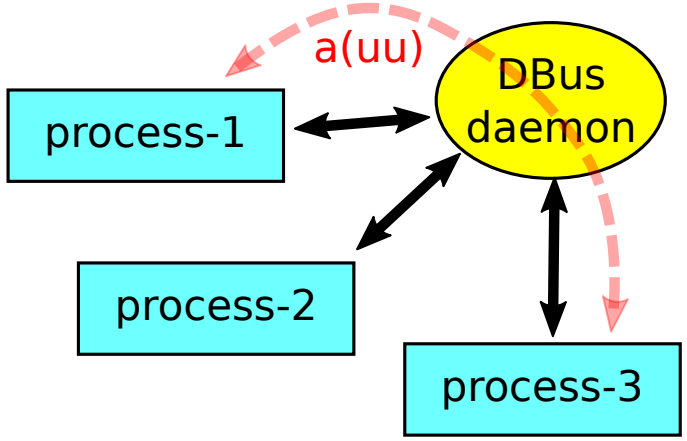
Unix File Descriptor Array Copy in Saftlib

Michael Reese

November 17, 2016

DBus



DBus provides

- typed high-level IPC between two processes
- daemon allows communication between all connected processes (function calls, signals, properties)
- low level C-API, rarely used directly

High-level APIs in various languages

- Gio: C-API, part of GTK+ support libraries
- Glibmm: C++ wrapper around Gio and Glib

Arrays on DBus

DBus Latency

- per transaction
  ≈ 0.4 ms on my computer
  ≈ 1.8 ms on a SCU3
- per amount of data
  ≈ 0.8 ms for 1000 uint32 on my computer
  ≈ 5 ms for 1000 uint32 on a SCU3

- Too slow for data transfer to the function generator.

C++ std::vector through unix pipe

writing process

```
vector<int> data(10000,42);
int size = data.size();
int fd[2];
pipe(fd); // pipe = 2 file
          // descriptors.
          // fd[0] for reading
          // fd[1] for writing
write(fd[1],&size,sizeof(size));
write(fd[1],&data[0],
           size*sizeof(data[0]));
```

reading process

```
vector<int> data;
int size = 0;
int fd[2] = ...  ;
// fd has to refer to same pipe
//   != same integer value


read(fd[0],&size,sizeof(size));
data.resize(size);
read(fd[0],&data[0],
             size*sizeof(data[0]));
```

- Use DBus to transfer the `fd[2]` array from one process to another and exchange `vector<T>` data through the pipe

Asynchronous DBus function call

client process

- prepare normal DBus-function arguments
- prepare a pipe `df[2]`
- asyncronous DBus function call with arguments and pipe
- put `vector` data into the pipe
- wait

  ...

- function returns: retrieve normal return values
- read `vector` data from the pipe

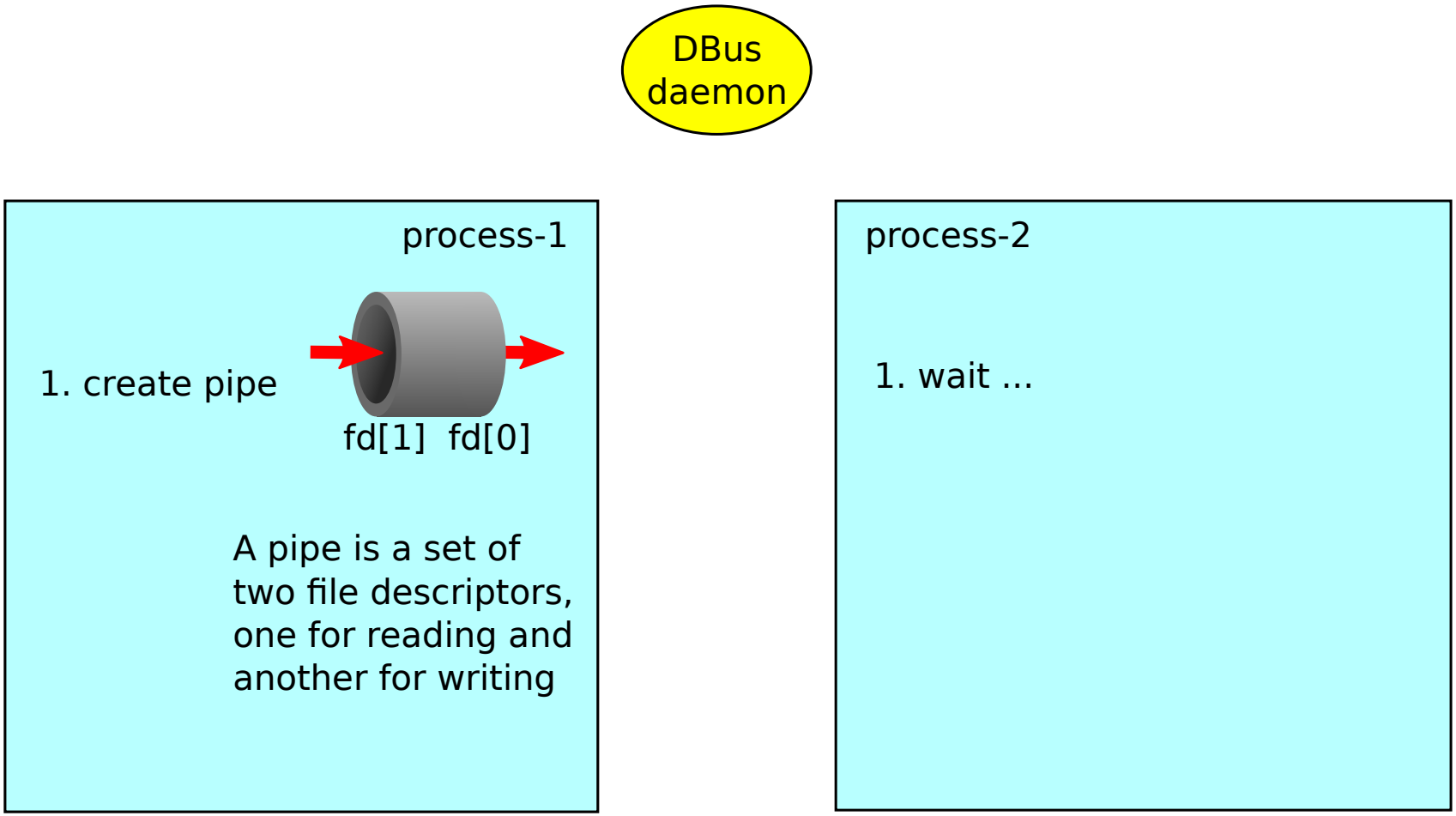
service process

- idle state

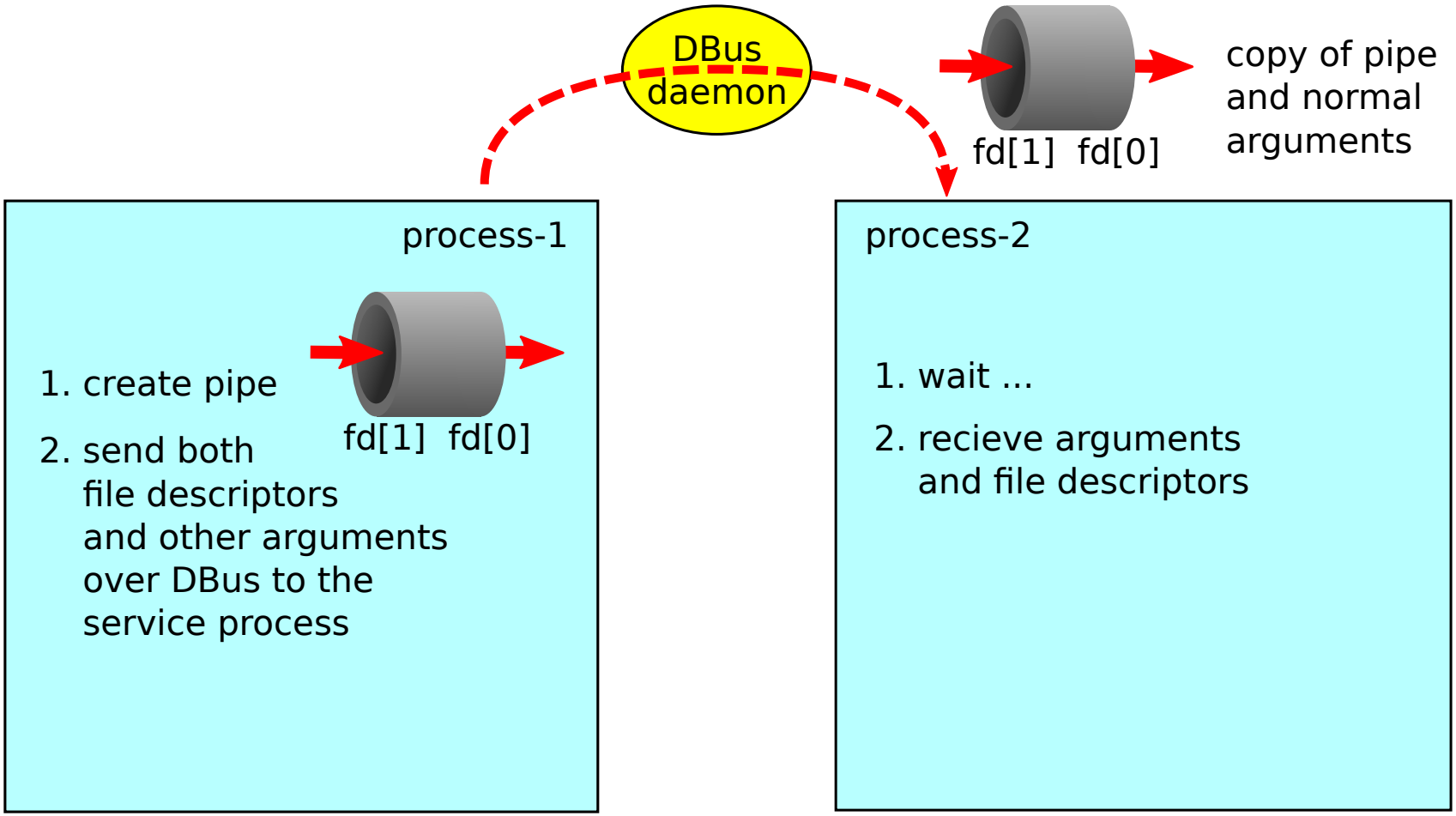
  ...

- receive a function call with arguments and a `fd[2]` pipe
- read vector data from the pipe
- calculate the result values and vectors
- prepare the result values as DBus arguments
- answer the DBus function call
- put vector data into the pipe

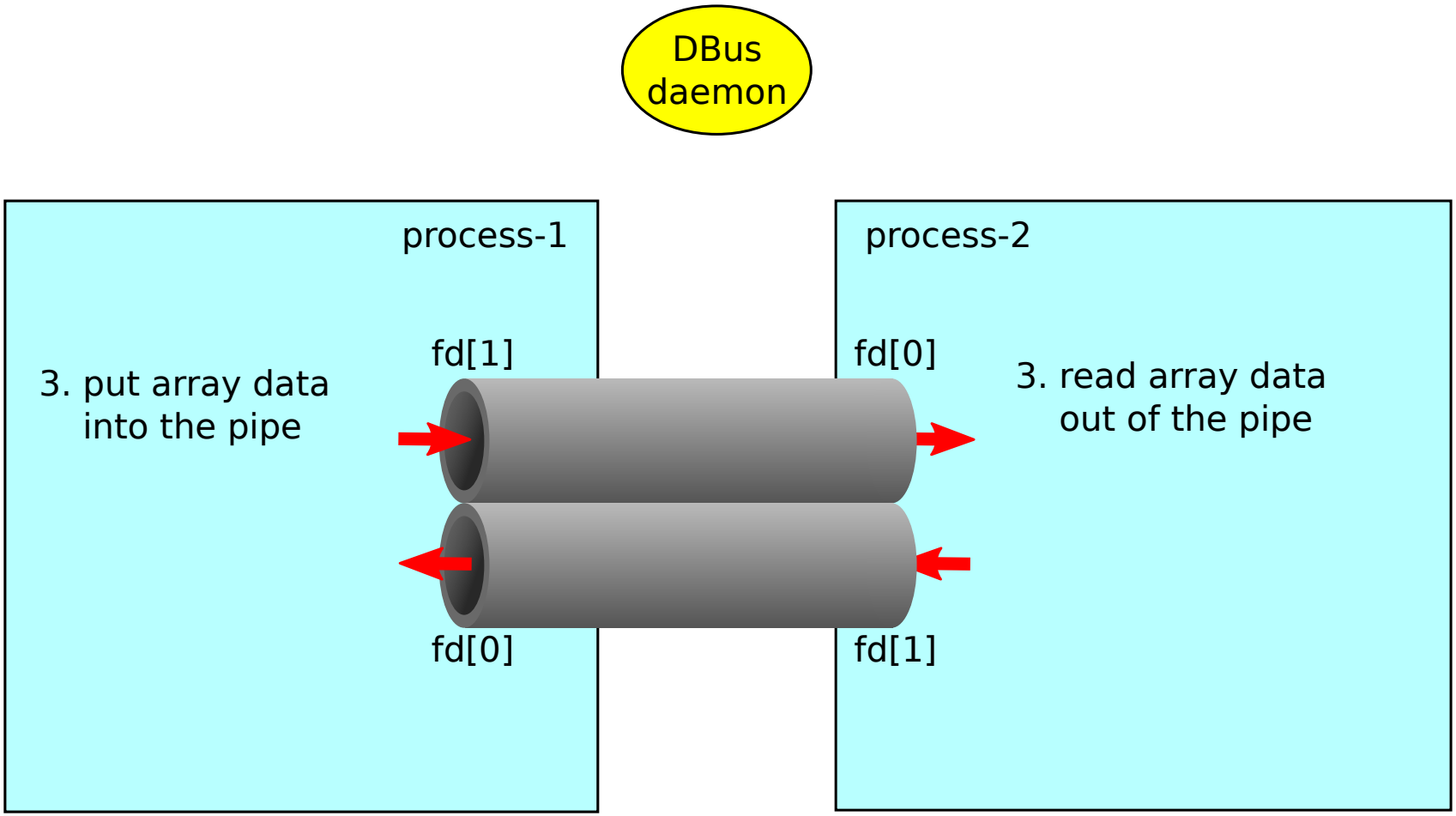
Asynchronous DBus function call (1)



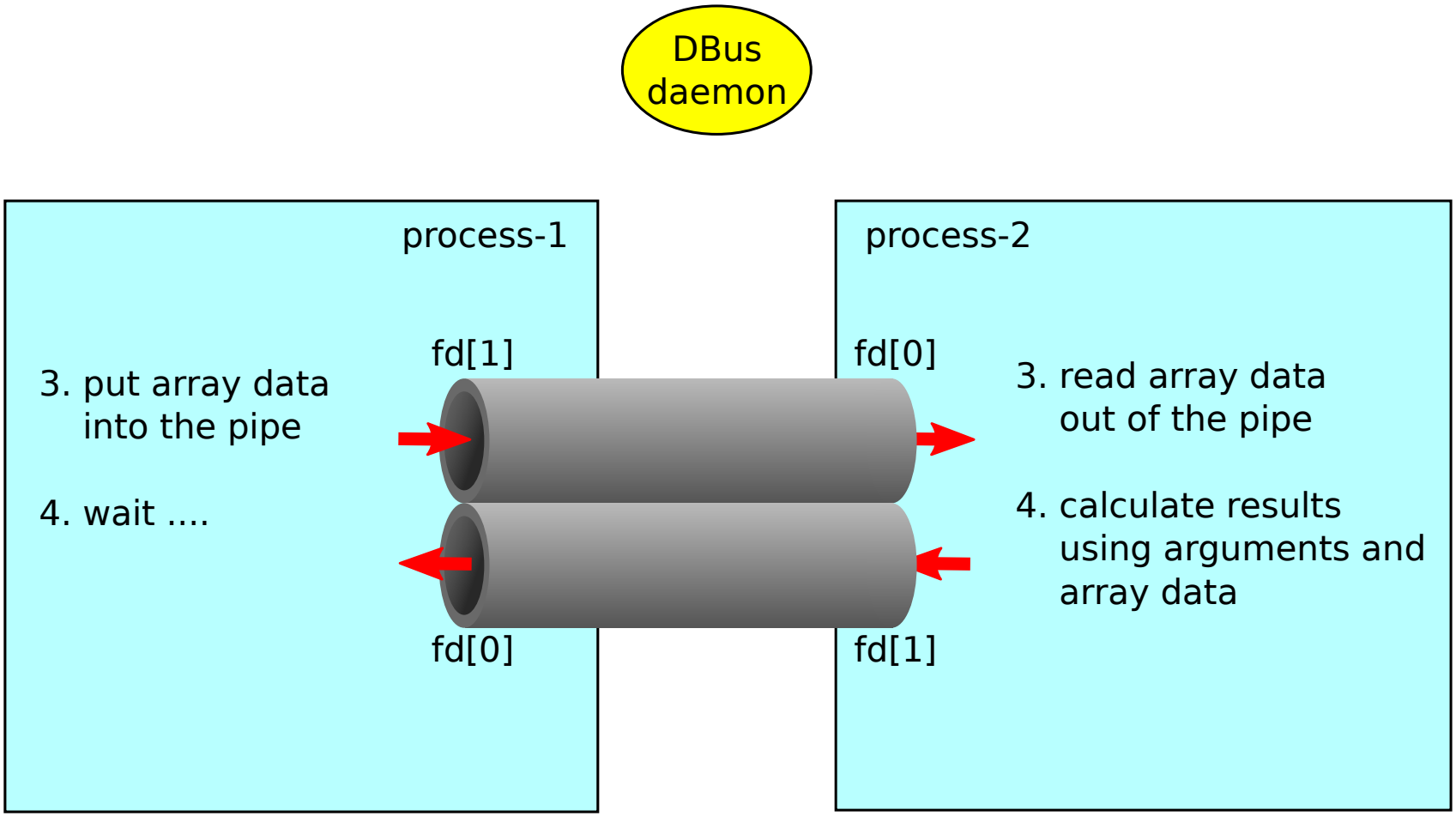
Asynchronous DBus function call (2)



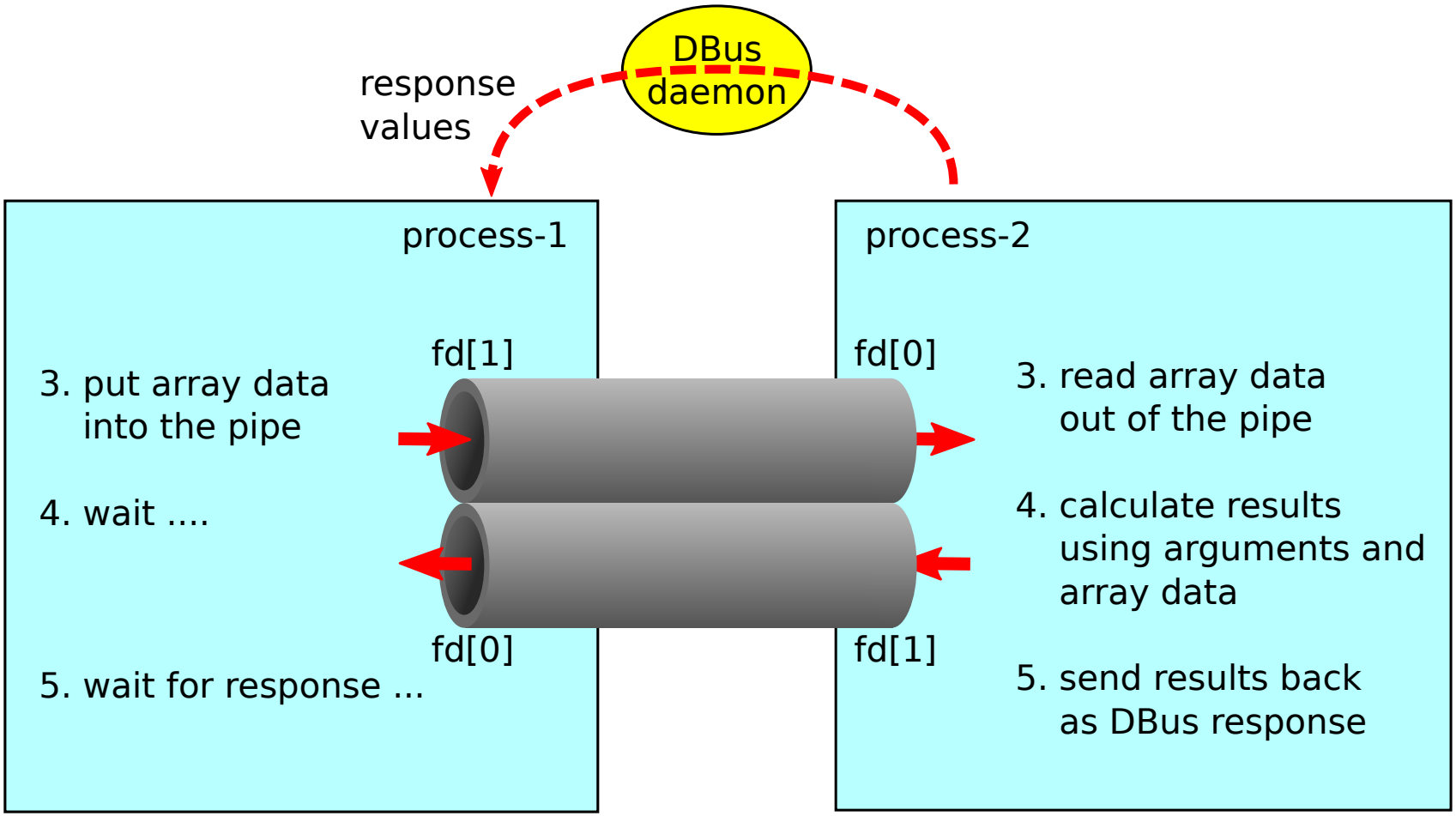
Asynchronous DBus function call (3)



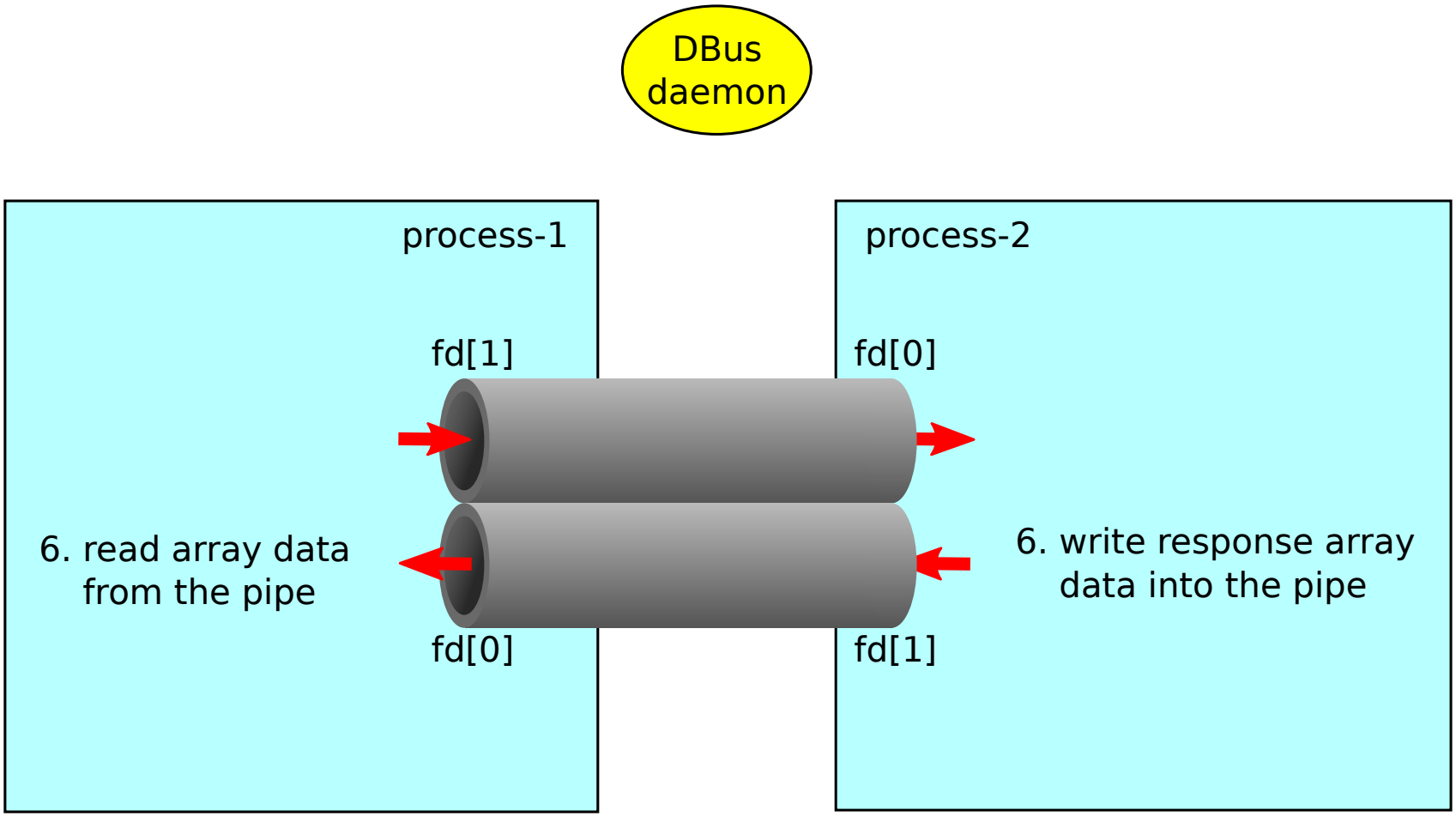
Asynchronous DBus function call (4)



Asynchronous DBus function call (5)



Asynchronous DBus function call (6)

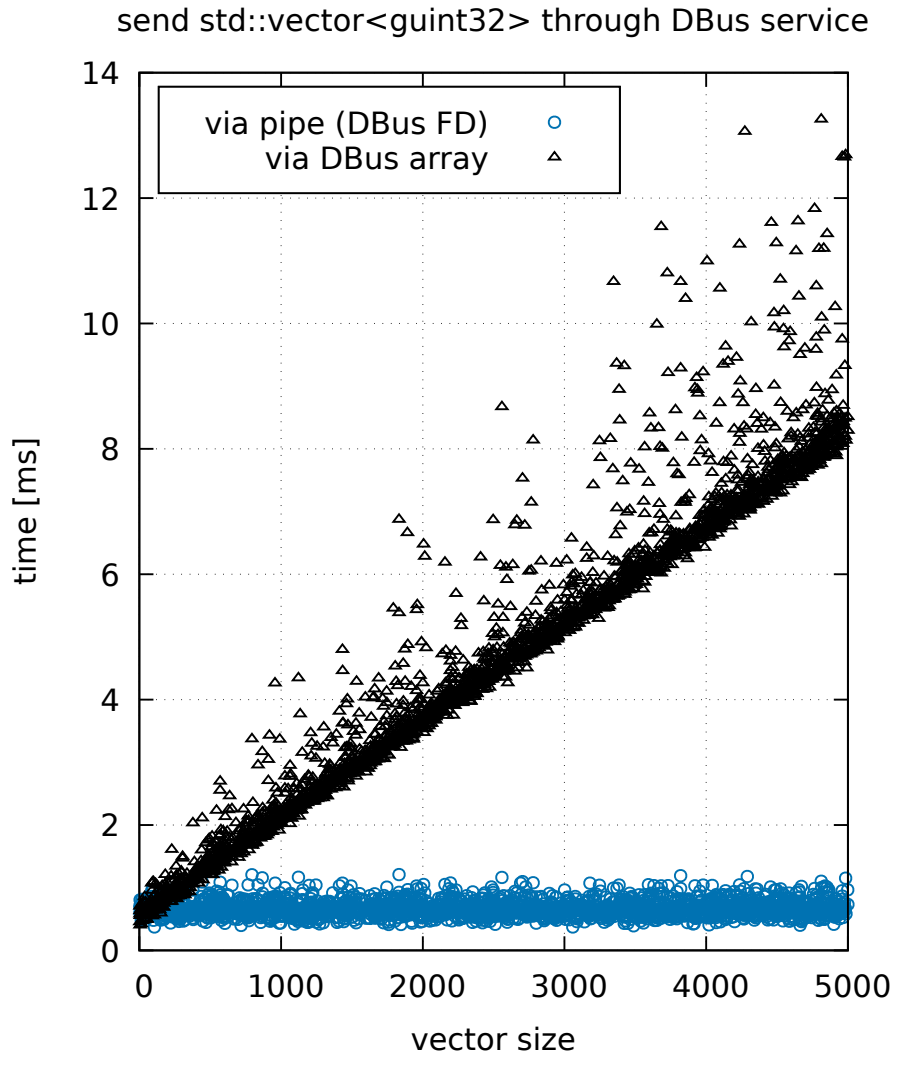


Implementation in Saftlib

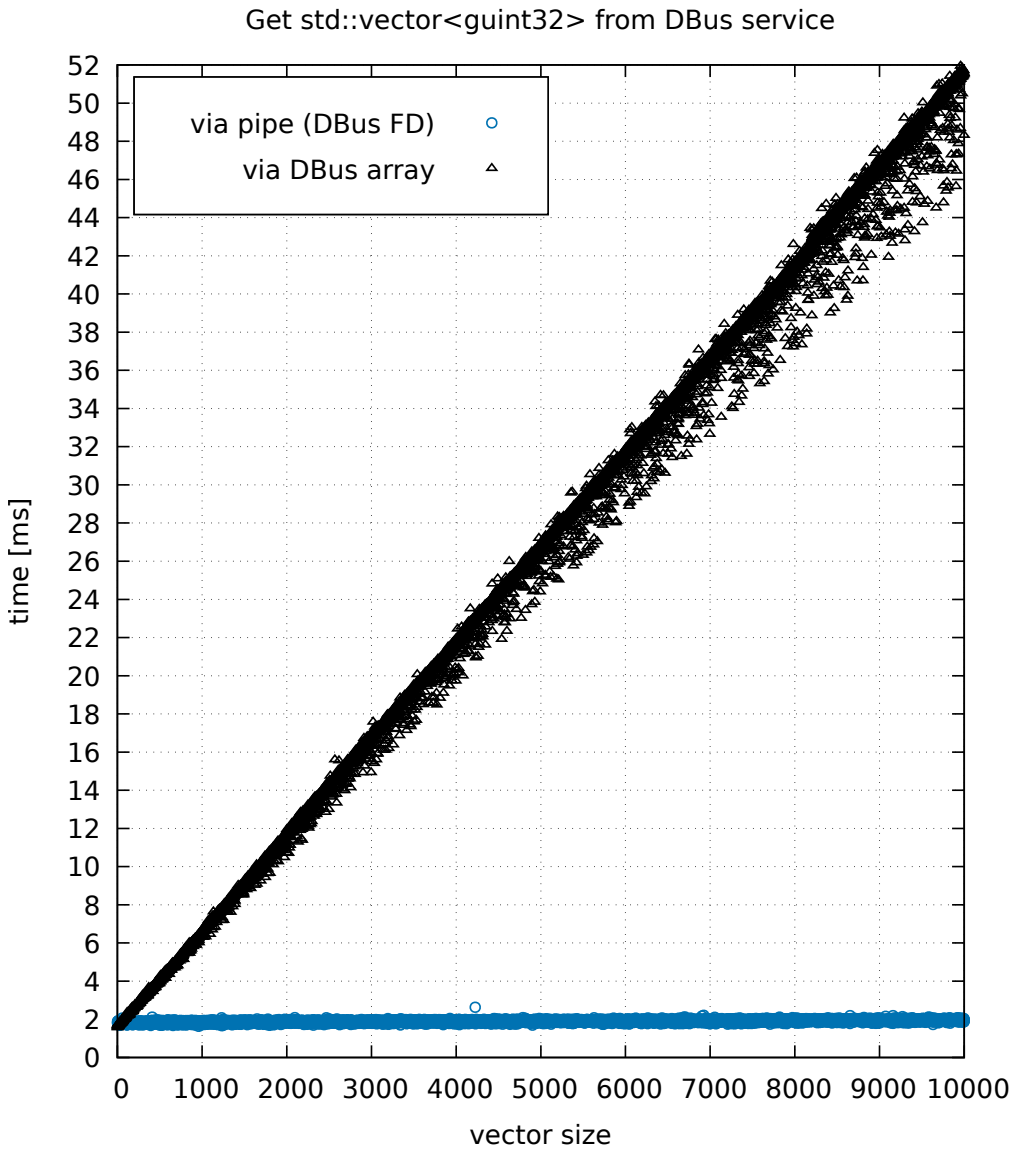
XML Interface “ax” → “Ax”

```
<method name="ArrayTest">
  <arg ...  type="au"/>
  <arg ...  type="au"/>
</method>
<method name="FDPipeTest">
  <arg ...  type="Au"/>
  <arg ...  type="Au"/>
</method>
```

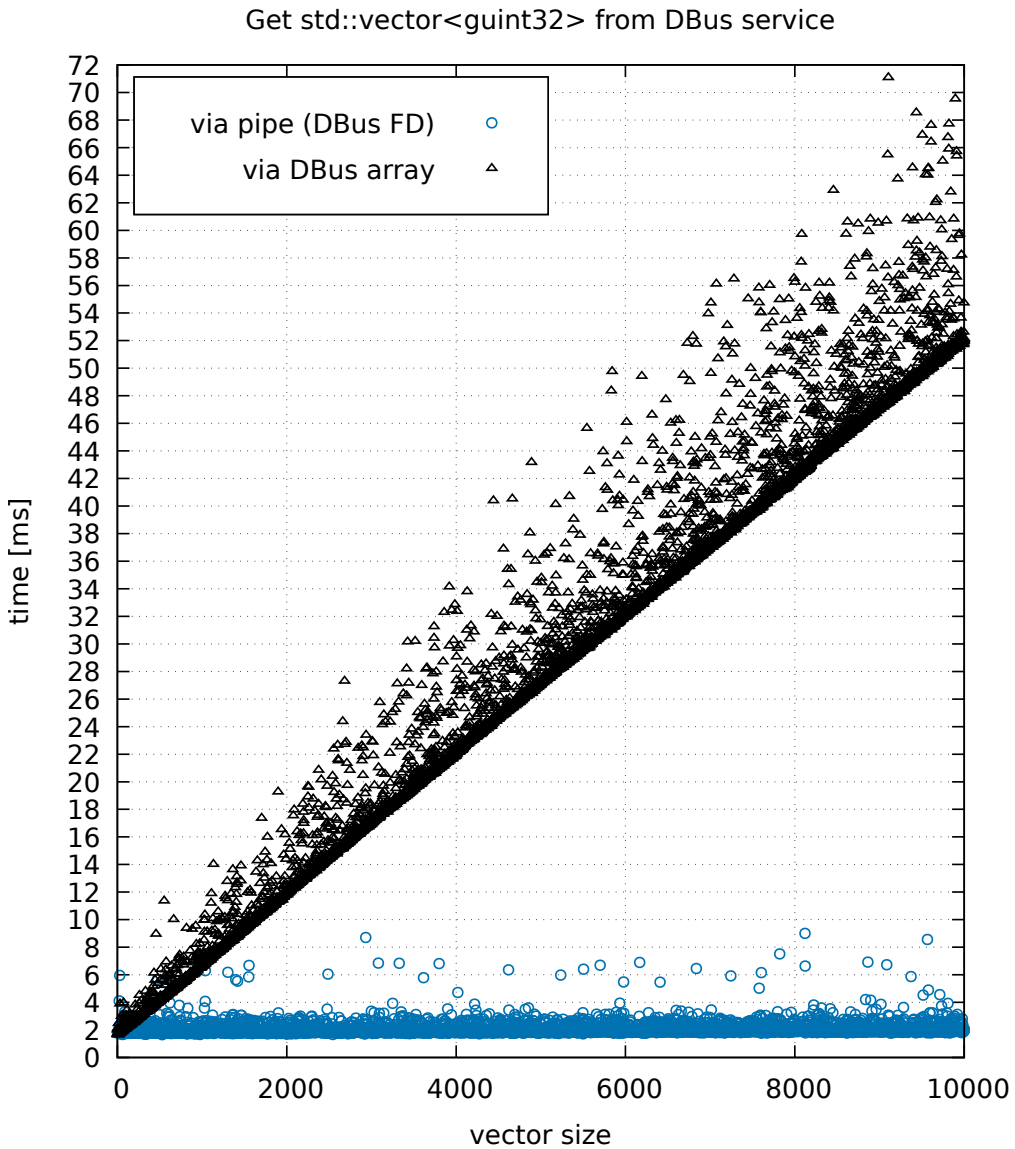
- large vectors are fast
- seems to work fine
- needs to be tested



SCU measurement: no traffic on DBus



SCU measurement: fg-ctl running



SCU measurement: pps-gen running