WR characterization tools

Measuring bandwidth, lost and corrupted packages

Author: Miguel Jiménez López
Email: klyone@correo.ugr.es
Supervisor: Javier Diaz Alonso
University: University of Granada

Table of contents

1 Introduction

These applications have been developed in the framework of White Rabbit project. White Rabbit is a technology that can give a performance and accuracy below the nanosecond. This project is led by CERN. You can see more information in [1]. We will use two SPEC cards connected to same PC by PCI bridge. You can obtain more information about SPEC cards in [2], [3], [4] and [5].

In this document, it will be presented a series of programs that measure link bandwidth to Ethernet and White Rabbit technology. **wr-sender** and **wr-receiver** are C programs developed for this goal. **wr-sender** sends bursts of packages to **wr-receiver** and latter receives all of them and calculates bandwidth, lost and corrputed packages statistics. **wr-sender** and **wr-receiver** are inspired in **wr-agent** and **wr-ruler** codes developed by Alessandro Rubini. If you want to get more information, you can visit [2]. To do easier this measurement, we are designed Python script that calls C programs with parameters needed. In next sections, we will explain C and Python programs in detail.

2 Hardware and software platform

In this section, we describe hardware and software tools used.

1. **Software:**
   1. **Tools:**
      1. gcc 4.6.3
      2. Python 2.7
      3. C Network library (to use RAW and TCP sockets)

   2. **OS:**
      1. Ubuntu 12.04 LTS (64 bits)

2. **Hardware:**
   1. Intel core i7-2600k CPU @ 3.40GHz (8 cores)
   2. 15.7 GiB RAM
   3. 46.0 GiB of Swapping partition
   4. 336 GiB of Disk
   5. 3 x Gigabit Ethernet port
   6. 2 x SPEC card (in PCI ports)
   7. Nvidia GeForce GTX550

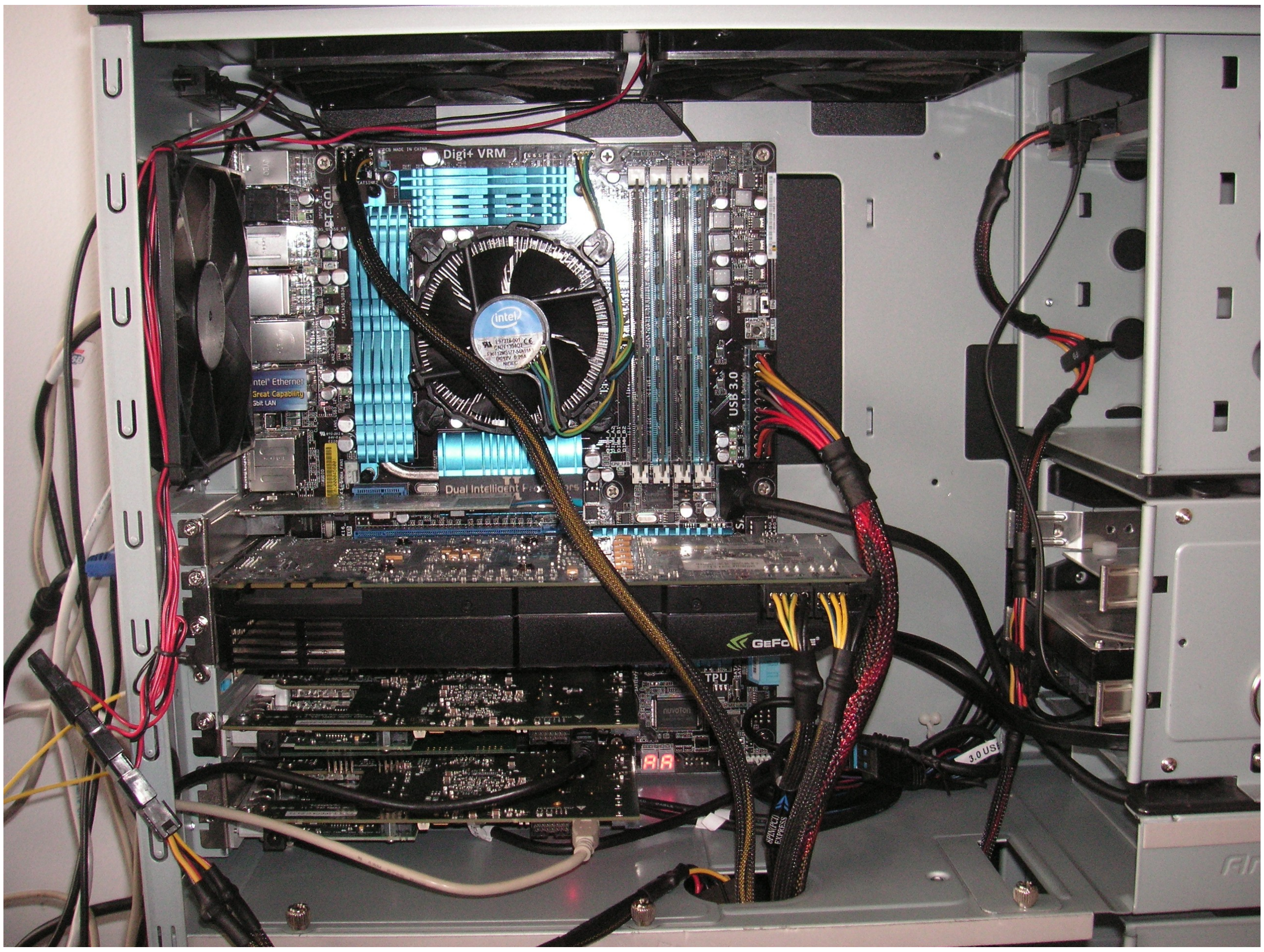


Image 1. PC used

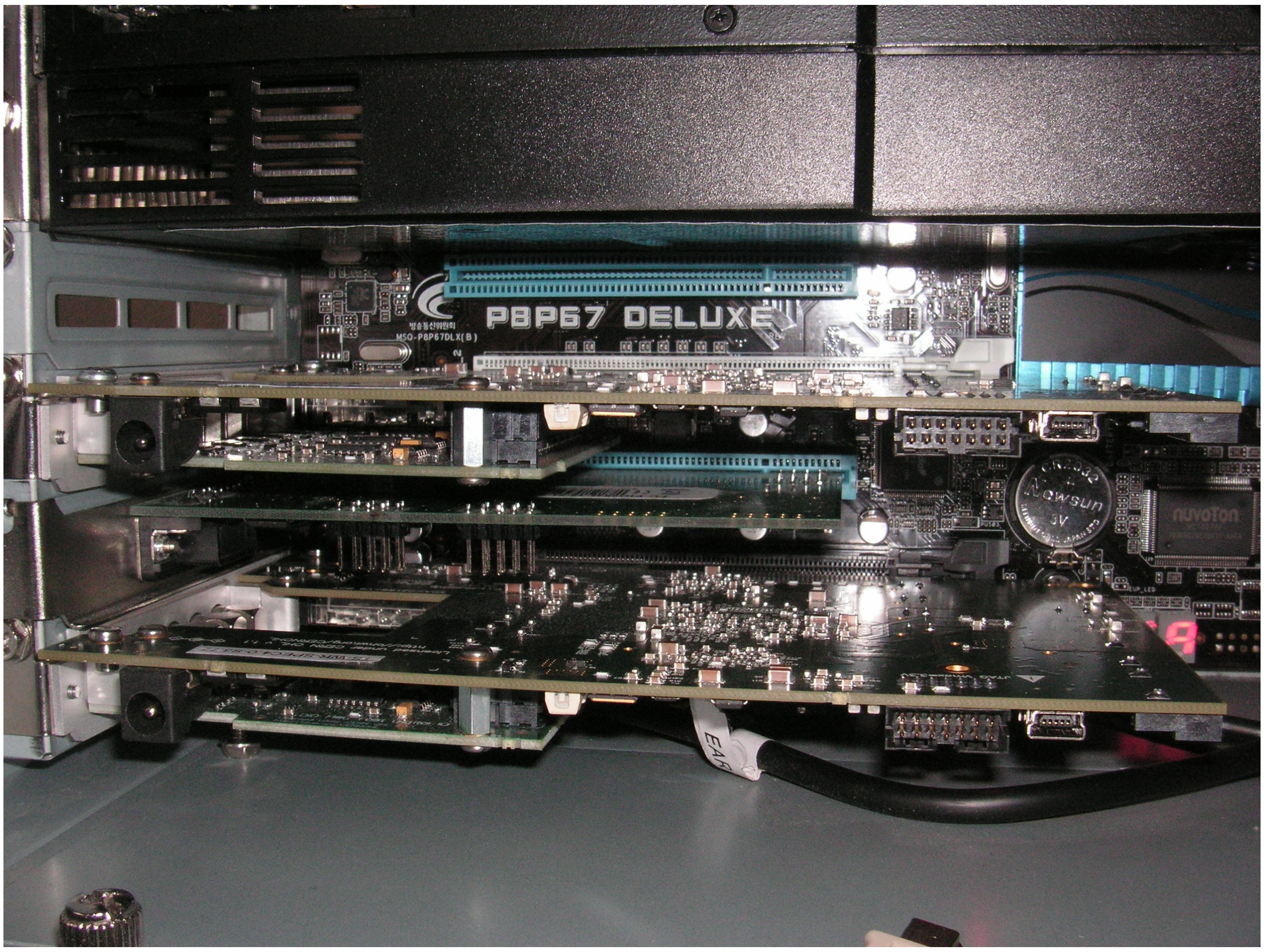


Image 2. SPEC cards in PCI bridge

3 Test tools

3.1 Initial considerations

- These applications are not intended to accurately measure the bandwidth.
- TCP sockets have been used to synchronization mechanism.
- RAW sockets have been used to send/receive packages in each test.
- It allows the sender and receiver are on the same machine or different.
- The application is robust against failure and that is collecting the temporary results of tests already taken.

3.2 Goals

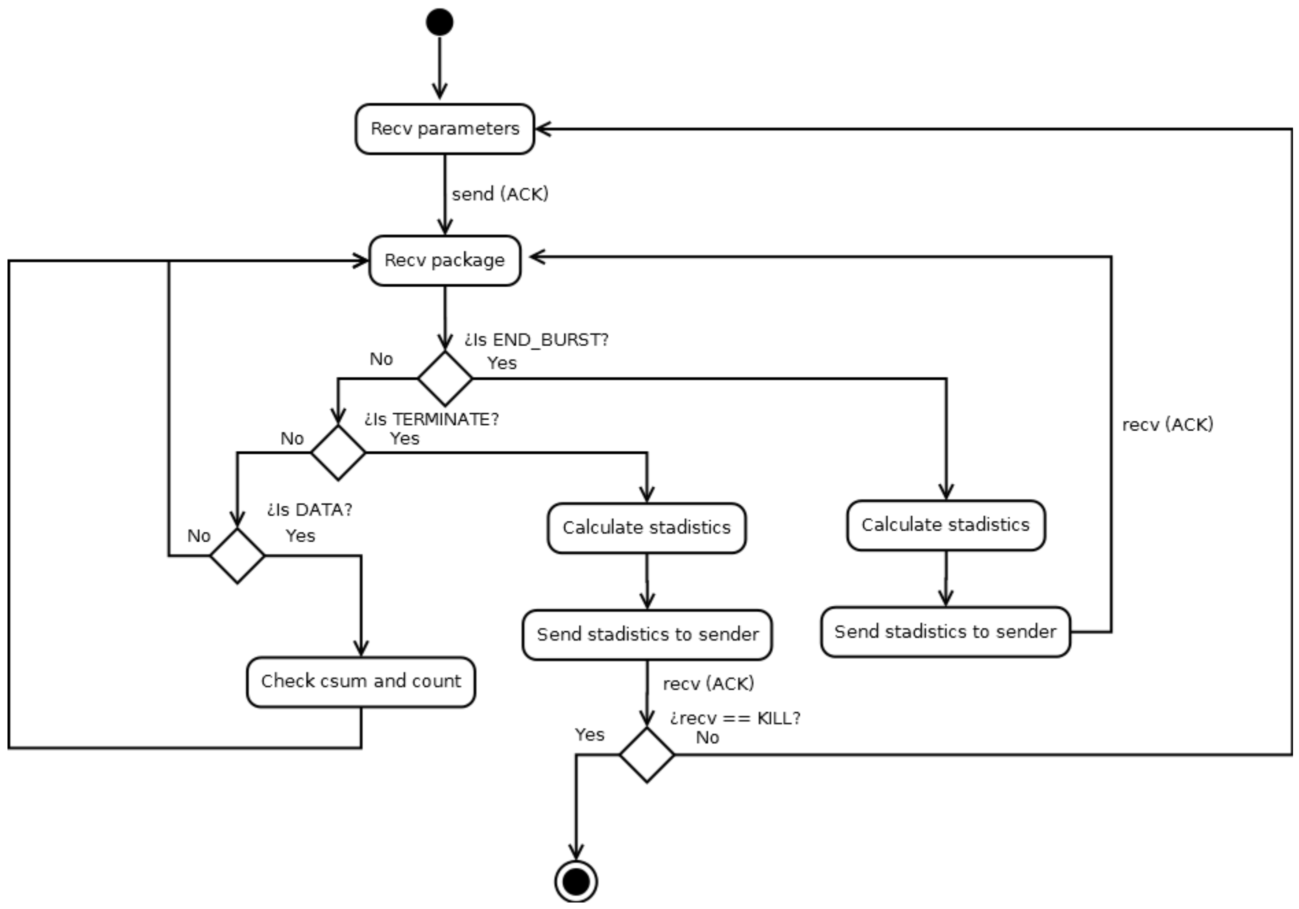
The ultimate goal of these applications is to give an idea of the performance provided by the system to perform the assigned task. Likewise, it attempts to detect bottlenecks and motivate further developments.

3.3 wr-sender and wr-receiver

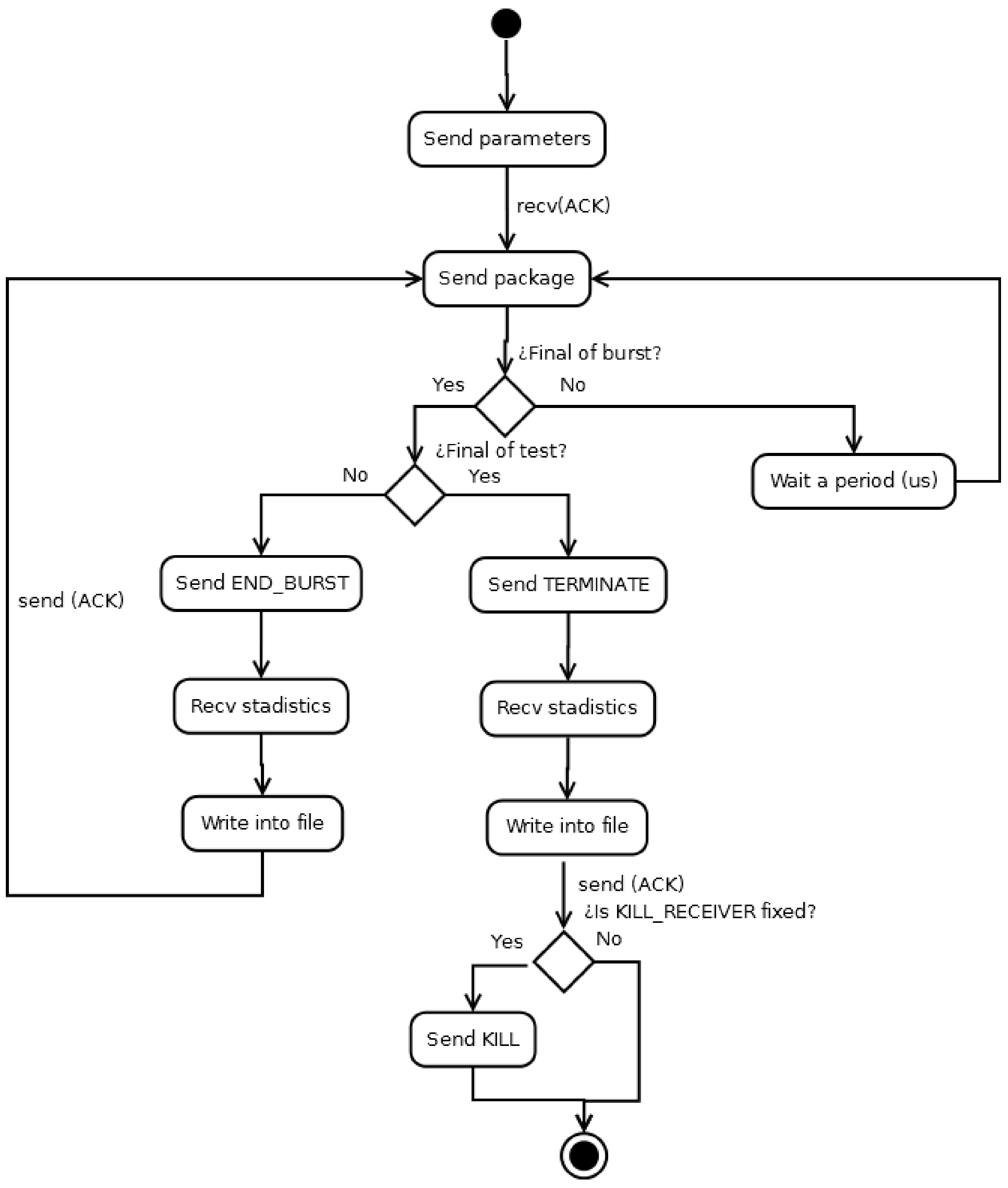
C programs implement network protocol measures bandwidth and other statistics.

wr-sender is responsible for sending bursts of packages to the receiver. It must inform receiver of the emission parameters: number of burst, number of packages per burst, package size and emission period. Once these parameters reported, transmitter starts sending each of the burst. **wr-sender** must wait for receiver acknowledge for each burst (it allows receiver can calculate statistics). Then, **wr-receiver** calculates bandwidth, lost and corrupted measures and it sends them to sender. **wr-sender** stores measures in file for each burst. Finally, **wr-sender** must inform receiver that all bursts have finished (it sends special package).

wr-receiver state diagram



wr-sender state diagram



In next section, we explain how you can use **wr-sender** and **wr-receiver**.

3.3.1 wr-sender

If you want to use **wr-sender**, you have to respect next syntax:

```
wr-sender <wr-if> <period (us)> <size package> <number packages per burst> <number of bursts> <kill_receiver>
```

```
wr-sender <wr-if> <receiver host name> <period (us)> <size package> <number packages per burst> <number of bursts>
<kill_receiver>
```

We are going to detail program parameters:

1. **wr-if:** Interface name (it may be ethernet interface or SPEC interface, see USE_SPEC for more information)

2. **receiver host name:** Hostname that will be used in case of wr-receiver is not in local machine

3. **period:** Emission period in microseconds

4. **size package:** Size of data frame

5. **number packages per burst:** Number of packages in each burst

6. **number of bursts:** Number of bursts in test

7. **kill_receiver:** it allows wr-sender to kill wr-receiver when test finishes

3.3.2 wr-receiver

If you want to use **wr-sender**, you have to respect next syntax:

```
wr-receiver <wr-if> <localhost>
```

We are going to detail program parameters:

1. **wr-if:** Interface name (it may be ethernet interface or SPEC interface, see USE_SPEC for

more information)

2. **localhost:** it indicates if **wr-sender** and **wr-receiver** are in same machine.

3.4 test-bw

We have developed **test_bw** to facilitate the task of taking measures of bandwidth, lost and corrupt packages. **test_bw** is a Python script that programs test with **wr-sender** and **wr-receiver.** You must create a configuration files to indicate test(es) will be executed by **test_bw**. When script finishes a test, it gets C program output and converts it into CSV file (to export Calc/Excel tables and graphs). This tool is robust against failure and that is collecting the temporary results of tests already taken.

3.4.1 Global configuration file

Global configuration file (**global.cfg**) contains global information for **test_bw.** You can write comments with # symbol before your message. We are going to describe parameters you can configure in this file:

1. **CSV_FILE:** It indicates output CSV name file to test

2. **TEST_FILE:** It indicates test configuration file. It contains all test that **test_bw** will execute

3. **RECEIVER_START:** It allows to start **wr-receiver** automatically. If you don't use this parameters, **test_bw** doesn't call **wr-receiver** (initiated supposed)

3.4.2 Test configuration file

Test configuration file contains information about test will be executed by **test_bw**. You can write comments just as **global configuration file**.

Each line of file is a test. Each test begins with **TEST** header. Then, you must specify test parameter (what are going to change in test). Finally, you must specify rest of values needed to test.

1. **TEST:** Header of each test line

2. **SIZE:** Number of bytes (frame)

3. **NBURSTS:** Number of bursts in test

4. **NPACKG:** Number of packages in burst

5. **PERIOD:** Emission period in microsecond

Note: If you want to execute **wr-receiver** in remote machine you must add **RUN_REMOTE** parameter.

```
TEST <parameter> <init value> <step> <final value> <rest of values>
```

Examples:

1. TEST SIZE 100 10 220 NBURSTS 10 NPACKG 1000 PERIOD 1000
2. TEST NBURSTS 1 1 10 SIZE 1400 NPACKG 1000 PERIOD 100

4 References

[1] White Rabbit project repository,Dietrich Beck, Erik van der Bij, Javier Serrano,Maciej Lipinski,2013,http://www.ohwr.org/projects/white-rabbit

[2] SPEC software repository,Alessandro Rubini,Juan David González Cobas, Matthieu Cattin,2013,http://www.ohwr.org/projects/spec-sw

[3] SPEC board repository,Erik van der Bij, Javier Serrano, Matthieu Cattin, Tomasz Wlostowski,2013,http://www.ohwr.org/projects/spec

[4] White Rabbit NIC repository, Benoit Rat, Javier Díaz,2013,http://www.ohwr.org/projects/wr-nic

[5] White Rabbit PTP Core – the sub-nanosecond time synchronization over Ethernet, Grzegorz Daniluk,2012,http://www.ohwr.org/documents/174