論文
7-3-13

서비스 단계별 확장 가능한 온라인 게임 서버 구조에 대한 연구 (A study on incrementally expandable online game server architecture)

김정훈(Jeong Hoon Kim)[1)]

요 약

본 논문에서는 사용자 수 증가로 인한 확장 가능한 온라인 게임 서버 구조를 제시하고 있다. 현재 상용 서비스 중인 대부분의 온라인 게임 서버에서는 로긴서버, 캐시서버, 데이터베이스서버, 게임서버, NPC서버 등으로 구성된 하나의 서버 그룹을 운영하다가 사용자 수 증가에 따라 같은 구조의 또 다른 서버 그룹을 추가 설치하고 있다. 그러나 본 논문에서 제안하는 서버 구조는 서버를 추가 설치할 때 로긴서버, 캐시서버, 데이터베이스서버, 게임서버, NPC서버 등의 한 그룹을 추가시키지 않고 게임서버만을 추가한다. 이후 캐시서버나 데이터베이스서버에 부하가 걸릴 때에만 또 다른 서버 그룹을 추가시켜 서버 추가 비용을 최대한 줄일 수 있게 하였다.

Abstract

The purpose of this study is to propose the online game server architecture which can expand as the number of users increases. In most online game servers, there is a server group composed of a login server, a cache server, a database server, a game server, and an NPC server, and when the number of users increases, an additional server group with the same structure is installed. The server architecture proposed in this study does not install a server group composed of a login server, a cache server, a database server, a game server, an NPC server, etc., but installs a game server only. When there is a need for the cache server and database server, the required servers will be additionally installed, thus reducing costs.

논문접수 : 2006. 5. 16.
심사완료 : 2006. 6. 4.

1) 정회원 : 용인송담대학 컴퓨터게임정보과

1.서론

단 하나의 게임으로 약 1조원을 벌어들인 엔씨 소프트의 리니지 게임은 대표적인 온라인 게임으로 손꼽힌다[1]. 리니지의성공으로 MMORPG라는 확고한 장르가 성립되었으며 후발 게임 업체들도 리니지와 비슷한 게임을 만들기에 여념이 없다.

리니지는 1998년 9월에 처음으로 서비스되었는데 이때 개발에 참여한 인원은 기획자 겸 프로그래머 1명과 그래픽 디자이너 1명, 모두 2명이었고 개발기간도 1년이 채 되지 않았다[2]. 따라서 제작비용은 2사람의 1년 인건비 정도였으니 1억원 남짓한 셈이다. 그러나 최근에 개발되고 있는 게임들은 비교적 제작비가 저렴하다는 캐주얼 게임이어도 10억원이 넘어가고 보통 대작이라고 하면 100억원이 넘어가는 경우도 많다[3]. 실제로 리니지의 뒤를 이은 리니지2는 50여명의 개발인력이 약 3년 정도의 개발기간을 들여 완성하였다.

이렇게 큰 금액이 투자되는 게임 제작에 효율적인 서버 확장이라는 문제는 아주 중요한 이슈이다. 동시 사용자 수가 얼마가 될지 모르는 서비스 초기에 수십만명의 동시 사용자가 접속할 것을 가정하고 서버를 설치해 놓을 수는 없는 일이다.

사용자 입장에서 하나의 서버로 보이더라도 그 구조를 들여다 보면 로긴서버, 캐시서버, 데이터베이스서버, 게임서버, NPC서버 등의 그룹으로 구성되는 경우가 많다. 하나의 서버 그룹이 약 3~5천명을 수용할 수 있기 때문에 십만명의 동시 사용자라면 (100,000/5,000) * 5(한 서버 그룹에 포함된 컴퓨터 수) = 100대 정도의 서버 컴퓨터가 필요하다[4]. 서버 컴퓨터의 경우는 한대 당 수천만원을 호가하는 경우가 보통이므로 대당 5천만원에 100대라면 50억원이 소요되는 셈이다. 서비스 초기에 이런 투자는 불가능하다.

서비스 초기에는 단지 몇 대의 컴퓨터로 서버를 운영하다가 동시 사용자 수가 늘어나면 그때마다 서버를 추가 설치하는 구조가 되어야 바람직하다. 이때 서버의 추가 설치가 어렵고 비효율적이라면 대단히 곤란해질 수 있다. 따라서 서비스 초기에는 단순한 형태로 시작하고 늘어나는 동시 사용자 수에 맞추어 서버도 추가 설치할 수 있는 구조가 필요하다[5].

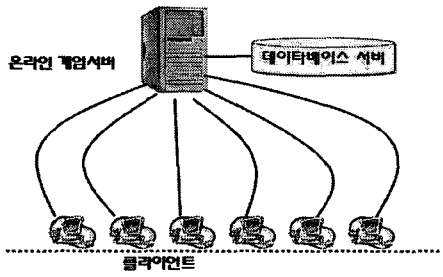
이에 본 논문에서는 서비스 단계별 추가 설치 가능한 서버 구조에 대한 연구를 하고자 한다. 늘어나는 사용자 수를 별 어려움 없이 수용할 수 있는 서버 구조를 고안한다면 투자대비 성능비를 한층 더 높일 수 있을 것이다. 본 논문은 다음과 같이 기술된다. 2장에서는 일반적인 서버 구조를 살펴보고, 3장에서는 본 논문에서 제안하는 서비스 단계별 추가 설치 가능한 온라인 게임 서버 구조에 대해서 상세히 기술한다. 마지막으로 4장에서 본 논문의 결론을 맺는다.

2.일반적인서버구조

2.1단일서버구조

초창기 온라인 게임에서 클라이언트와 서버 간에 많이 사용된 연결 방식은 2-tier 방식이었다. 2-tier 방식은 인터넷을 통해 클라이언트와 서버 간에 다대일 관계를 유지하는데, 이러한 방식을 단일 서버 구조라고 한다[6, 7].

단일 서버 구조인 경우 서버의 구현과 운영이 쉽다는 장점이 있으나, 수용 가능한 사용자 수가 제한되어 있어 사용자 수 증가에 따른 시스템 부하를 적절히 조정할 수 없다는 단점이 존재한다. CPU 및 네트워크 트래픽(network traffic), 메모리 사용량은 사용자 수 증가에 따라 급격히 증가할 수 있는데 단일 서버 구조인 경우는 규모의 확장에 따른 대비를 적절히 할 수 없다는 말이다.

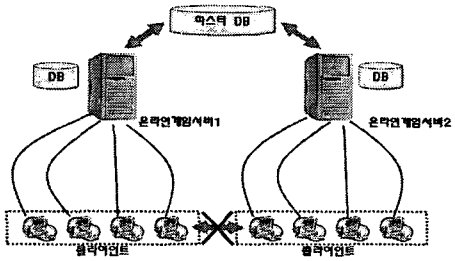


<그림 1> 단일 서버 구조

2.2분산서버구조

분산 서버 구조는 단일 서버 구조와는 달리 서버가 한 대 이상일 때 선택될 수 있는 구조이다[8,9]. 온라인 게임을 즐기는 사용자들이 급증하면서, 단일 서버로만 운영되는 시스템이 한계에 달하게 되었는데, 이러한 이유로 분산 서버 구조가 제안되었다. 분산 서버 구조는 2개 이상의 서버로 구성되며, 이 서버들은 서버들 간에 게임 데이터가 공유되는 방식과 그렇지 않은 방식으로 나누어 질 수 있다. 게임 데이터가 공유되는 방식은 replicated 서버라고 하고 그렇지 않은 경우를 non-replicated 서버라고 한다.

다음 그림에서 보여주는 replicated 분산 서버 구조는 단지 동일한 서버 구조를 가진 시스템을 2개 이상 연결하여, 클라이언트로 하여금 서버에 접속할 수 있는 사용자 수를 늘리게 해준다. 이 구조에서는 동일한 서버라고 해도 각각에 연결되어 있는 클라이언트는 다른 서버의 클라이언트와 커뮤니케이션을 할 수 없다는 단점이 있다.

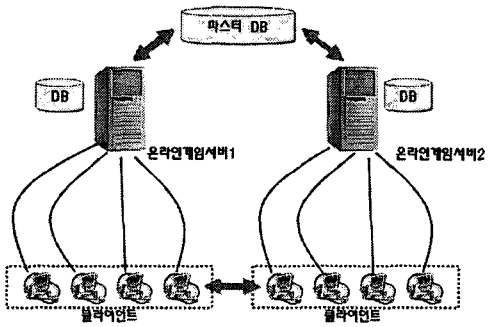


<그림 2> replicated 분산 서버 구조

replicated 분산 서버 방식은 동일한 서버를 추가하여 서비스 가능한 사용자 수를 늘리는 방식이므로 각 서버는 동일한 게임 맵 데이터를 가지게 된다. 서버는 클라이언트가 실행하는 도중에 클라이언트의 관리를 다른 서버로 이전하지 않는다.

replicated 분산 서버 방식을 선택하면 각 게임 서버에 걸리는 부하가 대체적으로 균등하게 분산되는 장점이 있다[10]. 결국 각 서버의 평균 부하가 감소되어, 증가되는 서버의 수만큼 많은 사용자를 수용할 수 있게 된다. 하지만 각 게임 서버 간의 동기화 문제가 발생될 수 있는데 이는 접속한 서버에 따라 게임 데이터가 다르기 때문에 발생된다.

non-replicated 분산 서버 구조는 데이터베이스를 통해 서버 간의 데이터를 공유할 수 있으며, 하나의 서버에서 다른 서버로의 연결전환이 가능하다. 그러나replicated 분산 서버와는 달리 서버 간의 부하가 균등하게 분산되지 않을 수 있다는 단점이 있다.



<그림 3> non-replicated 분산 서버 구조

지금까지 살펴본 바에 따르면 단일 서버 구조는 중앙 집중 구조로 운영 및 관리가 용이한 반면 많은 사용자를 수용하기 위한 확장성은 낮다. 반면 분산 서버 구조는 서버를 쉽게 추가할 수 있게 하여 확장성은 높였지만, 서버 간의 복잡한 동기화 과정과 데이터의 동시성 보장 등이 부담이 되는 단점이 있다.

3. 서비스 단계별 추가 설치 가능한 서버 구조

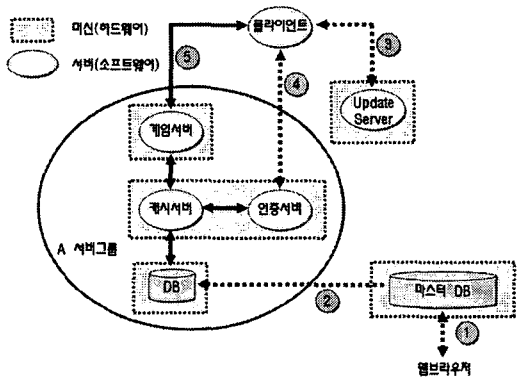
본 논문에서 제안하는 시스템은 replicated 분산 서버 구조를 변형한 것이다. 효율적인 로드 밸런싱을 위하여 서버 그룹(server group)이라는 개념을 도입한다. 서버 그룹은 사용자들에게는 하나의 서버로 보여지지만 각 서버 그룹은 업데이트서버, 로긴서버, 게임서버, 캐시서버, 로컬데이스베이스서버 등의 요소로 이루어진다.

제안 시스템 구조에서의 특징은 증가하는 사용자들을 수용하기 위해 게임서버 만을 추가로 설치하면 된다는 것이다. 추가되는 게임서버가 많아져서 캐시서버나 로컬데이터베이스서버에 부하가 걸리게 되면 그때 또 다른 서버 그룹을 추가한다. 이렇게 함으로써 최소한의 비용 투입으로 최대의 효과를 거둘 수 있다.

또한 non-replicated 분산 서버 구조의 단점인 서버들의 교착상태 문제도 본 제안 서버 구조에서는 서버 그룹을 추가함으로써 해결할 수 있다. 본 제안 서버 구조에서는 하나의 서버 그룹 내에서의 게임서버 간의 이동을 허용하지만 다수의 클라이언트와 연결되어 있는 게임서버들은 캐시서버를 통해 동일한 로컬데이터베이스를 사용하므로 replicated 분산 서버 구조에서 나타나는 문제들도 해결할 수 있다.

3.1 서비스 초기 서버 구성

서비스 초기에는 많은 장비가 투입될 수 없다. 게임의 성공 여부를 판단하기 힘든 시점이기 때문에 최소한의 장비만이 투자되어 서비스되어야 한다. 본 제안 시스템은 초기에 5개의 서버 머신(machine)이 설치된다. 물론 각 서버들은 물리적으로 같은 머신에 설치될 수도 있다. 이 경우 5개 미만의 머신으로도 서비스가 가능할 것이다. 다음은 이 내용을 그림으로 표현한 것이다. 웹브라우저를 통해 가입된 회원 정보는 마스터 데이터베이스에 저장된다. 아래 그림에서 점선은 필요에 의해 연결된 후 끊어지는 것을 표현하고 실선은 계속적으로 연결되어 있는 상태를 표현한다.



<그림 4> 초기 서비스 시 서버 구조

1단계

사용자는 웹브라우저를 통해 회원에 가입한다. 사용자는 회원 가입 시 서버 그룹을 선택하는데, 선택된 서버 그룹 정보를 포함하여 회원 정보는 마스터데이터베이스에 저장된다.

2단계

사용자가 선택한 서버 그룹의 로컬데이터베이스에 회원 정보가 전송된다.

3단계

클라이언트는 업데이트 서버에 연결해서 변경사항(추가된 서버 그룹 리스트 정보나 클라이언트의 업데이트 정보)이 있다면 변경사항을 다운로드 받는다. 이 작업이 끝나면 업데이트 서버와는 연결을 끊는다.

4단계

클라이언트는 인증서버에 연결되어 로컬데이베이스서버를 통해 인증을 받는다.

5단계

성공적으로 인증이 끝나면 게임서버에 연결된다.

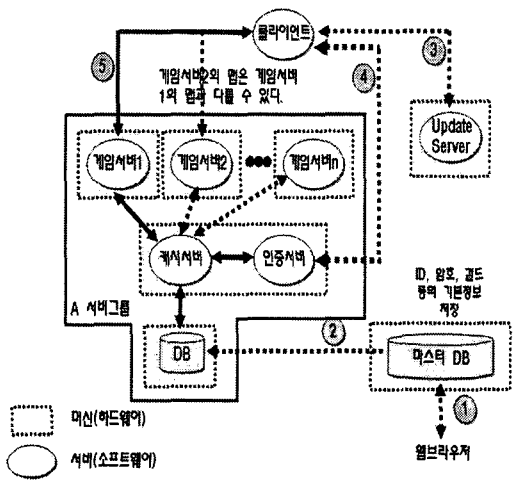
3.2 하나의 서버 그룹 내에서의 서버 확장시

위 3.1의 단계에서 사용자들이 증가하게 되면 이를 수용하기 위해 다음 그림과 같이 게임서

버를 추가한다. 다음 그림은 n 개의 게임서버를 추가로 설치한 그림이다. 이 구조에서는 게임서버1과 게임서버2가 서로 다른 맵 정보를 가질 수 있다. 따라서 이 부분만을 보면 위 2.2절에서살펴본 non-replicated 분산 서버 구조가 된다. 사용자들은 게임서버1에 접속되었다가 자연스럽게 게임서버2로 접속을 바꿀 수 있는데 이때 서버 접속 변경을 사용자가 알게 구현할 수도 있고, 모르게 구현할 수도 있다.

예를 들어 게임서버1이 던전1의 맵을 가지고 있고 게임서버 2가 던전2의 맵을 가지고 있을 때 어떤 사용자가 던전1에서 던전2의 입구를 통해 이동하면 자연스럽게 서버를 이동할 수 있다. 게임을 진행하고 있는 사용자들이 보기에는 단지 다른 장소로 이동하는 것처럼 보일 뿐이지만 접속된 서버가 바뀐 것이다. 만약 본 게임으로 들어가기 전 대기실과 같은 부분이 있다면 이 대기실에서 게임서버를 선택하게 할 수도 있다. 이러한 과정은 사용자가 직접 맵(서버를 의미)을 선택하는 형태이다. 이렇게 함으로써 로드 밸런싱을 자연스럽게 유도할 수 있고 더 많은 사용자들을 수용할 수 있다.

이 구조에서는 게임서버들 간의 아이템은 그대로 유지되고 아이템 변동 내용은 즉시 캐시서버를 통해 데이터베이스에 반영된다.



<그림 5> 게임서버의 추가

1단계

사용자는 웹브라우저를 통해 회원에 가입한다. 사용자는 회원 가입 시 서버 그룹을 선택하는데, 선택된 서버 그룹 정보를 포함하여 회원 정보는 마스터데이터베이스에 저장된다.

2단계

사용자가 선택한 서버 그룹의 로컬데이터베이스에 회원 정보가 전송된다.

3단계

클라이언트는 업데이트 서버에 연결해서 변경사항(추가된 서버 그룹 리스트 정보나 클라이언트의 업데이트 정보)이 있다면 변경사항을 다운로드 받는다. 이 작업이 끝나면 업데이트 서버와는 연결을 끊는다.

4단계

클라이언트는 인증서버에 연결되어 로컬데이베이스서버를 통해 인증을 받는다.

5단계

성공적으로 인증이 끝나면 게임서버에 연결된다.

6단계

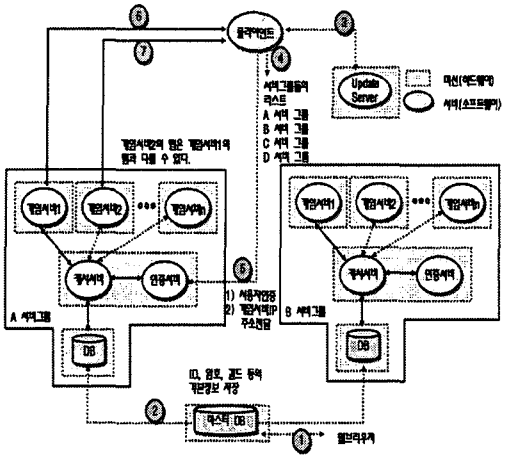
만약 사용자들이 다른 게임서버에 접속하고 싶으면 한 서버 그룹 내의 타 게임서버에 자유롭게 접속할 수 있다.

3.3또다른서버그룹이추가되는경우

별개의 서버 그룹이 추가되는 구조에서는 한 서버 그룹 내에서의 자유로운 이동은 허용하나 타 서버 그룹으로의 이동은 허용하지 않는다. 만약 A 서버 그룹에서 게임을 하던 사용자가 B 서버 그룹에서 게임을 하고 싶다면 A 서버 그룹에서 획득한 아이템이나 경험치를 B 서버 그룹에서는 포기해야 한다. 한 서버 그룹 내에서의 아이템 이동은 허용되나 다른 서버 그룹으로 이동시에는 기존 아이템들을 사용할 수 없다.

이와 같은 구조는 효율적인 서버 확장을 지원하면서도 하나의 서버 그룹 내에서는 게임서버 간의 이동을 허용한다는 장점이 있다. 서버 그룹을 독립적으로 운영하게 되면 A 서버 그룹

사용자는 B 서버 그룹에서 다른 게임을 즐기는 것과 같은 부가적인 효과를 누릴 수도 있다.



<그림 6> 별개의 서버 그룹이 추가된 경우

1단계
사용자는 웹브라우저를 통해 회원에 가입한다. 사용자는 회원 가입 시 서버 그룹을 선택하는데, 선택된 서버 그룹 정보를 포함하여 회원 정보는 마스터데이터베이스에 저장된다.

2단계
사용자가 선택한 서버 그룹의 로컬데이터베이스에 회원 정보가 전송된다.

3단계
클라이언트는 업데이트 서버에 연결해서 변경사항(추가된 서버 그룹 리스트 정보나 클라이언트의 업데이트 정보)이 있다면 변경사항을 다운로드 받는다. 이 작업이 끝나면 업데이트서버와는 연결을 끊는다.

4단계
사용자는 서버들의 리스트 중 하나를 선택한다(A서버 그룹, B서버 그룹, C서버 그룹 등).

5단계
클라이언트는 선택된 서버 그룹의 인증서버에 연결되어 로컬데이베이스서버를 통해 인증을 받는다.

6단계
성공적으로 인증이 끝나면 게임서버에 연결된다.

7단계
만약 사용자들이 다른 게임서버에 접속하고 싶으면 한 서버 그룹 내의 타 게임서버에 자유롭게 접속할 수 있다.

현재 상용 서비스되고 있는 대부분의 서버 구조는 이와 같이 서버 그룹 간의 이동을 허용치 않다는 점에서 본 논문에서 제안하는 서버 구조와 비슷하다. 그러나 본 논문의 서버 구조는 한단계 더 나아가 한 서버 그룹 내에서의 게임서버 간 이동을 허용하고 있다. 이는 서버 구조를 확장시킬 때 로긴서버, 캐시서버, 데이터베이스서버, 게임서버, NPC서버 등 모두를 추가시키지 않고 꼭 필요한 게임서버만을 추가하여 늘어나는 사용자 수를 수용할 수 있게 한다. 물론 몇 대의 게임서버를 추가한 후 캐시서버나 로컬데이터베이스서버 등에 부하가 증가되면 이때에는 또 다른 서버 그룹을 추가시켜야 할 것이다.

4.결 론

2장과 3장에서 소개한 여러가지 서버 구조들을 서로 비교 및 요약하면 다음과 같다. 먼저 단일 서버 구조는 구현과 운영이 용이하다는 특징을 가지고 있다. 또한 소수의 사용자일 경우 적합하다. 그러나 증가하는 사용자를 감당하기 어렵다는 단점이 있다. Replicated 분산 서버 구조는 접속이 분산되어 로드밸런싱이 잘 유지되는 특징이 있지만 서버 간의 커뮤니케이션이 불가능하다. Non-replicated 분산 서버 구조에서는 로드밸런싱이 잘 이루어지지 않으나 다른 서버와의 커뮤니케이션이 가능하다는 장점이 있다.

본 논문에서 제안하는 서버 구조는 사용자 수 증가에 의한 온라인 게임 서버 부하를 줄일 수 있는 효율적인 서버 구조이다. 이를 위해

replicated 분산 서버 구조와 non-replicated 분산 서버 구조를 적절히 혼용하는 방식을 채택하여 최소의 비용이 들어가게 하였다. 또한 본 제안은 초기 구축비가 저렴하여 손쉽게 도입할 수 있으며 대규모 사용자들을 무리없이 수용할 수 있다는 장점이 있다.

본 논문의 연구를 조금 더 확장하여 성능을 정확히 측정할 수 있는 도구를 고안한다면 게임 제작 업체들에게 실질적인 도움을 줄 수 있을 것으로 생각된다.

참고문헌

[1]. 국산 게임누적매출액 1000억 돌파 대작 속출, 전자신문, http://www.etnews.co.kr/news/detail.html?id=200601230184, 2006-01-24

[2] .리니지 5년이 남긴 것, 전자 신문http://news.naver.com/news/read.php?mode=LSD&office_id=030&article_id=0000044730&section_id=105&menu_id=105, 2003-09-27

[3]. 온라인게임 '빅3' 대격돌, 서울경제, http://news.naver.com/news/read.php?mode=LSD&office_id=011&article_id=0000118708&section_id=105&menu_id=105, 2006-02-05

[4]. 소프트웨어 산업 도약 원년 기반 조성의 해, ZDNet Korea, 2006-01-23 http://news.naver.com/news/read.php?mode=LSD&office_id=092&article_id=0000007126&section_id=105&menu_id=105

[5]. Dan Esbensen, Online Game Architecture: Back-end Strategies, Gamasutra, 2005-03-10

[6]. Bernier, Latency Compensating Methods in Client/Server In-Game Protocol Design and Optimization, GDC 2001

[7]. N. Baughman and B. Levine. Cheat-proof playout for centralized and distributed online games. In Proc. Infocom 2001, April 2001.

[8] C. Diot, L. Gautier, A Distributed Architecture for Multiplayer Applications on the Internet, IEEE Networks Magazine, vol. 13, no. 4.

[9] D. Saha, S. Sahu, A. Shaikh, A Service Platform for OnLine Games, in: Proceedings of NetGames, 2003.

[10]. D. Min, E. Choi, D, Lee, and B. Park. A load balancing algorithm for a distributed multimedia game server architecture. In Proceeding of IEEE Multimedia systems Conference, June 1999.

김정훈

1991년 2월 : 서울시립대학교 컴퓨터통계학과 (이학사)
1994년 2월 : 연세대학교 컴퓨터과학과(이학석사)
1994년 1월 ~ 1999년 5월 : (주)현대전자, (주)현대정보기술 선임연구원
1999년 6월 ~ 2001년 11월 : (주)엔씨소프트 리니지토너먼트 개발팀장
2002년 3월 ~ 2003년 8월 : (주)소프트젠 모바일 사업부 이사
2003년 8월 ~ 현재 : 용인송담대학 컴퓨터게임정보과 전임강사
1997년 12월 정보관리기술사
관심분야 : 온라인게임, 모바일게임