The Networking Claim and P2453R0

Document Number:	P4097R1
Date:	2026-05-01
Intent:	Inform
Audience:	WG21

Reply-to:	Vinnie Falco vinnie.falco@gmail.com C++ Alliance Proposal Team

Table of Contents

Abstract

The committee expressed consensus that sender/receiver is a good basis for networking. The published evidence behind that word is documented here.

In October 2021, LEWG polled: "The sender/receiver model (P2300) is a good basis for most asynchronous use cases, including networking, parallelism, and GPUs" (SF:24 / WF:16 / N:3 / WA:6 / SA:3 - consensus in favor). This paper searches the published record for the evidence that supported the word "networking" in that poll text - at the time of the vote and as of 2026.

Revision History

R1: May 2026 (pre-Brno mailing)

- Formatting corrections.

R0: April 2026 (post-Croydon mailing)

- Initial version.

---

1. Disclosure

The author provides information and serves at the pleasure of the committee.

This paper is part of the Network Endeavor (P4100R0), a project to bring coroutine-native I/O to C++.

The author developed and maintains Capy [1] and Corosio [2] and believes coroutine-native I/O is a practical foundation for networking in C++.

Coroutine-native I/O and `std::execution` are complementary. Each serves the domain where its design choices pay off.

This paper examines the published record. That effort requires re-examining consequential papers, including papers written by people the author respects.

The author is a co-author of P2469R0 [3], "Response to P2464: The Networking TS is baked, P2300 Sender/Receiver is not," which argued in October 2021 that the Networking TS was more mature than P2300. The reader should be aware that the author had a prior published position on the relationship between the Networking TS and P2300 [4].

The author's research method is systematic search of the published record - WG21 papers, published poll outcomes, and public mailing list archives. Every source cited in this paper is a published WG21 paper available on open-std.org or a published presentation available through wg21.link. The author also searched the LEWG reflector archives for evidence of sender-based networking deployments, prototypes, or implementations. No such deployment was identified. The reflector is not a public source and its contents are not quoted in this paper. The author acknowledges that absence of evidence is not evidence of absence. Committee discussions occur in rooms, hallways, dinners, and private channels that leave no public trace. Evidence that P2300 works for networking may exist in unpublished prototypes, internal deployments, or private communications. The author cannot prove that such evidence does not exist. If a reader is aware of a published document, deployment, or prototype that this paper's research did not reach, the author welcomes the correction and will update the

record in a future revision.

This paper asks for nothing.

---

2. The Poll

P2453R0 [5], "2021 October Library Evolution Poll Outcomes" (Bryce Adelstein Lelbach, Fabio Fracassi, Ben Craig, 2022), documents the following poll:

> *"The sender/receiver model (P2300) is a good basis for most asynchronous use cases, including networking, parallelism, and GPUs."*
>
> *SF:24 / WF:16 / N:3 / WA:6 / SA:3 - Consensus in favor.*

The chair's published interpretation:

> *"What this doesn't mean: It doesn't mean that S&R is going into C++23 (though it might). It doesn't mean that P2300R2 as it stands today will be sent forward to Library. It doesn't prevent us from adding new asynchronous models in the future."*
>
> *"What this means: Work will continue in Library Evolution on refining P2300R2, and Library Evolution will keep the various asynchronous use cases in mind while working on P2300R2."*

2.1 Context

In WG21, the people who write proposals are often the same people who present them, champion them, and participate in the polls that advance them. This is normal and expected - domain experts are the people best positioned to do the work. The following co-authorships are documented for completeness. Bryce Adelstein Lelbach served as LEWG Chair and is a co-author of P2300R10 [4]. Eric Niebler authored the P2300R10 [4] presentation slides (P2470R0 [6]) and is a co-author of P2300R10 [4] and P1525R0 [7]. Both contributed substantially to the design that the poll evaluated. The poll was open to all LEWG members and the results reflect the committee's collective judgment.

2.2 Selected Voter Comments

P2453R0 [5] Section 4.2 publishes selected comments from voters on Poll 2. The following comments address the networking component of the poll. All quotes are verbatim from the published paper.

Voters who stated they could not evaluate the networking claim:

> *"I think this is a good basis for parallelism/GPUs but can't judge its suitability for networking."*
>
> - Weakly Favor

> *"My vote is weakly in favor as I am not familiar enough with the networking requirements to be sure that it satisfies those, but for the other domains I am confident it does. If this were only for the parallelism and GPUs use case I would vote strongly in favour."*
>
> - Weakly Favor

> *"P2300R2 has not been around as long as Asio and hasn't been 'tried by fire' in the networking domain. I'm pretty familiar with special cases of networking for parallel computing, but not generally familiar with 'doing networking in C++,' so I'm voting WF instead of SF."*
>
> - Weakly Favor

> *"I think the general direction of sender/receiver more closely matches the semantics of the language and normal functions with regards to its handling of separate value/error channels... However, I do not think that sender/receiver is sufficiently well-baked to be included in C++23, and could do with some more refinement to address issues raised and more implementation experience."*
>
> - Weakly Favor

These votes counted toward the "consensus in favor" that included networking.

2.3 Poll 4

P2453R0 [5] documents a second poll that directly addresses networking:

> *"Networking in the C++ Standard Library should be based on the sender/receiver model (P2300)."*
>
> *SF:17 / WF:11 / N:10 / WA:4 / SA:6 - Weak consensus.*

The chair's published interpretation:

> *"In the short term, this poll result doesn't mean much. We don't have a paper in hand that proposes networking based on the [P2300R2] model."*

Selected voter comments on Poll 4:

> *"there is no concrete proposal to evaluate, let alone one that has been field-tested"*

> *"I would want to see more implementation experience with a sender/receiver-based networking design before being in favour of this"*

> *"at least a proof-of-principle implementation (maybe even in the form of a paper rather than code) is needed before changing the status quo"*

The most directly networking-specific poll produced a weaker result than Poll 2 and voter comments that explicitly called for evidence that did not yet exist.

---

3. The Evidence at the Time of the Vote

The poll was taken in October 2021. This section documents the published evidence that existed at that time regarding the sender/receiver model's suitability for networking.

3.1 Published Deployments

P2470R0 [6] (Niebler, 2021), the P2300R2 presentation slides, documented the following deployments:

> *"Sender/receiver as specified in P2300R2 is currently being used in the following shipping Facebook products: Facebook Messenger on iOS, Android, Windows, and macOS; Instagram on iOS and Android; Facebook on iOS and Android; Portal; An internal Facebook product that runs on Linux."*

> *"'NVIDIA is fully invested in P2300 senders/receivers; we believe it is the correct basis for portable asynchronous and heterogeneous execution in Standard C++ across all platforms, including CPUs, GPUs, and other accelerators...'"*

Deployment	Domain	Networking evidence
Facebook	Mobile apps, infrastructure	
NVIDIA	GPU dispatch, heterogeneous execution	
Bloomberg	Experimentation	

3.2 The P2300R2 Networking Example

P2300R2 [8] Section 1.4 contains a networking example using `NN::async_read_some` and `NN::async_write_some`:

> *"In this code, `NN::async_read_some` and `NN::async_write_some` are asynchronous socket-based networking APIs that return senders."*

The `NN::` namespace prefix is a placeholder. The example is a hypothetical illustration constructed by the proposal authors. It is not drawn from a deployed codebase.

3.3 Published Evidence Against

P2430R0 [9] (Kohlhoff, August 2021), "Partial success scenarios with P2300," was published two months before the poll. The paper documented that compound I/O results - an error code and a byte count - cannot be routed onto the sender's three completion channels without information loss:

> *"Due to the limitations of the set_error channel (which has a single 'error' argument) and set_done channel (which takes no arguments), partial results must be communicated down the set_value channel."*

3.4 Summary

Category	Evidence in the published record at the time of the vote
Sender-based networking deployment	None published. The production exchange described in P4125R1 [25] considered and rejected sender/receivers for architectural incompatibility (Section 7.2).
Sender-based networking prototype	
Sender-based networking code example	One hypothetical example in P2300R2 [8] Section 1.4 using placeholder `NN::` namespace
Published analysis of sender model for I/O	P2430R0 [9] (Kohlhoff, August 2021): compound I/O results cannot use `set_error` without information loss. Published before the poll.
Sender deployments (non-networking)	P2470R0 [6]: Facebook (mobile apps), NVIDIA (GPU dispatch), Bloomberg (experimentation)

4. The Evidence as of 2026

4.1 Papers by Other Authors

P2762R2 [10] (Kühl, 2023), "Sender/Receiver Interface For Networking," is the first published paper proposing sender-based networking APIs. It was published two years after the poll. Section 4.2 documents five routing options for I/O results onto sender channels and writes:

> *"some of the error cases may have been partial successes. In that case, using the set_error channel taking just one argument is somewhat limiting."*

The concern P2430R0 [9] raised in August 2021 - before the poll - remains documented in the literature as of 2023.

Paper	Year	What it documents
P2430R0 [9]	2021	Compound I/O results cannot use `set_error` without information loss
P2762R2 [10]	2023	First sender-based networking API proposal. Error channel "somewhat limiting" for partial success.
N4985 [11]	2024	St. Louis minutes: national body raised concerns about P2300R10 at the adoption vote.
P3801R0 [12]	2025	"Concerns about the design of std::execution::task." Urges the committee to reconsider P3552R3.
P3796R1 [13]	2025	"Coroutine Task Issues." Documents stack overflow, cancellation, and wording problems in the task.
Published sender-based networking deployment	2026	None. The only published production I/O evaluation (P4125R1 [25]) chose coroutines; sender/receivers rejected as architecturally incompatible.

N4985 [11], the St. Louis 2024 minutes, records the moment P2300R10 [4] was adopted into the C++ working paper:

> *"One national body reported that individual members had concerns. They would like to postpone this until Poland in order to increase concerns. The concerns are mostly about teachability and user story for beginners."*

P3801R0 [12] (Müller, 2025) writes:

> *"P3552R3, proposing the coroutine task type `std::execution::task`, was approved in Sofia for inclusion in C++26. I have strong concerns about its design and urge the committee to reconsider."*

4.2 The Author's Own Papers

The following papers are authored or co-authored by the author of this paper. They are listed separately so the reader can calibrate accordingly.

Paper	Title	What it documents
P4003R3 [14]	A Minimal Coroutine Execution Model	The coroutine executor concept for networking
P4007R3 [15]	Open Issues in `std::execution::task`	`AS-EXCEPT-PTR` converts routine `error_code` to `exception_ptr`
P4090R0 [16]	Sender I/O: A Constructed Comparison	Side-by-side sender vs. coroutine networking code
P4091R0 [17]	Two Error Models	The sender error channel vs. `error_code` for I/O
P4092R0 [18]	Consuming Senders from Coroutine-Native Code	Coroutine-to-sender interop bridge
P4093R0 [19]	Producing Senders from Coroutine-Native Code	Sender-to-coroutine interop bridge
P4088R0 [20]	What C++20 Coroutines Already Buy The Standard	The argument for coroutine-native I/O as the foundation for networking
P2583R4 [21]	Symmetric Transfer and Sender Composition	Symmetric transfer gap in sender task types

The author's position on coroutine-native I/O is a consequence of the findings documented in this series, not a premise.

4.3 The Timeline

- N1925 [22] (2005): first networking proposal.
- Networking TS published as ISO TS (2018).
- P2453R0 [5] (October 2021): "including networking." Consensus in favor.
- P2762R2 [10] (2023): first sender-based networking API proposal.
- 2026: no sender-based networking has shipped. Networking is not in the C++ standard. Twenty-one years from N1925 [22].

---

5. Anticipated Objections

Q: The poll said "good basis," not "ready to ship."

A: Section 2 quotes the chair's interpretation. The poll's outcome shaped the committee's direction. P2453R0 [5] Section 3 states: "The combination of this 'grand unified model' poll and Poll 4 heavily encourages the networking study group to produce a paper based on Senders and Receivers." The word "networking" in the poll text directed the committee's networking work toward the sender model.

Q: P2300 has been refined since 2021.

A: Section 4.1 documents the 2026 evidence. The error channel concern from P2430R0 [9] (2021) is documented again in P2762R2 [10] (2023). No sender-based networking deployment has been published.

Q: The poll included "parallelism and GPUs" which are validated.

A: This paper examines the word "networking." The parallelism and GPU evidence is documented in Section 3.1. The GPU deployments are real. The networking evidence column is empty.

Q: You are arguing for your own library.

A: Section 1 discloses this. Section 4.2 labels the author's own papers separately. The evidence in Sections 2 through 4 stands or falls on the published record, not on who assembled it.

Q: The committee voted with full information.

A: Jonathan Müller reported [23] from St. Louis: *"P2300 was adopted in the plenary vote and is now a part of the working draft which will become the C++26 standard, it was a very narrow vote with 1/3 voting against adoption."* One anonymous commenter during the 2021 electronic ballot wrote [24]: *"I don't think it's fair to consider standardizing S&R until there are at least a thousand codebases that use S&R. The probability of missing an important use-case, or an important gotcha is very very high if the actual quantity of 'Junior engineer + intern' experience in the field is low."*

Acknowledgments

The author thanks Bryce Adelstein Lelbach, Fabio Fracassi, and Ben Craig for the published poll outcomes in P2453R0; Christopher Kohlhoff for P2430R0, which documented the partial success problem before the poll was taken; Dietmar Kühl for P2762R2, the first sender-based networking API proposal; Eric Niebler for P2470R0 and the deployment documentation; and Steve Gerbino and Mungo Gill for Capy and Corosio implementation work.

References

[1] cppalliance/capy - Coroutine I/O primitives library.

[2] cppalliance/corosio - Coroutine-native networking library.

[3] P2469R0 - "Response to P2464: The Networking TS is baked, P2300 Sender/Receiver is not" (Christopher Kohlhoff, Jamie Allsop, Vinnie Falco, Richard Hodges, Klemens Morgenstern, 2021).

[4] P2300R10 - "std::execution" (Michał Dominiak, Lewis Baker, Lee Howes, Kirk Shoop, Michael Garland, Eric Niebler, Bryce Adelstein Lelbach, 2024).

[5] P2453R0 - "2021 October Library Evolution Poll Outcomes" (Bryce Adelstein Lelbach, Fabio Fracassi, Ben Craig, 2022).

[6] P2470R0 - "Slides for presentation of P2300R2: std::execution (sender/receiver)" (Eric Niebler, 2021).

[7] P1525R0 - "One-Way execute is a Poor Basis Operation" (Eric Niebler, Kirk Shoop, Lewis Baker, Lee Howes, 2019).

[8] P2300R2 - "std::execution" (Michał Dominiak, Lewis Baker, Lee Howes, Kirk Shoop, Michael Garland, Eric Niebler, Bryce Adelstein Lelbach, 2021).

[9] P2430R0 - "Partial success scenarios with P2300" (Christopher Kohlhoff, 2021).

[10] P2762R2 - "Sender/Receiver Interface For Networking" (Dietmar Kühl, 2023).

[11] N4985 - "WG21 2024-06 St Louis Minutes of Meeting" (Nina Ranns, 2024).

[12] P3801R0 - "Concerns about the design of std::execution::task" (Jonathan Müller, 2025).

[13] P3796R1 - "Coroutine Task Issues" (Dietmar Kühl, 2025).

[14] P4003R3 - "A Minimal Coroutine Execution Model" (Vinnie Falco, Steve Gerbino, Mungo Gill, 2026).

[15] P4007R3 - "Open Issues in std::execution::task" (Vinnie Falco, 2026).

[16] P4090R0 - "Sender I/O: A Constructed Comparison" (Vinnie Falco, 2026).

[17] P4091R0 - "Two Error Models" (Vinnie Falco, 2026).

[18] P4092R0 - "Consuming Senders from Coroutine-Native Code" (Vinnie Falco, Steve Gerbino, 2026).

[19] P4093R0 - "Producing Senders from Coroutine-Native Code" (Vinnie Falco, Steve Gerbino, 2026).

[20] P4088R0 - "What C++20 Coroutines Already Buy The Standard" (Vinnie Falco, 2026).

[21] P2583R4 - "Symmetric Transfer and Sender Composition" (Vinnie Falco, 2026).

[22] N1925 - "Networking proposal for TR2 (rev. 1)" (Gerhard Wesp, 2005).

[23] Trip Report: Summer ISO C++ Meeting in St. Louis, USA - Jonathan Müller, July 2024.

[24] r/cpp: C++ committee polling results for asynchronous programming - Oct 2021.

[25] P4125R1 - "Coroutine-Native I/O at a Derivatives Exchange" (Mungo Gill, 2026).