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

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

[1]pbdR Core Team

[2]National Institute for Computational Sciences,
University of Tennessee,
Knoxville, TN, USA

Contents

Acknowledgement ii

1. Introduction 1

2. Installation 1

3. Examples 3

4. Backward Comparable with rzmq 4

References 5

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

Warning: 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.

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) at `http://www.zeromq.org/` is a library for high-performance asynchronous messaging in scalable distributed applications. It provides APIs in several messaging patterns that essentially the basic way to form connections between devices including computers, mobile devices, and of course supercomputers. The APIs also simplify the complex calls to sockets and reduce the burden of knowing network communications. Several popular programming languages have also provided biding to utilize and take advantage from those APIs.

In **pbdZMQ**, those ZeroMQ APIs are in detail manners wrapped in R via lower level C language. **pbdZMQ** mainly implements on 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 designed in **pbdCS** (Schmidt and Chen 2015) that combines two different messaging libraries ZeroMQ and MPI and utilizes their advantages in

- interaction of user interface by ZeroMQ or **pbdZMQ**, and
- computation of statistical programming by MPI or **pbdMPI** (Chen *et al.* 2012).

2. Installation

There are three ways to install **pbdZMQ**

1. with system `libzmq` and `libzmq-dev` packages (as `pkg-config` is available),
2. with internal ZeroMQ library (4.1.0 rc1), or
3. with external ZeroMQ library (newer version).

The first way is easiest when a package managing system is available and produces both

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

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

The second way is tested under Linux, Mac OSX, and Windows systems, while the third way is only tested under Linux system. Solaris has been tested without any success in neither ways. FreeBSD is not tested yet.

Installation with system `libzmq` and `libzmq-dev` packages: it can be simply done with

Shell Command

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

Installation with internal ZeroMQ library: it can be simply done with

Shell Command

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

Example steps for installing with external ZeroMQ library:

- The minimum steps for installing **zeromq-4.1.2.tar.gz** is

  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`.

- Therefore, we can install **pbdZMQ** as next:

  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'"
  ```

Suppose **pbdZMQ** is installed correctly. One may run the next from *one* terminal to test the library.

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 to send and receive 5 times of 'Hello World' messages between two instances (simple server and client).

Note that one may want to use different polling system provided by ZeroMQ library. By default `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. The next may be used to change the polling method:

Shell Command

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

3. Examples

The package provides simplest examples based on the ZeroMQ guide for C developers by Pieter Hintjens (Hintjens 2013). The source codes next are all in the `pbdZMQ/demo/` directory, such as

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 example, 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. Backward Comparable with rzmq

The package currently has a few wrapper functions to server as the same functions in **rzmq** (Armstrong 2014). The original intension is only for backward comparable but without 100% fully functional guarantee. Users are recommended to use native `zmq.*()` functions provided by **pbdZMQ**.

The wrapper functions are listed in next

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

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.