NetTS, ASIO and Sender Library Design Comparison

Draft Proposal

Document #:	P2471R0
Date:	2021-10-06
Project:	Programming Language C++
Audience:	LEWG Library Evolution SG1 Concurrency and Parallelism SG4 Networking

Reply-to:	Kirk Shoop <kirk.shoop@gmail.com>

Contents

1 Changes

— first revision

2 Introduction

I have never seen the library designs of these libraries compared. This paper started as an email to the LEWG reflector. I was asked to make it a paper.

I used the following papers to fill in these tables:

— [N4771] “Working Draft, C++ Extensions for Networking”
— [P1322R3] “Networking TS enhancement to enable custom I/O executors”
— [P0958R3] “Networking TS changes to support proposed Executors TS”
— [P1943R0] “Networking TS changes to improve completion token flexibility and performance”
— [P2444R0] “The Asio asynchronous model”
— [P0443R14] “A Unified Executors Proposal for C++”
— [P2300R2] “std::execution”

I also searched in the ASIO repo

Corrections are welcome, especially for the ASIO and NetTS portions. I wish they had built something like this already.

I split these out into vertical tables because horizontal scrolling sucks.

The library designs compared in tables below are:

— Asynchronous Operation design
— Initiating function design
— Algorithm design
— Associated Values design
— Executor design
— Execution context design

Within each section are three tables:

— [N4771] as it currently stands.
— [ASIO] as it currently stands.
— [P2300R2] as it currently stands

3 Notes

The theme that revealed itself to me while compiling these is that ASIO & NetTS use traits and partial-specialization vs. Sender/Receiver use concepts and CPOs.

There are other disparities in specific places, but the approach to library design traits/concepts and specialization/CPOs are the repeated differentiators I saw.

Another thing that these tables revealed is all the changes in the ASIO design in the last 2-3 years (after sender/receiver was proposed).

4 Tables

4.1 Asynchronous Operation design

Table 1: NetTS - (N4771 is missing P1322, P0958, P1943, P2444)
13.2.7 Requirements on asynchronous operations

```
concept completion_token:
  async_result<
    completion_token,
    signature>
  ::completion_handler_type;

  async_result<
    completion_token,
    signature>
  ::return_type;
```

```
concept signature:
  (ErrorsAndValues...) -> void;

concept completion_handler_type:
  constructible<
    completion_handler_type,
    completion_token>;
  invocable<
    completion_handler_type,
    signature>;

concept result_type:
  constructible<
    result_type,
    completion_handler_type>;
```

Table 2: ASIO - (ASIO has P1322, P0958, P1943, P2444)

```
concept completion_token:
  async_result<
    completion_token,
    signature...>
  ::initiate(
    initiation,
    completion_token,
    Args...) -> Result;
```

```
concept signature:
  (ErrorsAndValues...) -> void;

concept initiation:
  (completion_handler, Args...) -> Result;

concept completion_handler:
  constructible<
    completion_handler,
    completion_token>;
  invocable<
    completion_handler,
    signature>...;
```

Table 3: Sender/Receiver - (P2300)

```
concept sender:
  connect(sender, receiver) -> operation_state;

concept operation_state:
  start(operation_state) -> void;

concept receiver:
  set_value(receiver, Values...) -> void;
  set_error(receiver, Error) -> void;
  set_done(receiver) -> void;
```

4.2 Initiating function design

Table 4: NetTS - (N4771 is missing P1322, P0958, P1943, P2444)
13.2.7 Requirements on asynchronous operations

```
Any function that takes a completion_token as the last
argument, and returns:

(..., completion_token)
  -> decltype(async_result<
    completion_token,
    signature>::result_type(
      async_result<
        completion_token,
        signature>
          ::completion_handler_type(
            completion_token)));
```

Table 5: ASIO - (ASIO has P1322, P0958, P1943, P2444)

```
Any function that takes a completion_token as the last
argument, and returns the result of:

(..., completion_token)
  -> decltype(async_result<
    completion_token,
    signature...>
  ::initiate(
    initiation,
    completion_token,
    Args...));
```

Table 6: Sender/Receiver - (P2300)

```
Any function returning a sender

(...) -> sender;
```

4.3 Algorithm design

Table 7: NetTS - (N4771 is missing P1322, P0958, P1943, P2444)

```
Unspecified, but without async_initiate the only option
I know of is:

concept algorithm:
  (completion_token) -> completion_token;

Happy to be corrected.
```

Table 8: ASIO - (ASIO has P1322, P0958, P1943, P2444)

```
Includes the above and:

Any specific completion_token can define a new
composable_type and return that.

The deferred completion_token is an example of this.

One of the infinite possible shapes for that new
composable_type could be:

concept algorithm
  (composable_type) -> composable_type;

concept composable_type:
  (completion_token) -> Result;
```

Table 9: Sender/Receiver - (P2300)

```
concept algorithm:
  (sender) -> sender
```

4.4 Associated values design

Table 10: NetTS - (N4771 is missing P1322, P0958, P1943, P2444)
13.2.2 Executor requirements

```
concept associated_executor:
  associated_executor<Source, Default>::type;
  associated_executor<Source, Default>
    ::get(source, default) -> executor; // static

concept associated_allocator:
  associated_allocator<Source, Default>::type;
  associated_allocator<Source, Default>
    ::get(source, default) -> allocator; // static
```

Table 11: ASIO - (ASIO has P1322, P0958, P1943, P2444)

```
Includes the above and:

concept associated_cancellation_slot:
  associated_cancellation_slot<Source, Default>::type;
  associated_cancellation_slot<Source, Default>
    ::get(source, default) -> cancellation_slot; // static
```

Table 12: Sender/Receiver - (P2300)

```
concept scheduler_provider:
  get_scheduler(scheduler_provider) -> scheduler;

concept allocator_provider:
  get_allocator(allocator_provider) -> allocator;

concept stop_token_provider:
  get_stop_token(stop_token_provider) -> stop_token;
```

4.5 Executor design

Table 13: NetTS - (N4771 is missing P1322, P0958, P1943, P2444)
13.2.2 Executor requirements

```
concept executor:
  executor::context() -> execution_context;
  executor::on_work_started() -> void;
  executor::on_work_finished() -> void;
  executor::dispatch(()->void, Allocator) -> void;
  executor::post(()->void, Allocator) -> void;
  executor::defer(()->void, Allocator) -> void;
```

Table 14: ASIO - (ASIO has P1322, P0958, P1943, P2444)

```
concept executor:
  execute(executor, ()->void) -> void;
```

Table 15: Sender/Receiver - (P2300)

```
concept scheduler:
  schedule(scheduler) -> sender;
```

4.6 Execution Context design

Table 16: NetTS - (N4771 is missing P1322, P0958, P1943, P2444)
13.2.3 Execution context requirements

```
concept execution_context:
  execution_context::executor_type;
  execution_context::get_executor()
    -> execution_context::executor_type;
```

Table 17: ASIO - (ASIO has P1322, P0958, P1943, P2444)

```
concept execution_context:
  no-requirements
```

Table 18: Sender/Receiver - (P2300)

```
concept execution_context:
  no-requirements
```

5 References

[ASIO] Christopher Kohlhoff. ASIO github.
https://github.com/chriskohlhoff/asio

[N4771] Jonathan Wakely. 2018-10-08. Working Draft, C++ Extensions for Networking.
https://wg21.link/n4771

[P0443R14] Jared Hoberock, Michael Garland, Chris Kohlhoff, Chris Mysen, H. Carter Edwards, Gordon Brown, D. S. Hollman. 2020-09-15. A Unified Executors Proposal for C++.
https://wg21.link/p0443r14

[P0958R3] Christopher Kohlhoff. 2021-03-15. Networking TS changes to support proposed Executors TS.
https://wg21.link/p0958r3

[P1322R3] Christopher Kohlhoff. 2021-02-15. Networking TS enhancement to enable custom I/O executors.
https://wg21.link/p1322r3

[P1943R0] Christopher Kohlhoff. 2019-10-07. Networking TS changes to improve completion token flexibility and performance.
https://wg21.link/p1943r0

[P2300R2] Michał Dominiak, Lewis Baker, Lee Howes, Kirk Shoop, Michael Garland, Eric Niebler, and Bryce Adelstein Lelbach. std::execution.
https://wiki.edg.com/pub/Wg21telecons2021/LibraryEvolutionWorkingGroup/P2300R2.html

[P2444R0] Christopher Kohlhoff. 2021-09-15. The Asio asynchronous model.
https://wg21.link/p2444r0