A Quick Guide for the pbdZMQ Package (Ver. 0.1-0)

Wei-Chen Chen[1] **and Drew Schmidt**[2]

[1]pbdR Core Team

[2]Business Analytics and Statistics,
University of Tennessee,
Knoxville, TN, USA

Contents

Acknowledgement

Chen was supported in part by the project "Bayesian Assessment of Safety Profiles for Pregnant Women From Animal Study to Human Clinical Trial" funded by U.S. Food and Drug Administration, Office of Women's Health. The project was supported in part by an appointment to the Research Participation Program at the Center For Biologics Evaluation and Research administered by the Oak Ridge Institute for Science and Education through an interagency agreement between the U.S. Department of Energy and the U.S. Food and Drug Administration.

Schmidt was supported in part by the project "Harnessing Scalable Libraries for Statistical Computing on Modern Architectures and Bringing Statistics to Large Scale Computing" funded by the National Science Foundation Division of Mathematical Sciences under Grant No. 1418195.

Disclaimer

The findings and conclusions in this article have not been formally disseminated by the U.S. Department of Health & Human Services, U.S. Food and Drug Administration, nor Oak Ridge Institute for Science and Education. They should not be construed to represent any determination or policy of University, Agency, Administration and National Laboratory.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Warning: This document is written to explain the main functions of **pbdZMQ** (Chen and Schmidt 2015), version 0.1-0. Every effort will be made to ensure future versions are consistent with these instructions, but features in later versions may not be explained in this document.

Information about the functionality of this package, and any changes in future versions can be found on website: "Programming with Big Data in R" at `http://r-pbd.org/` (Ostrouchov *et al.* 2012).

1. Introduction

ZeroMQ (ØMQ) Hintjens (2013b)[1] is a library for high-performance asynchronous messaging in scalable distributed applications. It provides APIs in several messaging patterns that, enabling developers a standardized way to form connections between different devices, including laptop computers, mobile devices, servers, clusters, and supercomputers. The APIs also simplify the complex calls to sockets and reduce the burden for developers of handling low-level network communications. Several popular programming languages provide bindings to these APIs.

In **pbdZMQ**, those ZeroMQ APIs are carefully wrapped in R via lower level C code and offers a few ZeroMQ patterns, including

- request-reply, in particular, one client and a server, and
- push-pull, in particular, one client and a set of servers.

These patterns are useful communication frameworks utilized in the **pbdCS** (Schmidt and Chen 2015) that combines two different messaging libraries, namely ZeroMQ and MPI, and utilizes their respective advantages in:

- user-to-server communication via **pbdZMQ**, and
- server-to-server computations for statistical programming via **pbdMPI** (Chen *et al.* 2012).

2. Installation

2.1. Installing

The **pbdZMQ** package requires an installation of the ZeroMQ library. So before we may discuss particulars of installing the R package, we take a moment here to describe the various ways in which you may install ZeroMQ itself. For convenience, we distribute with the package a distribution of ZeroMQ, although if you have access to a system installation, that may be preferable. We separate installation of ZeroMQ into 3 cases:

1. system package manager, such as the `libzmq` and `libzmq-dev` packages in Debian-derived systems,

[1] Available at `http://www.zeromq.org/`

2. **pbdZMQ**'s internal ZeroMQ library (4.1.0 rc1), or
3. an external ZeroMQ library (4.1.0 rc1 or later).

With System Package Manager This method is perhaps the easiest when a package managing system is available, such as on Linux, and returns the locations of the `libzmq` include and library paths via:

- `pkg-config -variable=includedir libzmq` and
- `pkg-config -variable=libsdir libzmq`.

In this setup, installation is very straightforward. From a shell, you can execute:

Shell Command

```
R CMD INSTALL pbdZMQ_0.1-0.tar.gz
```

Using Internal ZeroMQ This method uses the ZeroMQ library bundled with **pbdZMQ**, and should be fairly simple. This method has been successfully tested under Linux, Mac OSX, Windows, and FreeBSD. Solaris has been tested with no success.

Installation in this way can be simply done by adding the configure argument `-enable-internal-zmq`. In practice, this might look something like:

Shell Command

```
R CMD INSTALL pbdZMQ_0.1-0.tar.gz \
  --configure-args="--enable-internal-zmq"
```

Using External ZeroMQ This method assumes you have built your own ZeroMQ library somewhere, or perhaps one is offered to you by your system administrator. In any event, this method is only tested under Linux systems. As with the previous method, we were unsuccessful in our attempts to build on Solaris.

To build ZeroMQ yourself, you might do something like the following:

Shell Command

```
./configure \
   --prefix=/usr/local/zmq \
   --enable-shared=yes \
   --with-poller=select \
   --without-documentation \
   --without-libsodium
make -j 4
make install
```

which will install the library to `/usr/local/zmq/` where `/usr/local/zmq/include/` will have the header file `zmq.h` and `/usr/local/zmq/lib/` will have the shared library file `libzmq.so`.

With an external ZeroMQ available, we can install **pbdZMQ** via:

Shell Command

```
R CMD INSTALL pbdZMQ_0.1-0.tar.gz \
  --configure-vars="ZMQ_INCLUDE='-I/usr/local/zmq/include' \
                    ZMQ_LDFLAGS='-L/usr/local/zmq/lib -lzmq'"
```

2.2. Testing the Installation

To make sure that **pbdZMQ** is installed correctly, one may run a simple "hello world" test from *one* terminal to test the library as follows:

Shell Command

```
Rscript -e "demo(hwserver,'pbdZMQ',ask=F,echo=F)" &
Rscript -e "demo(hwclient,'pbdZMQ',ask=F,echo=F)"
```

This will run 5 iterations of sending and receiving 'Hello World' messages between two instances (simple server and client).

2.3. Polling System

Note that one may want to use different polling system provided by the ZeroMQ library. By default, the `select` method is used in **pbdZMQ** for Linux, Windows, and Mac OSX. However, users may want to use `autodetect` or try others for better polling. Currently, the options as given by ZeroMQ may be `kqueue`, `epoll`, `devpoll`, `poll`, or `select` depending on libraries and system. You may set the polling method at compile time via:

Shell Command

```
R CMD INSTALL pbdZMQ_0.1-0.tar.gz \
  --configure-vars="ZMQ_POLLER='autodetect'"
```

See the ZeroMQ manual for more details.

3. Examples

The package provides several simple examples based on *the ZeroMQ guide for C developers* by Pieter Hintjens (Hintjens 2013a). These are located in the `demo/` subdirectory of the **pbdZMQ** package source, and they include:

Examples	Descriptions
`hwclient.r`	hello world client
`hwserver.r`	hello world server
`tasksink.r`	task sink from two workers
`taskvent.r`	task ventilator send jobs to two workers
`taskwork.r`	task workers
`wuclient.r`	weather updating client
`wuserver.r`	weather updating server

For instance, the task examples can be run by

Shell Command

```
Rscript taskwork.r &
Rscript taskvent.r
Rscript tasksink.r

### Remember to kill two worker processors at the end, such as
ps -x|grep "file=task.*\.r"|sed "s/\(.*\) pts.*/\1/"|xargs kill -9
```

Or, via `demo()` function as the hello world example in Section 2.

The weather updating examples can be run by

Shell Command

```
Rscript wuserver.r &
Rscript wuclient.r
rm weather.ipc
```

Or, via `demo()` function as the hello world example in Section 2.

4. Backwards Compatibility with rzmq

This package currently has a few wrapper functions to offer the same API as that of the **rzmq** package (Armstrong 2014). The intent is to offer backwards compatibility as much as possible, but possibly with a reduced functionality set. Users are encouraged to use native `zmq.*()` functions provided by **pbdZMQ**.

The wrapper functions are:

Functions
`send.socket()`
`receive.socket()`
`init.context()`
`init.socket()`
`bind.socket()`
`connect.socket()`

References

Armstrong W (2014). "rzmq: R Bindings for ZeroMQ." R Package, URL http://cran.r-project.org/package=rzmq.

Chen WC, Ostrouchov G, Schmidt D, Patel P, Yu H (2012). "pbdMPI: Programming with Big Data – Interface to MPI." R Package, URL http://cran.r-project.org/package=pbdMPI.

Chen WC, Schmidt D (2015). "pbdZMQ: Programming with Big Data – Interface to ZeroMQ." R Package, URL http://cran.r-project.org/package=pbdZMQ.

Hintjens P (2013a). "The ZeroMQ Guide – for C Developers." URL http://zguide.zeromq.org/page:all.

Hintjens P (2013b). "ZeroMQ: The Guide."

Ostrouchov G, Chen WC, Schmidt D, Patel P (2012). "Programming with Big Data in R." URL http://r-pbd.org/.

Schmidt D, Chen WC (2015). "pbdCS: pbdR Client/Server Utilities." R Package, URL http://github.com/wrathematics/pbdCS.