Wrapyfi: A Python Wrapper for Integrating Robots, Sensors, and Applications across Multiple Middleware

Fares Abawi, Philipp Allgeuer, Di Fu and Stefan Wermter

Knowledge Technology, Department of Informatics, University of Hamburg

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

KNOWLEDGE TECHNOLOGY
www.knowledge-technology.info

Objectives

Wrapyfi is a middleware communication wrapper for transmitting data across nodes, without altering the operation pipeline of your Python scripts. Switching between middleware, such as **ROS** [1], **ROS 2** [2], **YARP** [3], and **ZeroMQ** [4] can be easily achieved by simply replacing one middleware with another. Wrapyfi also aims to:

- Minimize modifications to existing code.
- Support the **Publish/Subscribe** and **Request/Reply** communication patterns.
- Support multiple deep learning frameworks.
- Serialize images, audio, arrays and tensors for enabling a vast array of applications.
- Introduce three communication schemes for flexibly adapating code structure.

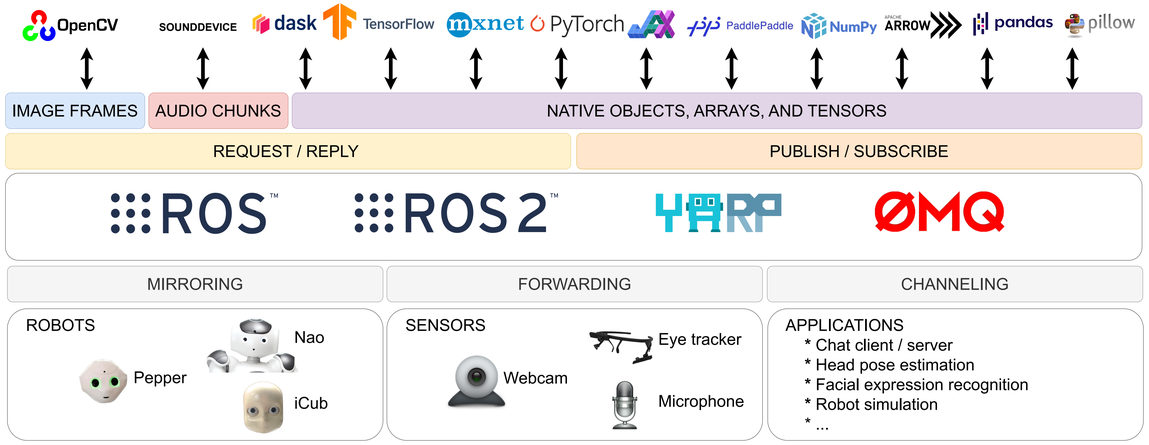
Communication Schemes

Mirroring enables synchronized communication across clients and servers or publishers and subscribers, ensuring method returns are consistently replicated on all sides.

Forwarding acts as a relay between entities unable to communicate directly, facilitating message exchange by passing information through intermediary servers or publishers.

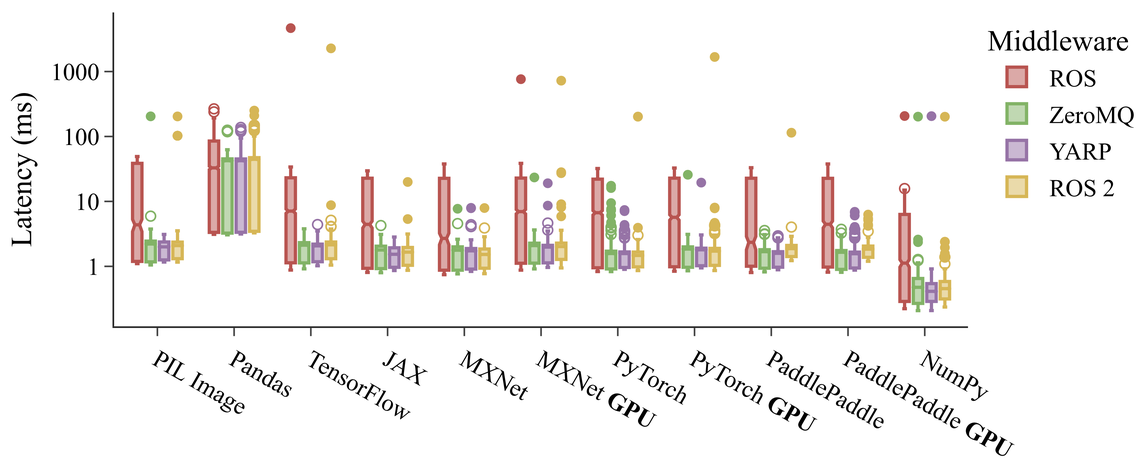
Channeling combines elements of mirroring and forwarding, enabling the selective transmission of multiple objects across a network, where objects can be mirrored to maintain consistency while also allowing for the targeted forwarding of specific data streams as required.

Overview



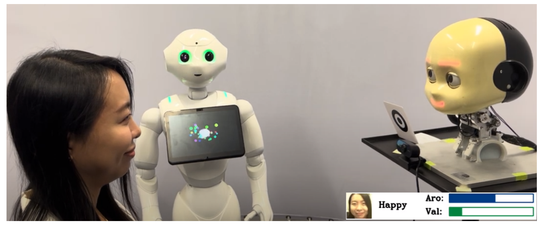
Overview of the Wrapyfi framework. From top to bottom: 1) Data types are encoded or decoded depending on the transmission mode; 2) Encoded objects are prepared for transmission using the Request/Reply or Publish/Subscribe communication pattern; 3) Messages are transmitted through the selected middleware protocol; 4) Messages sequenced according to the communication scheme.

Evaluation

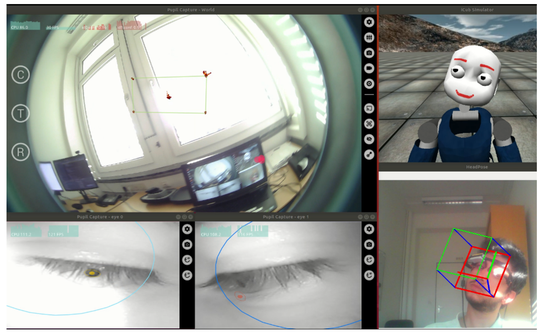


Latency between publishing and receiving 200×200 tensors of 32-bit floats, transmitted using each middleware independently with blocking methods (subscriber awaits publisher). 2000 trials are conducted with a publishing rate of 100 Hz for each middleware and plugin combination. Latency indicates the time difference in milliseconds between transmission and reception, including de/serialization.

Use Cases



Facial expression imitation on the Pepper and iCub robots by utilizing the forwarding scheme.



Head and eye movement imitation using either an IMU-fitted eye tracker or a head pose estimation model by utilizing the channeling scheme.

References

[1] Quigley et al. "ROS: An open-source Robot Operating System". In: *IEEE International Conference on Robotics and Automation Workshop on Open Source Software (ICRAOSS)*. 2009.

[2] Macenski et al. "Robot Operating System 2: Design, architecture, and uses in the wild". In: *Science Robotics* 7.66 (2022), eabm6074.

[3] Metta et al. "YARP: Yet Another Robot Platform". In: *International Journal of Advanced Robotic Systems* 3.1 (2006), p. 8.

[4] Hintjens. *ZeroMQ: Messaging for Many Applications*. O'Reilly Media, Inc., 2013.

Acknowledgements & Code

The authors gratefully acknowledge partial support from the German Research Foundation DFG under project CML (TRR 169).