eVLBI at 4 Gb/s with the 40 m radiotelescope

P. de Vicente, R. Bolaño, L. Barbas, J. González

Informe Técnico IT-OAN/CDT 2012-22

Revision history

Version	Date	Author	Updates
1.0	15-05-2012	P. de Vicente	First version

Contents

1 Introduction: goal of the test

Nexpress work package 5 aims for cloud correlation with high-speed (>1 Gbps) recording and transmitting capabilities. One of the most important steps in this work package is to achieve the first VLBI observation at 4 Gb/s in real time. The planned observation in June 20nd 2012 tried to send the data in real time through the network and record at the same time at 4 Gb/s on a diskpack for a later correlation. 4 Gb/s is an important rate towards which the EVN aims in the near future since it is required to observe 512 MHz with both polarizations:

$$\underset{\text{Band}}{512} \times \underset{\text{Nyquist}}{2} \times \underset{\text{Polarizations}}{2} \times \underset{\text{Bits}}{2} = 4096 \text{ Gb/s}$$

Radiotelescopes that took part in the observation were the 100 m telescope at Bonn, the 20 m telescope in Onsala and the 40 m radiotelescope in Yebes. This observation was prepared and coordinated by JIVE in close collaboration with the personel in each observatory. Up to 6 teleconferences took place since some months before the experiment. Two tests were run before the June 20nd session to test different technical parts involved in the test.

In this report we summarize the preparation works performed in Yebes and the results of the observations. Compilation of all the relevant information on a single document for future usage will be useful.

2 Required instrumentation

2.1 Frontends at the telescopes and the LO issue

A frontend with a minimum 512 MHz instantaneous bandwidth and two polarizations is required. Three ranges were available: C band, X band and 22 GHz. The frequency of observation was chosen to be a common one for Yebes, Onsala and Effelsberg to avoid any problem with the local oscillator setup and taking into account that 500 MHz is the current simultaneous bandwidth on at least Yebes and Onsala. 22 GHz and C band were disregarded because Onsala only has one polarization and an IF 512 MHz wide at these frequencies.

The chosen frequency range was X band. Since Yebes does not have a standard setup, Onsala modified their local oscillator to have a common band with Yebes. Effelsberg has frequency flexibility since their frequency range is between 7900 and 9000 MHz. The standard setup for X band requires a local oscillator frequency of 8080 MHz and a bandwidth between 8180 and 8580 (the IF band is in the first Nyquist zone). Yebes uses a LO oscillator at a fixed frequency of 7650 MHz which allows a 500 MHz frequency band between 8150 and 8650 MHz (second Nyquist zone). All telescopes set their LO to that frequency which was tested on May 25th, in between the EVN session at X band.

A second test was performed at 22 GHz on June 5th, also previous to the 22 GHz EVN session. The limitation with the LO frequency came from 2 telescopes: Effelsberg whose LO has to be a multiple of 4 MHz and Yebes whose LO has to be a multiple of 7 MHz once substracted 12800 MHz. Hence the common setting has to be every 28 MHz (minimum common multiple of both). The usual LO is 21500 MHz at Effelsberg and 21550 MHz at Yebes. In between both

values, 21536 MHz complies with both conditions. Onsala can set any LO value, however their IF works in the 0 to 500 MHz band (first Nyquist zone). This is not a problem when using the DDC mode in the DBBC, but it is a problem in the polyphase mode (PFB) as we will see later. In the PFB the 0 to 500 MHz uses Nyquist zone 1 and the upper side band, while 500-1000 MHz uses the second Nyquist zone and the lower side band

The final test in June required all three stations to use the same LO (7650 MHz) since 4 Gb/s is only achieved using the PFB mode at the DBBC.

2.2 Network connnection: 4 Gb/s

An internet connection which allows a flow rate of 4096 Mb/s was required at all stations. The connection at Onsala and Effelsberg was provided by LOFAR. The Low Frequency Array has antennas in Onsala and Effelsberg which are connected to JIVE via a high speed connection (10 Gb/s lines). These lines were exceptionally used for the test. Onsala required a special connection between Sunet and Nordunet.

At Yebes, 4 Gb/s was achieved using a 10 Gb/s connection between Yebes node (OANY) and the spanish NREN (RedIris) which is part of the new high speed reseach network, NOVA. This connection was implemented with a dark fiber. The 10 Gb/s connection is operative since May 7th, 2012.

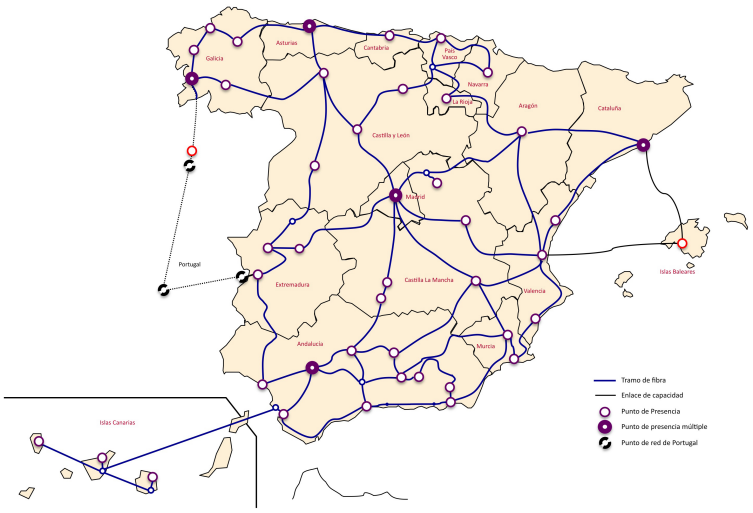

Figure 1: Topological map of RedIris Nova as of March 2012

RedIris is connected to GEANT via two independent 10 Gb/s connections and a 2.5 Gb/s backup circuit. Three 10 Gb/s lambda services with GEANT are also available. Fig. 2 shows the topology of the european network as of March 2012.

Between May 7th and June 7th, Yebes was connected to GEANT using the primary 10 Gb/s circuit, which showed small 8 Gb/s peaks. On June 7th, 2012 the traffic from Yebes was moved to the secondary 10 Gb/s link, to avoid the overload of the primary link.

The RedIris NOVA access point at Yebes consists on 4 racks: three of them can be seen in Fig. 4 (left panel). The right most rack contains batteries, a rectifier and a DC power supply for

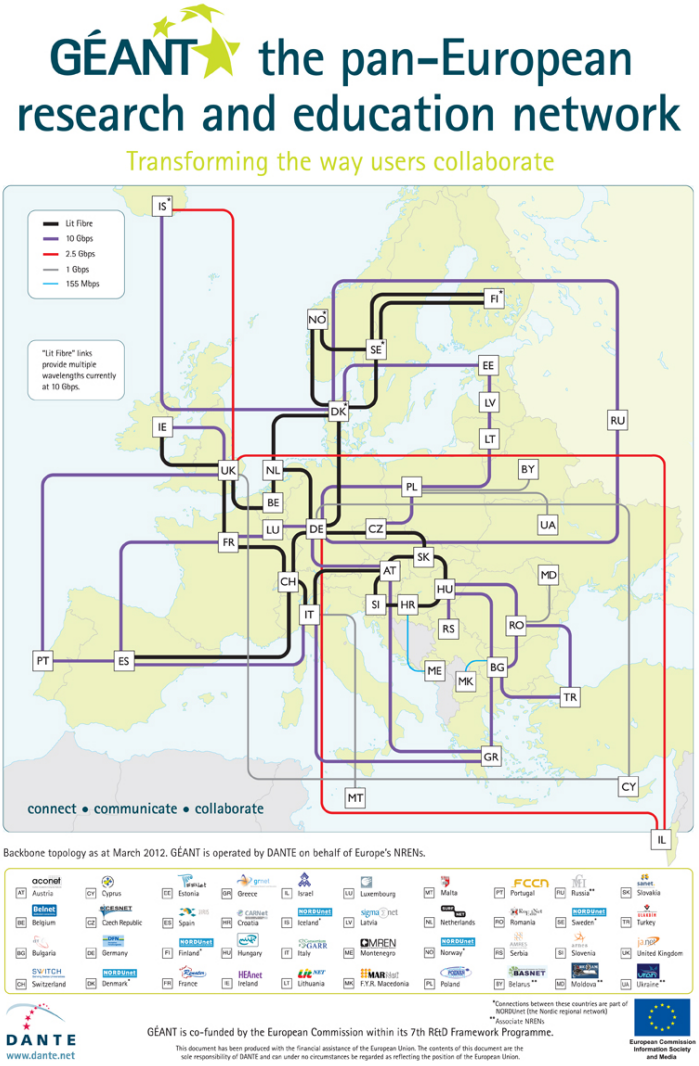

Figure 2: Connections to GEANT

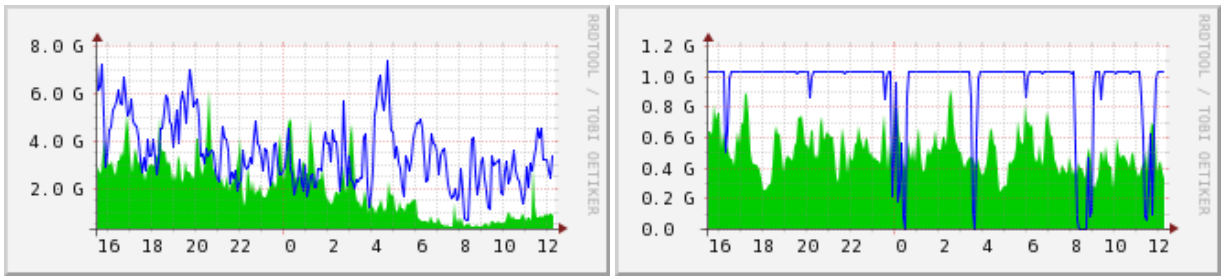

Figure 3: *Left: 24 hour traffic in the primary RedIris-GEANT link at the time this report was written. Right: same information for the secondary link. The sustained 1 Gb/s comes from an eVLBI session taking place at Yebes. The secondary link has less occupancy*

the electronic equipment. The central rack contains electronic equipment from Alcatel-Lucent. The left most rack contains a tray with optical fiber connections from outside at the bottom and a patch panel drawer at the top. The patch panel contains 72 SC-APC optical connectors in 12 rows of 6 connectors each. Fig. 4 shows the patch panel from above with all connectors except two, unused.

Figure 4: Left: racks at the RedIris NOVA access point. Right: patch panel at the left most rack

We laid an OM3 double patch 25 m long between the patch panel and the router: the patch panel end used SC connectors and the router end, LC connectors. The router, an HP 3500yl, is installed in a communications rack in the control room of the 40 m radiotelescope. It was upgraded in December 2011 with a 10 GB E board with 2 SFP+ ports and 2 CX4 ports. The board is inserted in the back of the switch and required a firmware upgrade to work correctly. The upgrade was a two step procedure and required downloading the firmware from HP onto an USB memory stick which was inserted in the switch.

The firmware was upgraded from K12.16 to K13.68 and later to K15.06. Firmware files were downloaded from
`https://h10145.www1.hp.com/downloads/SoftwareReleases.aspx?ProductNumber=J9310A`.

This address can be reached from `www.procurve.com` after selecting ”quick links” and ”software updates” and entering the switch model: 3500-24G-PoE+ yl (J9310A). To upgrade we connected via ssh to the switch and used the command line interface. An option allows to choose the USB pen as source of the firmware file. Each firmware upgrade required a reboot of the switch. After two resets the firmware version was checked:

```
jucar# show flash
Image               Size (bytes) Date     Version
------------------ ------------ -------- --------------------
Primary Image     :     14844423 10/09/11 K.15.06.0008
Secondary Image   :      6610294 05/29/07 K.12.16

Boot ROM Version : K.15.19
Default Boot     : Primary
```

The graphical web interface for the new firmware differs from the old one. Fig. 5 displays a section of the page showing the new 10 Gb/s available ports: A1 to A4.

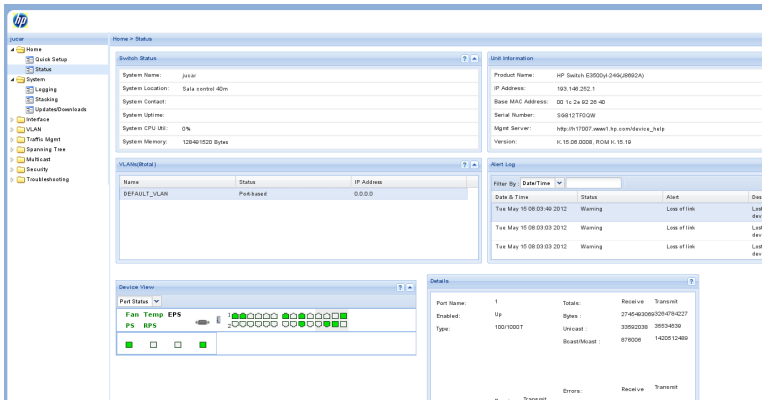

Figure 5: Section of the graphical web interface from the 3500yl ProCurve. The four 10 Gb/s ports can be seen at the low left side of this snapshot (only A1 and A4 are in green)

2.3 VLBI equipment: DBBC

The conversion between the analog intermediate frequency (IF) to digital data (VSI output) was performed by the DBBC with a sampling rate of 64 Mb/s per channel. The analog BBCs plus the VSI4 module, usually used for all VLBI observations at Yebes 40 m dish, is limited to a sample rate of 32 Mb/s per channel and therefore was disregarded. The dBBC has two modes of operation: DDC (Digital Down Converter) and PFB (Polyphase Filter Bank). In the first one, the frequency for individual channels can be set independently. This mode is similar to the mode used at the old VLBA BBCs backend. The second mode consists on setting 16 contiguous channels in frequency per polarization, each channel with a bandwidth of 32 MHz. Each CORE can handle 512 MHz from one polarization, and therefore only two COREs are required to generate 4 Gb/s.

The dBBC accepts IF signals in the 0-500 MHz or in the 500-1000 MHz IF range. Any of these options is selected by software. In Yebes we usually use the 500-1000 MHz IF range. Only under exceptional conditions we can use the lower band IF range. In the latter case we need to use the FFTS preprocessor to lower the band and two attenuators to achieve the correct IF input power.

2.4 VLBI equipment: Mark5B+, Mark5C and Fila10G

The observation was devised to record and transmit data simmultaneously in real time. Recording on disks is performed by Mark5 devices. The latest versions, Mark5B, Mark5B+ and Mark5C are fed with data along 32 channels in parallel which comply with the VSI-H (VLBI Standard Interface for Hardware) standard.

This test required a Mark5C using both disk banks. A Mark5C is a Mark5B+ without an I/O card (VSI-H inputs) and a different Mezzazine card that allows to route the data from a CX4 input port to both disk banks. The Mark5C runs under Linux and requires a control program called DRS which manages its operation. This program has the same goal as "dimino" in Mark5Bs. Mark5Cs can generate data in Mark5B compatible format and in VDIF format.

Mark 5B+s can record at 2048 Mb/s on one disk bank if the VSI-H source sends the data with a rate of 64 Mb/s. This rate is not available with a VSI4 sampler module, but it is with a Digital BBC. In order to generate two 2048 Mb/s flows, two Mark5B+ recorders and two VSI connectors are required. However Mark5Cs as Mark5B+s are limited to send a maximum rate of 1024 Mb/s through their NIC while recording in a disk pack. Hence it is not possible to record and send data at 4096 Mb/s simmultaneously. In any case the usage of 2 Mark5Bs was disregarded by JIVE.

This limitation can be overcome if the DBBC is connected to a Fila 10G board or a Fila10G standalone module. The Fila 10G has 4 VSI ports and two 10 GbE optical output ports. The VSI ports can be configured as all inputs or 2 inputs and 2 outputs, and two 10 GbE optical output ports. The most common setup uses 2 input VSI ports and 2 VSI outputs, which are a copy of the former. The output is also sent to both 10 GbE optical ports. Figs. 6 and 7 show an opened Fila 10G standalone module from above and the rear cover from outside respectively. The standalone module is a 2U device, with its own power supply, which can be placed in a rack close to the DBBC.

The final connection using one Mark5C is depicted in Fig. 8. The dBBC output was injected into a Fila 10G board using two VSI ports. The Fila 10G board redirected the data flow to its two optical ports. One port sent the data to Internet and the correlator through a router. The other port was connected to a Mark5C for recording on two diskpacks. However the Mark5C only accepts a CX4 10 Gb/s input which uses copper. Since the Fila 10G does not have a CX4 output, a converter is required between the 10 Gb optical port and the CX4. This may be achieved either by a glass to copper device (Glapper, as in Effelsberg) or using a switch with CX4 and 10 GbE ports properly configured (explained later).

The Fila10G can produce Mark5B or VDIF compatible data from the VSI-H input flow. During the test the more reliable and tested format was used: Mark5B.

In the future the Fila10G will be able to split the data in chunks which allow to send these subbands with lower speeds to different correlators, or to a distributed correlator. Since this

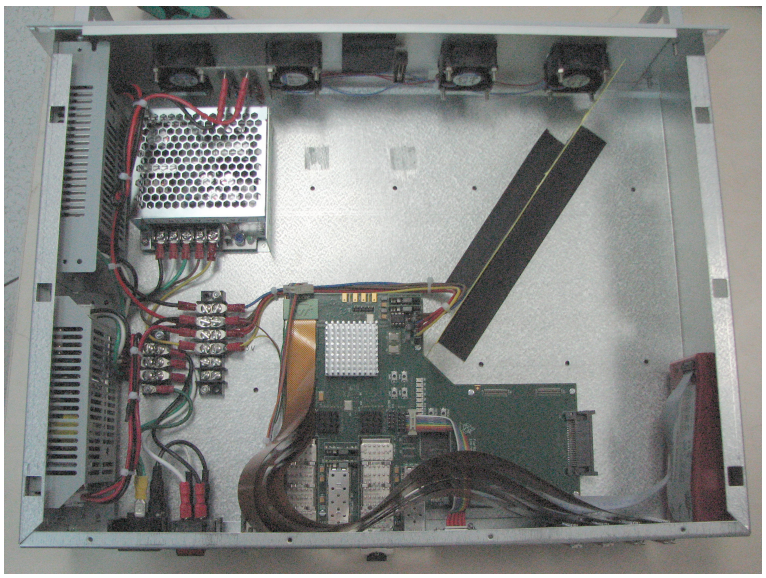

Figure 6: Fila10G view from above.

Figure 7: Rear side of the Fila10G. Available ports are: 2 XFPs, 1 serial port, 1 ethernet port, 1 USB port and 4 VSI input/output connectors.

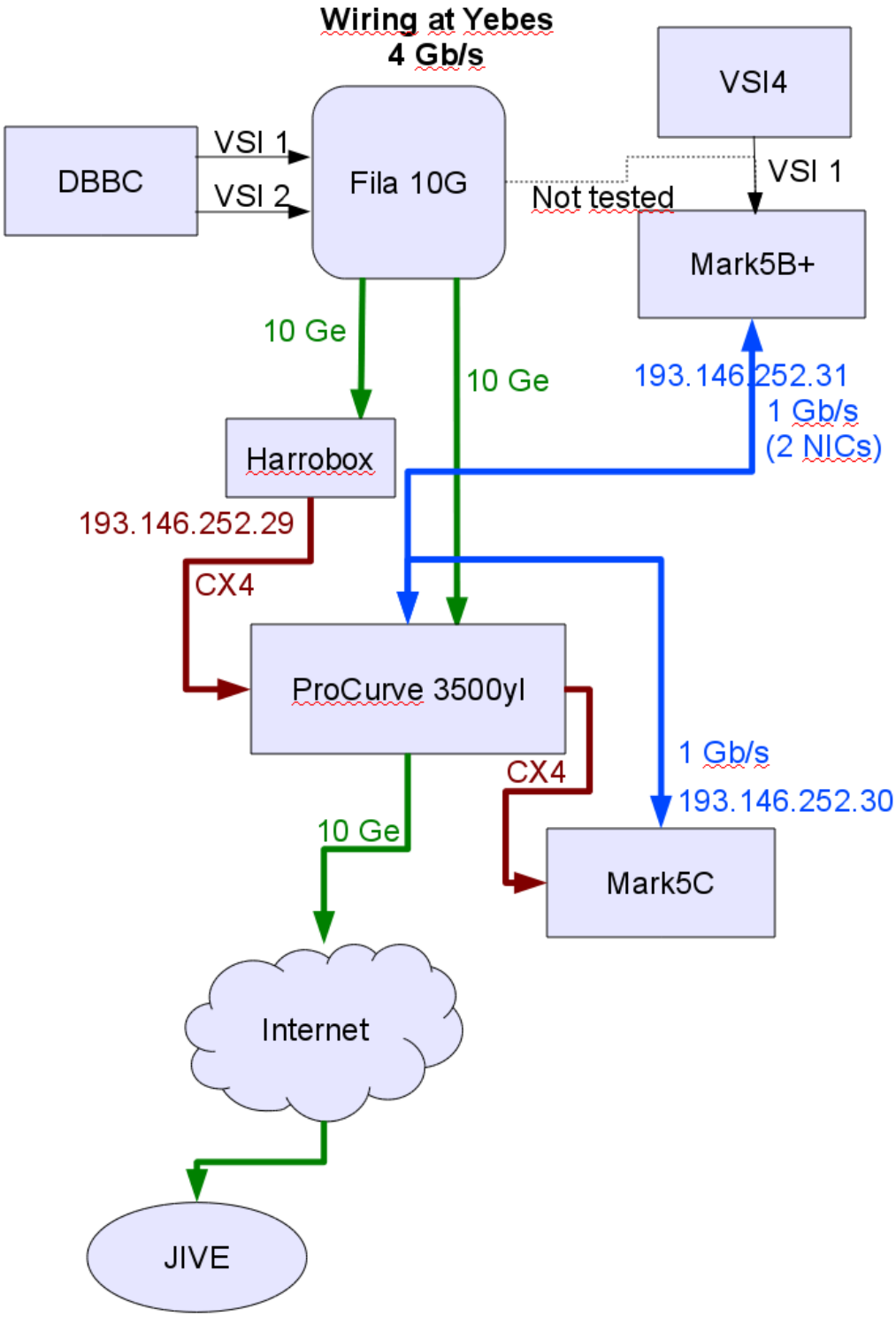

Figure 8: Wiring at Yebes during the 4 Gb/s test.

feature was not available by the time of the test, another equipment was required.

2.5 Harrobox and the VDIF data format

One of the requirements for this experiment was the so called “cornerturning”. Cornerturning means splitting the data flow in frames with the same frequency band and sending frames with that same frequency to a distributed correlator, where the computing power is shared among CPUs which can handle data rates up to 1 Gb/s.

This feature is foreseen in the VDIF format specification, which allows different data threads, and it is graphically represented in Fig. 9. Mark5B format is not ready for this feature.

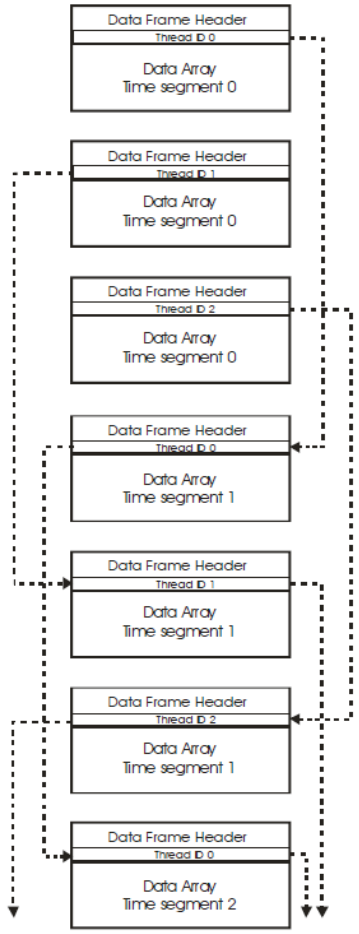

Figure 9: Data threads in VDIF.

As mentioned above, cornerturning should be performed by the Fila 10G but by the time of the experiment the firmware was not ready for this possibility yet. In the meantime Harro Verkouter from JIVE developed software to perform this function in a dedicated PC, which was named as “Harrobox”. The Harrobox is a 1 U PC equipped with a very powerful CPU (eight core Intel Xeon CPUs) and at least 6 ethernet ports: four 1 Gb/s copper NICs, one 10 GbE port and one 10 Gb CX4 port. The PC runs Linux Debian Squeezy. The eth0 port is used for

management and should get its IP address from a DHCP server. This PC obtains the data flow from the CX4 port, does cornerturning and sends the data through the 10 GbE port which is connected to a 10 GbE optical fiber towards Internet.

3 Deployment of equipment in Yebes

All the VLBI equipment were installed in a rack in the backends room. Fig. 10 shows the physical distribution in the rack and Fig. 8 the schematics. In order to avoid a power overload, equipment was plugged in sockets with different current limit switches. Later, and some days before the observation, the Mark5C had to be moved close to the switch (explained later).

Since the router/switch is in the control room, two CX 4 cables and two optical fiber patches 10 m length were necessary. The optical fiber patches, OM3 and with LC connectors in both ends, were supplied by JIVE. The equipment may have different transceivers on both ends: XFP in the Fila 10G and SFP+ in the HP 3500yl switch. The 2 XFP transceivers in the Fila 10G, together with the optical patches, CX4 cables, the Harrobox and the Mark 5C were supplied by JIVE. The FILA 10G was borrowed from Metsahovi.

Figure 10: Rack with all VLBI equipment used in the 4 Gb/s tests. From up to down: Field System computer, Mark5C, Mark5B, Common monitor, Harrobox, Fila10G and DBBC.

3.1 DBBC Cabling

The DBBC was cabled as follows:

- 1 PPS in. The 1 PPS signal was injected using a BNC cable connected to the Station Sync output at the front of the VLBA DAR 32 MHz module.
- 10 MHz in. The DBBC requires a 10 MHz reference signal which is obtained from the Quartzlock distributor.
- RF In 1 (IF A to D). These connectors were used to feed the DBBC with the IF outputs. The regular connection is:
  - IFA is connected to the RCP FFTS preprocessor output
  - IFB is connected to the VLBA IF B output
  - IFC is connected the LCP FFTS preprocessor output
  - IFD is connected to the VLBA D output

  However for the final test only two COREs and therefore only IFs A (RCP from X band) and B (LCP from X band) were used. COREs 3 and 4 were removed from the stack. The input for both RF connectors came from the RCP and LCP FFTS preprocessor outputs.
- 1 RS232 output. There was a connection between the DBBC and the Fila10G, through the serial port, which allowed the former to upload the firmware on the latter and set the Fila 10G configuration. Once the Fila10G was configured the cable was unplugged.

The PFB mode of the DBBC only uses the last 2 COREs in the stack. If there are 4 boards these will be number 3 and 4, which are connected to IFS C and D. In case there are only two COREs, one has to make sure to which IF the CORE is connected. This is done with aselector on the board. Each CORE can generate a 512 MHz bandwidth signal. Each signal, sampled with 2 bits, generates a flow of 2 Gb/s on one VSI cable. Two output VSI cables were used to send the signals to the Fila10G standalone module.

3.2 Fila10G cabling

The Fila10G was equipped with two XFP+ optical transceivers used to send the data at 10 Gb/s to the Mark5C and the Harrobox using OMG3 optical fiber patches. The Fila10G also has two control ports: one RS232 and one ethernet port. Both ports are internally connected to the same place, but they cannot be used simmultaneously. To enable one or the other it is necessary to make a connection inside the box. The Fila10G at Yebes was using the RS232 connection and the ethernet connector was disabled. As already mentioned in the previous subsection, the RS232 is used to upload the firmware and set the required options, and after, during the observation, to start and stop the output flow of data and to synchronize the equipment. This connection was done using a Lantronix with an IP address. Every time the recording required to be stopped a command was sent to Fila10G (via the Lantronix) that stopped the output flow fo data for 10 seconds. After the 10 seconds the flow of data was enabled again. This ackward behaviour was required by the Mark5C as explained below.

The firmware was uploaded from the DBBC with this command:

```
%XILINX%\bin\nt\impact -batch FILA10GSA_conf.cmd
```

The command file `FILA10GSA_conf.cmd` content is include below:

```
setMode -bs
setCable -port usb21 -esn 00001510273301 -baud 6000000
Identify
IdentifyMPM
assignFile -p 2 -file c:/DBBC_CONF/FilesDBBC/fila10g_v2_0.bit
erase -p 2
program -p 2
quit
```

The signals with data were injected in VSI ports 1 and 3. Since the cables were very short, the Fila10G standalone module was stacked on top of the DBBC. Fig. 11 shows a picture of the cable mess behind both the DBBC and the Fila10G.

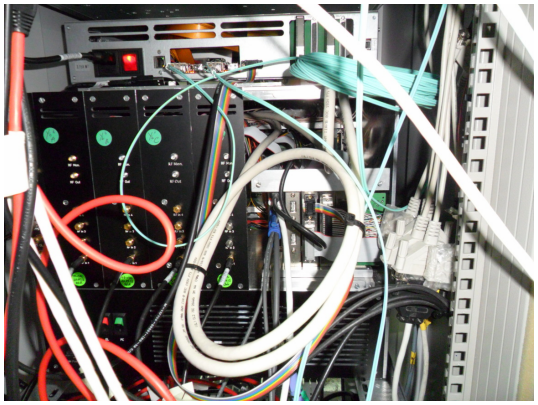

Figure 11: Back of the rack with VLBI equipment. The 2 VSI cables and the 2 optical patches are visible.

3.3 Mark5C cabling

As we have explained above, the Mark5C only accepts a CX4 cable to inject the data. At Yebes we used the HP Procurve switch (3500yl) as interface between the Fila10G and the Mark5C. We discovered that long CX4 cables (10 m) prevented the Mark5C working correctly. Some users have reported that long cables (up to 5 m) can be used if the ground connection is properly done. We moved the Mark5C close to the switch and used a 0,5 m cable that worked successfully.

4 Fila10G, Mark5C and switch setup

The Mark5C has two serious drawbacks which required a special configuration to transfer the data from the DBBC:

- Its 10 Gb CX4 port does not have ethernet firmware and hence cannot be assigned neither a MAC address nor an IP address. This means that the CX4 cable has to be directly

connected from the acquisition terminal (DBBC in this case) to the recorder (using a Glapper) or devise a special way to do this, which involves using a switch.

- The Mark5C is very slow responding to commands when the flow of rate is high (4 Gb/s). In order to stop a recording it is necessary first to stop the flow of data and later issue a stop command. Hence the flow of data has to be stopped at the source of data: the Fila10G.

In order to make a connection between the Mark5C and the Fila10G a switch with an optical 10Gb E port and a CX4 was used. Since the Mark5C does not have an IP address, the switch cannot route the data. The switch was configured with a exclusive VLAN that connected both 10 Gb/s ports and assigned a (even) static mac address to the CX4 port that connected with the Mark5C. The two port VLAN prevented flooding all switch ports making the switch become unresponsive when the 4 Gb/s flow was started. The configuration is included below:

```
vlan 200
   name "VLBI"
   untagged A2,A4
   ip address 10.88.1.1 255.255.255.0
   jumbo
   exit

static-mac 002244-6688aa vlan 200 interface A4
```

The fake mac address has to be chosen with care since not any will do. Apparently, valid mac addresses need the first byte to begin by an even number.

The Fila10G was configured according to the following command file:

```
@echo off

set F10GCOM=COM1
echo Assuming FILA10G is on serial port %F10GCOM%

echo ---- Changing FPGA clock: 80MHz,OSC,TVG=4Gbps -----
echo regwrite regbank0 1 x40000003 | sendstr %F10GCOM%
echo reset | sendstr %F10GCOM%

echo ---- Configuring station name to "Ap"
rem http://www.asciitable.com/ and note fila10g hex format is xFF not 0xFF
echo regwrite regbank0 10 x4170 | sendstr %F10GCOM%


rem echo -- Switching to VSI1/VSI2(2Gbps) or VSI1-2(4Gbp)
echo inputselect vsi1-2 | sendstr %F10GCOM%
echo reset | sendstr %F10GCOM%

echo ---- Setting system clock
timesyncFILA10G %F10GCOM%
```

```
echo ---- Setting system clock
vdif_timesyncFILA10G %F10GCOM%

echo arp off | sendstr %F10GCOM%

echo ---- Source MAC, IP, Port for eth0
echo tengbcfg eth0 mac=ba:dc:af:e4:be:e1 | sendstr %F10GCOM%
echo tengbcfg eth0 ip=10.88.1.120 gateway=10.88.1.1 nm=27 | sendstr %F10GCOM%
echo tengbcfg eth0 port=46227 | sendstr %F10GCOM%

echo ---- Source MAC, IP, Port for eth1
echo tengbcfg eth1 mac=ba:dc:af:e4:be:e2 | sendstr %F10GCOM%
echo tengbcfg eth1 ip=10.88.1.121 gateway=10.88.1.1 nm=27 | sendstr %F10GCOM%
echo tengbcfg eth1 port=46227 | sendstr %F10GCOM%

echo -- Setting destination IP address 10.88.1.125 for eth0
echo regwrite regbank0 11 0x0A58017D | sendstr %F10GCOM%

echo -- Setting destination IP address 10.88.1.124 for eth1
echo regwrite regbank0 6  0x0A58017C | sendstr %F10GCOM%

echo -- Adding fake MAC for Mark5C (through switch port A4) in eth0 ARP
echo tengbarp eth0 125 00:22:44:66:88:aa | sendstr %F10GCOM%

echo -- Adding MAC for harrobox in eth1 ARP
echo tengbarp eth1 124 90:e2:ba:0c:34:0c | sendstr %F10GCOM%

echo -- dataformat select 5b or vdif
echo dataselect 5b | sendstr %F10GCOM%

rem look at the 'VDIF size-maxnumber table', (here set to 8000+32)
echo -- VDIF data packet size-1 in 8-bytes unit as 0x05...
echo regwrite regbank0 2 0x050003e7
echo -- VDIF data packet number-1 in 1sec
echo regwrite regbank0 3 0x0400f9ff

echo ---- Check from the 10G info that multicast MAC
echo tengbinfo eth0 | sendstr %F10GCOM%
echo tengbinfo eth1 | sendstr %F10GCOM%
```

The previous script assigns a mac address and an IP address to the Fila10G optical ports 1 and 2, eth0 and eth1 respectively. It also assigns a couple of mac addresses and IP addresses to the destination of the data: the Mark5C and the Harrobox. The mac address (00:22:44:66:88:aa) for the Fila10G is fake and should match the one set in the switch to allow the correct route of the data. Its IP address is also fake: 10.88.1.125.

During observations, two kind of commands were required to control the Fila10G: stopping the flow of data and time syncing. Both of them were issued thorugh the serial port and hence required the Lantronix as interface. The first command was automatic and triggered by the schedule, whereas the second one required human intervention.

The Fila10G flow was stopped for 10 seconds to allow the Mark5C stop the recording and restarted 10 seconds later. The script, written by Jan Wagner at Effelsberg, was run from the FS computer. The script can be found in Appendix 1 of this report. Below we include the procedure ran from the FS to activate the script:

```
define  stopstream    12172135900x
" stop fila10g data stream for 10 sec
" Please insert correct IP address of your FILA10G
sy=exec /usr2/oper/bin/stopfila10g.py 192.168.0.123 10002 &
```

Syncronization was achieved by sending the following command to the Lantronix:

```
timesync <sec> <mjd>
```

where `<sec>` is seconds of day and `<mjd>` the modified julian day. The time was chosen and issued in coordination with the correlator which delivered instructions to all the stations.

The Mark5C was controlled by the FS, version 9.10.3. Since this version does not have support for this equipment, the recorder was deselected in the control file `/usr2/control/equip.ctl` and set to `none`. However the Mark5C IP address was kept at `/usr2/control/mk5ad.ctl` to allow its control and monitor using FS low level commands (those which begin my `mk5=`).

5 Tests prior to the observation: speed, LOs and PFB mode

5.1 May 25th test

Two tests were planned for this session:

- fr013a: its goal was to test the LO at 7650 MHz at all stations and the Yebes DBBC. The maximum data rate was 512 Mb/s and the DBBC was used in DDC mode.
- fr013b: its goal was to test the Polyphase filter bank mode in the DBBC. The maximum data rate was 2048 Mb/s

A wiring sketch is in Fig. 12. A Mark5B was used as recorder to avoid introducing more unknowns (a Mark5C).

5.2 fr013a

Experiment fr013a used the DDC mode: different BBC channels each with a different frequency. Channels were grouped in pairs for each polarization. The DDC mode was started launching the server program (double clicking on the icon named `control_net.exe`)

Prior to starting the schedule we established the connection with the DBBC from the Field System:

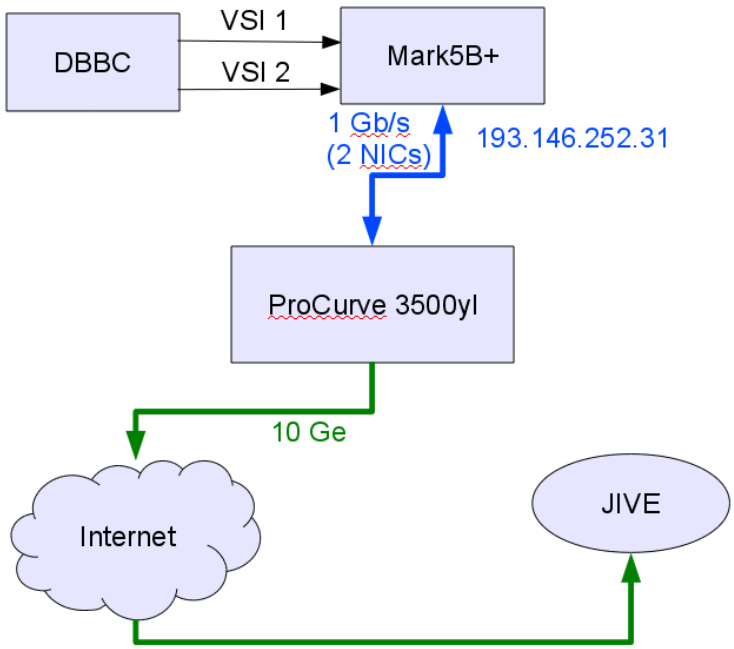

Figure 12: Hardware setup for the May 25th test. No Fila10G was used

```
> dbbcinit
```

and then loaded the schedule.

Since the Field System does not have support for the DBBC, the prc files need to be modified using a perl script written by Dave Graham. Below we include it. The most important issue to take into account is to set correctly the input filter for the DBBC: 2 is for 0-512 MHz and 1 for 512-1024 MHz (command `dbbc=dbbcifa=1,agc,1`. Version used in Yebes:

```
define  exper_initi   12146074358x
proc_library
sched_initi
mk5=DTS_id?
mk5=OS_rev?
mk5=SS_rev?
mk5=status?
dbbc=pps_sync
!+1s
mk5=mode=ext:0xffffffff:1
!+1s
mk5=tvr=off
!+1s
mk5=clock_set=32:ext:32
!+1s
mk5=1pps_source=vsi
!+1s
```

```
mk5=dot?
!+1s
mk5=dot_set=:force
!+1s
mk5=dot?
enddef
define  setup01       12146074442x
pcalon
tpicd=stop
pcald=stop
mk5b_mode=ext,0xFFFFFFFF,2
mk5b_mode
dbbc=dbbcform=astro
dbbc=dbbcform
bbc018
ifd01
bank_check
tpicd=no,2000
tpicd
enddef
define  bbc018        12146074443x
dbbc=dbbc01=741.49,a,8.000,8.000
dbbc=dbbc05=741.49,c,8.000,8.000
dbbc=dbbc02=757.49,a,8.000,8.000
dbbc=dbbc06=757.49,c,8.000,8.000
dbbc=dbbc03=773.49,a,8.000,8.000
dbbc=dbbc07=773.49,c,8.000,8.000
dbbc=dbbc04=789.49,a,8.000,8.000
dbbc=dbbc08=789.49,c,8.000,8.000
!+1s
dbbc=dbbcifa=1,agc,1
!+1s
dbbc=dbbcifc=1,agc,1
enddef
define  ifd01         12146074445x
ifdab=0,0,nor,nor
ifdcd=0,0,nor,nor
lo=
newlo=0
lo=loa,7650.00,usb,rcp,1.000
newlo=a
lo=loc,7650.00,usb,lcp,1.000
newlo=c
enddef
```

This schedule did not generate fringes with Ys in real time due to a failure in the correlator. As a consequence the second part of the observation was not performed. However, after the

experiment, data fringes were found in all three baselines: Yebes, Effelsberg and Onsala in two different scans. (see Fig. 13. This success demonstrated for the first time that the DBBC, in DDC mode, worked correctly.

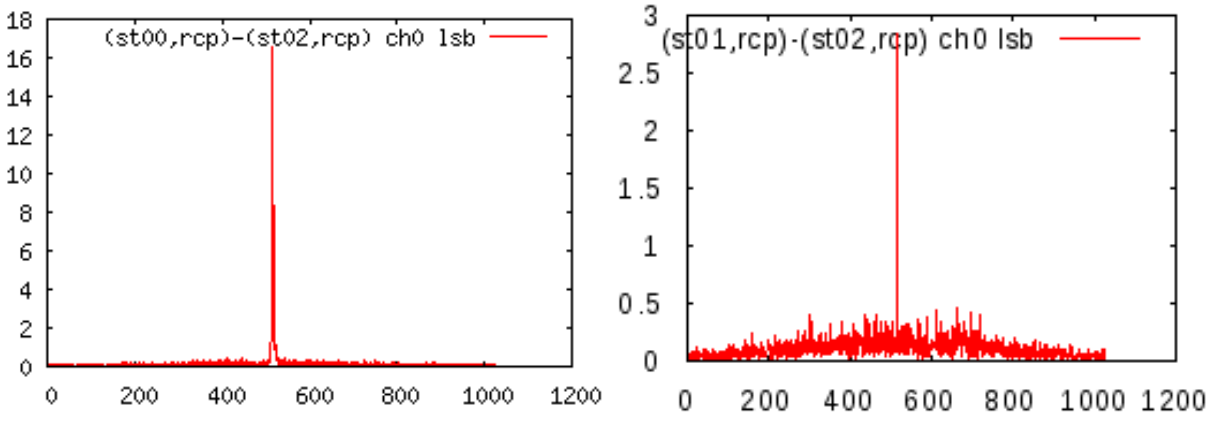

Figure 13: fr013a. Left: Effelsberg - Yebes fringe on channel 1. Right: Onsala - Yebes fringe on channel 1

5.3 June 5th test

An observation at 21900 MHz using the 0 to 500 MHz (Nyquist zone 1) was scheduled on June 5th. Three different experiments were programmed:

- fr014a. This was a standard NME mode in which the BBC was working in the DDC mode.
- fr014b. This was a 2 Gb/s experiment with the DBBC working in PFB mode.
- fr014c. This was a 4 Gb/s experiment with the DBBC working in PFB mode. For this part telescopes should switch to Mark5C.

The first test was successful. We used the FFTS preprocessor to convert from the second Nyquist zone (500-1000 MHz band) to the first Nyquist zone (0-500 MHz band). Attenuators were used in both polarizations to get the correct power at the DBBC input.

The second test was unsuccessful. Yebes DBBC did not work on PFB mode. It was discovered that two of the COREs were not working as expected. Since the second test was unsuccessful the third part was also discarded.

The PFB mode is loaded at the DBBC stopping before *DBBC_Control_net.exe* and starting *DBBC_Control_poly_net_v12.exe*. The Field System procedure generated by drudg for this experiment was modified by hand. Three important parameters are set manually:

- `mk5=mode=ext:0xffffffff:1`
- `mk5=clock_set=64:ext:32` To generate 2048 Mb/s the Mark5B+ needs to know that the VSI input flow rate is 64 Mb/s

- `dbbc=dbbcifdc=1,agc,1` and `dbbc=dbbcifd=1,agc,1`

```
define  exper_initi   00000000000x
proc_library
sched_initi
mk5=DTS_id?
mk5=OS_rev?
mk5=SS_rev?
mk5=status?
dbbc=pps_sync
!+1s
mk5=mode=ext:0xffffffff:1
!+1s
mk5=tvr=off
!+1s
mk5=clock_set=64:ext:32
!+1s
mk5=1pps_source=vsi
!+1s
mk5=dot?
!+1s
mk5=dot_set=:force
!+1s
mk5=dot?
enddef
define  setup01       00000000000x
pcalon
tpicd=stop
pcald=stop
bbc01m
ifd01
bank_check
tpicd=no,2000
tpicd
enddef
define  bbc01m        00000000000x
dbbc=dbbcifc=1,agc,1
!+1s
dbbc=dbbcifd=1,agc,1
!+1s
dbbc=power=04
!+1s
dbbc=power=03
enddef
define  ifd01         00000000000x
ifdab=0,0,nor,nor
ifdcd=0,0,nor,nor
```

```
lo=
newlo=0
lo=lod,21900.00,usb,rcp,5.000
newlo=d
enddef
```

PFB tests between Onsala and Effelsberg did not produce fringes and hence part 3 was discarded.

6 The observation

The final observation took place on June 20th. Two Field System computers were used, the standard one to operate the antenna and start and stop the flow of data at the Fila10G and a second one to control the Mark5C. In principle, a single Field System instance would have been enough. Two instances are required when a parallel recording has to be done on two Mark5s, or it is necessary to rapidly switch between a Mark5B and a Mark5C with different IP addresses. The only control of the Mark5C was used to start and stop the recordings. The setup of the Mark5C was removed from the procedure file and was done by hand before the observation:

```
mk5=personality=mark5c:dualbank;
mk5=fill_pattern=464374526;
mk5=mode=unk;
mk5=packet=36:0:5008:0:0;
```

A typical cycle at the snap file can be seen below:

```
source=0234+285,023752.41,284809.0,2000.0,
setup01
!2012.172.07:59:50
preob
!2012.172.08:00:00
mk5=record=on:no0001_ys_fr015b;
data_valid=on
midob
!2012.172.08:14:00
data_valid=off
stopstream
mk5=record=off;
postob
```

`stopstream` is defined in the procedure file as:

```
define  stopstream    12172135900x
" stop fila10g data stream for 10 sec
" Please insert correct IP address of your FILA10G
sy=exec /usr2/oper/bin/stopfila10g.py 192.168.0.123 10002 &
enddef
```

The observation did not perform fringes and the causes are unknown yet.

7 Appendix

Script to stop for 10 seconds the Fila10G flow of oupt data and restart it again:
/usr2/oper/bin/stopfila10g.py

```
#!/usr/bin/python
#
# Python version of FILA10G system setup Windows batch file (*.bat).
#
# Unlike the Windows C++ version, this Python script can handle
# only UDP and TCP links to the FILA10G (RS232<=>IP converters).
#
# Example: ./stopfila10g.py <ip> <port> <datasource>
#          ./stopfila10g.py 134.104.79.24 23 vsi1   -- Effelsberg FILA10GSA, 2 G
#

import os
import time
import math
import sys
import socket

# Infos
def usage_exit():
        print 'Usage: stopfila10g.py [-u|--udp] <ip_address> <ip_port>'
        sys.exit(-1)

# Commands to send
_cmds_stop=[]
_cmds_stop.append('') # any reply?
_cmds_stop.append('write 1 x9C31C012 0')                    # stop stream on et

_cmds_start=[]
_cmds_start.append('') # any reply?
_cmds_start.append('write 1 x9C31C012 1')                    # restart stream o

# Return a socket:  conn=(ip,port,use_tcp)
def getSocket(conn):
        try:
                if conn['isTCP']:
                        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
                        s.connect((conn['ip'],conn['port']))
                else:
                        s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
                s.settimeout(1.0)
                conn['socket'] = s
```

```
        except:
                print 'Connection to %s:%u failed' % (conn['ip'],conn['port'])
                return None
        return conn

# Send a command, wait for reply
def sendRcv(conn,cmdstr):
        buffer_size = 2048
        cmdstr = cmdstr.strip() + '\n'
        if conn['isTCP']:
                conn['socket'].send(cmdstr)
        else:
                conn['socket'].sendto(cmdstr, (conn['ip'],conn['port']))
        reply = ''
        try:
                while True:
                        data = conn['socket'].recv(buffer_size)
                        reply = reply + data
        except:
                print 'Reply: %s' % (reply.strip())
                # print 'Timed out with no reply'


# Prepare and send commands to FILA10G
def main(argv):
        global _cmds

        if len(argv)<3:
                usage_exit()

        if (argv[1][0]=='-'):
                if (argv[1]=='--udp' or argv[1]=='-u') and len(argv)==4:
                        conn_TCP = False
                        conn_IPaddr = str(argv[2])
                        conn_IPport = int(argv[3])
#                       fila_source = str(argv[4])
                else:
                        usage_exit()
        else:
                conn_TCP = True
                conn_IPaddr = str(argv[1])
                conn_IPport = int(argv[2])
#               fila_source = str(argv[3])
        conn = {'ip':conn_IPaddr, 'port':conn_IPport, 'isTCP':conn_TCP, 'socket'

        print 'Connecting...'
        conn = getSocket(conn)
```

```
        if (conn==None):
                sys.exit(-1)

        # First stop sending data
        # Append data source command(s)
        _cmds = _cmds_stop
#       _cmds.append('inputselect %s' % (fila_source)) # choose of data source
#       _cmds.append('reset')                          # purge FIFOs and resync

        # Send commands
        print 'The following commands will be sent:'
        for cmd in _cmds:
                print 'Command: ', cmd.strip()
        for cmd in _cmds:
                sendRcv(conn, cmd+'\r\n')

        # Wait 10 seconds for the recording to stop
        print 'Wait 10 seconds for the recording to stop'
        time.sleep(10)
        # Then start sending data again
        # Append data source command(s)
        _cmds = _cmds_start
#       _cmds.append('inputselect %s' % (fila_source)) # choose of data source
#       _cmds.append('reset')                          # purge FIFOs and resync

        # Send commands
        print 'The following commands will be sent:'
        for cmd in _cmds:
                print 'Command: ', cmd.strip()
        for cmd in _cmds:
                sendRcv(conn, cmd+'\r\n')
        print 'Done!'

if __name__ == "__main__":
    main(sys.argv)
```