추가 내용

MM4220 게임서버 프로그래밍

정내훈

내용

- BOOST/ASIO
- 모바일 게임 서버
  - 쿠키런
- 유용한 라이브러리
- Case Study (Eve, Tera)

BOOST

- 개요
  - C++ 프로그래밍을 위한 각종 라이브러리 모음
    - 선형 대수, 난수, 멀티쓰레딩, 영상 처리, 정규식 표현, 유닛 테스트, 비동기 I/O, reflection
- 성격
  - 대부분이 template로 구현되어서 따로 header 파일만 포함하면 되는 것들이 많다.
    - asio는 library도 필요하다.
- 특징
  - C++표준을 채택된 것들이 많다.
- 위치
  - Www.boost.org

ASIO

- ASIO(ASynchronous Input Output)
- Boost에 존재하는 비동기 I/O를 위한 라이브러리
- Windows의 IOCP에 많은 영향을 받음
  - Visual studio로 컴파일 하면 내부가 IOCP로 구현되어 있음
- 모든 동작이 call back함수를 사용한 비동기로 이루어짐

ASIO

- 구성 클래스
  - class boost::asio::io_service
  - 사용법
    - tcp::acceptor 에 연결
    - async_accept 호출
      - call back에서 async_read_some 호출
    - io_servce::run()호출 후
      - async_read_some의 call back함수를 중심으로 동작.

ASIO

- IOCP와의 차이

```
worker_thread()
{
    GQCS(g_iocp,…);
    if (is_recv)
        ProcessBuffer();
}
```

```
worker_thread()
{
    io_service->run();
}
```

```
WSARecv(sock, WSABUF)
```

```
async_read(sock,
    buffers,
    ProcessBuffer);
```

ASIO

- Receive

```
socket.async_read_some(boost::asio::buffer(data, size), handler);
```

또는

```
boost::asio::async_read(socket, boost::asio::buffer(data, size), handler);
```

  - Socket에 대해 recv를 요청하고 결과를 handler 함수를 통해서 얻음.
  - Handler 함수는 worker 스레드에서 실행

```
void handler(
 const boost::system::error_code& error, // Result of operation.
 std::size_t bytes_transferred // Number of bytes read.
);
```

암호

- “algorithm”

ASIO

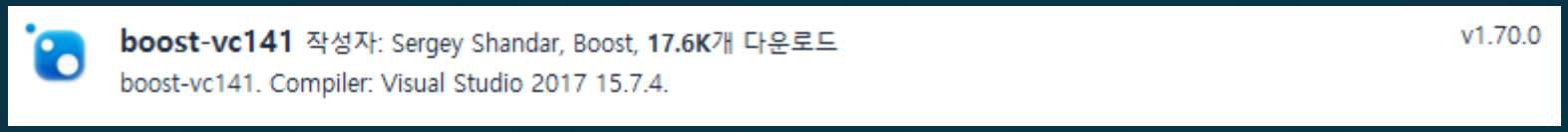
- 특징
  - async_read와 async_read_some의 차이
    - async_read는 읽어야 할 byte 수를 지정해 주어야 한다.
    - async_read는 내부적으로 여러 개의 async_read_some으로 구현되며 이때 실행 순서 동기화를 해야 하고, strand 객체가 필요하다.
    - strand : 함수의 실행을 lock을 걸지 않고 kernel내부에서 순차적으로 실행, (꼭 같은 쓰레드에서 실행은 가능하나 동시에 여러 쓰레드는 불가능)

ASIO

- 특징
  - async_write와 async_write_some의 차이
    - async_write는 async_write_some들로 구현 됨
    - 여러쓰레드에서 async_write_some을 할 때 순서가 서로 섞일 수 있음
    - 따라서 동기화를 해야 하고, strand 객체가 필요하다.
    - async_write_some에 패킷 하나씩을 매핑하면 strand가 필요 없음.

ASIO

- 실습
  - 설치
    - www.boost.org에서 다운받아서 설치
    - 또는, Nuget사용



모바일 게임 서버

- 특징
  - OffLine으로 게임 진행 후 OnLine으로 정산
    - 해킹에 매우 취약.
    - Web 서비스로 구현 가능
  - 서버 분리 없이 통합 world
    - Database의 부하가 큼
    - Web서비스의 특징 상 서버에 상태가 저장되지 않으므로 Database의 부하가 특히 큼
      - NoSQL을 사용한 data caching이 필요.

모바일 게임 서버

- 온라인 플레이의 추가
  - 실시간 PvP, 실시간 길드전, FPS, MMORPG
  - MMORPG는 PC와 동일
  - MMORPG이외는 인던 형식의 플레이 (스타의 배틀넷과 비슷)
    - HotSpot이 없는 서버, 시야의 개념이 없는 서버
    - 구현난이도가 낮음
- SPOF(Single Point Of Failure) 배제가 필수

모바일 게임 서버

- 구조 : 기능별로 복수의 서버가 분산 처리
  - Auth Server
  - Web Game Server
  - L4 Switch
  - DB Server
  - Match Server
  - Game Server or Relay Server
  - Chatting Server
  - CDN Server

모바일 게임 서버

- Case Study
  - 쿠키런 서버 개발기
    - 클라우드 서버를 사용하는 이유와 방법
    - http://www.slideshare.net/serialxnet/1-35304689

유용한 라이브러리

- ProudNet
  - 저렴한 게임 엔진
  - 성능 좋고, 쓰기 쉬움
  - 홀펀칭에 특화
- ProtocolBuffer
  - 플랫폼과 언어에 독립적인 패킷 기술 라이브러리
  - 사용하기 쉽고, Configuration 파일로도 사용하기 좋음
- FlatBuffer
  - ProtocolBuffer의 단점인 성능을 보완
  - 사용 편의성을 희생