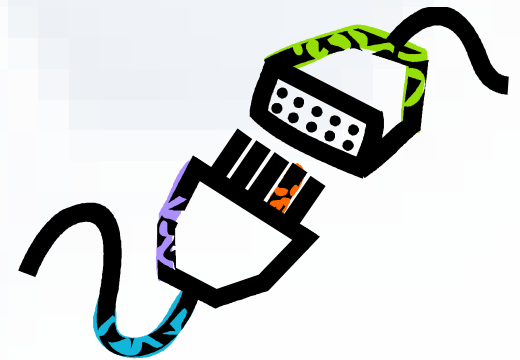


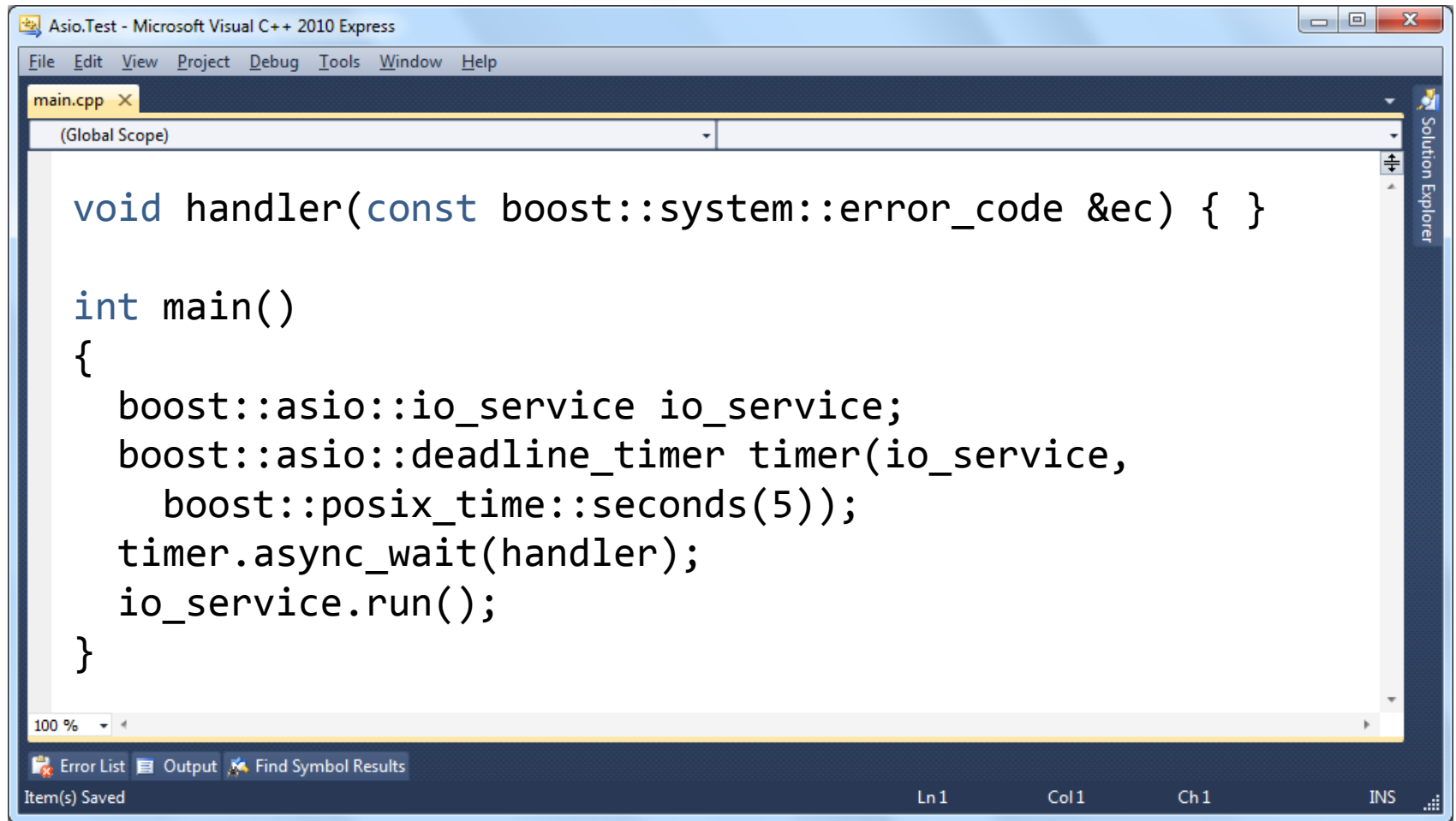
Creating Boost.Asio extensions

Boris Schäling, May 2011, www.highscore.de

- How does Boost.Asio look like internally?
- What are I/O service objects, I/O services and I/O objects?
- How do I access platform-specific I/O services?



Boost.Asio: Asynchronous functions



The screenshot shows a code editor window titled "Asio.Test - Microsoft Visual C++ 2010 Express". The editor displays the following C++ code in a file named "main.cpp":

```
void handler(const boost::system::error_code &ec) { }

int main()
{
    boost::asio::io_service io_service;
    boost::asio::deadline_timer timer(io_service,
        boost::posix_time::seconds(5));
    timer.async_wait(handler);
    io_service.run();
}
```

The code defines a handler function and a main function. The main function creates an `io_service`, a `deadline_timer` set to 5 seconds, and calls `timer.async_wait(handler)` followed by `io_service.run()`. The status bar at the bottom shows "Item(s) Saved", "Ln 1", "Col 1", "Ch 1", and "INS".

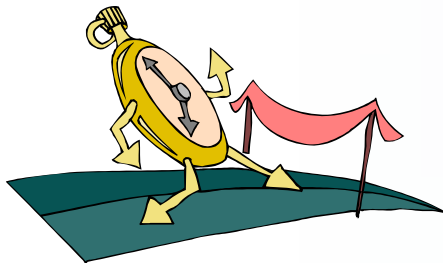


Boost.Asio: Asynchronous functions

```
asio.Test - Microsoft Visual C++ 2010 Express  
File Edit View Project Debug Tools Window Help  
main.cpp x  
(Global Scope)  
boost::asio::deadline_timer timer(io_service,  
    boost::posix_time::seconds(5));  
timer.async_wait(handler);
```

deadline_timer is one of many classes in Boost.Asio which make it possible to call functions asynchronously

Calling asynchronously means that `async_wait()` doesn't block – after the time expired the function is called which is passed as a parameter (here handler)



Boost.Asio: Asynchronous functions

`deadline_timer`

`async_wait()` to wait until some time is expired

`ip::tcp::acceptor`

`async_accept()` to accept TCP/IP connections

`ip::tcp::resolver`

`async_resolve()` to resolve hostnames

`ip::tcp::socket`

`async_read_some()` and `async_write_some()`
to send and receive data

Boost.Asio provides different classes which turn different blocking functions into asynchronous functions



Boost.Asio internals

```
asio.Test - Microsoft Visual C++ 2010 Express
File Edit View Project Debug Tools Window Help
main.cpp x
(Global Scope)

void handler(const boost::system::error_code &ec) { }

int main()
{
    boost::asio::io_service io_service;
    boost::asio::deadline_timer timer(io_service,
        boost::posix_time::seconds(5));
    timer.async_wait(handler);
    io_service.run();
}

100 %
Error List Output Find Symbol Results
Item(s) Saved Ln1 Col1 Ch1 INS
```

The screenshot shows a code editor window for 'main.cpp' in Visual C++ 2010 Express. The code defines a handler function and a main function. In the main function, two Boost.Asio objects are created: a `boost::asio::io_service` and a `boost::asio::deadline_timer`. The `io_service` is annotated with a callout box labeled 'I/O service object'. The `deadline_timer` is annotated with a callout box labeled 'I/O object'. The `timer` is constructed with the `io_service` and a 5-second deadline. The `timer` is then used to asynchronously wait for the handler to be called, and finally, the `io_service` is run.

Boost.Asio internals



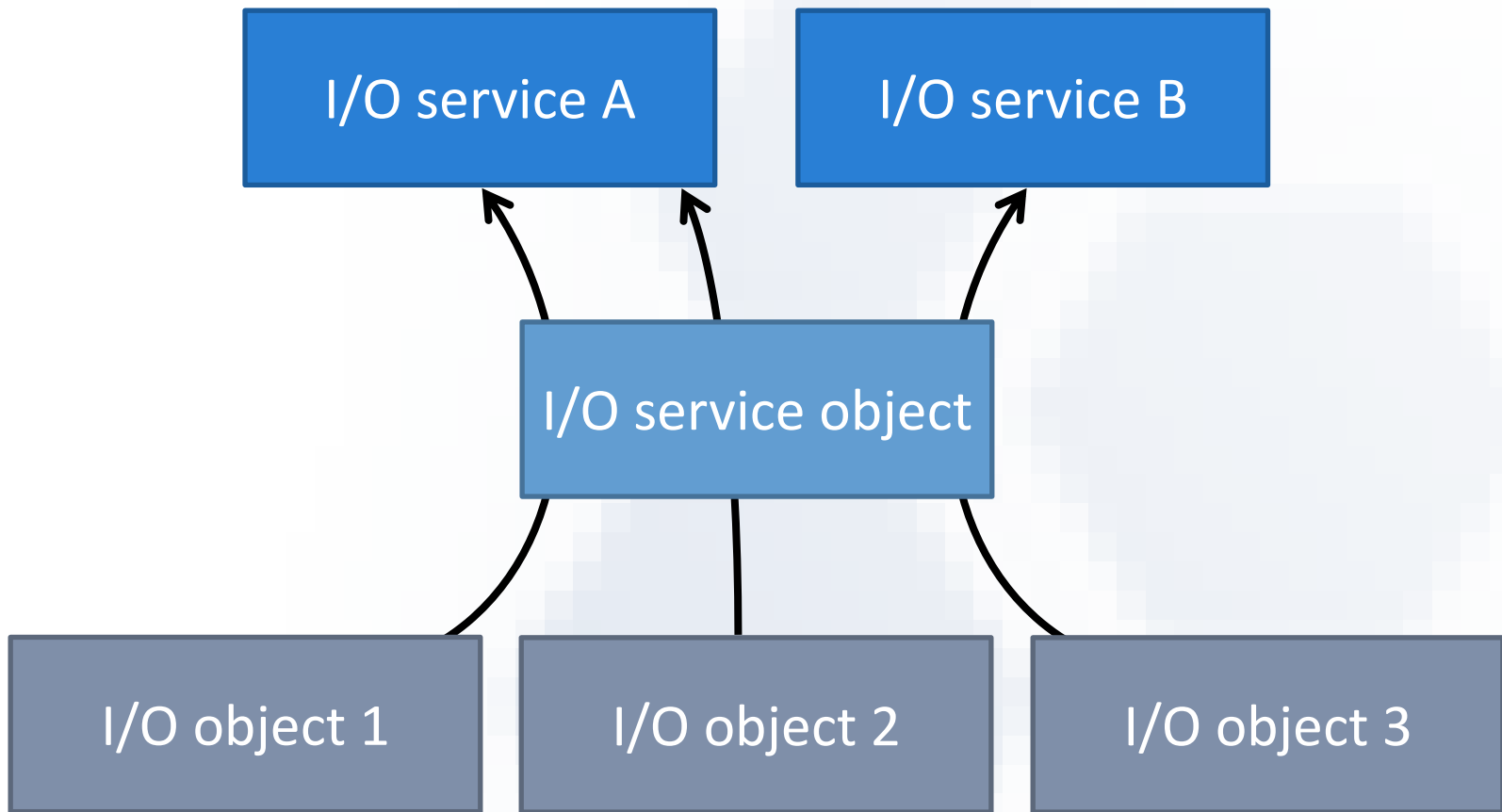
An I/O object is initialized with an I/O service object. It doesn't use the I/O service object though – it uses services provided by the I/O service object.

An I/O service object provides services to I/O objects. Think of it as a set of services – set because there is maximum one instance of each and every service.

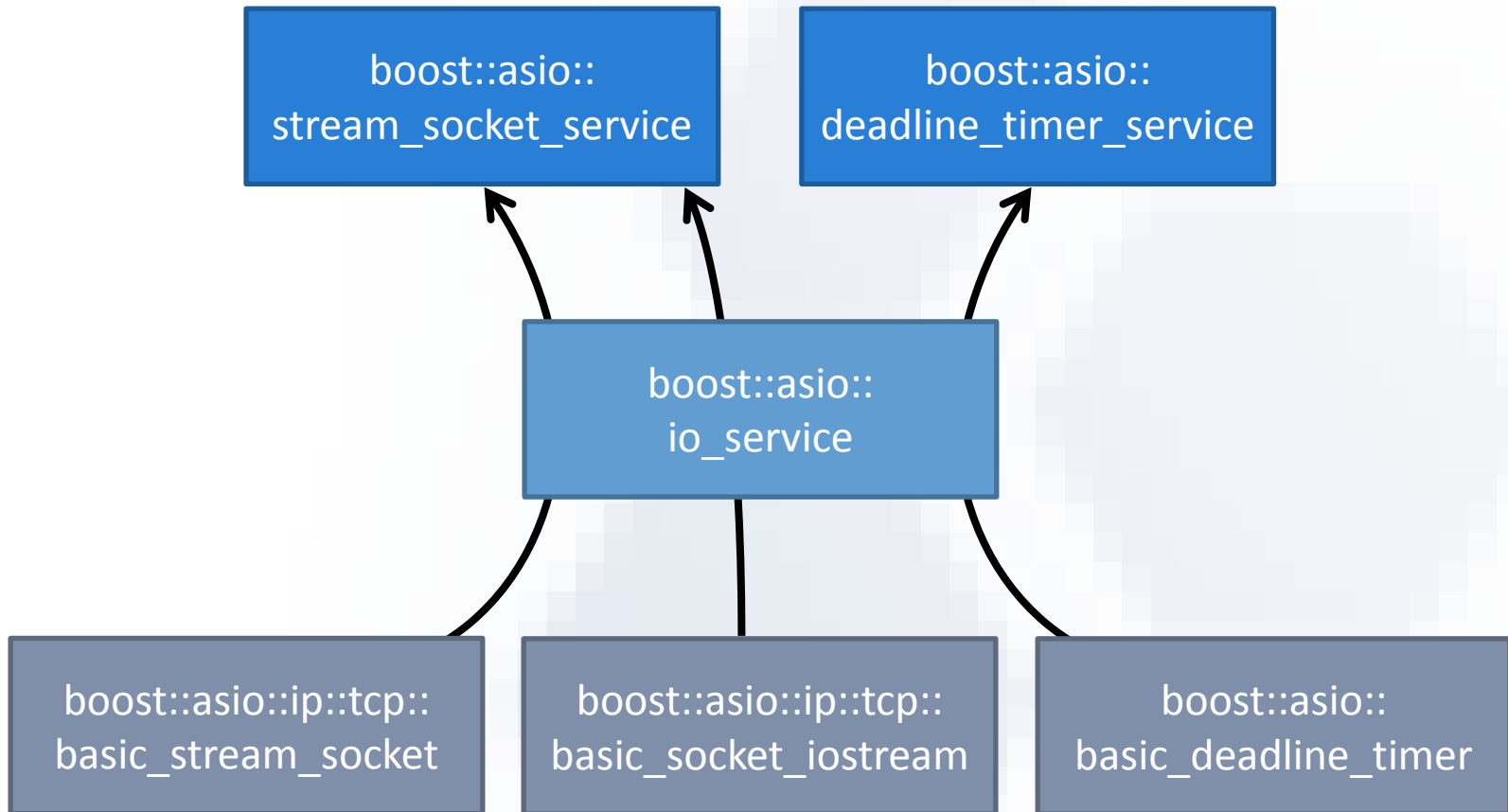
While I/O objects and I/O service objects are visible in user code, I/O services do the hard work in the background.



Boost.Asio internals

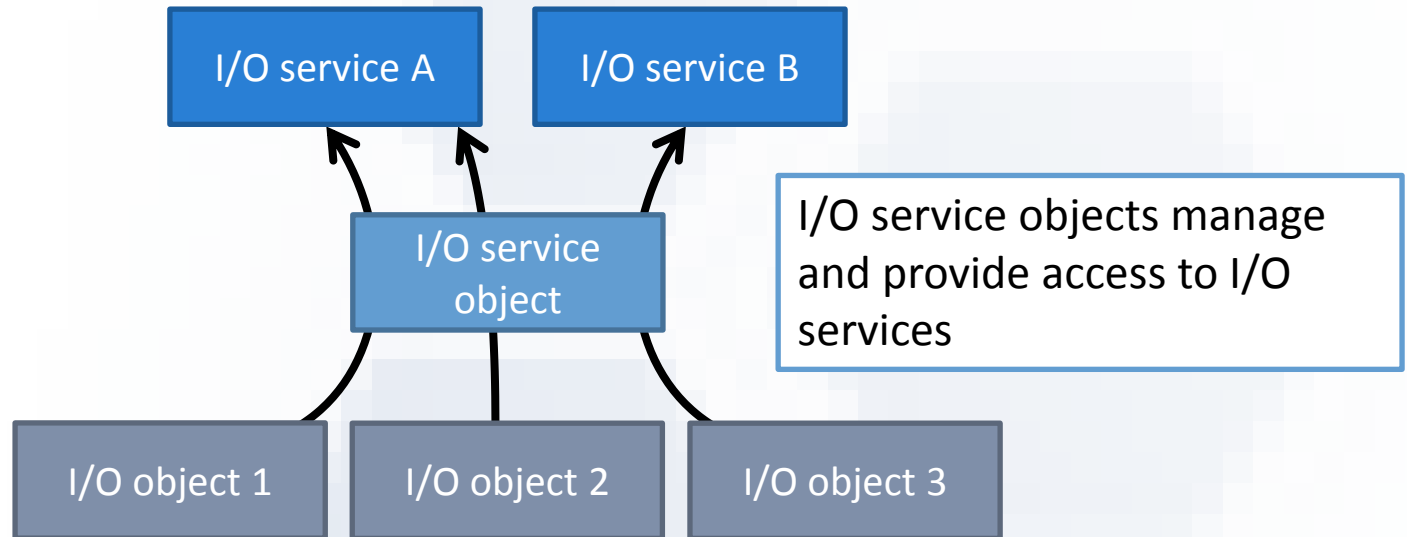


Boost.Asio internals



Boost.Asio internals

I/O services are based on system functions to provide a service to one or several I/O objects

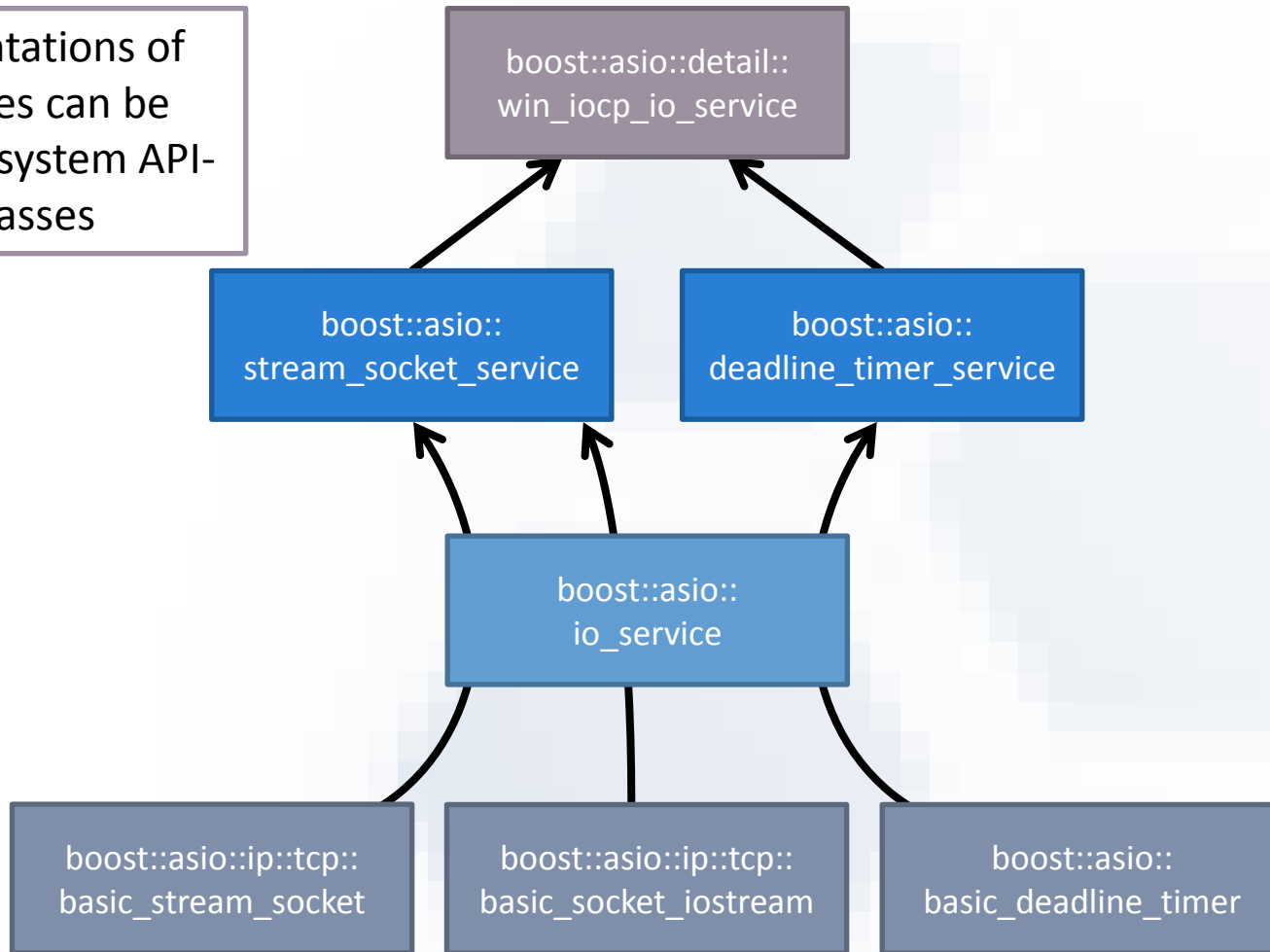


I/O objects get a reference to an I/O service object when instantiated

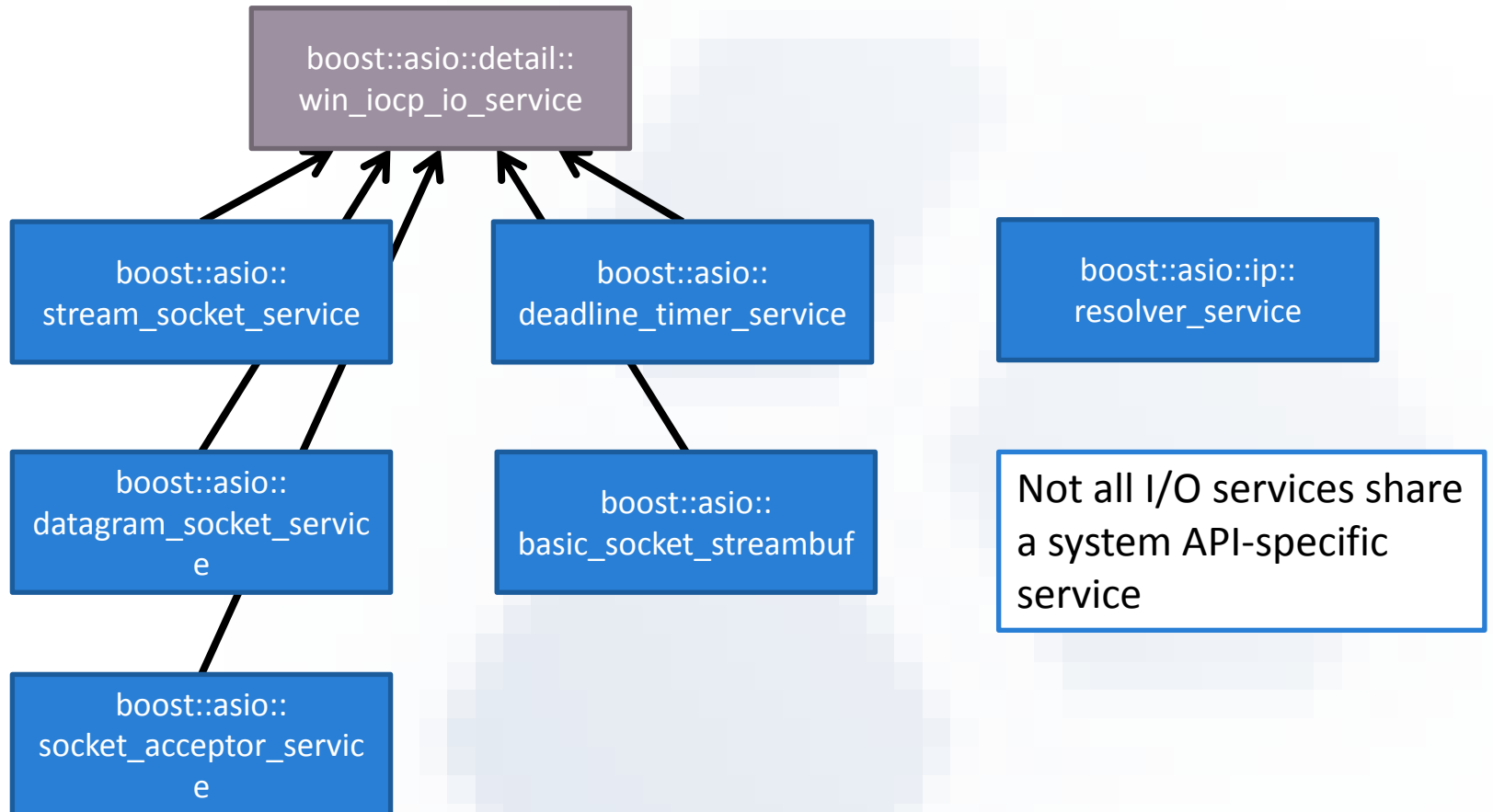


Boost.Asio internals

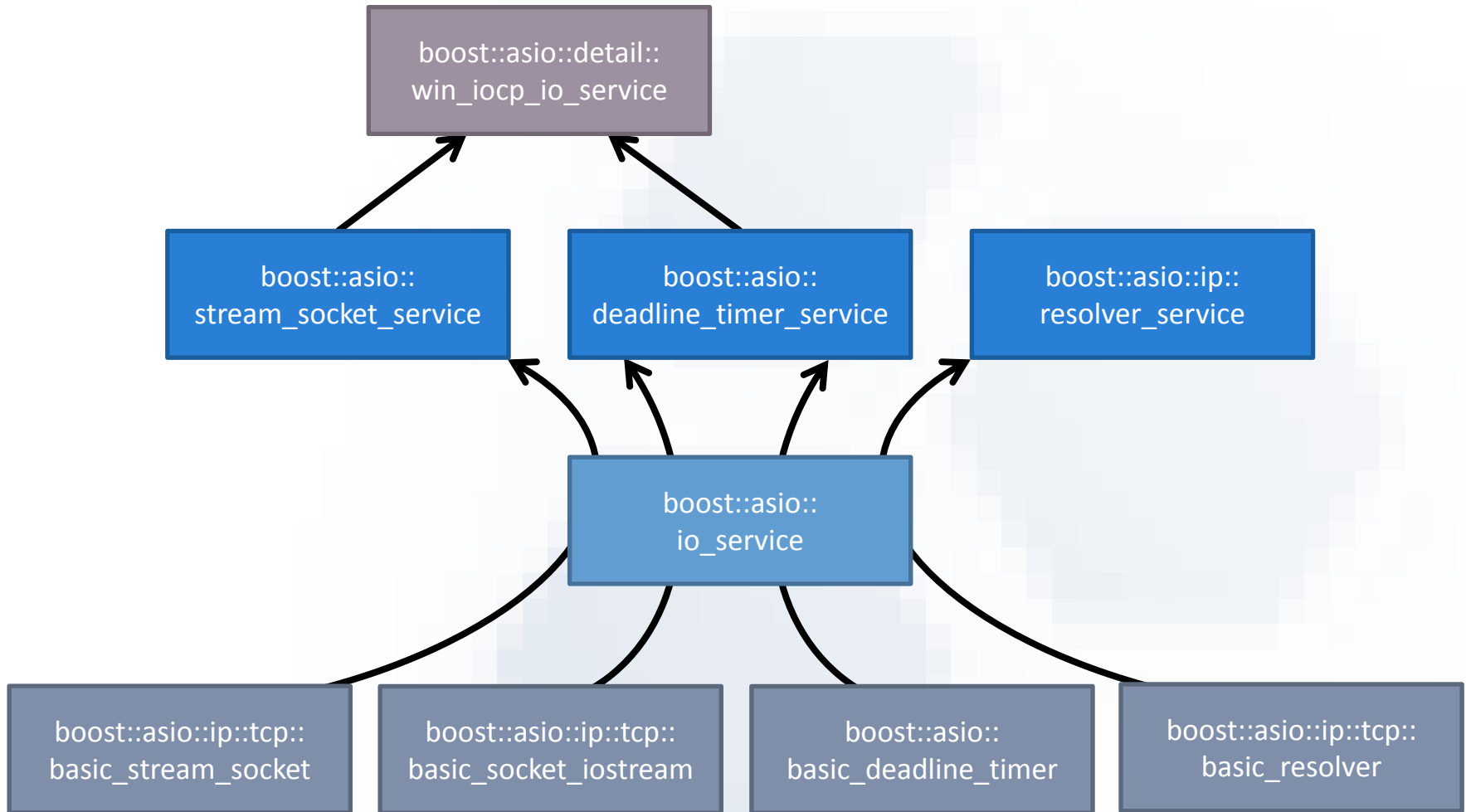
Implementations of I/O services can be based on system API-specific classes



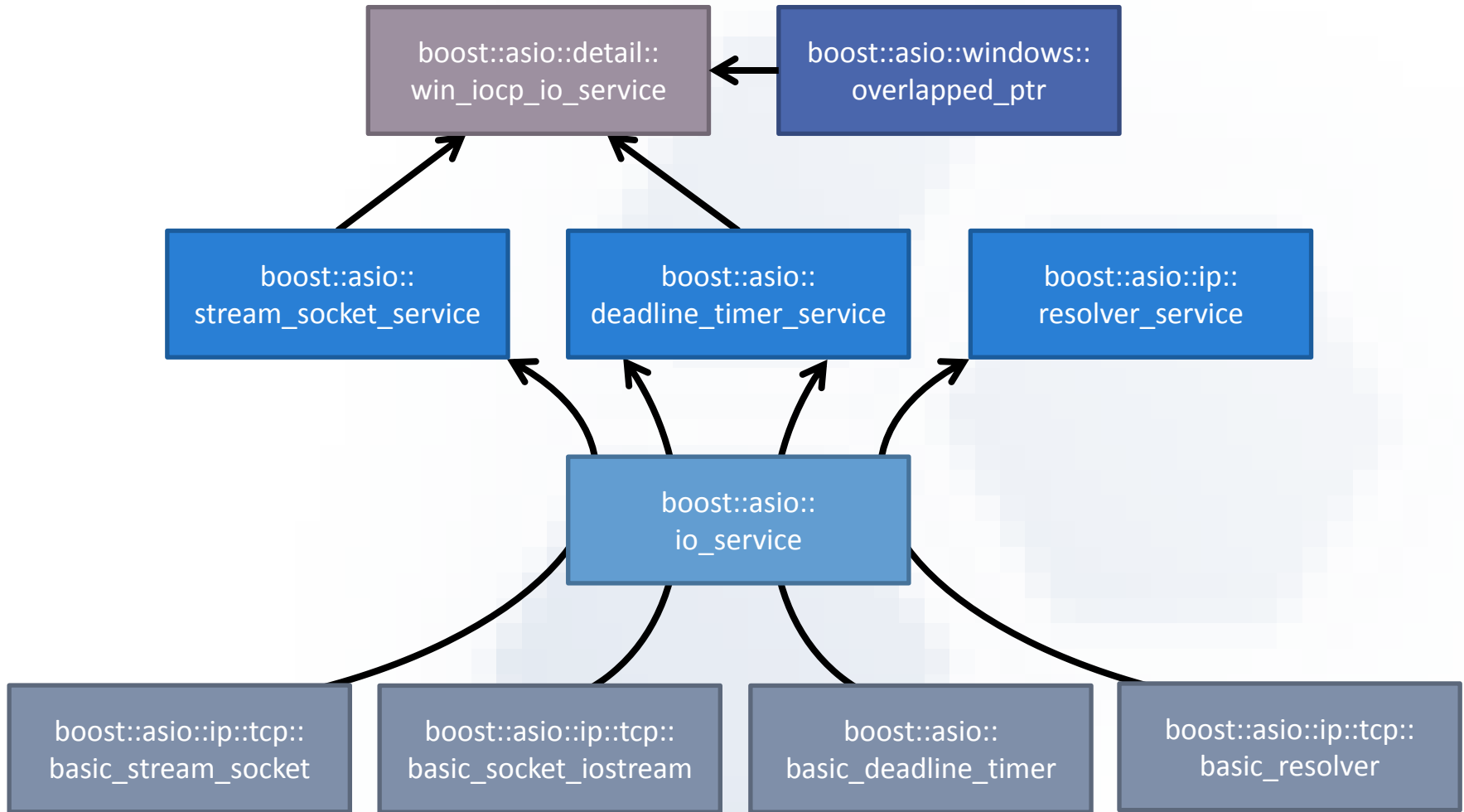
Boost.Asio internals



Boost.Asio internals



Boost.Asio internals



Boost.Asio internals

