서 말했듯이 3D에 대한 높은 품질의 나오게 때문에 네트워크 시스템에 대한 대용량 데이터 처리가 가장 우선시 발전되어야만 이러한 발전 방향으로 나아갈 수 있다고 생각합니다.

참고자료

- 차세대 MMORPG 서비스 아키텍처 PPT

(<http://www.slideshare.net/freefish/nextgeneration-mmorpg-service-architecture>)