Using VDIF in astronomy VLBI observations at the 40m RT. Flexbuff and eVLBI operations

P. de Vicente, L. Barbas, J. González

Informe Técnico IT-CDT 2016-12

Revision history

Version	Date	Author	Updates
0.8	20-02-2016	P. de Vicente	First draft

Contents

1 Introduction

VDIF stands for VLBI Data Interchange Format and it is an universal standardized VLBI raw-data format that can be used for both eVLBI (network transmision) and data storage (recording systems). It was ratified in 2009 and it is increasingly being adopted by the VLBI community. Prior to it the most widely used format was VLBI Standard Interface (VSI), a standard electrical/timing (VSI-H) and supporting software (VSI-S) specification which was ratified in 2000. This latter format was the one used for Mark5Bs. Information on VDIF can be found in Whitney et al. (2009) in http://www.vlbi.org/vdif/docs/Whitney-VDIF_paper_for_IVS_GM_Tasmania.pdf

VDIF has started been adopted in the EVN since 2013 with the usage of Flexbuffs and Mark6 at Onsala and Effelsberg. The adoption takes time because stations and correlators mainly use Mark5Bs. The first recorder being able to use VDIF is the Mark5C, but this is not a widely adopted recorder except in the VLBA by NRAO (only Mc uses a Mark5C in the EVN). However the usage of Flexbuff and Mark6 will boost its usage worldwide.

Flexbuffs are computers equipped with many SAS disks that act as storage servers. Its main advantage is that they are Commercial off-the-shelf products (COTS) and use open and free software as opposed to Mark5Bs which use propietary firmware. Data are recorded and transferred locally using optical fiber trasmission and transferred later to the correlator using high speed lines. The Observatorio de Yebes has recently bought 2 Flexbuffs to replace its Mark5B recorders.

VDIF format is generated by the DBBC2 Fila10G unit which can produce two VDIF data flows up to 4 Gbps each through two optical fibers. The Field System supports Fila10G commands since version 9.11.8 and that has eased the adoption of VDIF format in eVLBI.

This report describes our experience using the Fila10G, installing and using Flexbuffs and how we have adopted "VDIF" eVLBI at Yebes. Obviously all the the work described here is a collective effort of the Technical and Operations Group of the EVN. Spectial mentions should go to **Harro Verkouter** (JIVE), **Paul Boven** (JIVE), **Mark Kettenis** (JIVE), **Bob Campbell** (JIVE), **Bob Eldering** (JIVE), **Simon Casey** (Onsala), **Jun Yang** (Onsala), **Uwe Bach** (MPIfR), **Guiseppe Macafferi** (INAF) and **Ed Himwich** (NASA-NVI).

2 Fila10G: a formatter

The DBBC2 Fila10G acts as a formatter which generates VDIF data. The Fila10G gets the data flow from the CORE2s through two high speed buses, VSI1 and VSI2, which can transport 2 Gbps at most and generates two VDIF data flows up to 4 Gbps in total through two optical ports equipped with XFP transceivers. There are two types of Fila10G units: internal and external. The DBBC2s at Yebes are equipped with internal units. The internal units can be directly connected to the CORE2s with flat high speed cables at a rate of 4 Gbps per CORE, but these cables can't be used with the standard VSI cables. By the time of this report we have never used them.

If the internal Fila10G is connected to both buses, the VSI outputs towards the Mark5Bs can't be used. So using VDIF and VSI are mutually exclusive, However there are cases in which we can have one VSI ouput and a "reduced" VDIF (only one VDIF port is available).

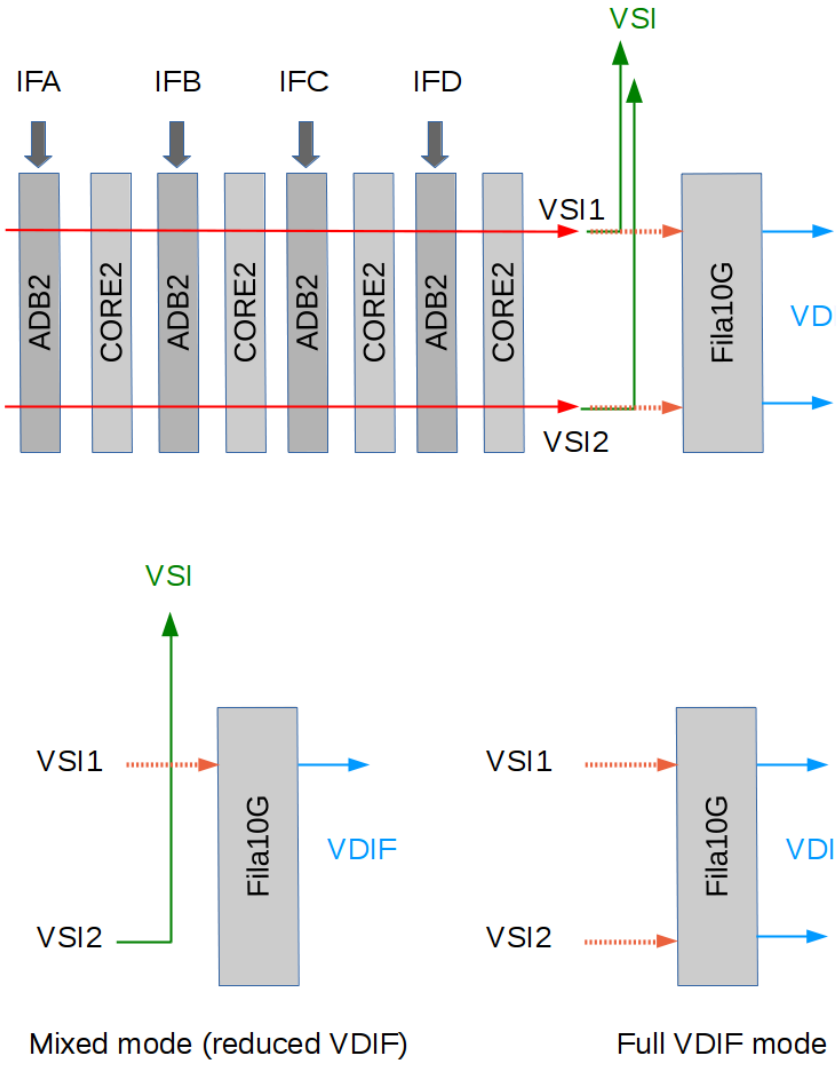

Figure 1: *Internal connections in a DBBC2 with an internal Fila10G unit. The green arrows indicate the data connection when using 2 VSI outputs towards a Mark5B. The blue arrows when using the VDIF outputs. By the time of this report we are using the "mixed" mode which has one VSI output and one VDIF output*

Fig. 1 shows a schematics with the possible connections.

The “mixed” mode can be used in “astro”, “astro2” and “geo” modes since the internal VSI2 bus is a copy of the internal VSI1 one and we only use one VSI channel with a recording rate of at most 2 Gbps, both in DDC and PFB. If we use PFB mode at 4 Gbps rates composing the signals from VSI1 and VSI2 the “full VDIF mode” is required since both VSI1 and VSI2 should feed the Fila10G. The Fila10G output can be such that it mixes the inputs from both VSI channels and get the data through VDIF1 and a copy in VDIF2 (the standard way) or it is also possible to have the VDIF1 input connected to the internal VSI1 bus and VDIF2 to the VSI2 internal bus and hence have separate outputs.

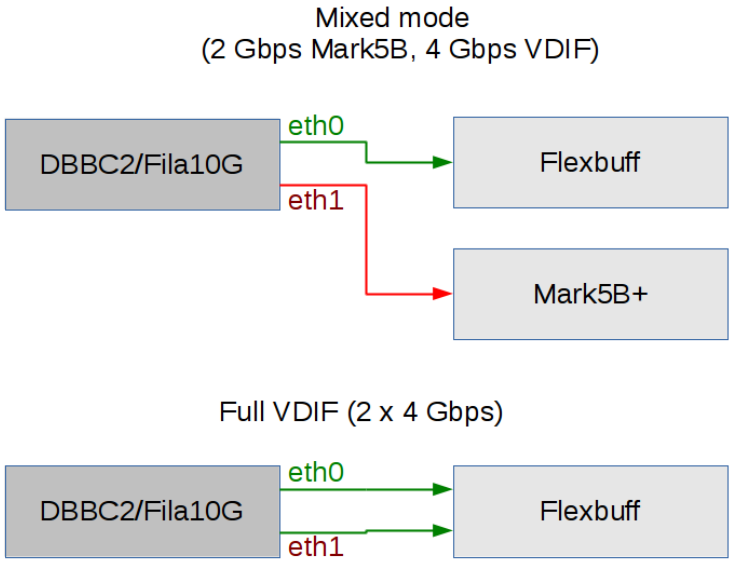

Figure 2: *“Mixed mode” and “full VDIF mode”.*

External Fila10G units have 2 input VSI connectors and 2 output VSI connectors. It is possible to have all four VSI connectors as inputs, or copy the two VSI inputs into two VSI outputs. These units as the internal ones require XFP transceivers in the optical ports.

Some Fila10G units have a GPS receiver inside which allows to keep the time. The receiver needs to be connected to an antenna and an external BNC connector is available at the DBBC2 (internal Fila10G) or at the Fila10G box (external Fila10G).

Communication, that is monitor and control, to the Fila10G is done via a serial RS232 connection. External units also have an ethernet port, but it can’t be used with the serial port simultaneously. It is necesary to choose between the two input options. Since DBBC DDC version 105 and PFB V15, Fila10G commands are routed through the DBBC2 Control software. For this to work the Fila10G has to be connected through a serial port to the DBBC2. The serial port to be used is included in the DBBC2 configuration file. For example, file `dbbc_config_file_105.txt` has a line starting by 2 which indicates that there is a Fila10G unit, followed by the firmware to be loaded and the serial port to be used.

```
2 fila10g_v3.3.2_1.bit com2
```

The Field System supports Fila10G commands since version 9.11.8 and the control files need to be modified to take it into account. Section 7 describes in detail all modifications required in the control files and procedures to use the Fila10G.

3 Flexbuff: hardware characteristics

At the time of this report we have two units, one to be used at Yebes and a similar one to be sent to JIVE correlator. A third one with a storage capacity of 216 TB is expected soon.

The Flexbuff is a Supermicro general purpose host prepared to be mounted in a rack, 4 Us in size and equipped with 36 4 TB disks. The unit is rather heavy and was delivered with two telescopic stainless steel guides to mount the equipment. The guides require a deep rack ($>$ 800 cm) and allow the unit to be extracted easily. The list of components was investigated at Yebes (Barbas & de Vicente) after being advised by P. Boven at JIVE and Simon Casey at Onsala. The final list of components is included below:

- One (1) box SUPERMICRO 4U 36 bays Ref. SC847AR1400LP – Main board: SUPERMICRO X9DRI-3F dual XEON ES-2600
- Two (2) processors INTEL XEON 6 CORES 2,6GHz 15Mb Cache LGA2011 Ref. ES-2630V2
- Two (2) radiators XEON E5-2600 2U, narrow ILM 2011 Ref. SNKP0048PS
- Four (4) DDR3 8GB 1600MHz ECC Reg LOW VOLTAGE Ref. DDR3M8GBECC
- Four (4) controllers SAS-2 LSI 2308 8 ports 6GB/S (8 internal) RAID 0, 1,10 Ref. AOCS2308LLBI
- Nine (9) cable SUPERMICRO CBL-0281L 75cm iPass to iPass Ref. CBL0281L
- Thirty-six (36) hard disks 4TB SATA 3 6Gb/s CONSTELLATION ES.3 Ref. HD4TBSATAllns
- One (1) card SUPERMICRO 10Gb ETHERNET 2 PUERTOS RJ45 Ref. AOCSTGI2T– CHIPSET INTEL X540 CON I/OAT, PCI-E x8 (JIVE)
- One (1) card SUPERMICRO 10Gb OPTICS 2 PORTS SPF+ - Intel Corporation 82599ES 10-Gigabit SFI/SFP+ Network Connection (YEBES)
- Four (4) cable SUPERMICRO 10.5”, 4 PIN FOR FAN EXTENSION WITH PWM Ref. CBL0088L
- One (1) SATADOM SUPERMICRO 64GB Ref. SSDDM064PHI

The internal hard disk is the SATADOM, a solide state device. The unit for the correlator is equipped with a 10 Gb RJ45 port, whereas the unit at Yebes has a dual optical port card. Two SFP+ transceivers were also delivered. The ethernet port for primary communications is a 1

Gbs ethernet port in the main board. The two optical ports are to be used with the DBBC2, either directy or via a 10 Gbs switch.

Figs. 3 and 4 shows two photograps: the front and the rear view of the Flexbuff in the backends room. The unit takes some time to boot and it is rather noisy. All disks have a blue LED indicating that they are switched on.

Figure 3: *Front view of the Flexbuff unit at Yebes. 24 disks are mounted on the front.*

Figure 4: *Rear view of the Flexbuff unit at Yebes. 12 disks are mounted at the back.*

4 Connecting the Fila10G, the flexbuff and the correlator

Fig. 5 shows the connections between the DBBC2/Fila10G, the Flexbuff, the switches and the router.

The Fila10G has two optical fiber outputs, eth0 and eth1. eth0 is directly connected to one of the Flexbuff optical ports. The connection is easy to do since the LEDs above the Flexbuff port immediately light if the connection is correct and the two optical pairs are not swapped. eth1 is connected to one HP Procurve 2900 switch with four 10 Gb/s ports, two of them optical and the other two CX4. One of its CX4 ports is used to connect that switch to the HP Procurve 3500yl router that provides the connectivity to Internet. This CX4 connection allows to have the two switches stacked and have four 10 Gb/s optical input/outputs in total.

The usage of two switches allows to connect a second Flexbuff and use the DBBC2 eth1 output for e-VLBI or for the second Flexbuff eth0 once the routing is changed in the control

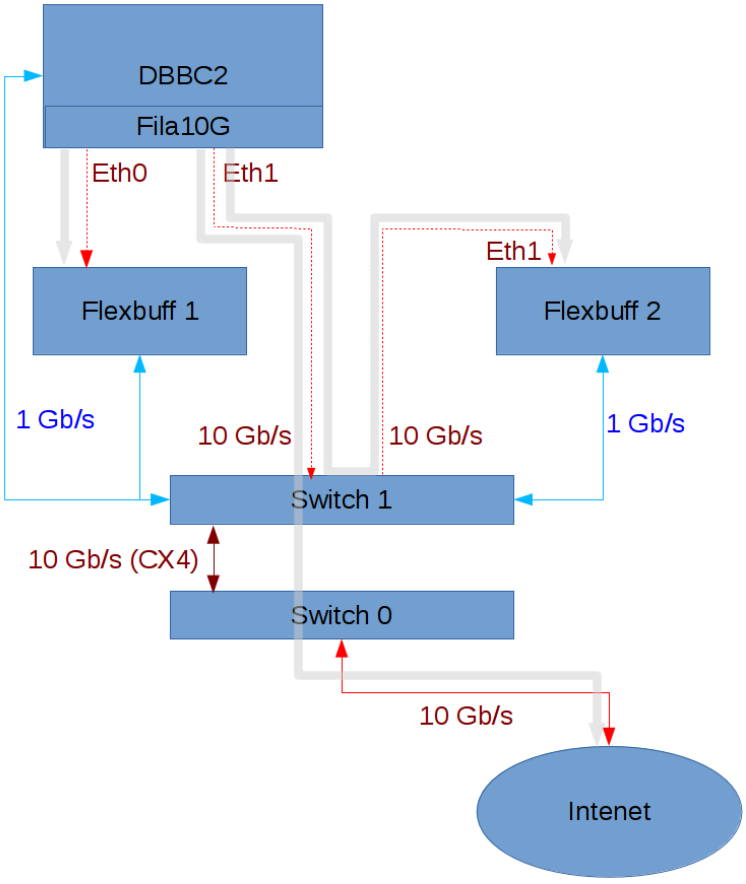

Figure 5: *Data flow for astronomy VLBI observations. DBBC2 eth1 may be connected either to the seconds Flexbuff or to JIVE via Internet*

configuration of the Fila10G. The second Flexbuff provides additional data storage and allows to use it for Radioastron and Geodetic observations. In this way EVN observations get stored in the first Flexbuff via DBBC2 eth0 and the rest of observations in the second Flexbuff via DBBC2 eth1.

5 Debian installation and configuration at the Flexbuff

The Flexbuff was delivered with a Linux distribution: CentOS (user:flytech - password:flytech), installed in the SATA-DOM. The file system is xfs and has logical virtual LVM partitions for /boot, /, /home and swap.

Disks can be listed with `lsblk`:

```
root@flexbuff:~# lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda           8:0    0  3,7T  0 disk
sdb           8:16   0  3,7T  0 disk
sdc           8:32   0  3,7T  0 disk
sdd           8:48   0  3,7T  0 disk
sde           8:64   0  3,7T  0 disk
sdf           8:80   0  3,7T  0 disk
sdg           8:96   0  3,7T  0 disk
sdh           8:112  0  3,7T  0 disk
sdi           8:128  0  3,7T  0 disk
sdj           8:144  0  3,7T  0 disk
sdk           8:160  0  3,7T  0 disk
sdl           8:176  0  3,7T  0 disk
sdm           8:192  0  3,7T  0 disk
sdn           8:208  0  3,7T  0 disk
sdo           8:224  0  3,7T  0 disk
sdp           8:240  0  3,7T  0 disk
sdq          65:0    0  3,7T  0 disk
sdr          65:16   0  3,7T  0 disk
sds          65:32   0  3,7T  0 disk
sdt          65:48   0  3,7T  0 disk
sdu          65:64   0 59,6G  0 disk
??sdu1       65:65   0  500M  0 part /boot
??sdu2       65:66   0 59,1G  0 part
  ??cl-root 254:0    0 35,7G  0 lvm  /
  ??cl-swap 254:1    0    6G  0 lvm  [SWAP]
  ??cl-home 254:2    0 17,5G  0 lvm  /home
sdv          65:80   0  3,7T  0 disk
sdw          65:96   0  3,7T  0 disk
sdx          65:112  0  3,7T  0 disk
sdy          65:128  0  3,7T  0 disk
sdz          65:144  0  3,7T  0 disk
sdaa         65:160  0  3,7T  0 disk
sdab         65:176  0  3,7T  0 disk
sdac         65:192  0  3,7T  0 disk
sdad         65:208  0  3,7T  0 disk
sdae         65:224  0  3,7T  0 disk
sdaf         65:240  0  3,7T  0 disk
sdag         66:0    0  3,7T  0 disk
sdah         66:16   0  3,7T  0 disk
sdai         66:32   0  3,7T  0 disk
sdaj         66:48   0  3,7T  0 disk
sdak         66:64   0  3,7T  0 disk
```

We replaced this system by Debian 8.3 using a standard installation from an USB stick, but keeping the same partitions and the same file system (XFS). The same users as the Mark5B were

created. For safety reasons we do not include neither the user nor the passwords. Instructions for installing Debian from a USB stick are available at the Debian web page and documentation.

As mentioned above this unit has 36 disks 4 TB, which provides a raw data storage of 144 TB that can be organized in different ways:

- RAID0: all disks form a block and there is no redundancy. No space is lost but if one disk fails all the block is useless.
- RAID5: all disks form a block and there is redundancy. Some space is lost, but this time two disks have to fail to render the block useless.
- Independent disks. The data access is not so fast but no space is lost and if one disk fails it does not affect the remaining disks.

The last option was chosen for the unit at the Observatorio de Yebes. The unit at the correlator used a different policy and the total space availability was reduced to 108 TB.

The file system for each disk was created using:

```
mkfs.xfs /dev/sda
```

```
meta-data=/dev/sda               isize=256    agcount=4, agsize=244188662 blks
         =                       sectsz=4096  attr=2, projid32bit=1
         =                       crc=0        finobt=0
data     =                       bsize=4096   blocks=976754646, imaxpct=5
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0 ftype=0
log      =internal log           bsize=4096   blocks=476930, version=2
         =                       sectsz=4096  sunit=1 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
```

and the mount point:

```
mkdir /mnt/disk0
mount -t xfs /dev/sda /mnt/disk0
```

The mounting point of the disks must be in file `/etc/fstab`, but before we need to determine the disk UUID:

```
blkid /dev/sda
```

```
/dev/sda: UUID="085fca55-8bdc-4fc4-ba48-21d8bae43abb" TYPE="xfs"
```

And the line in `/etc/fstab/` should read like:

```
UUID=085fca55-8bdc-4fc4-ba48-21d8bae43abb /mnt/disk0  xfs  defaults  0  2
```

The network configuration requires setting two IP addresses. The primary network interface is a 1 Gbs RJ45 and it is used for monitor and control and to transfer the data to the correlator via Internet after the observation is completed. The IP address could be obtained via DHCP from the Observatory server, but we have set it to a fixed address to load an MTU of 9000 bytes. In case the IP address is obtained via DHCP it is possible to force the usage of an MTU of 9000 bytes issuing the following command:

```
ifconfig eth0 mtu 9000 up
```

The second network interface is an optical 10 Gb/s port. It has to be configured belonging to a private VLAN since it is directly connected to the Fila10G. Its MTU also has to be set to 9000 bytes since VDIF frames from the Fila10G are 8192 bytes long. Being a second IP address there may be routing problems if it is not properly configured. To avoid them the network and the gateway should be left without configuration as shown below. Two other interfaces, one copper and one optical, are available but not configured for the time being.

`/etc/network/interfaces` file content:

```
# The primary network interface
auto eth0
iface eth0 inet static
        address 193.146.252.24
        netmask 255.255.255.128
        network 193.146.252.0
        broadcast 193.146.252.127
        gateway 193.146.252.1
        mtu 9000

# eth1 is SFP+ 10 Gb/s
#allow-hotplog eth1
auto eth1
iface eth1 inet static
        address 192.168.2.11
        netmask 255.255.255.128
        #network 192.146.252.0
        broadcast 192.168.2.127
        #gateway 192.168.2.1
        mtu 9000

# eth2 is copper

# eth3 is also SFP+ 10 Gb/s
```

6 Flexbuff: installing jive5ab and some utility tools

jive5ab does not exist as a Debian package for Debian Jessie (8.3) and sources must be compiled. Compilation requires installing previously some packages:

```
aptitude install make automake g++ gcc module-assistant
module-assistant prepare
```

The code should be downloaded from Harro Verkouter webpage in JIVE. It is advisable to create a `downloads` directory to store external software:

```
mkdir downloads
cd downloads
wget http://www.jive.nl/%7Everkout/evlbi/jive5ab-2.7.1.tar.gz
cd
tar xzvf downloads/jive5ab-2.7.1.tar.gz
```

Compilation should be done for a 64 bit architecture. That requires using flag `B2B=64`:

```
cd jive5ab-2.7.1/
make B2B=64
```

Installation should be done from "root" and the correct permissions should be applied to allow the program be run from "oper:"

```
su
make install B2B=64
chown oper.oper /opt/jive5ab-2.7.1-64bit
```

jive5ab provides some utilities that should be placed in the `bin` subdirectory. The `PATH` should include this directory:

```
export PATH=$PATH:/home/oper/bin
cd
cd jive5ab-2.7.1/
cp DirList.py /home/oper/bin/
cp m5copy /home/oper/bin/
cp SSErase.py  /home/oper/bin
cp StartJ5 /home/oper/bin
```

To make the `PATH` definitive, the following line should be included in `.bashrc`:

```
PATH=$PATH:/home/oper/bin
```

jive5ab should be started from script `StartJ5` which needs to be configured with the name of the station.

```
StartJ5
```

On start `jive5ab` looks for all mount sites in `/mnt/disk*` and uses all of them to record the data scattering it.

During recording jiveab records every scan in files 256 Mb size across different disks which get rotated. For example, the output of command `tree` from the `/mnt` location and after recording scan 1, would yield:

```
disk0
 n16m1_yd_no0001
  n16m1_yd_no0001.00000009
disk10
 n16m1_yd_no0001
  n16m1_yd_no0001.00000001
disk13
 n16m1_yd_no0001
  n16m1_yd_no0001.00000002
disk15
 n16m1_yd_no0001
  n16m1_yd_no0001.00000018
disk16
 n16m1_yd_no0001
  n16m1_yd_no0001.00000015
disk18
```

```
 n16m1_yd_no0001
  n16m1_yd_no0001.00000012
disk20
 n16m1_yd_no0001
  n16m1_yd_no0001.00000007
disk21
 n16m1_yd_no0001
  n16m1_yd_no0001.00000000
disk22
 n16m1_yd_no0001
  n16m1_yd_no0001.00000004
disk27
 n16m1_yd_no0001
  n16m1_yd_no0001.00000017
disk3
 n16m1_yd_no0001
  n16m1_yd_no0001.00000003
disk32
 n16m1_yd_no0001
  n16m1_yd_no0001.00000006
disk33
 n16m1_yd_no0001
  n16m1_yd_no0001.00000013
disk34
 n16m1_yd_no0001
  n16m1_yd_no0001.00000005
disk35
 n16m1_yd_no0001
  n16m1_yd_no0001.00000014
disk6
 n16m1_yd_no0001
  n16m1_yd_no0001.00000011
disk7
 n16m1_yd_no0001
  n16m1_yd_no0001.00000010
disk8
 n16m1_yd_no0001
  n16m1_yd_no0001.00000008
disk9
 n16m1_yd_no0001
  n16m1_yd_no0001.00000016
```

6.1 vsfsb tool

Locating and manipulating the files from a scattered system is quite unpractical and a utility called vsfsb should be downloded and installed. This is a FUSE file system which presents FlexBuff "vbs" recordings as single files for use with standard UNIX tools. Download and install the package

```
cd
cd downloads
wget http://www.jive.nl/~verkout/flexbuff/vbsfs-0.4.5.deb
dpkg --install
```

To make it work a mount point should be created and “activated”:

```
mkdir /mnt/virtualDisk
vbs_fs /mnt/diskpack
```

Since vbs_fs works as fuseMk5a (and other FUSE file systems) the procedure to unmount it is:

```
fusermount -u /mnt/diskpack
```

Alternatively if vbs_fs is run in the foreground (using option -f) a CTRL-C will unomunt the file system.

6.2 Listing and copying file within on the Flexbuff

Two more scripts are available for listing and removing files:

- `vbs_ls` lists the scans recorded. Multiple scan files across the disks are hidden and only scans are shown.
- `vbs_rm` removes a scan or several scans across the disks.

Care should be taken with `vbs_fs` which keeps a snapshot index of the disk state when it is started up. If files are removed without using vbs_rm, it will not know about it and trying to read the data will fail.

`vbs_ls` has been overhauled while this report was written and provides several options to list files and sort them according to different conditions. To get help:

```
vbs_ls --help
```

```
usage: vbs_ls [--help] [-6] [-v] [-F] [-h] [-l] [-r] [-R ROOTDIRS] [-S] [-t]
              [-T] [--version]
              [patterns [patterns ...]]

List FlexBuff recording details, much like ls(1), allowing for filtering
and/or sorting by size and/or time. In addition vbs_ls can accumulate totals
(sizes) per pattern, e.g. to compute total size per experiment. By default
vbs_ls searches the FlexBuff mountpoints for recordings. Using the '-6' flag
and/or the '-R' argument (see below) the search path may be altered. Shell-
style wildcards on the -R arguments are supported.

positional arguments:
  patterns     Only show recordings matching these pattern(s). Shell-style
               wildcards ('*?') are supported

optional arguments:
  --help       Show help for this program and exit succesfully
  -6           Look for FlexBuff recordings in Mark6 mountpoints
  -v           Look for FlexBuff recordings in FlexBuff mountpoints (default)
  -F           Show type of recording in short format using a trailing
               character: '/' = directory = FlexBuff recording, '' (nothing) =
               file = Mark6 recording, '+' if the recording was found in both
               formats and '?' for unknown type
  -h           Display file sizes in human readable format using base-1024
               unit prefixes kMGTPE
  -l           (The lowercase letter ``ell''. List in long format
  -r           Reverse the order of the sort to get reverse lexicographical
               order or the oldest entries first (or largest files last, if
               combined with sort by size)
  -R ROOTDIRS  Append directories matching the pattern to the vbs_ls search
               path. Shell-style wildcards ('*?') are supported. This option
               may be present multiple times to add multiple patterns
  -S           Sort recordings by size
  -t           Sort by time modified (most recently modified first) before
               sorting the operands by lexicographical order.
  -T           Compute size total(s) per pattern. Only useful if used with
               ``-l'' and one or more patterns
  --version    Print current version and exit succesfully
vbs_ls --help
```

A typical usage is listing all scans from a given experiment and sort them in reverse time order and use human readable units:

```
vbs_ls -lht n16m2_ys*
```

To get the grand total size used by the experiment:

```
vbs_ls -lhtT n16m2_ys*
```

```
Found 1 recordings in 569 chunks,  134.73G
drw-r--r-- oper      oper       134.73G Jun 03 15:00 n16m2_ys*
```

A python script called dirvbs.py was also developed locally, prior to the new `vbs_ls` version. To get information from the script type:

```
dirvbs.py -h
```

```
usage: dirvbs.py [-h] [-d {asc,desc}] [-t] [-s] [-f EXPERIMENT]

optional arguments:
  -h, --help            show this help message and exit
  -d {asc,desc}, --date {asc,desc}
                        Files order by date asc or desc [default asc]
  -t, --tsize           Shows the experiment total size. Experiment name is
                        required
  -s, --size            Shows the size per scan
  -f EXPERIMENT, --file EXPERIMENT
                        Experiment name
```

Both scripts `vbs_ls` and `dirvbs.py` can be used, but we recommend using `vbs_ls` since it will be maintained and provides extra functionality not available at `dirvbs.py`

Scans can be copied using m5copy using the correct format-prefix. For example:

```
m5copy vbs:///n16m1_yd_no0001 mk5://mark5b-40m.oan.es/
```

would copy the scan from the Flexbuff into the activated diskpack in the specified Mark5B.

A grapical tool from JIVE, jive5ab copy manager, is also available for transfers between tow hosts. This tool is used for internal copies, like between the Flexbuff and the PCVSIB host. Fig 6 shows a snapshot of the application. To run it:

```
cd jive5ab_copy_manager
./jcm.py
```

Its usage is simple: select the mode, the IP address and click “reload” on the left panel. The same operation should be done on the right one. To copy, one should select the files on the left panel, click on the destination directory on the right panel and click with the right mouse button over the files to transfer. A new window with a list of copy commands will appear. Each line corresponds to an individual file transfer. It is possible to change the command on the window prior to clicking “Go”.

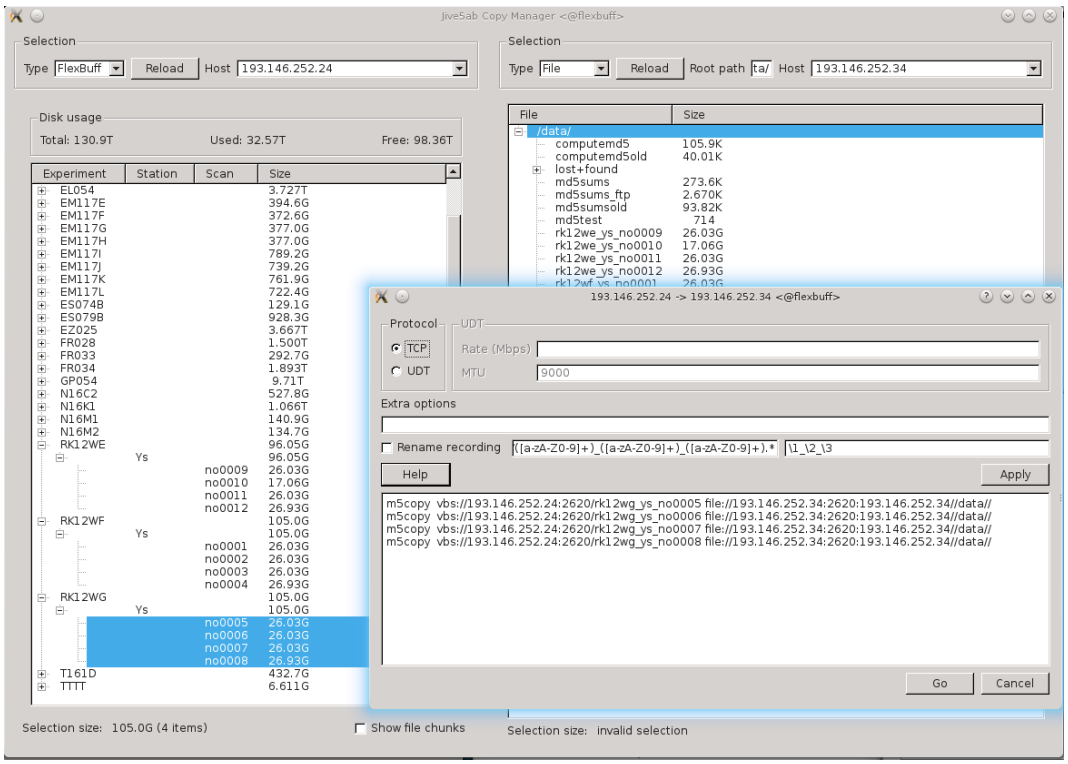

Figure 6: *Graphical jive5ab copy manager. Snapshot taken about to copy from the Flexbuff to a host (PCVSIB)*

6.3 Transfers to Bonn and Washington correlators

Although this is a report devoted to astronomy VLBI observations we also include for completeness information about transfers to the geodetic correlators. Since June 22, 2016, geo experiments are written in VDIF. L. Barbas has written a python script that manages transfers using m5copy to Bonn and Washington correlators. The goal is to offer a simple procedure to the operators and update the web page with transfers towards the correlators: `http://www3.mpifr-bonn.mpg.de/cgi-bin/showtransfers.cgi`. An example of usage is shown below. It was used to transfer EUR142 files to Bonn correlator. The script started a jive5ab instance on port 9209 at Bonn correlator and transfered data on port 9230. The port used for the local jive5ab was 2620.

```
./run_m5copy.py
Nombre del experimento(SIN indicar estacion ys, yj): eur142

Tipo de observacion en Yebes
 1.- 40m (ys)
 2.- 13m - RAEGE (yj)
[1-2]: 1

Lugar correlacion --> direcciones IP (nombre disco)
Bonn     --> 195.37.231.39 (io2)
             195.37.231.34 (io3)
             195.37.231.36 (io10)
             195.37.231.37 (io11)
             195.37.231.38 (io13)
             195.37.231.41 (io14)
Waco     --> 198.116.24.178 (san01)
Direccion IP del correlador: 195.37.231.39

******* Relacion de directorios destino *******
PRINCIPAL:
195.37.231.39  -> /data2
195.37.231.34  -> /data3
195.37.231.36  -> /data10
195.37.231.37  -> /data11
195.37.231.38  -> /data13
195.37.231.41  -> /data14
198.116.24.178 -> /data1, /data2,..., /data9
SECUNDARIO:
 Experimentos astro --> /astro/yebes/nombre_experimento/ o /astro/raegyeb/nombre_experimento/
 Experimentos r1 --> /r1/yebes/nombre_experimento/ o /r1/raegyeb/nombre_experimento/
 Experimentos euro --> /euro/yebes/nombre_experimento/ o /euro/raegyeb/nombre_experimento/
 Experimentos t2 --> /t2/yebes/nombre_experimento/ o /t2/raegyeb/nombre_experimento/
 Experimentos r4 --> /evlbi0#/nombre_experimento_ys/ o /evlbi0#/nombre_experimento_yj/

**********************************************************************
*****       Comprobar que existe el directorio de destino        *****
*****       si no crealo antes de continuar con la transferencia *****
**********************************************************************

Directorio de destino [/data2/euro/yebes/eur142/]:

Velocidad de transferencia [200]: 100


Indica el conjunto de ficheros para transmitir [eur142*]:

[list of files here]

.........................................................
Enviando el fichero de start a Bonn...
```

```
Comando: touch /tmp/20160624111833_eur142_ys_bonn_100_9230_start

Comando: ncftpput ftp.mpifr-Bonn.mpg.de /incoming/geodesy/transfers/ /tmp/20160624111833_eur142_ys_bonn_100_92
/tmp/20160624111833_eur142_ys_bonn_100_9230_start:       0.00 B     0.00 B/s
/tmp/20160624111833_eur142_ys_bonn_100_9230_start:       0.00 B     0.00 B/s

Comando: rm -rf /tmp/20160624111833_eur142_ys_bonn_100_9230_start

.........................................................
Arrancando jive5ab en el correlador bonn
Comando: ssh evlbi@195.37.231.39 " ps afx | grep 'jive5ab -m 3 -b -p 9209' "
13777 ?        Ss     0:00          \_ bash -c  ps afx | grep 'jive5ab -m 3 -b -p 9209'
13779 ?        S      0:00              \_ grep jive5ab -m 3 -b -p 9209

Comando: ssh evlbi@195.37.231.39 'jive5ab -m 3 -b -p 9209'
```

For transfers to WACO the philosophy is slightly different; previous to starting the script *run_m5copy.py* jive5ab is started on port 46224 using script *StartJ5_3* and the target remote directory at the correlator needs to be created.

6.4 DifX mark5access

Plotting the spectra of the VLBI channels is an excellent tool for diagnosing problems. Mark5access is a collection of tools from DifX which performs the autocorrelation of all recorded channels and provides additional utilities like, zero baseline correlation, extraction of phase cal phase, analysis of data statistics or checking the integrity of data.

Traditionally we have used DifX 2.1 in the Mark5Bs but the usage of 4 GBps PFB modes which use 32 channels requires version 2.4.1 and further. The installation is simple once we download the package using SVN.

```
aptitude install automake autoconf pkg-config libtool libfftw3-dev
aclocal
libtoolize --copy --force
autoconf
autoheader
automake -a -c
./configure
make
su
make install
exit
export LD_LIBRARY_PATH=/usr/local/lib
```

The following apps get installed in `/usr/local/bin`. We only comment those used at Yebes:

- `directory2filelist`
- `fixmark5b`
- `m5bstate` Shows the bit statistics

- `m5bsum`
- `m5d`
- `m5fold`
- `m5timeseries`
- `m5tsys`
- `m5test` Checks the frames in a scan and its integrity
- `m5time` Decodes the start time of the recorded data
- `m5slice`
- `m5findformats` Tries different MarkIV/VLBA/Mark5B formats. VDIF not included.
- `test5b`
- `test_mark5_stream`
- `test_unpacker`
- `m5pcal` Extracts the amplitude and phase of the phase cal tones. Example where the tones are spaced each 5 MHz and start at 0.4 MHz:

  ```
  m5pcal pct174_yj_174-1216_2.vdif VDIF_8192-2048-16-2 0.4 pcal_data -i 5 -n 2
  ```

- `m5spec` Performs the autocorrelation of the recorded channels. Example to create an ascii file *spec.out* with the autocorrelation for all channels: The spectrum has 1024 points and stacks 4000 frames.

  ```
  m5spec pct174_yj_174-1216_2.vdif VDIF_8192-2048-16-2 4000 1024 spec.out -dbh
  ```

- `m5fb`
- `zerocorr` Does a zero correlation, that is data from two backends connected to the same antenna.

7 Configuring the FS

The usage of a Flexbuff requires tuning some control files, the station procedure library and at least one script:

- *skedf.ctl*. File that is used by DRUDG when the procedure and snap files are being generated. If there is no backend equipment specified in the schedule file DRUDG will use the ones specified in the line that begins with the keyword *equipment*. If the line *equipment_override* is specified then the equipment in the control file becomes the default regardless of what is in the schedule. Whenever Flexbuff is going to be used, the following line should be uncommented (FS versions before 9.11.13):
  `equipment DBBC/Fila10g Flexbuff none.`
  That will set a DBBC with a Fila10g as the DAS (Data Adcquisition System) and a Flexbuff as the recorder. No second recorder is available. For FS versions above 9.11.13 and DDC mode the line should be:
  `equipment DBBC_DDC/FILA10G Flexbuff none.`
  and for PFB mode:
  `equipment DBBC_PFB/FILA10G Flexbuff none.`

- *equip.ctl*. Receiver configuration that Field System uses on startup. Under VLBI equipment select `dbbc/fila10g` (FS versions prior to 9.11.13) as rack and flexbuff as recorder 1. As a sanity check verify that the firmware version for the DBBC is the one that you really want to use and is already running on the DBBC. For FS versions 9.11.13 and later select `dbbc_ddc/fila10g` for DDC mode or `dbbc_pfb/fila10g` for PFB mode.

- *midob*. It is a procedure that can be found at *station.prc* library. The following lines should be commented:

```
mk5b_mode
!+1s
mk5=dot?
```

  The last generation recorders (flexbuff and mark6) do not use these commands any more.

- *checkmk5*. Another procedure that plots the spectrum of every channel so the user can easily check if the observation is running properly. The following lines should be commented:

```
disk_stats
mk5=get_stats?
```

  Line asking for recorder mode
  `mk5b_mode`
  should be replaced by
  `mk5=mode?`
  or
  `mk5c_mode?`

- *checkdata.py*. This Python script is invoked by the `midob` procedure. It retrieves some relevant information from the logfile (data rate, channel distribution) and plots the spectra of each channel for the subscan time interval extracted in *systest.m5b* by the previous procedure. To be used with a flexbuff the corresponding IP address must be assigned to the *rec_id* variable.

8 FiLa10g configuration

The fila10g configuration is basically done within *fila10g_cfg*. This file basically activates VDIF encoding and sets the MAC and IP addresses of the two DBBC2 ethernet ports, the gateway and the mac address and IP address of the destination.

```
define  fila10g_cfg   00000000000x
"general setup
fila10g=arp off
fila10g=vdif_enc on
"
"customize ips, ports, gateways, macs, nms, and destinations,
"for your station, configure appropriate ethxs
"
"the fila10g does not automatically map ipv4 addresses to ethernet
" mac addresses (the arp table).see "arp off" above, although that has
" other (positive) consequences as well.
"the arp table must be programmed manually using 'tengbarp' below.
"parameters are the last number from the destination's dotted-quad
" ipv4 address ('11' here, from '192.168.2.11')
"and xx:xx:xx:xx:xx:xx is the mac address of the flexbuff interface
" with this ipv4 address.
"
"the destinaton port must agree with the flexbuff, which uses 2630 by default.
" if you change this you will need to program the flexbuff with
" "mk5=net_port = y;" where "y" is the port you used. you could put
" that command in "initi" and "sched_initi" to try to make it reliable
" but it would better to just not change the port.
"
"eth0 follows
fila10g=tengbcfg eth0 ip=192.168.2.21 gateway=192.168.2.1
fila10g=tengbcfg eth0 mac=00:60:dd:45:66:69
fila10g=tengbcfg eth0 nm=27
"flexbuff ip address eth1
fila10g=destination 0 192.168.2.11:2630
fila10g=tengbarp eth0 11 0c:c4:7a:bd:4a:80
"
"fila10g=destination 0 none
"
"eth1 follows
fila10g=tengbcfg eth1 ip=193.146.252.21 gateway=193.146.252.1
fila10g=tengbcfg eth1 mac=00:60:dd:45:66:79
fila10g=tengbcfg eth1 nm=27
"jive for evlbi
"fila10g=destination 1 192.42.120.86:2630
"fila10g=tengbarp eth1 86 00:1c:2e:92:26:40
"use the entry below to stop the flow
fila10g=destination 1 none
"flexbuff ip address eth1
"fila10g=destination 0 192.168.2.11:2630
"fila10g=tengbarp eth1 11 0c:c4:7a:bd:4a:80
enddef
```

The following command

```
fila10g_mode=,0xffffffff,,4.000
```

sets the payload to 8192 bytes and uses singled thread multichanel VDIF. Command

```
fila10g_mode
```

starts the data flow, internally commanding:

```
fila10g=start vdif
```

The data flow can be stopped issuing:

```
fila10g=stop
```

9 e-VLBI observations. A proxy

Traditionally e-VLBI observations have been done from the stations but the data flow was always controlled by JIVE which could access remotely the station Mark5Bs. The Mark5B recorders have a public IP address and JIVE can control jivea5b running there. The maximum data rate towards the network at the Mark5Bs is 1 Gb/s due to hardware limitations. A recording rate of 2 Gbps required a direct connection to the DBBC from JIVE. Some tests have been done within the EVN involving the TOG to test this mode. After some tests it has been demonstrated that multi-channel single thread transmission is possible with MTUs of 8192 bytes and single channel multi thread transmission is only possible with small transfers: 1500 bytes.

Most of the DBBCs at the stations use private IP addresses and this prevents JIVE from directly commanding them. Besides the DBBC server software only accepts one connection at a time and multiple connections are disallowed. The solution devised by JIVE and developed by Harro Verkouter is to run a proxy that allows more than one connection in a host reachable from JIVE. The FS computer talks to that proxy server and JIVE correlator also talks to the proxy during e-VLBI observations. Fig. 7 shows a diagram with the data flow during eVLBI.

To get the last version and install it:

```
wget http://www.jive.eu/~verkout/evlbi/dbbc_proxy
chmod a+x dbbc_proxy
```

The proxy is transparent for the FS. We only need to change `dbbad.ctl` with the new IP address. At Yebes the proxy runs in the Harrobox, a dedicated server reused from the 4 Gbps tests performed several years ago. The proxy is started running:

```
./dbbc_proxy -l 0.0.0.0:2620 -r 193.146.252.26:4000 -f -d
```

which means that it listens on port 2620 locally on all interfaces and forwards traffic to the real DBBC at IP address 193.146.252.26 on port 4000. This proxy server logs information on the screen useful for debugging purposes (-d) and does not daemonize itself (-f). Options can be looked up by typing:

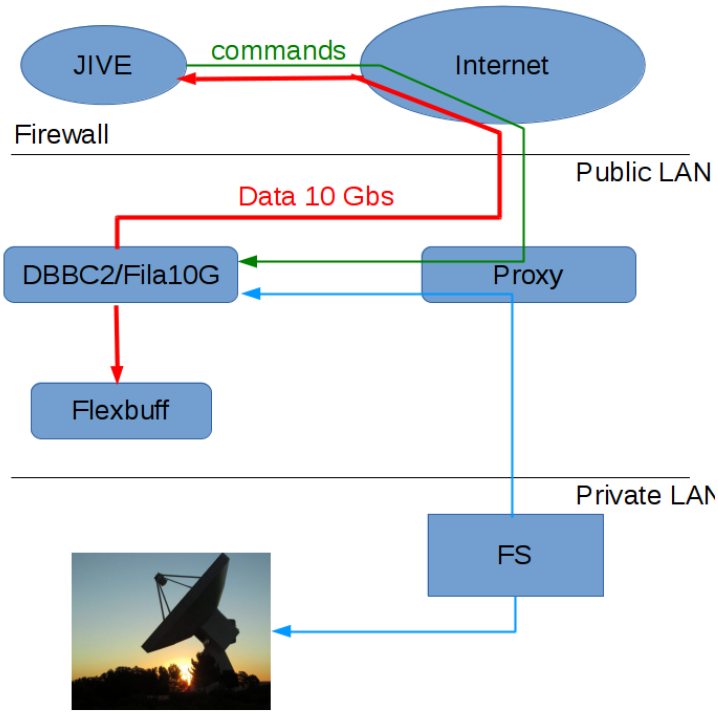

Figure 7: *Data flow for e-VLBI observations*

```
./dbbc_proxy -h
```

In order to perform eVLBI observations, *fila10g_cfg* should not set the destination IP address and leave it as "none". Each EVN station is associated to a different IP address at JIVE correlator but to avoid overflowing it with unexpected data, it is the correlator that will set the destination IP address remotely when ready. Therefore the configuration at Yebes, assuming that eVLBI is done via eth1 should look like below:

```
"eth1 follows
fila10g=tengbcfg eth1 ip=193.146.252.21 gateway=193.146.252.1
fila10g=tengbcfg eth1 mac=00:60:dd:45:66:79
fila10g=tengbcfg eth1 nm=27
"jive for evlbi
"fila10g=destination 1 192.42.120.86:2630
"fila10g=tengbarp eth1 86 00:1c:2e:92:26:40
"use the entry below to stop the flow
fila10g=destination 1 none
```

10 Observing with the Flexbuff

Observations with the Flexbuff require some simple operations and checks which may change with newer FS observations. By the time of this report version FS 9.11.13 is installed and options are as we describe below:

- Prior to starting a schedule a `fila10g_cfg` should be run. That will load the Fila10G network configuration.
- Immediately after, `fmset` should be executed with option `s`. Synchronization will take some seconds. In some ocassions if the synchronization is lost a firmware reload may be required.
- Monitor that there is data traffic into the Flexbuff ethernet port. This is done using a Linux network tool which may be installed as a Debian package:

  ```
  bmon
  ```

  bmon is a tool that displays the TX and RX rates of all ethernet connections. Moving up and down with the arrows through the port names shows a simple histogram as a function of time. If there is no data flowing into the Flexbuff it may be possible that the Fila10G has not received the command to start yielding VDIF data. The FS command that starts this is `fila10g_mode`
- If the data does not flow into the Flexbuff it can be forced typing:

  ```
  fila10g=start vdif
  ```

In order to check the spectra of the channels and the bit statistics the schedule has a procedure called `checkmk5` which in Yebes has been customized as follows:

```
scan_check
mk5c_mode
jive5ab=mode?
"write out some bytes from beginning of the last scan
!+1s
mk5=scan_set=::+20000000
!+1s
mk5=disk2file=/home/oper/data/systest.vdif:::w
!+3s
sy=exec /usr2/oper/bin/checkdata.py `lognm` /home/oper/data/systest.vdif &
```

`checkdata.py` is a Python script derived from J. Quick and U. Bach's one used for Mark5 recorders. The changes are listed below:

```
for line in fileContent:
        #
        # Find mode
        #
        # jive5ab/!mode? 0 : VDIF_8000-128-16-2 : VDIF : 32 : 4000000.000 : 8000 ;
        if "/jive5ab/!mode?" in line:
                newline=line.split(":")
                vdifformat = newline[3].strip()

vdiflist = vdifformat.split('-')
aux = int(vdiflist[0].split('_')[1])
mbps = int(vdiflist[1])
bbcs = int(vdiflist[2])
bits = int(vdiflist[3])
BW = mbps/(2*bbcs*bits)
```

10.1 Extracting data from the correlator

Once the observation is completed the correlator gets the data using m5copy and stores it in a station Flexbuff at the correlator. In order to separate the logs of jive5ab operations from the jive5ab transfers we decided to start a second jive5ab instance listening on port 2621. This is achieved with command:

```
StartJ5_2
```

The logs are stored in directory `jive5ab_transfer.logs`. The correlator needs access to the oper account from JIVE and to reach port 2621 for issuing commands.

The copy application started from the correlator will read file `oper/.JIVE.allow` to see if it is allowed to do a data transfer and at which data rate. A file without restrictions looks like this:

```
2014-07-01 00:00:00 2042-07-30 00:00:00 0
```

where the first two columns are the start date and time, and the third and fourth column are the stop date and time and the last column the data rate in Mbps where zero means there is no limitations. The file can have multiple lines and in case of time overlap on the ranges the most restrictive data rate will be used.

The name of transferred experiments are added to file `oper/.JIVE.transferred`. For example:

```
EM117F
EM117E
ES079B
```

11 Tuning the Flexbuff to minimize losses

B. Eldering warned us that experiment GP054A showed 0.1% packet losses between the DBBC2 and the Flexbuff. This number is very small but we identified the necessity to tune the system to minimize the packet loss during experiments recording (transmission between the DBBC2 and the Flexbuff).

The tuning is summarized by **Harro Verkouter** in this document: *http://www.jive.eu/ verkout/flexbuff/flexbuf.recording.txt* and it was performed in Yebes together with Harro using a remote connection with *tmux*. Below we explain some preliminary steps acomplished to configure a shell script that will be executed every time the host is rebooted.

- Install package `ethtool`.

  ```
  aptitude install ethtool
  ```

- Disable hyperthreading. This can be done either in the BIOS or using shell commands (Verkouter 2016). The first option is the best one and it required a flexbuff reboot. To check if hyperthreading is active type:

  ```
  grep -e '\(processor\|core id\)' /proc/cpuinfo | vim -
  ```

  The answer at Yebes was like

  ```
  processor       : 0
  core id         : 0
  processor       : 1
  core id         : 1
  processor       : 2
  core id         : 2
  processor       : 3
  core id         : 3
  processor       : 4
  core id         : 4
  processor       : 5
  core id         : 5
  processor       : 6
  core id         : 0
  processor       : 7
  core id         : 1
  .....
  ```

  up to 24 processors were listed.

  To check how many CPUs are present in the Flexbuff and how many COREs per CPU we type:

  ```
  cat /proc/cpuinfo | grep 'physical id'
  cat /proc/cpuinfo | grep 'core id'
  ```

  The output, manually sorted, is like this:

```
physical id     : 0  core id         : 0
physical id     : 0  core id         : 1
physical id     : 0  core id         : 2
physical id     : 0  core id         : 3
physical id     : 0  core id         : 4
physical id     : 0  core id         : 5
physical id     : 1  core id         : 0
physical id     : 1  core id         : 1
physical id     : 1  core id         : 2
physical id     : 1  core id         : 3
physical id     : 1  core id         : 4
physical id     : 1  core id         : 5
physical id     : 0  core id         : 0
physical id     : 0  core id         : 1
physical id     : 0  core id         : 2
physical id     : 0  core id         : 3
physical id     : 0  core id         : 4
physical id     : 0  core id         : 5
physical id     : 1  core id         : 0
physical id     : 1  core id         : 1
physical id     : 1  core id         : 2
physical id     : 1  core id         : 3
physical id     : 1  core id         : 4
physical id     : 1  core id         : 5
```

According to the previous output the Flexbuff has two CPUs with 6 cores each. Since there appear 24 processors, hyperthreading is active. Disabling it show be done at the BIOS.

- Manage the network memory and UDP. This is done from *tune_flexbuff.sh* shell script.
- Dedicate one physical CPU to manage ethernet interrupts. This is the most important tuning. File */proc/irq/<irg>/smp_affinity* lists the (hexadecimal) processor mask which CPUs (logical) may handle the interrupt <irq> and */proc/interrupts* is the irq table. To extract all IRQs that eth1 (the connection port from the DBBC2) can generate we type:

```
for irq in `grep eth1-TxRx /proc/interrupts | awk -F: '{print $1;}'`; do cat
```

- Copy *tune_flexbuff.sh* to /tt /home/oper/bin/
- Modify */etc/rc.local* to include the following line:

```
/home/oper/bin/tune_flexbuff.sh
```

File *tune_flexbuff.sh* contains these commands (comments inside the script have been removed in this report):

```
/sbin/sysctl net.core.netdev_max_backlog=8192
/sbin/sysctl net.core.rmem_max=201326592
/sbin/sysctl net.core.wmem_max=201326592
/sbin/sysctl net.ipv4.udp_mem="16384 32768 49152"
for irq in `grep eth1-TxRx /proc/interrupts | awk -F: '{print $1;}'`; do
    echo 1 > /proc/irq/${irq}/smp_affinity;
done
/sbin/ethtool -C eth1 rx-usecs 5
/sbin/ethtool -G eth1 rx 4096
```

12 Real observations: N16C1 and N16M1 and EVN session 2016-2

The first tests with the Flexbuff were N16C1 and N16M1 during EVN session 1 and the whole EVN session 2016-2 was the first EVN session fully recorded on the Flexbuff at Yebes. Fringe plots from the correlator are available at JIVE web pages. For example N16M2 is shown in Fig. 8. Statistics for all Yebes recorded experiments are correct. Fig. 9 shows a snapshot of the bin statistics for scan 8, N16M2. The distribution of the bits is as expected.

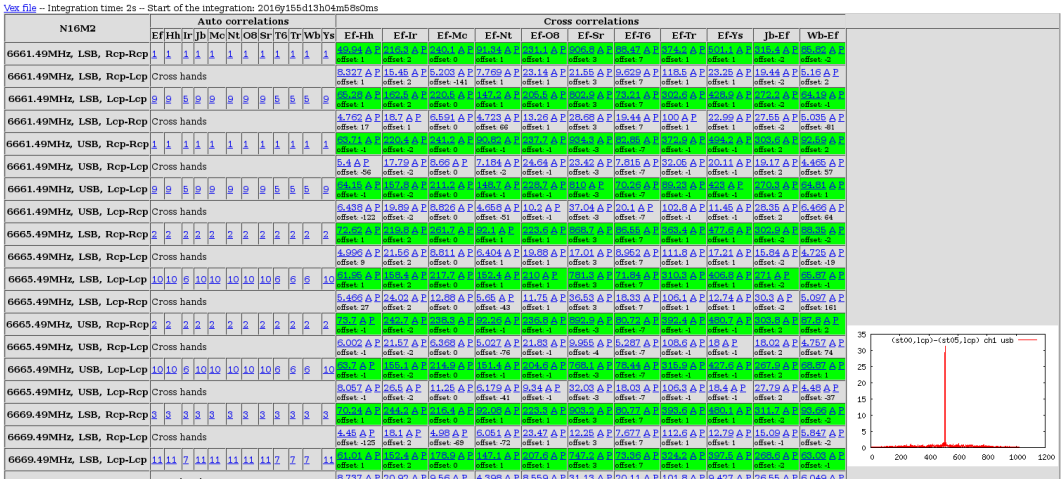

Figure 8: *N16M2 scan 8. Fringes from some channels and all participating stations.*

Ys	- -	- +	+ -	+ +	invalid	avg sign bit	avg mag bit
6661.49MHz, LSB, Rcp	17.08%	31.35%	32.16%	18.63%	0.7794%	0.5119	0.5037
6661.49MHz, LSB, Lcp	17.01%	31.12%	32.06%	19.04%	0.7794%	0.515	0.5055
6661.49MHz, USB, Rcp	18.33%	32.07%	31.5%	17.32%	0.7794%	0.492	0.4978
6661.49MHz, USB, Lcp	18.97%	31.76%	30.95%	17.54%	0.7794%	0.4887	0.4968
6665.49MHz, LSB, Rcp	17.11%	31.35%	32.12%	18.63%	0.7794%	0.5116	0.5038
6665.49MHz, LSB, Lcp	17.01%	31.11%	32.12%	18.98%	0.7794%	0.5151	0.5049
6665.49MHz, USB, Rcp	18.31%	32.06%	31.54%	17.31%	0.7794%	0.4924	0.4976
6665.49MHz, USB, Lcp	18.88%	31.72%	31.1%	17.52%	0.7794%	0.49	0.4963
6669.49MHz, LSB, Rcp	17.16%	31.3%	32.11%	18.65%	0.7794%	0.5116	0.5034
6669.49MHz, LSB, Lcp	17.1%	31.01%	32.09%	19.02%	0.7794%	0.5151	0.5042
6669.49MHz, USB, Rcp	18.3%	32.07%	31.49%	17.35%	0.7794%	0.4923	0.4981
6669.49MHz, USB, Lcp	18.63%	31.95%	31.34%	17.31%	0.7794%	0.4903	0.4964
6673.49MHz, LSB, Rcp	17.08%	31.4%	32.14%	18.6%	0.7794%	0.5114	0.5039
6673.49MHz, LSB, Lcp	17.07%	31.05%	32.1%	19%	0.7794%	0.515	0.5045
6673.49MHz, USB, Rcp	18.34%	32.02%	31.5%	17.37%	0.7794%	0.4925	0.4977
6673.49MHz, USB, Lcp	18.7%	31.89%	31.29%	17.35%	0.7794%	0.4902	0.4963

Figure 9: *Statistics for N16M2 scan 8. The two bit distribution is correct and as expected.*

References

[1] A. Whitney, 1 , M. Kettenis 2 , C. Phillips 3 and M. Sekido. http://www.vlbi.org/vdif/docs/Whitney-VDIF_paper_for_IVS_GM_Tasmania.pdf

[2] H. Verkouter. http://www.jive.eu/ verkout/flexbuff/flexbuf.recording.txt