Ping Simple

Table of contents

List of Figures

Most applications will require more than one operator. In this example, we will create two operators where one operator will produce and send data while the other operator will receive and print the data. The code in this example makes use of the built-in **PingTxOp** and **PingRxOp** operators that are defined in the `holoscan::ops` namespace.

In this example we'll cover:

- how to use built-in operators
- how to use `add_flow()` to connect operators together

Note

The example source code and run instructions can be found in the examples directory on GitHub, or under `/opt/nvidia/holoscan/examples` in the NGC container and the debian package, alongside their executables.

Operators and Workflow

Here is a example workflow involving two operators that are connected linearly.

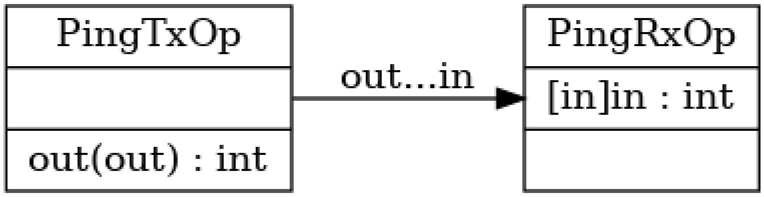

Fig. 5 *A linear workflow*

In this example, the source operator **PingTxOp** produces integers from 1 to 10 and passes it to the sink operator **PingRxOp** which prints the integers to standard output.

Connecting Operators

We can connect two operators by calling `add_flow()` (`C++` / `Python`) in the application's `compose()` method.

The `add_flow()` method (`C++` / `Python`) takes the source operator, the destination operator, and the optional port name pairs. The port name pair is used to connect the output port of the source operator to the input port of the destination operator. The first element of the pair is the output port name of the upstream operator and the second element is the input port name of the downstream operator. An empty port name ("") can be used for specifying a port name if the operator has only one input/output port. If there is only one output port in the upstream operator and only one input port in the downstream operator, the port pairs can be omitted.

The following code shows how to define a linear workflow in the `compose()` method for our example. Note that when an operator appears in an `add_flow()` statement, it doesn't need to be added into the workflow separately using `add_operator()`.

Ingested Tab Module

Running the Application

Running the application should give you the following output in your terminal:

```
Rx message value: 1 Rx message value: 2 Rx message value: 3 Rx message value: 4
Rx message value: 5 Rx message value: 6 Rx message value: 7 Rx message value: 8
Rx message value: 9 Rx message value: 10
```