Module06-06
C++ ACE: Proactor 框架

C++ ACE - Proactor 框架

- ACE 简介
- I/O 相关对象
- Reactor 框架
- Service Configuration 框架
- Task 框架
- Acceptor-Connector 框架
- → Proactor 框架
- 杂项

C++ ACE - Proactor 框架 - 概要

- Proactor 框架：
  - 概要
  - 异步 I/O 工厂类
  - ACE_Handler
  - 前摄式 Acceptor-Connector
  - ACE_Proactor

Proactor 框架 - 概要

- 关于 Proactor 框架
  - Proactor 框架引入异步 I/O 机制，既保留了反应式框架的事件多路分离，避免多线程的开销，同时还缓和了反应式的同步 I/O 的瓶颈效应，该模式允许应用通过以下两个阶段来执行 I/O 操作：
    - 应用可以在多个 I/O 句柄上并行的发起一个或多个异步 I/O 操作，且不需等待它们完成
    - 每个操作完成时， OS 会通知应用定义的完成处理器，由它随后对已经完成的 I/O 操作的结果进行处理

Proactor 框架 - 概要

- 关于 Proactor 框架（续）
  - Proactor 框架负责：
    - 发起异步 I/O 操作
    - 保存每个操作的参数，并将它们中继给完成处理器
    - 等待指示这些操作的结束的完成事件
    - 将完成事件多路分离给与其相关联的完成处理器，并且
    - 分派处理器上的挂钩方法，以通过应用定义的方式处理这些事件

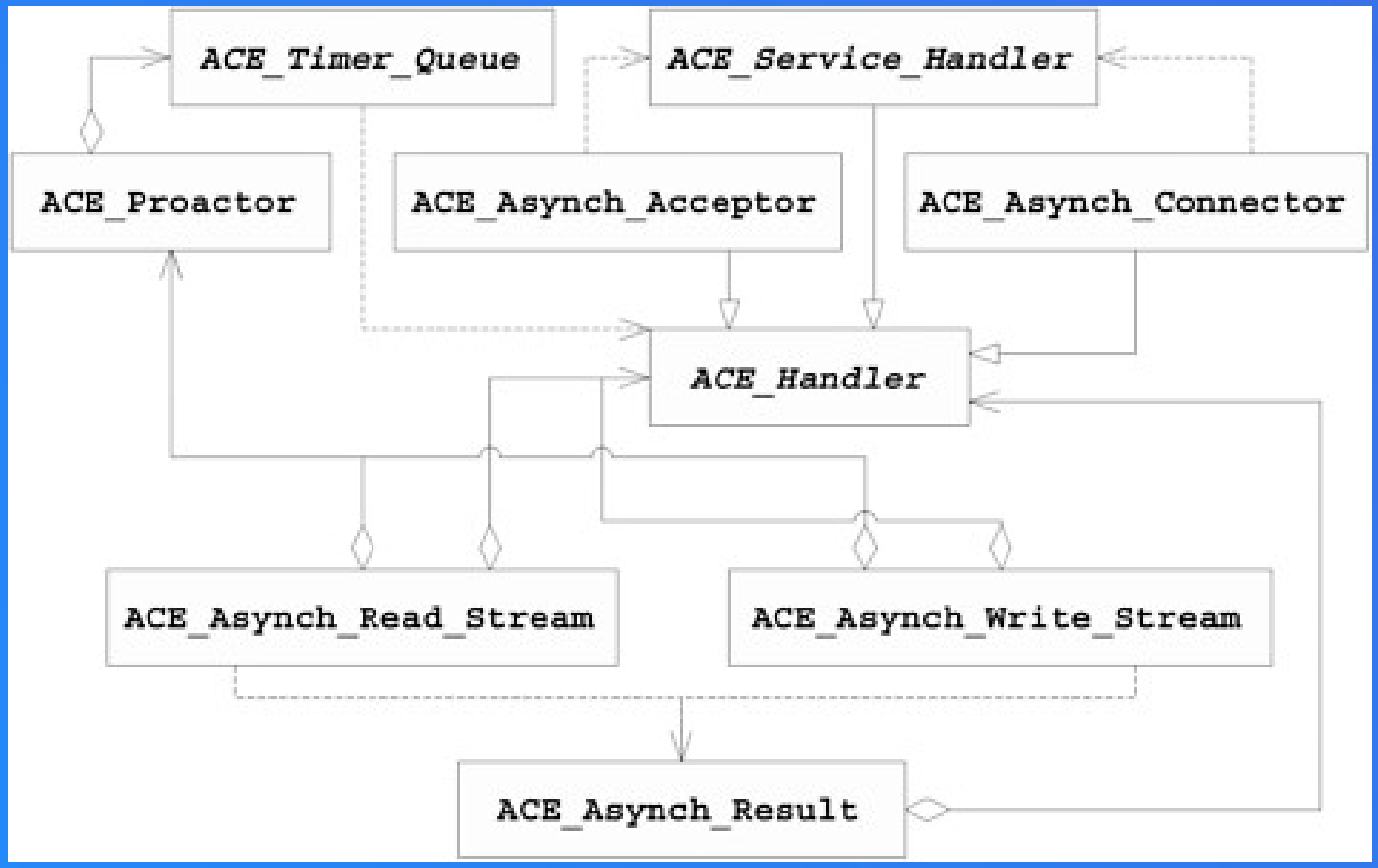
Proactor 框架 - 概要

- Proactor 框架的主要参与者

ACE 类	说明
ACE_Handler	定义用于接受异步 I/O 操作的结果和处理定时器到期的接口
ACE_Asynch_Read_Stream ACE_Asynch_Write_Stream ACE_Async_Result	在 I/O 流上发起异步读和异步写操作，并使每个操作与一个用于接收操作结果的 ACE_Result 对象关联起来
ACE_Asynch_Acceptor ACE_Asynch_Connector	Acceptor-Connector 模式的一种实现，它异步的建立新的 TCP/IP 连接的服务
ACE_Service_Handler	定义 ACE_Asynch_Acceptor 和 ACE_Asynch_Connector 连接工厂的目标，且提供挂钩方法来初始化通过 TCP/IP 连接的服务
ACE_Proactor	管理定时器和异步 I/O 完成事件多路分离。这个类是 ACE_Reactor 框架中 ACE_Reactor 类的类似物

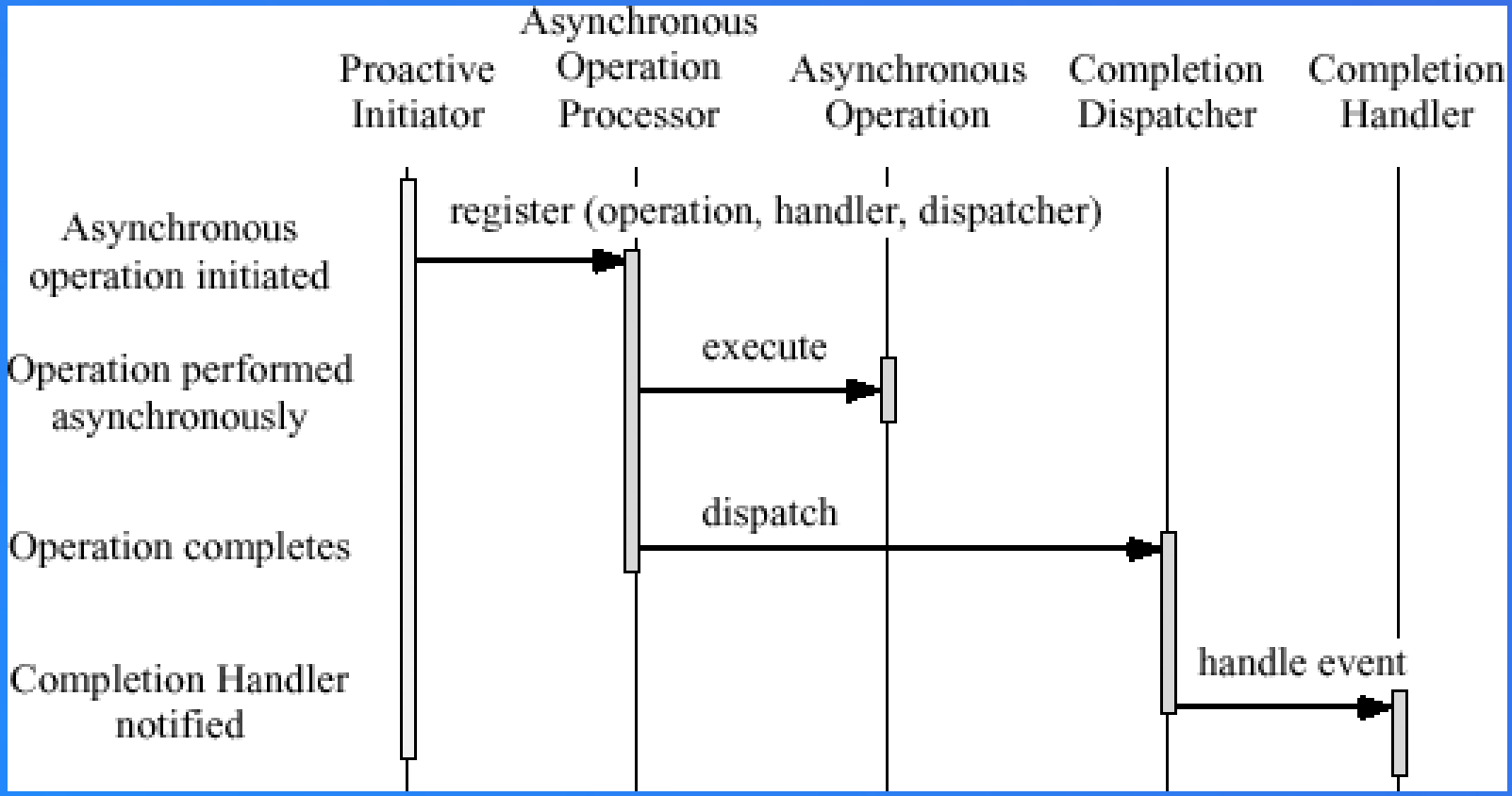
Proactor 框架 - 概要

- Proactor 框架类关系图



Proactor 框架 - 概要

- Proactor 框架交互图示



Proactor 框架 - 概要

- Proactor 框架类交互过程
  - 前摄发起器发起操作：为执行异步操作，应用在 Asynchronous Operation Processor 上发起操作。例如， Web 服务器可能要求 OS 在网络上使用特定的 socket 连接传输文件。要请求这样的操作， Web 服务器必须指定要使用哪一个文件和网络连接。而且， Web 服务器必须指定（ 1 ）当操作完成时通知哪一个 Completion Handler ，以及（ 2 ）一旦文件被传输，哪一个 Completion Dispatcher 应该执行回调。
  - 异步操作处理器执行操作：当应用在 Asynchronous Operation Processor 上调用操作时，它相对于其他应用操作异步地运行这些操作。现代操作系统（比如 Solaris 和 Windows NT ）在内核中提供异步的 I/O 子系统。

Proactor 框架 - 概要

- Proactor 框架类交互过程（续）
  - 异步操作处理器通知完成分派器：当操作完成时， Asynchronous Operation Processor 取得在操作被发起时指定的 Completion Handler 和 Completion Dispatcher 。随后 Asynchronous Operation Processor 将 Asynchronous Operation 的结果和 Completion Handler 传递给 Completion Dispatcher ，以用于回调。例如，如果文件已被异步传输， Asynchronous Operation Processor 可以报告完成状态（比如成功或失败），以及写入网络连接的字节数。
  - 完成分派器通知应用： Completion Dispatcher 在 Completion Handler 上调用完成挂钩，将由应用指定的任何完成数据传递给它。例如，如果异步读取完成，通常一个指向新到达数据的指针将会被传递给 Completion Handler 。

Proactor 框架 - 异步 I/O 工厂类

- Proactor 框架：
  - 概要
  - 异步 I/O 工厂类
  - ACE_Handler
  - 前摄式 Acceptor-Connector
  - ACE_Proactor

Proactor 框架 - 异步 I/O 工厂类

- 关于异步 I/O 工厂类
  - 异步 I/O 机制在各种操作系统下，实现的接口差异较大：
    - Windows： ReadFile() 和 WriteFile() 系统函数既可以执行同步 I/O 。也可以发起 overlapped I/O 操作
    - POSIX： aio_read() 和 aio_write() 函数分别用于发起异步读写操作，这两个函数使用的 read() 和 write() （以及 Socket recv() 和 send() 函数）是分开的

Proactor 框架 - 异步 I/O 工厂类

- 异步 I/O 工厂类能力
  - ACE_Asynch_Read_Stream 和 ACE_Asynch_Write_stream 两个工厂类，使用它们可以发起异步的 read() 和 write() 操作，这个两个类提供以下能力：
    - 它们可在面向流的 IPC 机制（如 TCP Socket ）上发起异步操作
    - 它们将 I/O 句柄、 ACE_Handler 对象，以及 ACE_Proactor 绑定在一起，以正确而高效的处理 I/O 完成事件
    - 它们创建一种将操作的各参数从 ACE_Proactor 框架携带到它的完成处理器 的对象
    - 它们派生自 ACE_Asynch_Operation ，后者提供的接口可用于初始化对象和请求取消未完成的 I/O 操作

Proactor 框架 - 异步 I/O 工厂类

- 关于 Result 对象
  - ACE_Asynch_Read_Stream 和 ACE_Asynch_Write_stream 内部定义了一个嵌套类型 Result，其派生于 ACE_Asynch_Result。用于表示操作和其参数之间的绑定

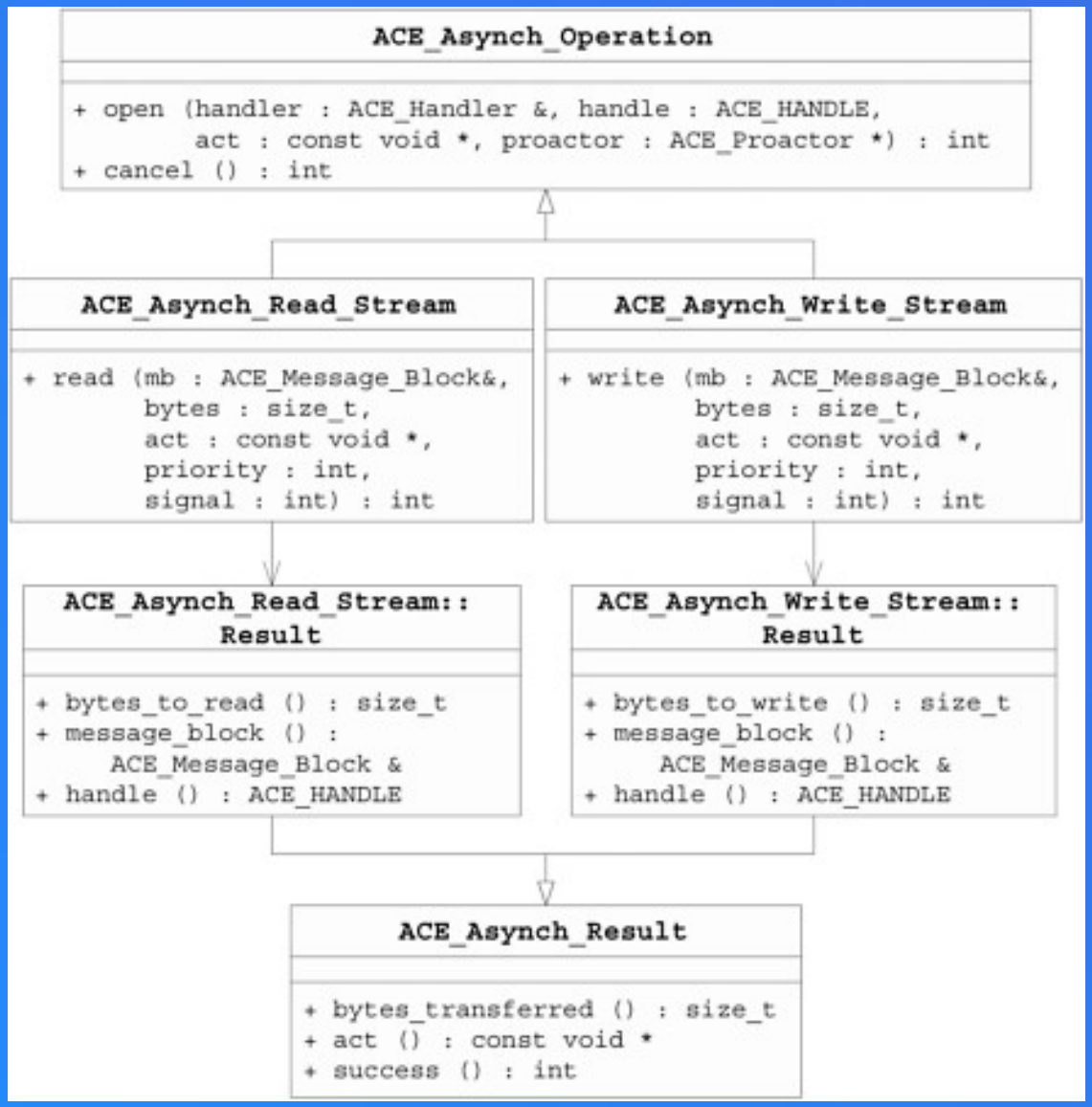
Proactor 框架 - ACE_Handler

- Result 类关键方法
  - 调用 handle_read_stream() 和 handle_write_stream() 挂钩方法时传入的是指向已完成的异步操作相关联的 Result 对象的引用，该对象的一些有用方法：

方法	说明
success()	指示异步操作是否成功
handle()	获取异步 I/O 操作所使用的 I/O 句柄
message_block()	获取指向在操作中所使用的 ACE_Message_Block 的引用
bytes_transferred()	指示在异步操作中实际传输了多少字节
bytes_to_read()	指示请求 read() 操作读取的字节是多少
bytes_to_write()	指示请求 write() 操作写出的字节是多少

Proactor 框架 - 异步 I/O 工厂类

- 异步 I/O 对象类图



Proactor 框架 - 异步 I/O 工厂类

- 异步 I/O 对象与 IPC 对象
  - ACE_Asynch_Read_Stream 和 ACE_Asynch_Write_Stream 并没有封装任何底层的 IPC 机制，仅仅定义了发起异步 I/O 操作的接口，其好处：
    - 允许我们在 ACE Proactor 框架中复用 ACE 的 IPC 包装类，如 ACE_SOCK_Stream 和 ACE_SPIPE_Stream，避免重新创建一组平行的、只能用于 Proactor 框架的 IPC 类
    - 通过只开放必需的 I/O 操作发起器，强制性的建立了一个结构来避免 I/O 句柄的误用
    - 通过将 I/O 句柄给予异步操作工厂，方便了开发者将相同的 IPC 类用于同步和异步 I/O

      ACE Proactor 框架还包含有用于发起数据报 I/O 操作的工厂类 (ACE_Asynch_Read_Dgram 和 ACE_Asynch_Write_Dgram), 以及用于发起文件 I/O 操作的工厂类 (ACE_Asynch_Read_File、 ACE_Asynch_Write_File 等)

Proactor 框架 - ACE_Handler

- Proactor 框架：
  - 概要
  - 异步 I/O 工厂类
  - ACE_Handler
  - 前摄式 Acceptor-Connector
  - ACE_Proactor

Proactor 框架 - ACE_Handler

- 关于 ACE_Handler
  - ACE_Handler 是 ACE Proactor 框架中的所有异步完成处理器的基类，这个类提供以下能力：
    - 它提供了多个挂钩方法来处理 ACE 中定义的所有异步 I/O 操作的完成，包括连接建立和在 IPC 流上的 I/O 操作
    - 它提供了一个挂钩方法来处理定时器到期

Proactor 框架 - ACE_Handler

- ACE_Handler 类图

```
                    ACE_Handler
# proactor_ : ACE_Proactor *
+ handle () : ACE_HANDLE
+ handle_read_stream (result :
          const ACE_Asynch_Read_Stream::Result &)
+ handle_write_stream (result :
          const ACE_Asynch_Write_Stream::Result &)
+ handle_time_out (tv : const ACE_Time_Value &,
                   act : const void *)
+ handle_accept (result :
          const ACE_Asynch_Accept::Result &)
+ handle_connect (result :
          const ACE_Asynch_Connect::Result &)
```

Proactor 框架 - ACE_Handler

- ACE_Handler 类关键方法

方法	说明
handle()	获取这个对象所用的句柄
handle_read_stream()	挂钩方法，在 ACE_Asynch_Read_Stream 发起的 read() 操作完成时调用
handle_write_stream()	挂钩方法，在 ACE_Asynch_Write_Stream 发起的 write() 操作完成时调用
handle_time_out()	挂钩方法，在通过 ACE_Proactor 调度的定时器到期时调用

Proactor 框架 - 前摄式 Accepter-Connector

- Proactor 框架：
  - 概要
  - 异步 I/O 工厂类
  - ACE_Handler
  - 前摄式 Acceptor-Connector
  - ACE_Proactor

Proactor 框架 - 前摄式 Accepter-Connector

- 关于 ACE_Asynch_Acceptor
  - ACE_Asynch_Acceptor 是 Acceptor-Connector 模式中的接受器角色的另一种实现，该类提供了以下能力：
    - 发起异步的被动连接建立
    - 充当工厂，为每个被接受的连接创建一个新的服务处理器
    - 可以取消先前发起的异步 accept()
    - 提供了一个挂钩方法来在新的连接建立时获取对端的地址
    - 提供了一个挂钩方法来在初始化新服务处理器前确认对端

Proactor 框架 - 前摄式 Accepter-Connector

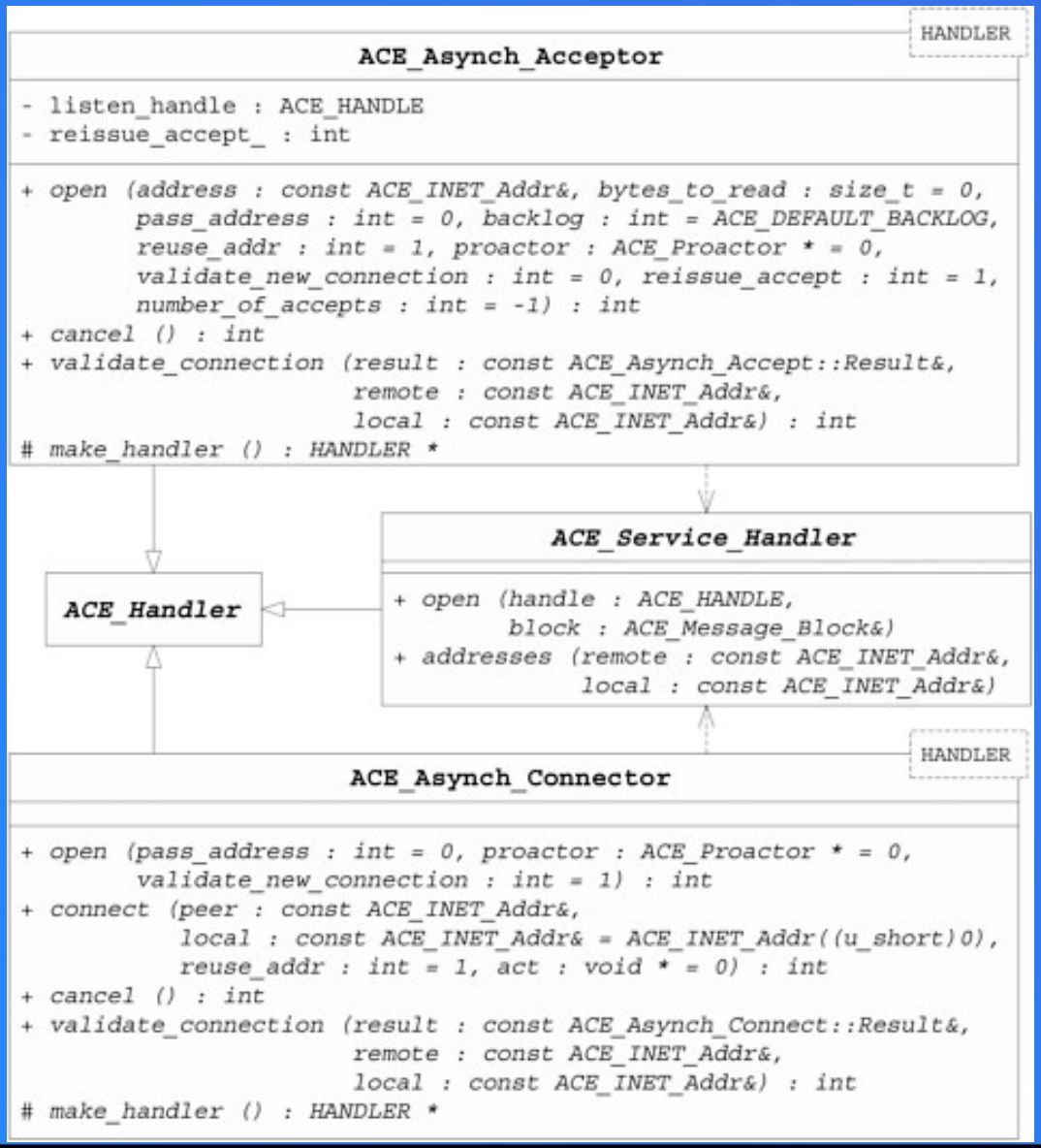
- 关于 ACE_Asynch_Connector
  - ACE_Asynch_Connector 在 ACE_Proactor 框架的实现中扮演了 Acceptor-Connector 模式中的连接器角色，该类提供了以下能力：
    - 发起异步的主动连接建立
    - 充当工厂，为每个已完成的连接创建一个新的服务处理器
    - 可以取消先前发起的异步 connect() 操作
    - 提供了一个挂钩方法来在新连接被建立时获取对端地址
    - 提供了一个挂钩方法来在初始化新服务处理器之前确认对端
  - 由于 ACE Proactor 没有封装 I/O 句柄，不依赖于具体的 IPC I/O 对象，所以可以使用于无连接的 IPC 机制（如 UDP 和文件 I/O ），不需进行连接设置，可以直接与 ACE Proactor 的 I/O 工厂类一起使用

Proactor 框架 - 前摄式 Accepter-Connector

- 关于 ACE_Service_Handler
  - 该类提供了以下能力：
    - 它为初始化和实现网络化应用服务提供基础，充当 ACE_Asynch_Acceptor 和 ACE_Asynch_Connector 连接工厂的目标
    - 接收相连对端的地址
    - 同继承了处理异步完成事件的能力，因为该类继承于 ACE_Handler

Proactor 框架 - 前摄式 Accepter-Connector

- 前摄式 Acceptor-Connector 的类关系图



- ACE_Asynch_Acceptor 关键方法

方法	说明
open()	初始化和发出一个或多个异步 accept() 操作
cancel()	取消所有由该接受器发起的异步 accept() 操作
validate_connection()	挂钩方法，用于在为新连接打开服务之前确认对端地址
make_handler()	挂钩方法，用于为新连接获取处理器对象

Proactor 框架 - 前摄式 Accepter-Connector

- ACE_Asynch_Acceptor 的 handle_accept() 方法
  - 收集代表每个新连接的端点的 ACE_INET_Addr
  - 如果传给 open() 的 validate_new_connection 参数为 1，调用 validate_connection() 方法，将相连对选的地址作为参数传给它。如果 validate_connection() 返回 -1，连接被终止
  - 调用 make_handler() 挂钩方法为每个新连接获取服务处理器。缺省实现使用 operator new 来动态分配新的处理器
  - 设置新处理器的 ACE_Proactor 指针
  - 如果传给 open() 的 pass_address 参数为 1，用本地和对端地址做参数，调用 ACE_Service_Handler::address() 方法
  - 设置新连接的 I/O 句柄，并调用新的服务处理器的 open() 方法

- ACE_Asynch_Connector 关键方法

方法	说明
open()	初始化用于主动连接工厂的信息
connect()	发起一个异步的 connect() 操作
cancel()	取消所有由该接受器发起的异步 accept() 操作
validate_connection()	挂钩方法，用于在为新连接打开服务之前确认对端地址
make_handler()	挂钩方法，用于为新连接获取处理器对象

- ACE_Service_Handler 关键方法

方法	说明
open()	挂钩方法，用于在新的连接建立之后初始化服务
address()	挂钩方法，用于捕捉服务连接的本地和远程地址

Proactor 框架 - 前摄式 Accepter-Connector

- 示例： Proactor 版本的 Echo Server

```
#include "ace/Asynch_IO.h"
#include "ace/Asynch_Acceptor.h"
#include "ace/INET_Addr.h"
#include "ace/Proactor.h"

class EchoService: public ACE_Service_Handler {
public:
    ~EchoService() {
        if (this->handle() != ACE_INVALID_HANDLE)
            ACE_OS::closesocket(this->handle());
    }
```

- 示例： Proactor 版本的 Echo Server （续 1 ）

```
virtual void open(ACE_HANDLE h, ACE_Message_Block&) {
    handle(h);
    if (this->reader_.open(*this) != 0 ||
        this->writer_.open(*this) != 0) {
        ACE_ERROR((LM_ERROR, "%p\n", "open()"));
        delete this;
        return;
    }

    ACE_Message_Block* mb;
    ACE_NEW_NORETURN (mb, ACE_Message_Block (512));
    if (this->reader_.read(*mb, mb->space()) != 0) {
        ACE_ERROR((LM_ERROR, "%p\n", "read()"));
        mb->release();
        delete this;
        return;
    }
}
```

- 示例： Proactor 版本的 Echo Server （续 2 ）

```
    virtual void handle_read_stream(
            const ACE_Asynch_Read_Stream::Result& result) {
        ACE_Message_Block &mb = result.message_block();
        if (!result.success() || result.bytes_transferred() ==
0) {
            mb.release();
            delete this;
        } else {
            if (this->writer_.write(mb, mb.length()) != 0) {
                ACE_ERROR ((LM_ERROR, "%p\n", "write()"));
                mb.release();
            } else {
                ACE_Message_Block* mblk;
                ACE_NEW_NORETURN (mblk, ACE_Message_Block
(512));
                this->reader_.read(*mblk, mblk->space());
            }
        }
    }
```

Proactor 框架 - 前摄式 Accepter-Connector

- 示例： Proactor 版本的 Echo Server （续 3 ）

```
    virtual void handle_write_stream(
            const ACE_Asynch_Write_Stream::Result& result) {
        result.message_block().release();
    }

private:
    ACE_Asynch_Read_Stream reader_;
    ACE_Asynch_Write_Stream writer_;
};
```

Proactor 框架 - 前摄式 Accepter-Connector

- 示例： Proactor 版本的 Echo Server （续 4 ）

```
int main() {
    ACE_INET_Addr listen_addr(8868);
    ACE_Asynch_Acceptor<EchoService> aio_acceptor;
    if (0 != aio_acceptor.open(listen_addr, 0, //
bytes_to_read
            0, // pass_addresses
            ACE_DEFAULT_BACKLOG, 1, // reuse_addr
            0, // proactor
            0)) // validate_new_connection
        ACE_ERROR_RETURN((LM_ERROR, "%p\n", "write()"), 1);

    ACE_Proactor::instance()->proactor_run_event_loop();
}
```

Proactor 框架 - ACE_Proactor

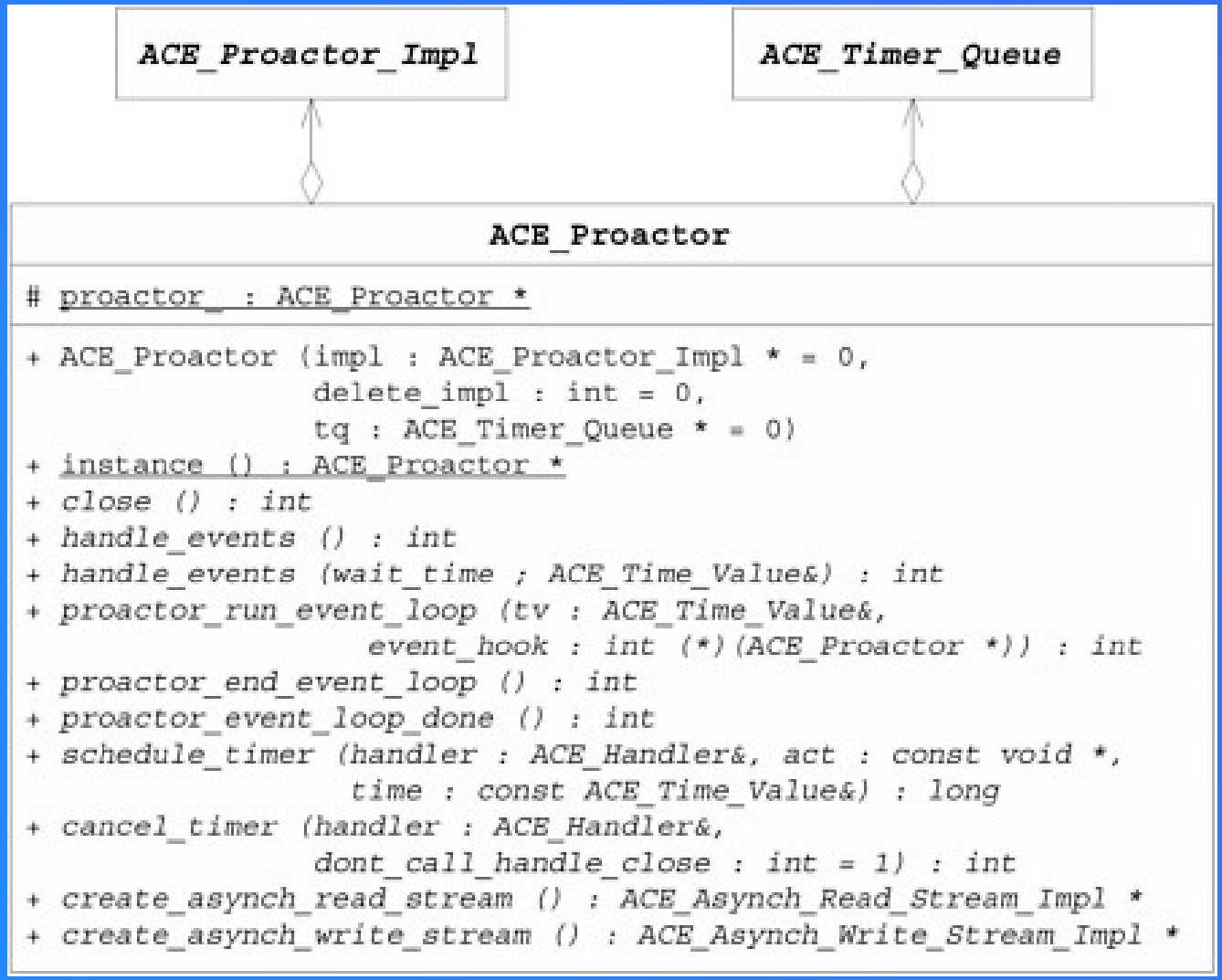
- Proactor 框架：
  - 概要
  - 异步 I/O 工厂类
  - ACE_Handler
  - 前摄式 Acceptor-Connector
  - ACE_Proactor

Proactor 框架 - ACE_Proactor

- 关于 ACE_Proactor
  - ACE_Proactor 类提供以下能力：
    - 将前摄式应用中的事件循环处理集中在一起
    - 将定时器到期分派给与其相关联的 ACE_Handler 对象
    - 将完成事件多路分离给完成处理器，并在完成处理器上分派适当的挂钩方法，由这些方法执行应用定义的处理，以对完成事件进行响应
    - 可以解除执行完成事件侦测、多路分离和分派线程与发起异步操作的线程的耦合
    - 在发起 I/O 操作的类和平台特有的异步 I/O 实现细节之间进行协调

Proactor 框架 - ACE_Proactor

- ACE_Proactor 类图



Proactor 框架 - ACE_Proactor

- ACE_Proactor 创建、销毁方法

方法	说明
`ACE_Proactor()` `open()`	
`~ACE_Proactor()` `close()`	
`instance()`	静态方法，返回执行一个单例 ACE_Reactor 的指针

Proactor 框架 - ACE_Proactor

- ACE_Proactor 事件循环管理方法

方法	说明
handle_events()	等待完成事件发生，并随即分派与其相关联的完成处理器。可以使用超时参数限制花在等待事件上的时间
proactor_run_event_loop()	反复调用 handle_events() 方法，直到发生下列情况之一：该方法失败、 proactor_event_loop_done() 返回 true 或是发生超时
proactor_end_event_loop()	指示前摄器关闭其事件循环
proactor_envent_loop_done()	如果前摄器的事件循环已被结束（例如，通过调用 proactor_end_event_loop() ），则返回 1

Proactor 框架 - 小结

- ACE Proactor 框架与 Reactor 框架类似，底层均离不开操作系统支持的事件多路分离机制
- 与 ACE Reactor 框架不同， Proactor 框架的利用的是各个操作系统的异步 I/O 机制，更好的支持高并发的网络化应用
- 与 Reactor 框架的同步 I/O 方法不同， Proactor 的异步 I/O 的方法调用均无阻塞，事件完成后需由完成处理器处理