Fearless Asynchronous Communications with Timed Session Types in Rust (Artifact)

Nicolas Lagaillardie
Imperial College London, UK

Ping Hou
University of Oxford, UK

Nobuko Yoshida
University of Oxford, UK

Abstract

We introduce `MultiCrusty`^{T}, a Rust toolchain designed to facilitate the implementation of safe affine timed protocols. `MultiCrusty`^{T} leverages generic types and the `time` library to handle timed communications, integrated with optional types for affinity. This artifact allows to evaluate our approach by running examples from the literature, real-world use cases and protocols from real-time systems, featured in the related article, showcasing the *correctness by construction* of our approach. We allow to see the performance of our solution by running benchmarks and generating graphs, highlighting a less than 10% compile-time and runtime overhead compared with an untimed implementation. We also demonstrate how to implement, step by step, your own timed protocols, from a very basic one with only two parties and simple interactions, to complex ones with more than three parties, choices and recursion.

2012 ACM Subject Classification Software and its engineering → Software usability; Software and its engineering → Concurrent programming languages; Theory of computation → Process calculi

Keywords and phrases session types, affine types, π-calculus, asynchrony, timeouts, failures, Rust

Digital Object Identifier 10.4230/DARTS.10.2.27

Funding Work supported by: EPSRC EP/T006544/2, EP/K011715/1, EP/K034413/1, EP/L00058X/1, EP/N027833/2, EP/N028201/1, EP/T014709/2, EP/V000462/1, EP/X015955/1, NCSS/EPSRC VeTSS, and Horizon EU TaRDIS 101093006.

Acknowledgements We thank the anonymous reviewers for their useful comments and suggestions.

Related Article Ping Hou, Nicolas Lagaillardie, and Nobuko Yoshida, "Fearless Asynchronous Communications with Timed Multiparty Session Protocols", in 38th European Conference on Object-Oriented Programming (ECOOP 2024), LIPIcs, Vol. 313, pp. 29:1–29:29, 2024.
`https://doi.org/10.4230/LIPIcs.ECOOP.2024.29`

Related Conference 38th European Conference on Object-Oriented Programming (ECOOP 2024), September 16–20, 2024, Vienna, Austria

1 Scope

The purpose of this document is to describe in detail the steps required to assess the artifact associated with our paper. We claim our artifact to be *functional*, *reusable* and *available* as follows:

1. **Functionality**:
   - `MultiCrusty`^{T} can be used for safe affine timed communication programming in Rust. In particular, you should be able to verify claims from the paper:
     - use `MultiCrusty`^{T} to write and verify affine timed protocols with ATMP and νScr^T, following the top-down approach explained in Section 2 of the related paper;

- observe the detected errors due to incompatible types, as explained in Section 2 of the related paper.

- With this artifact, you can reproduce the benchmarks in Section 6 of the related paper (i.e., Figure 11):
  1 claim on expressiveness (Section 6.2 of the related paper): examples in Figure 11 can be implemented using **`MultiCrusty`**^{T}.
  2 claim on compile-time performance (Section 6.1 of the related paper, Figure 11):
    - the more participants there are, the higher is the compilation time for **`MultiCrusty`**^{T}.
  3 claim on run-time performance (Section 6.1 of the related paper, Figure 11):
    - **`MultiCrusty`**^{T} is nearly as fast as `MultiCrusty`, the AMPST implementation, when there are a large number of interactions and participants (in the full-mesh protocol);
    - the worst-case scenario for **`MultiCrusty`**^{T} occurs in protocols with many participants but no causal dependencies between them, resulting in a slowdown compared to `MultiCrusty` (in the ring protocol);
    - **`MultiCrusty`**^{T} has a negligible overhead in comparison to `MultiCrusty`.

  Instructions for reproducing Figure 11 are available in the `README.md` file.

2. **Reusability**: **`MultiCrusty`**^{T} can be used to verify custom communication protocols and programs, follow the instructions in the `README.md` file.
3. **Availability**: The artifact is available under the Creative Commons CC BY license (`https://creativecommons.org/licenses/by/4.0/`).

For more details, please consult Section 6 in the related article, Appendix G in the full version [1], and the `README` file in the artifact.

2 Content

The Docker image serves as the artifact submission. It contains:

- **`mpst_rust_github/`**: Source code, Examples, Benchmarks, and Tests of **`MultiCrusty`**^{T}:
  - **`src/`**: Source code.
  - **`examples/`**: Examples:
    - **`timed/`**: Examples of Table 2 in the related paper.
    - **`artifact_atmp/`**: Functionality badge examples.
  - **`scripts/`**: Result reproduction scripts.
  - **`benches/`**: Benchmarks:
    - **`mesh/timed/`** and **`ring/timed/`**: Figure 11 (top) examples.
    - **`timed/`**: Figure 11 (bottom) benchmarks.
  - **`tests/`**: Tests.
  - **`graphs_bench/`**: Generated graphs from the benchmarks (Figure 11).
- **`nuscr/`**: νScr^T library source code.

The artifact contains additional subfolders that are not essential for understanding the toolchain's functionality but are necessary for its operation.

3 Getting the artifact

The artifact endorsed by the Artifact Evaluation Committee is available free of charge on the Dagstuhl Research Online Publication Server (DROPS). In addition, the artifact is also available at: `https://zenodo.org/doi/10.5281/zenodo.11032195`. The source files can be accessed at `https://github.com/NicolasLagaillardie/mpst_rust_github`.

4 Tested platforms

The artifact has been tested under:

- Windows 10 (22H2)
- Linux (Arch Linux, XFCE 4.18, kernel 6.6.25-1)
- MacOS (Sonoma 14.4.1, M2)

All machines had at least 16 GB of RAM and 50 GB of disk space. In principle, the artifact should be able to run under a correct installation of Docker.

5 License

The artifact is available under the Creative Commons CC BY license (`https://creativecommons.org/licenses/by/4.0/`).

6 MD5 sum of the artifact

06073a8a89f4c520990f87157ca132ed

7 Size of the artifact

16.1 GB

A Additional Information

For additional information, readers are invited to consult the `README.md` file accompanying the Docker image, which provides instructions on how to use the artifact. Alternatively, the `README` file is available online at `https://github.com/NicolasLagaillardie/ECOOP24-Artefact/blob/main/README.md`.

References

1 Ping Hou, Nicolas Lagaillardie, and Nobuko Yoshida. Fearless asynchronous communications with timed multiparty session protocols, 2024. URL: `https://arxiv.org/abs/2406.19541`, `arXiv:2406.19541`.