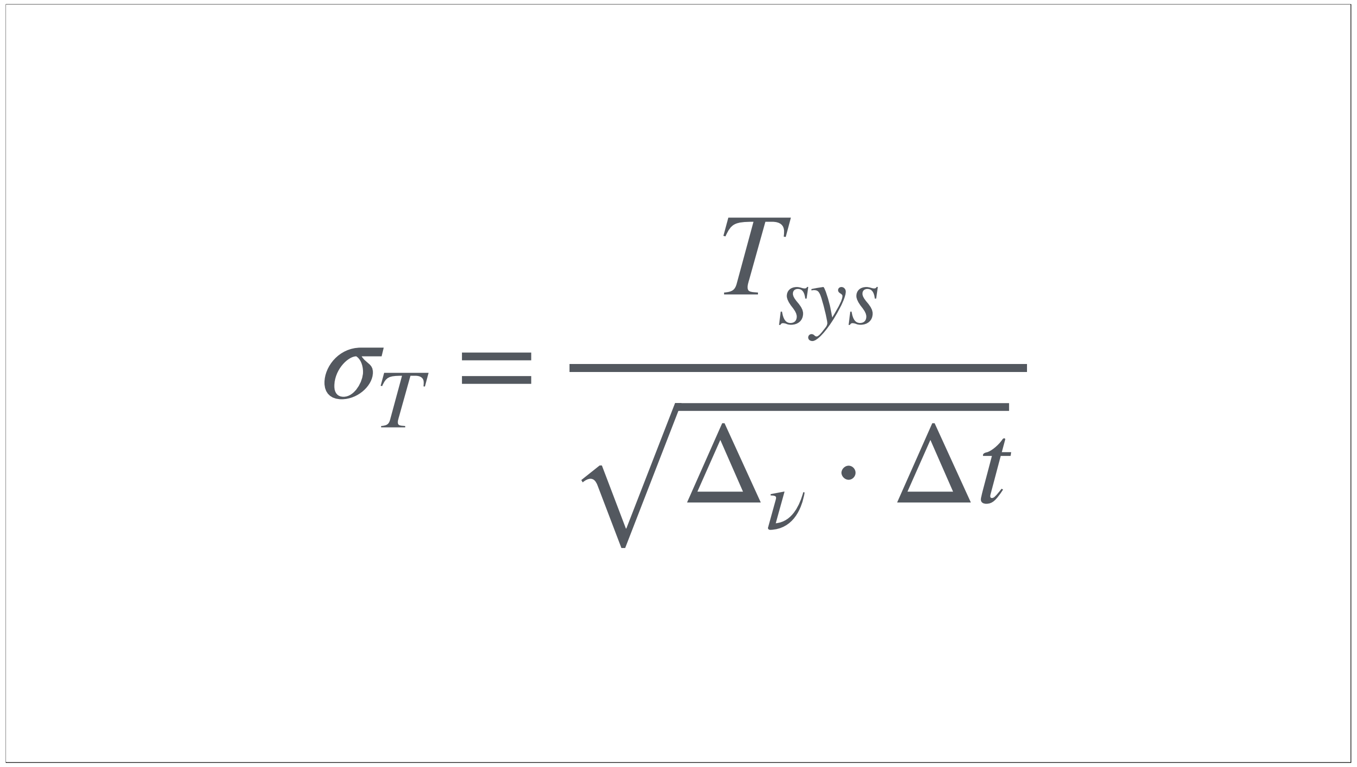
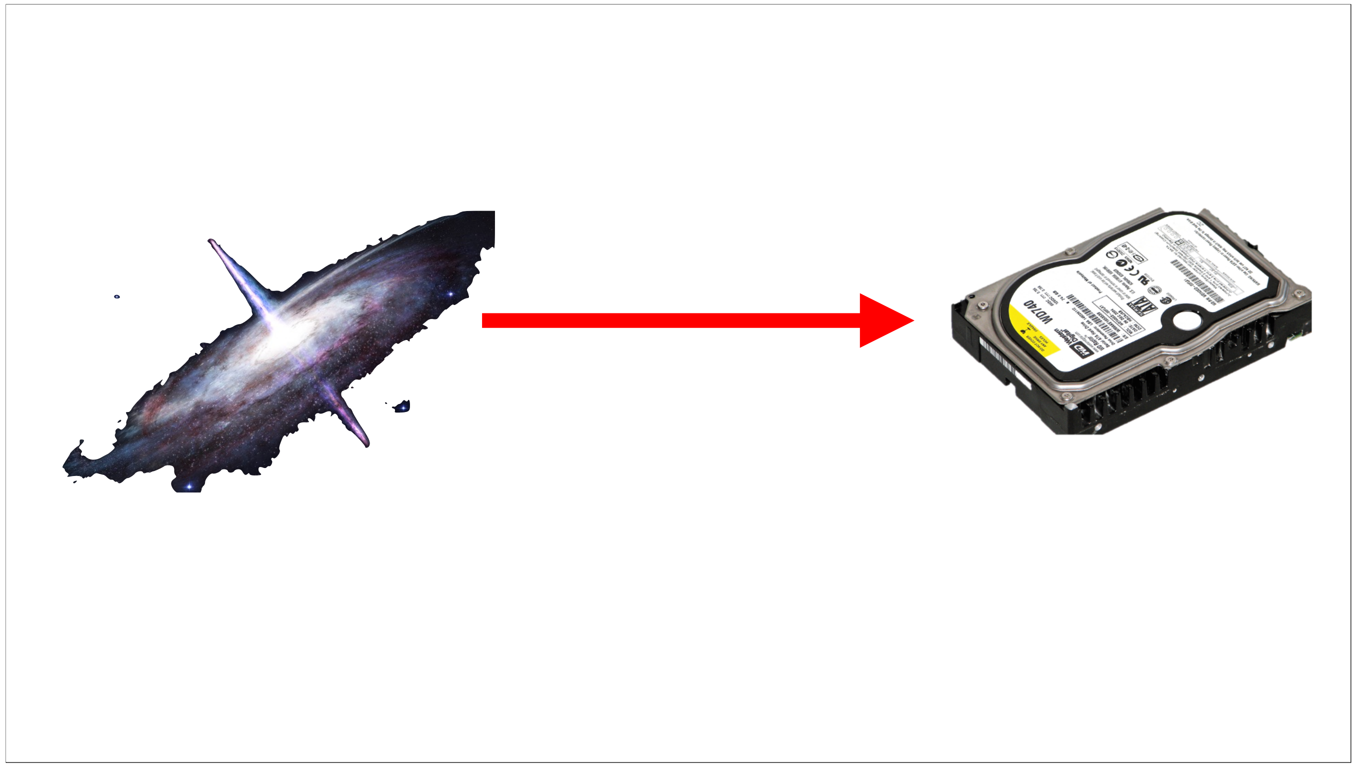


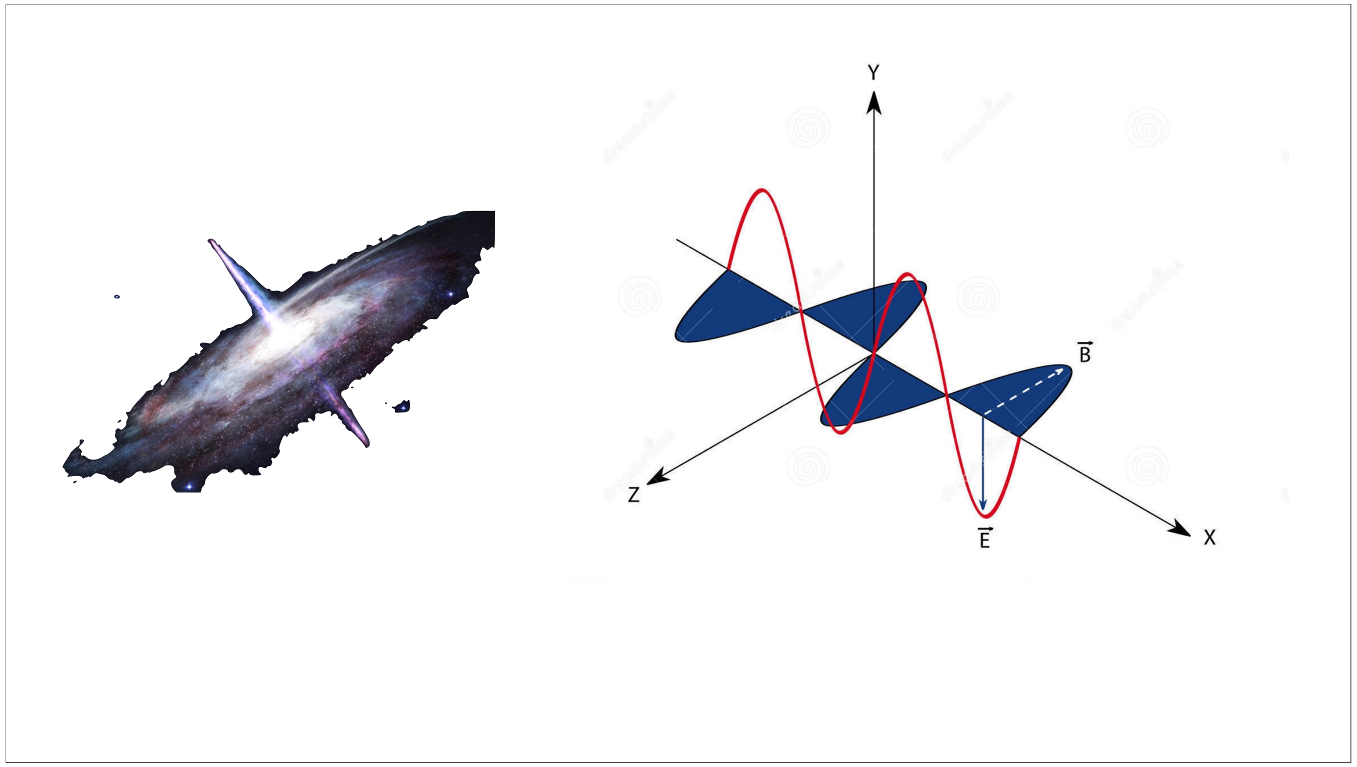
$$\sigma_T = \frac{T_{sys}}{\sqrt{\Delta_\nu \cdot \Delta t}}$$

hello everyone, good time of day whichever time zone you are in!

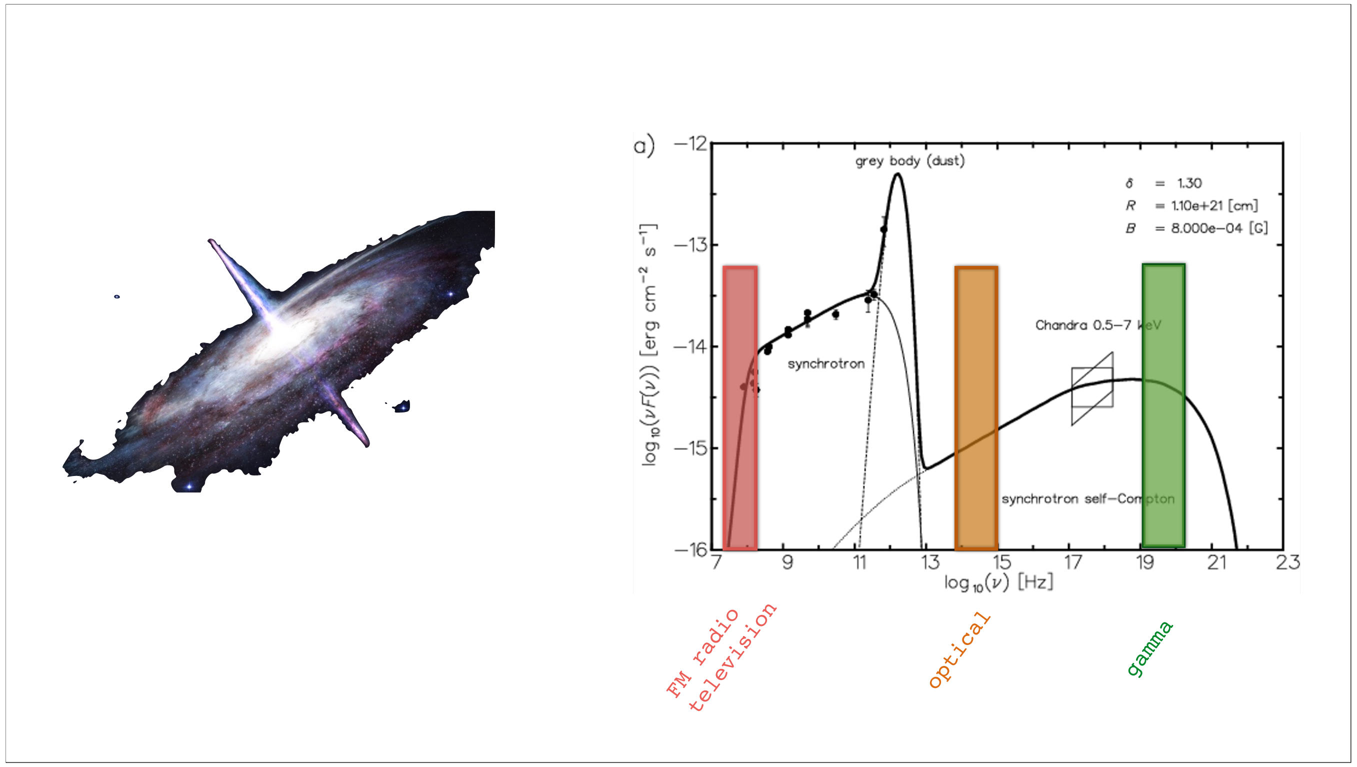
This is an impression of a quasar. A strongly emitting astronomical source.



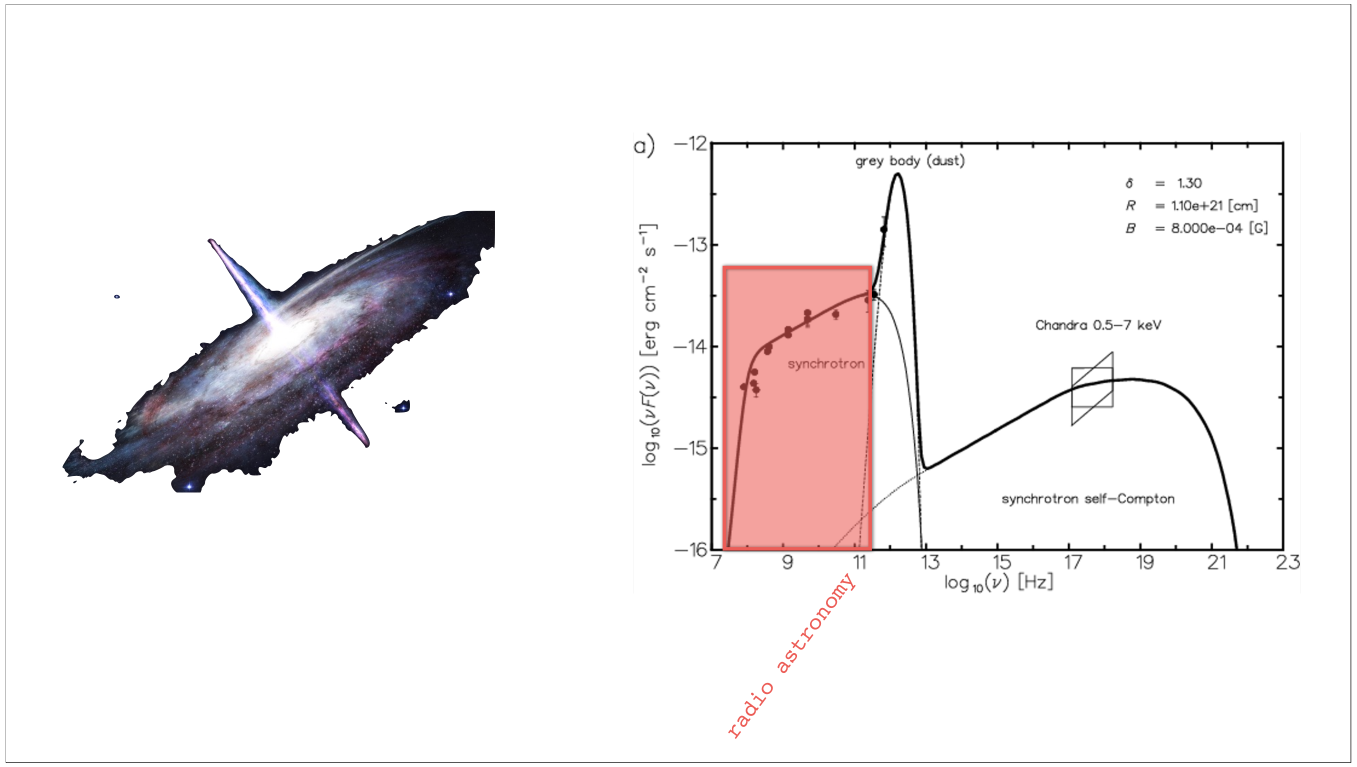
Our goal in this lecture is to get that thing on a hard disk.



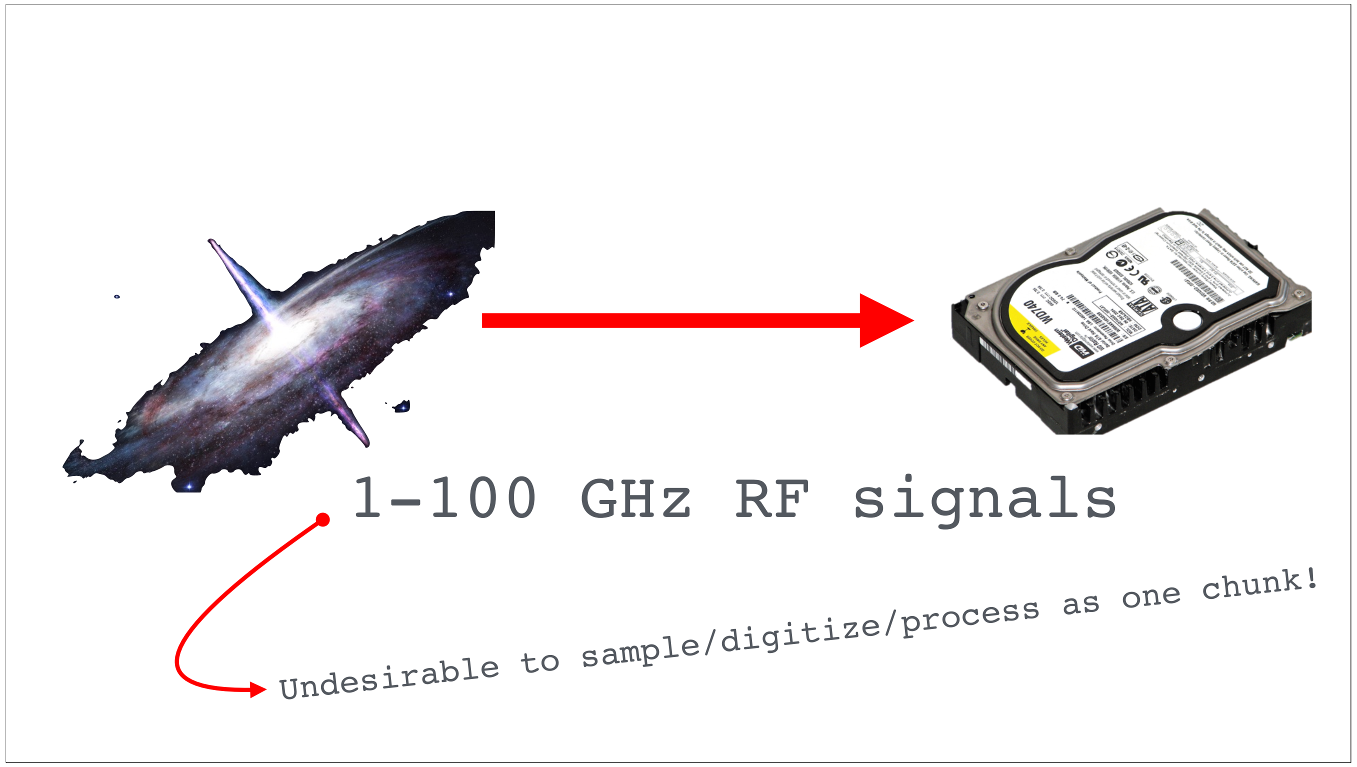
The quasar emits electromagnetic radiation



across all of the electro-magnetic spectrum, [click] from high energy gamma rays, through [click] optical, down to [click] FM radio



In radio astronomy we're mainly concerned with this part of the spectrum



1-100 GHz RF signals

Undesirable to sample/digitize/process as one chunk!

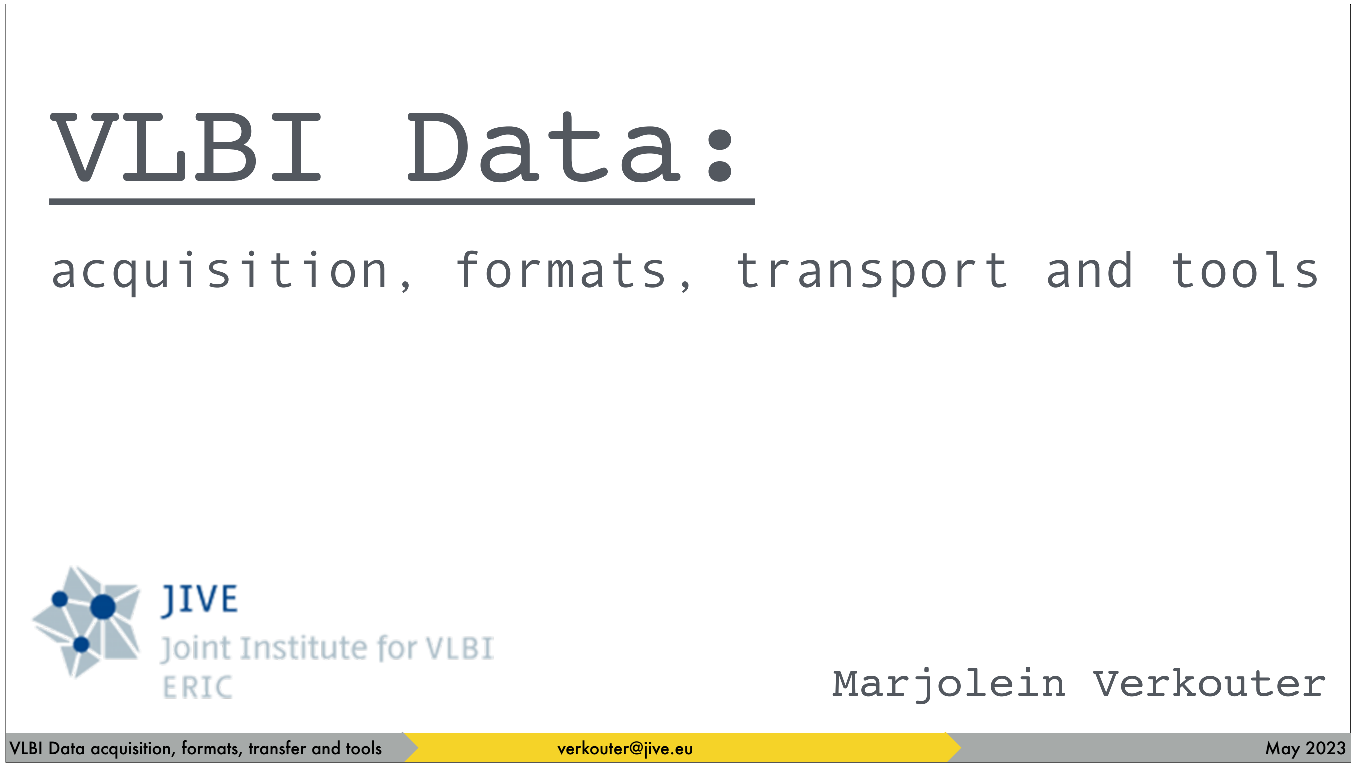
so we’re looking at radio frequency signals from MHz to 100 GHz. For many reasons [click] we cannot or want to process this as one chunk of data.

analog

?

digital

basically something needs to happen inbetween.



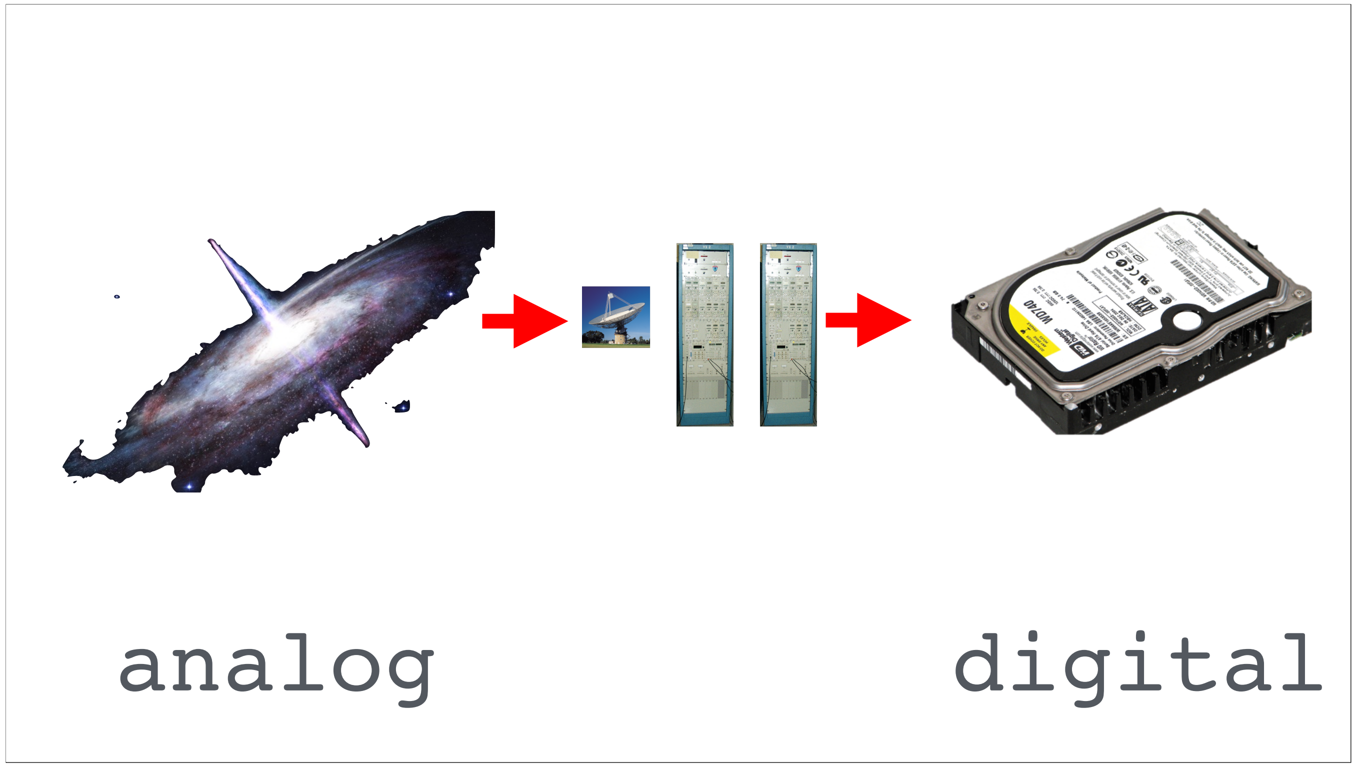
VLBI Data:

acquisition, formats, transport and tools

JIVE
Joint Institute for VLBI
ERIC

Marjolein Verkouter

My name is Marjolein Verkouter; I work at the Joint institute for VLBI and this is a lecture on these here topics

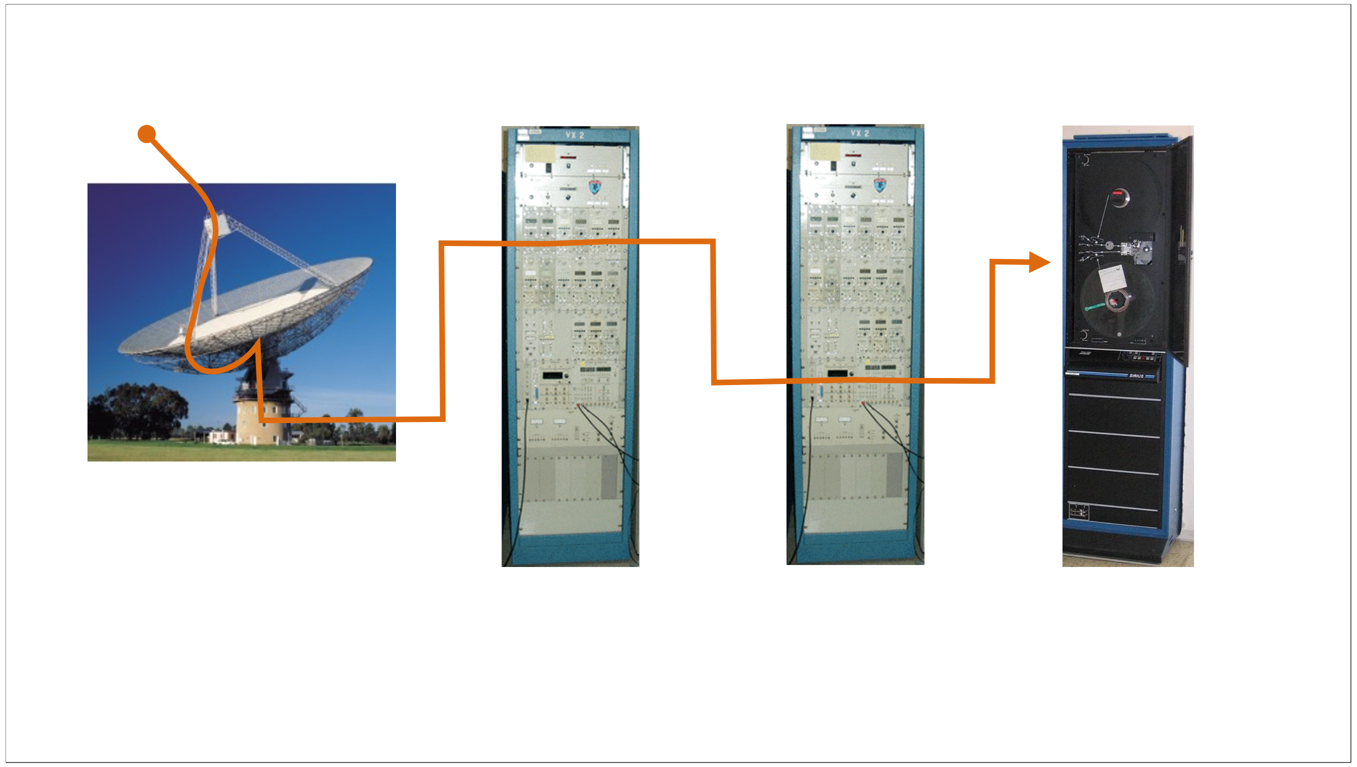


analog

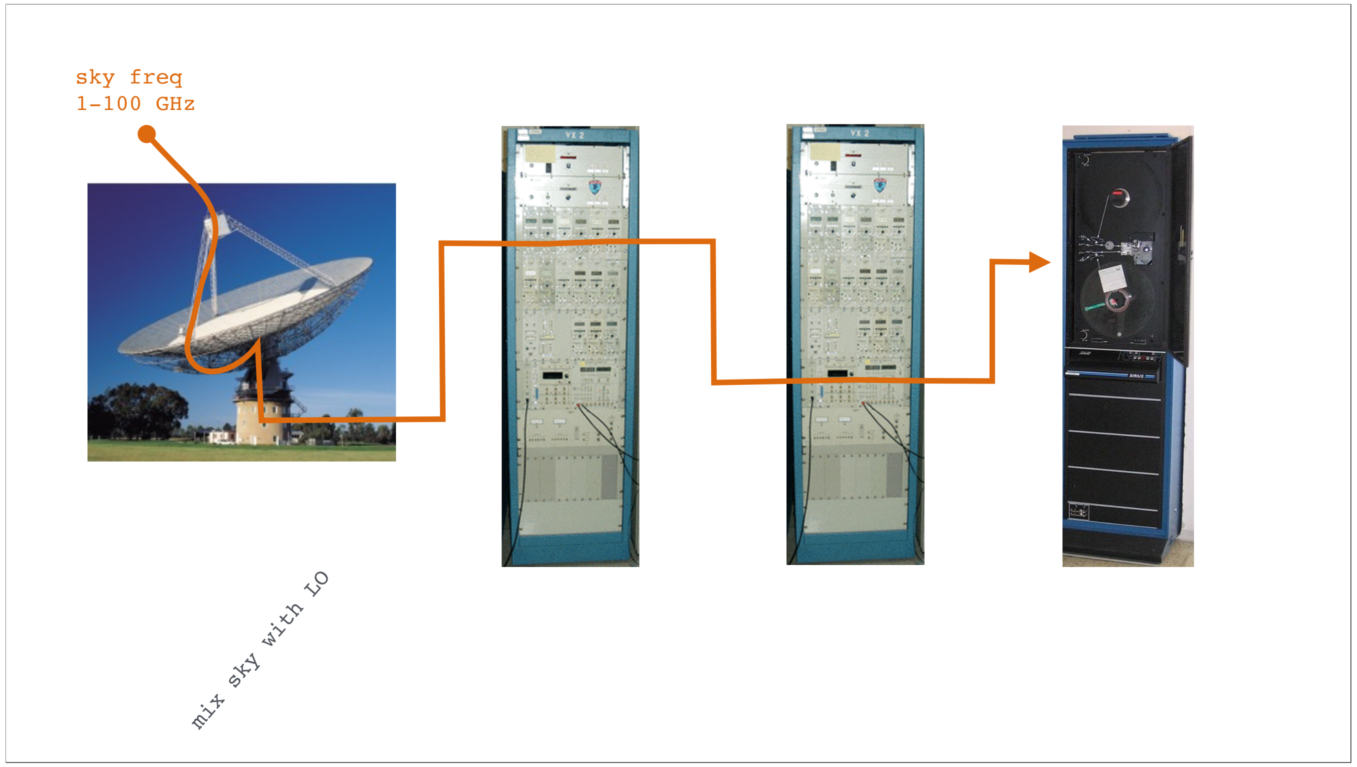
digital

What happens inbetween is a signal detection and processing chain.

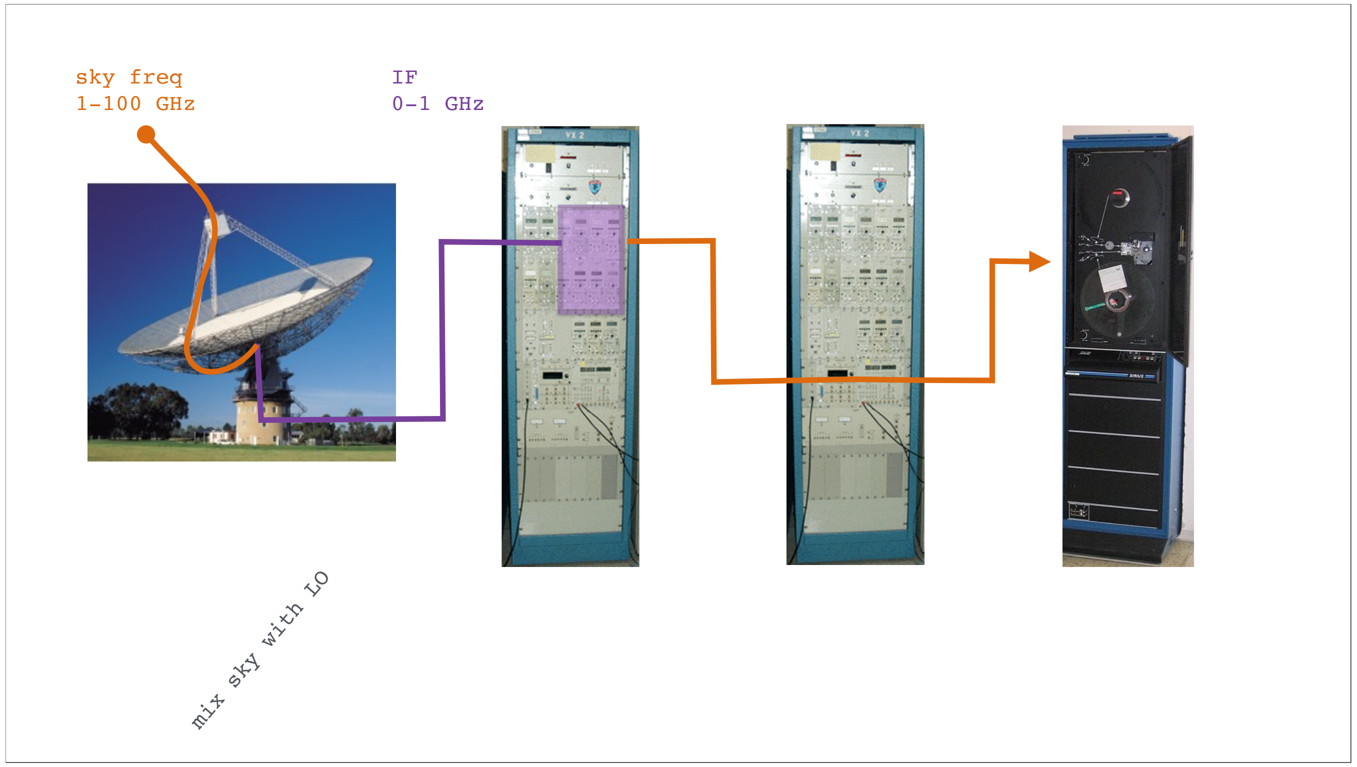
I use the old analog signal chain for illustration, because it allows us to really _point_ at the individual signal processing steps …



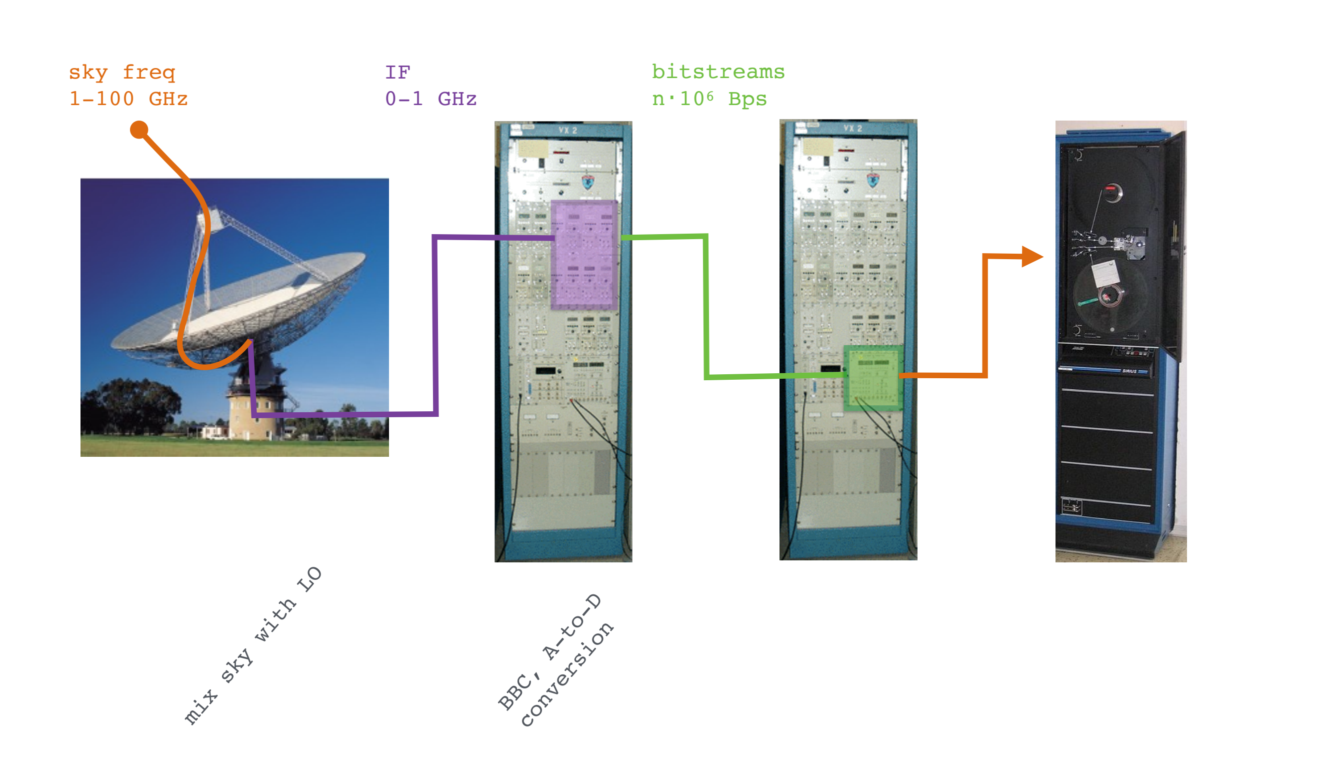
… and the path the signal takes through them



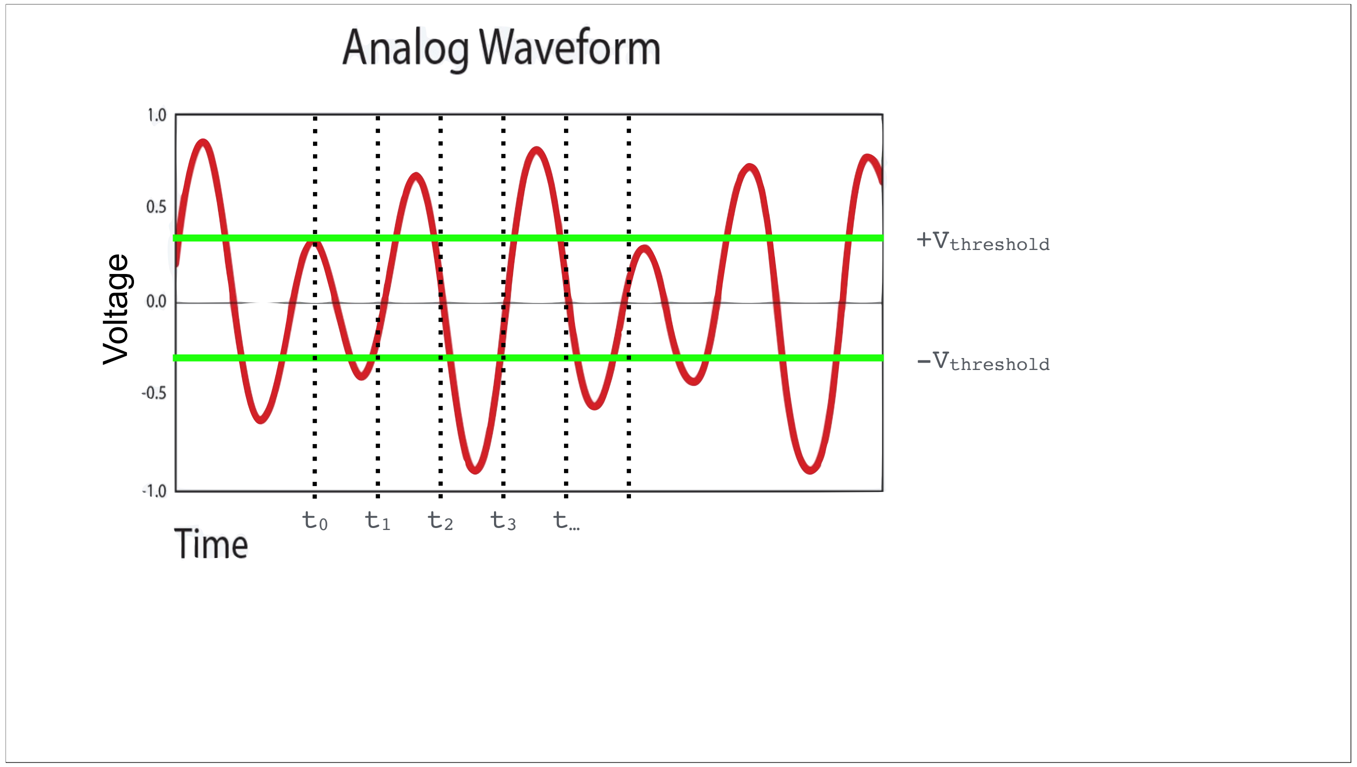
we start off by taking the sky signal and mixing it with a very precise “local oscillator” frequency in the telescope’s frontend.



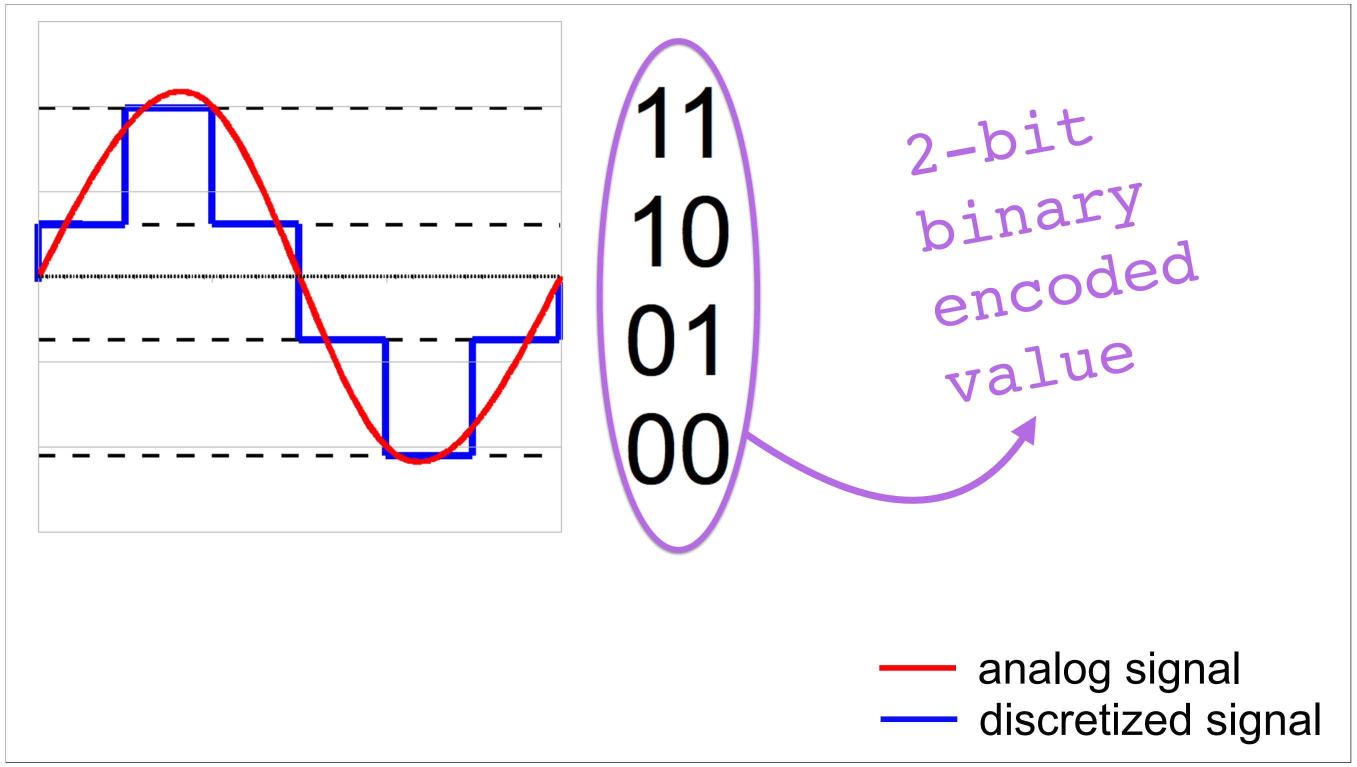
this brings down the sky signal to a frequency band called the “intermediate frequency”, or “eye ef”, which has a manageable bandwidth.



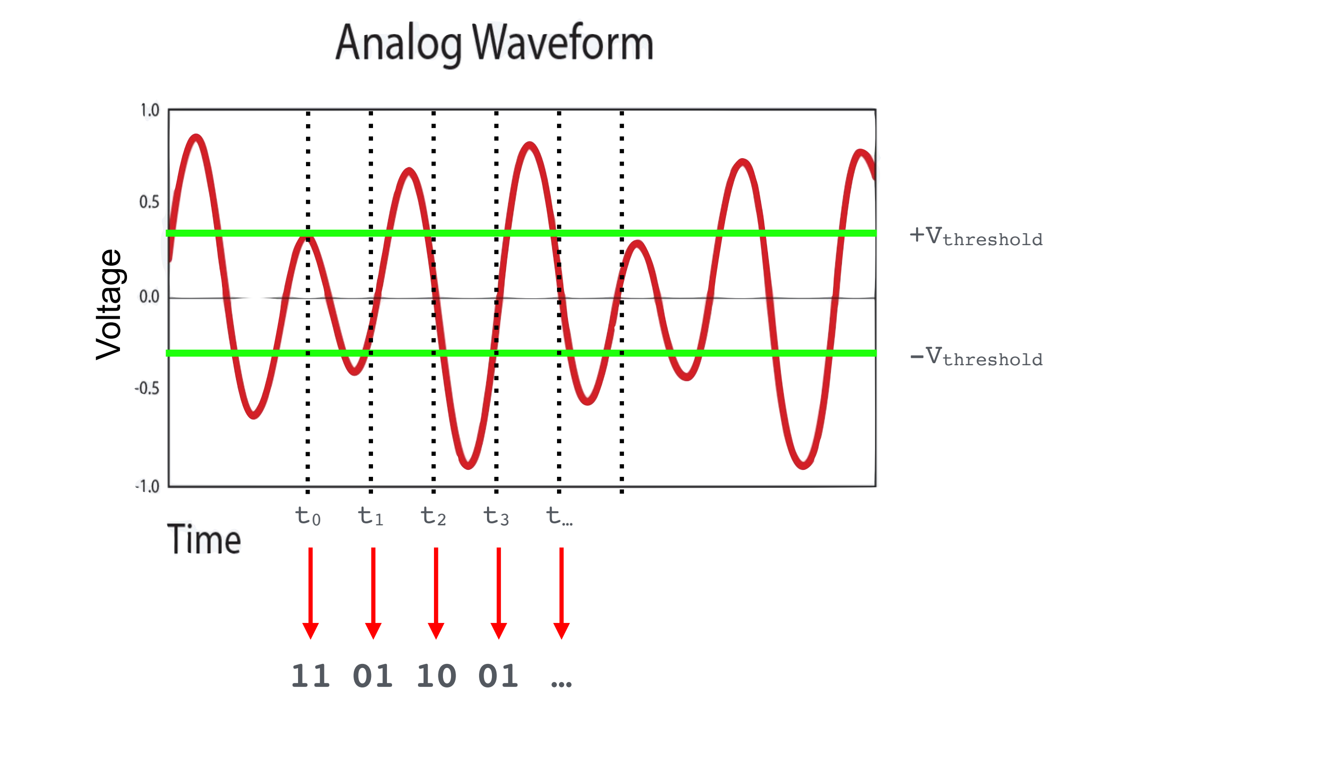
The IF bandwidth is further subdivided in smaller bandwiths still, a process called base band conversion (BBC), and the signal is converted from analog to digital.



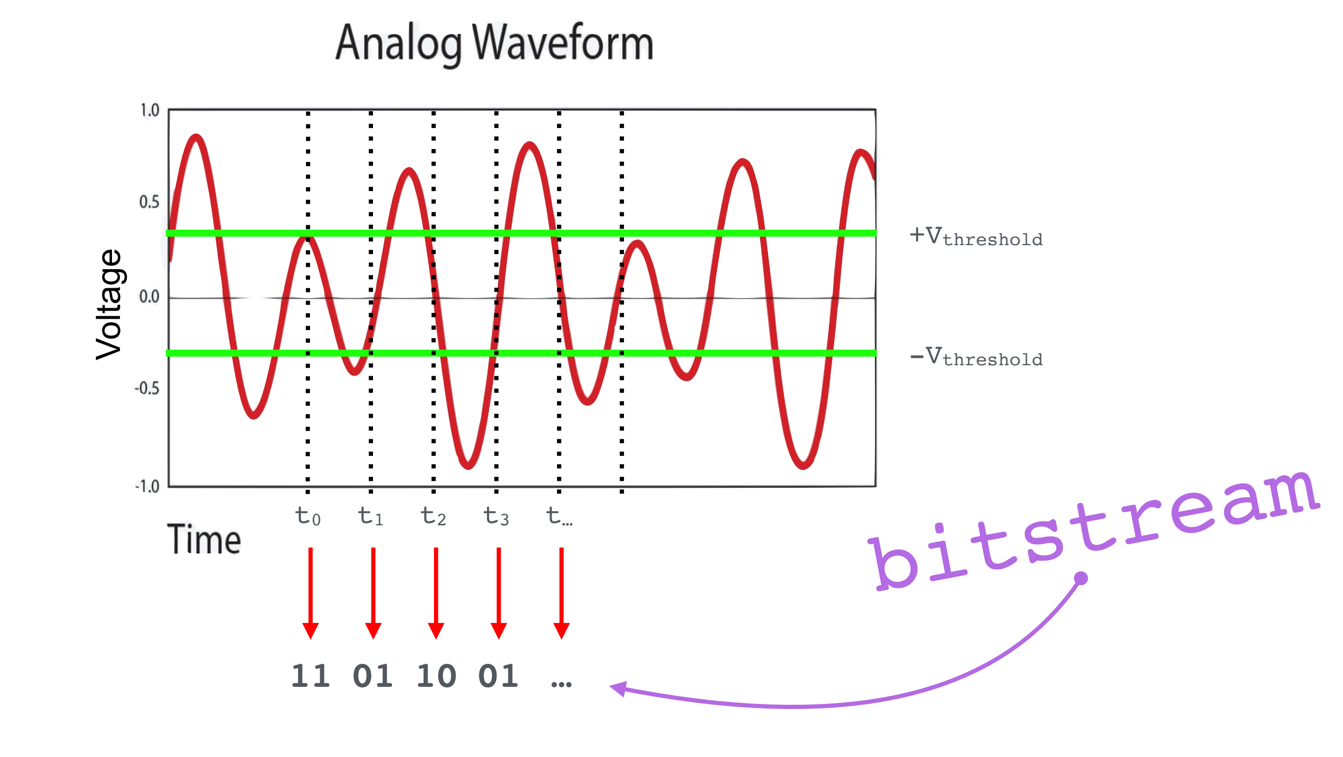
This conversion is quite simple. The analog voltage as function of time is [click] compared to a threshold voltage at [click] regular times.



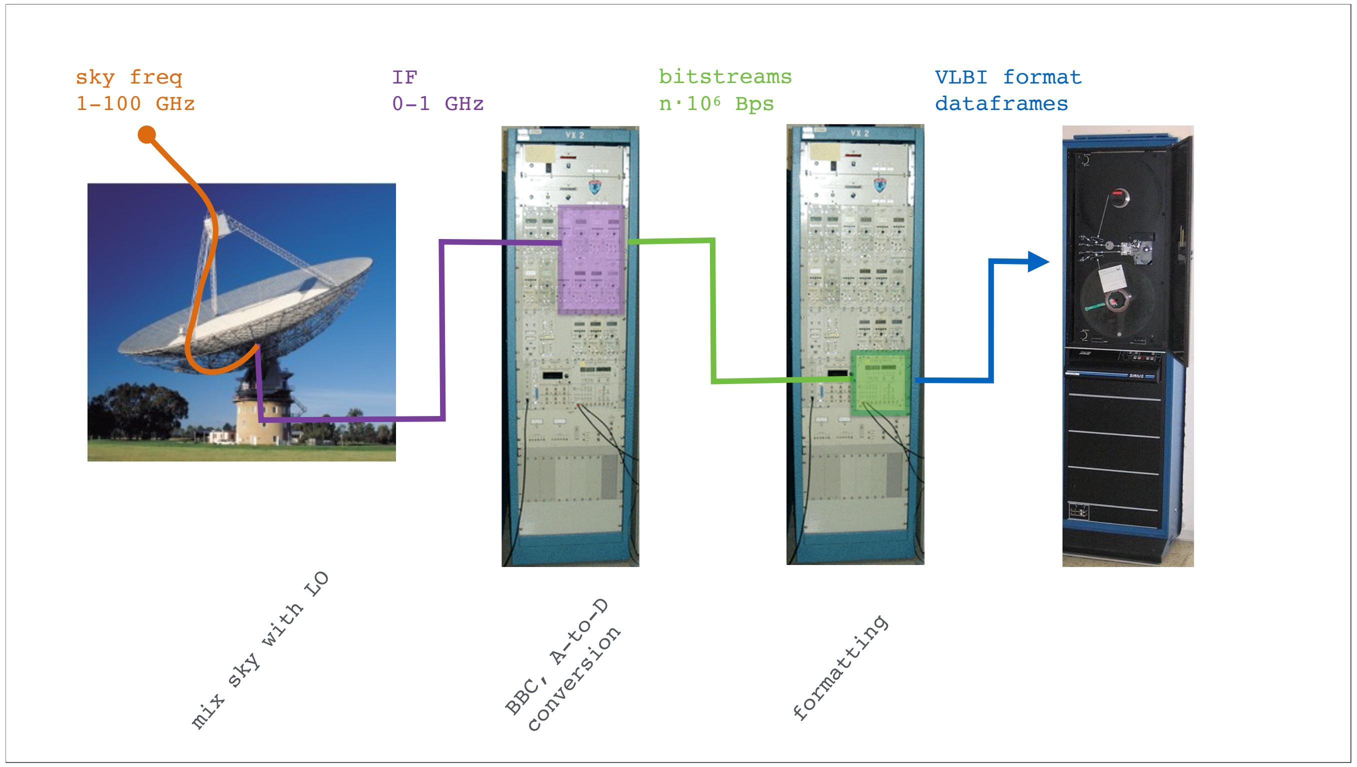
The value of the signal is encoded through a simple table as shown here for the 2-bit discretization



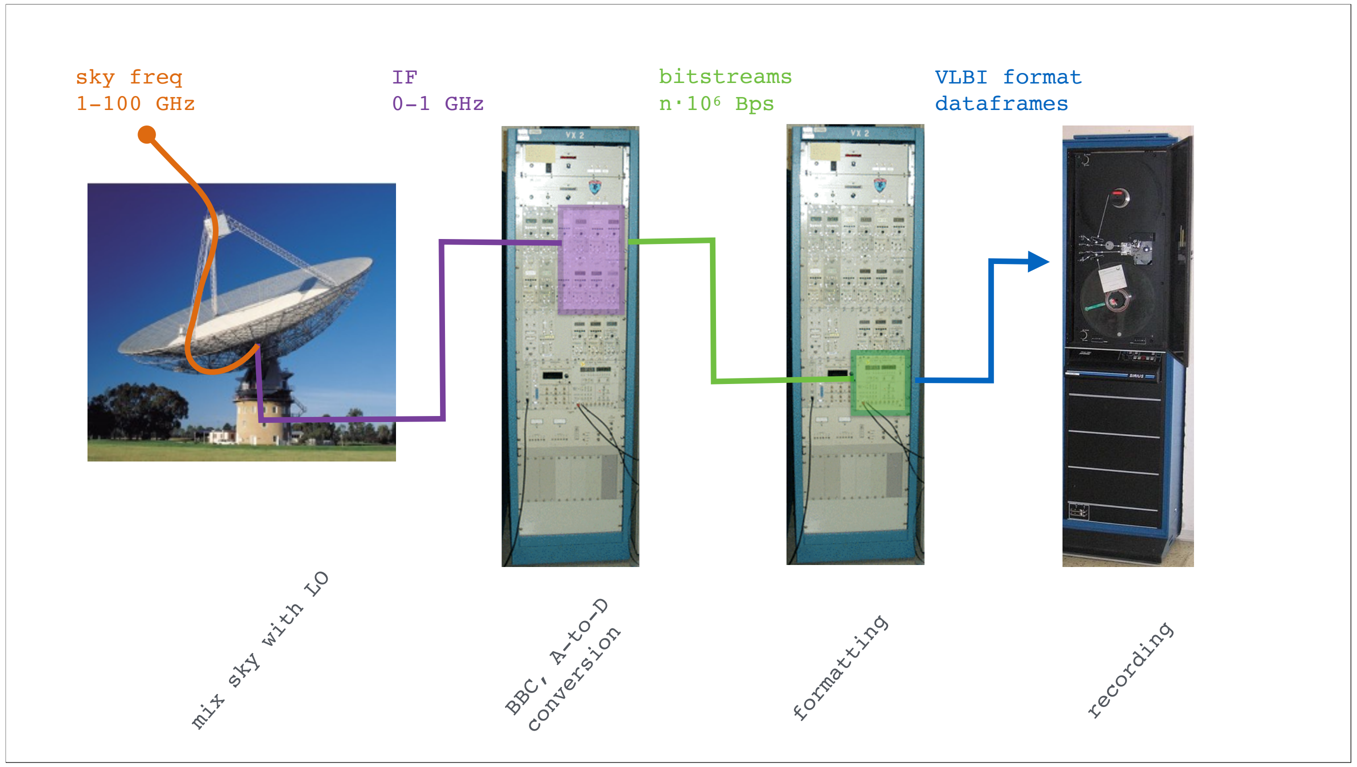
Thus, when repeatedly sampling the voltage, the …



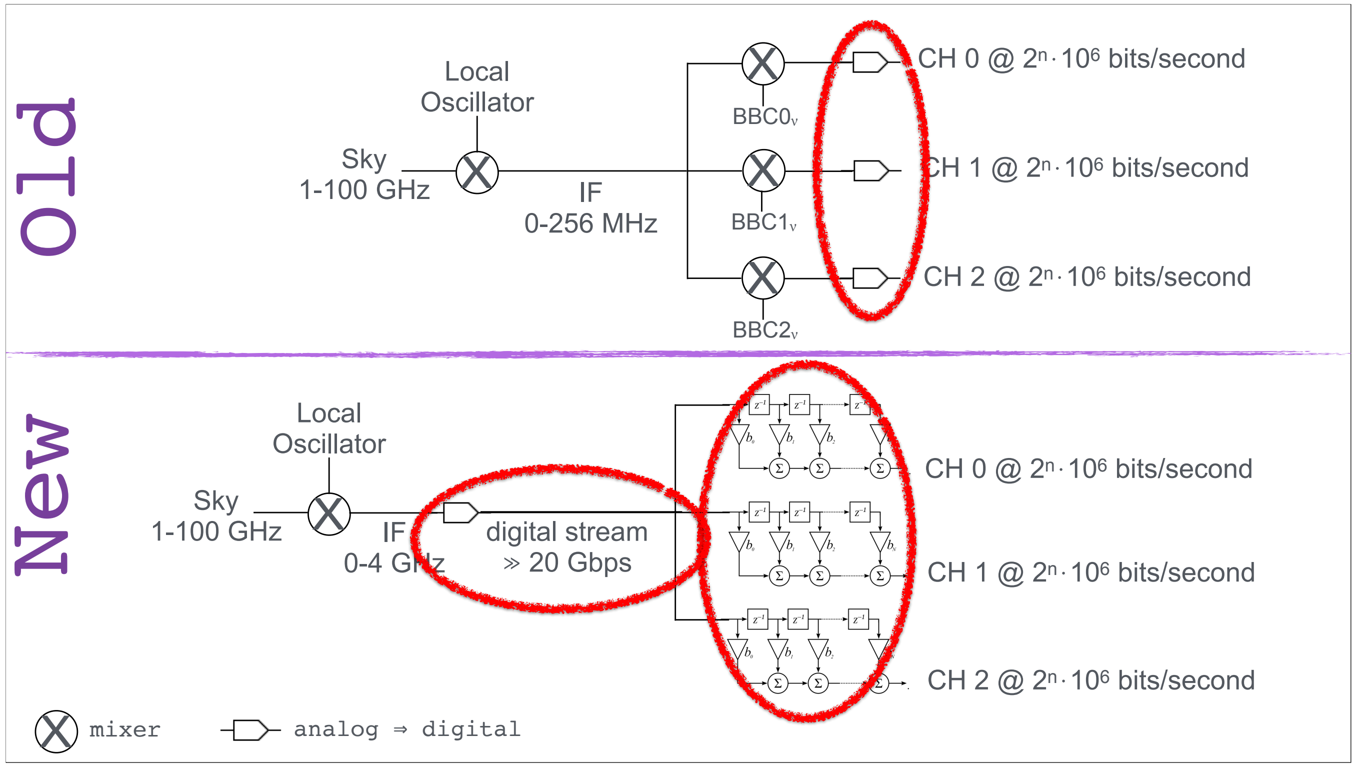
… concatenation of those 2-bit samples is a bitstream, representing the sampled signal.



Those sampled bitstreams are time tagged and frame headers are inserted at regular intervals; a process called "formatting"



and those frames, finally, are written to a persistent medium by a recorder.



Old

Local Oscillator

Sky 1-100 GHz

IF 0-256 MHz

BBC0$_v$

BBC1$_v$

BBC2$_v$

CH 0 @ $2^n \cdot 10^6$ bits/second

CH 1 @ $2^n \cdot 10^6$ bits/second

CH 2 @ $2^n \cdot 10^6$ bits/second

New

Local Oscillator

Sky 1-100 GHz

IF 0-4 GHz

digital stream » 20 Gbps

CH 0 @ $2^n \cdot 10^6$ bits/second

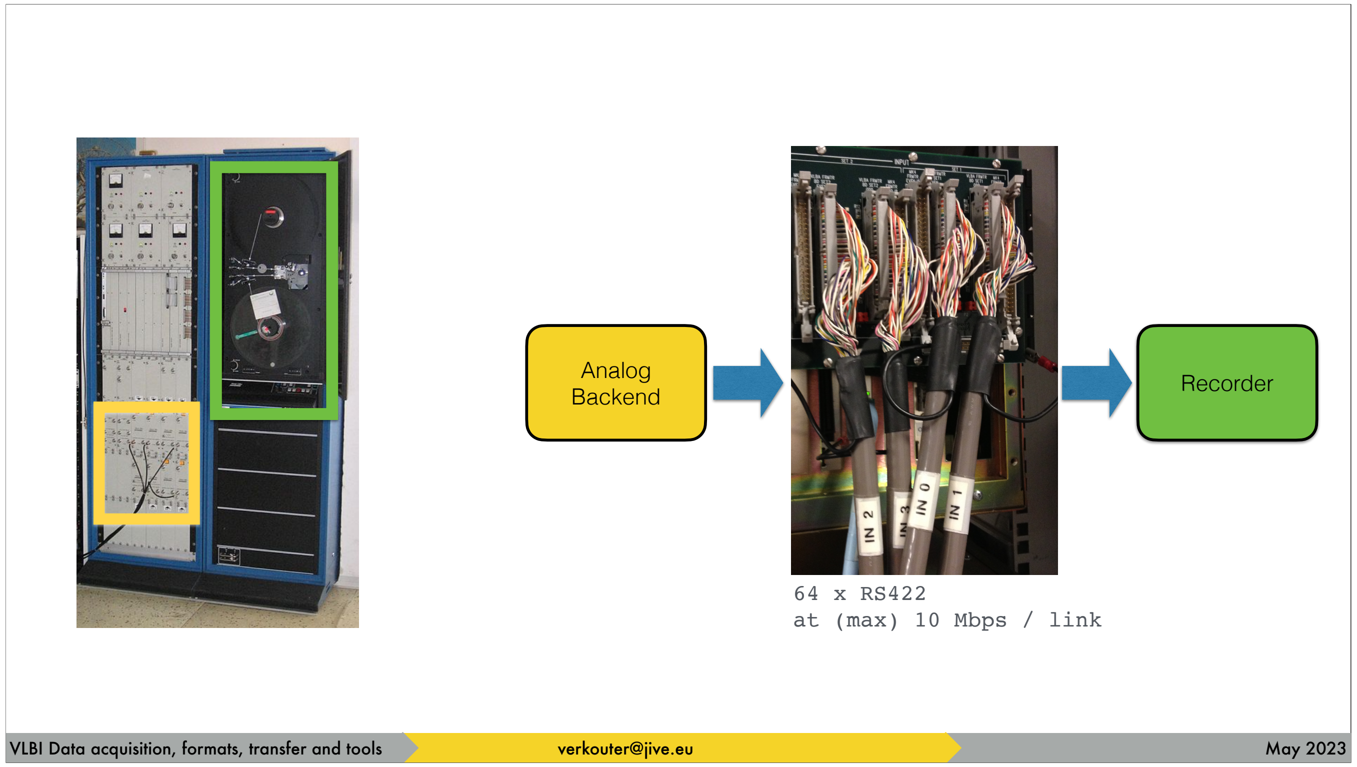
CH 1 @ $2^n \cdot 10^6$ bits/second

CH 2 @ $2^n \cdot 10^6$ bits/second

mixer

analog ⇒ digital

Interestingly enough, from a signal processing point of view you can rearrange some of the steps without affecting the end result. In the old analog system the analog to digital conversion happens here [click] after the frequency channels have been formed. Modern digital backends convert the IF to a digital stream first [click] and then form the frequency channels by digital signal processing here [click], which is much easier to implement and offers more flexibility.



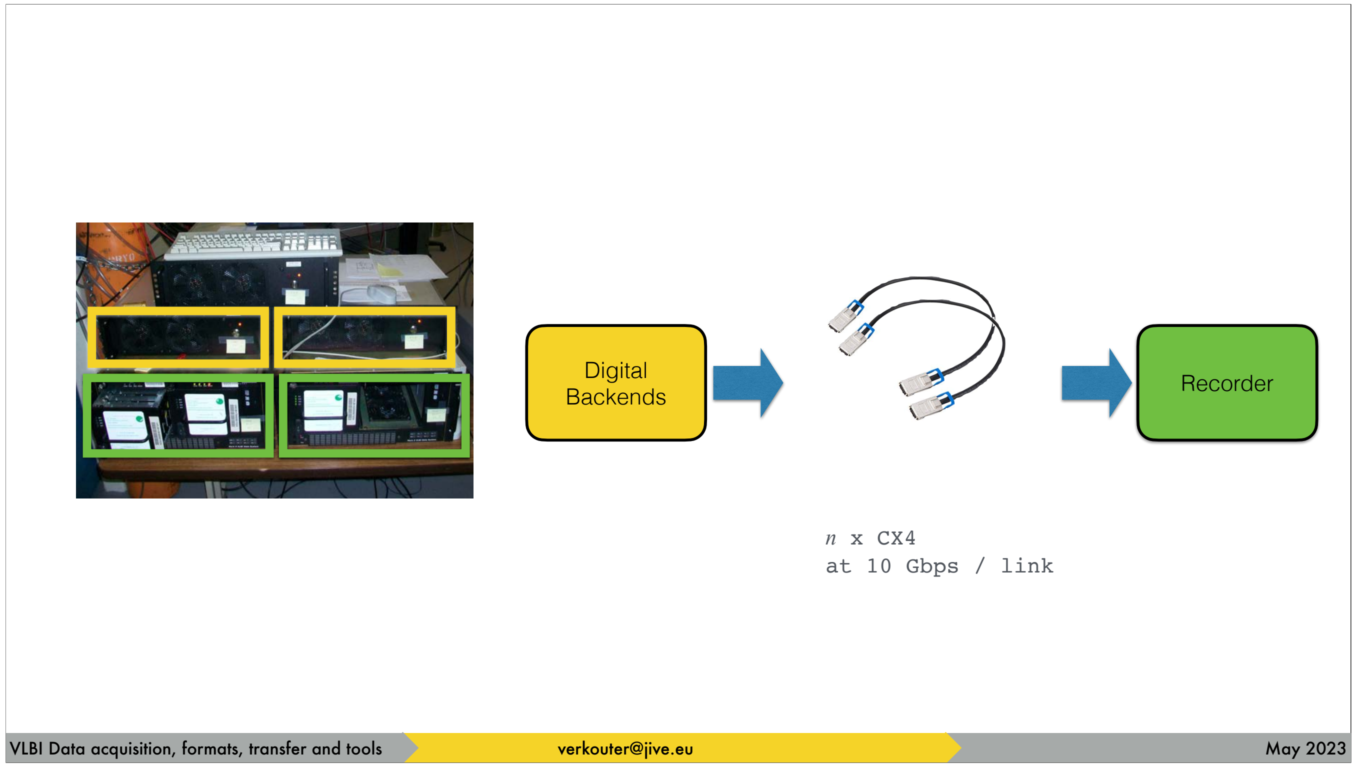
Analog Backend

Recorder

64 x RS422
at (max) 10 Mbps / link

VLBI Data acquisition, formats, transfer and tools verkouter@jive.eu May 2023

In the days of the MarkIV/VLBA system the [click] digitizers were connected to the recorder [click] via a multitude [click] of parallel serial RS422 connections

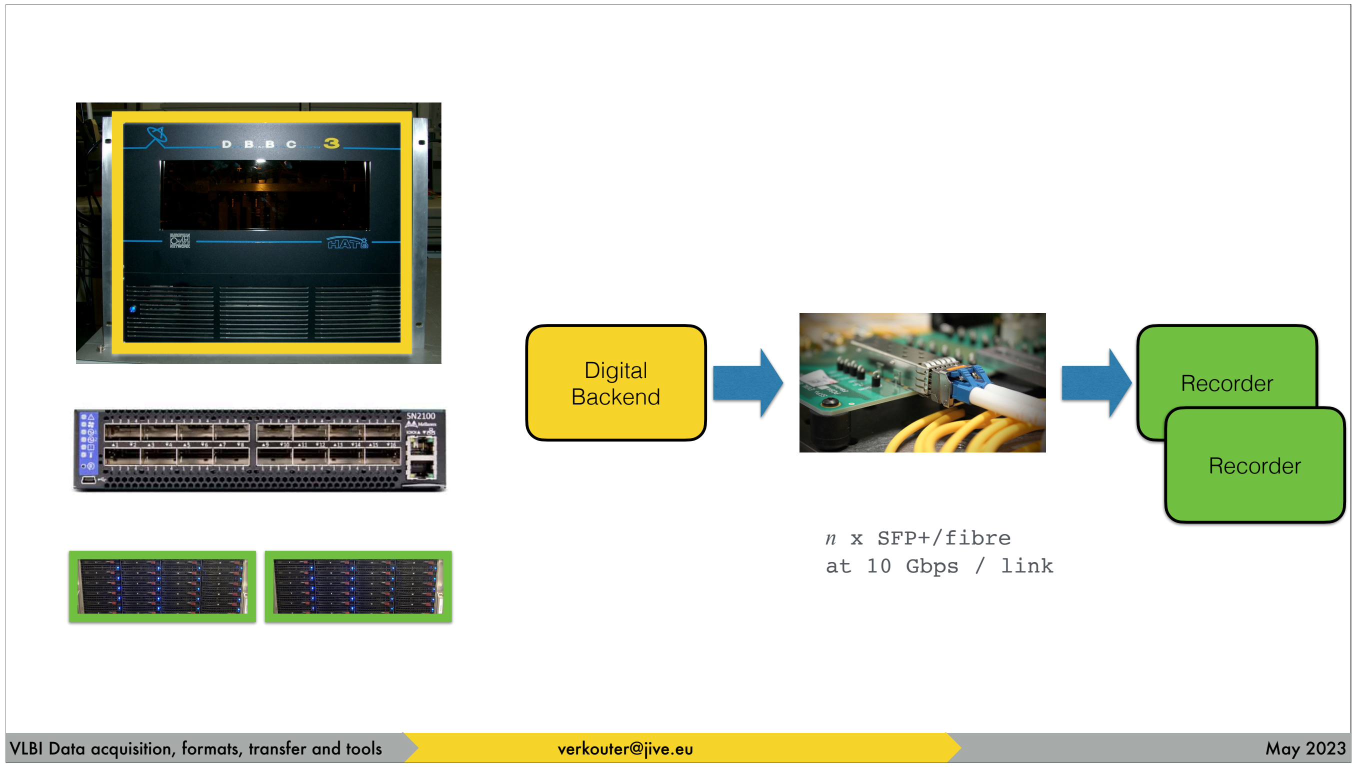


Digital Backends

Recorder

n x CX4
at 10 Gbps / link

Slightly more modern digital backends [click] send their data to [click] multiple recorders over [click] 10 Gbps CX4 ethernet cables,



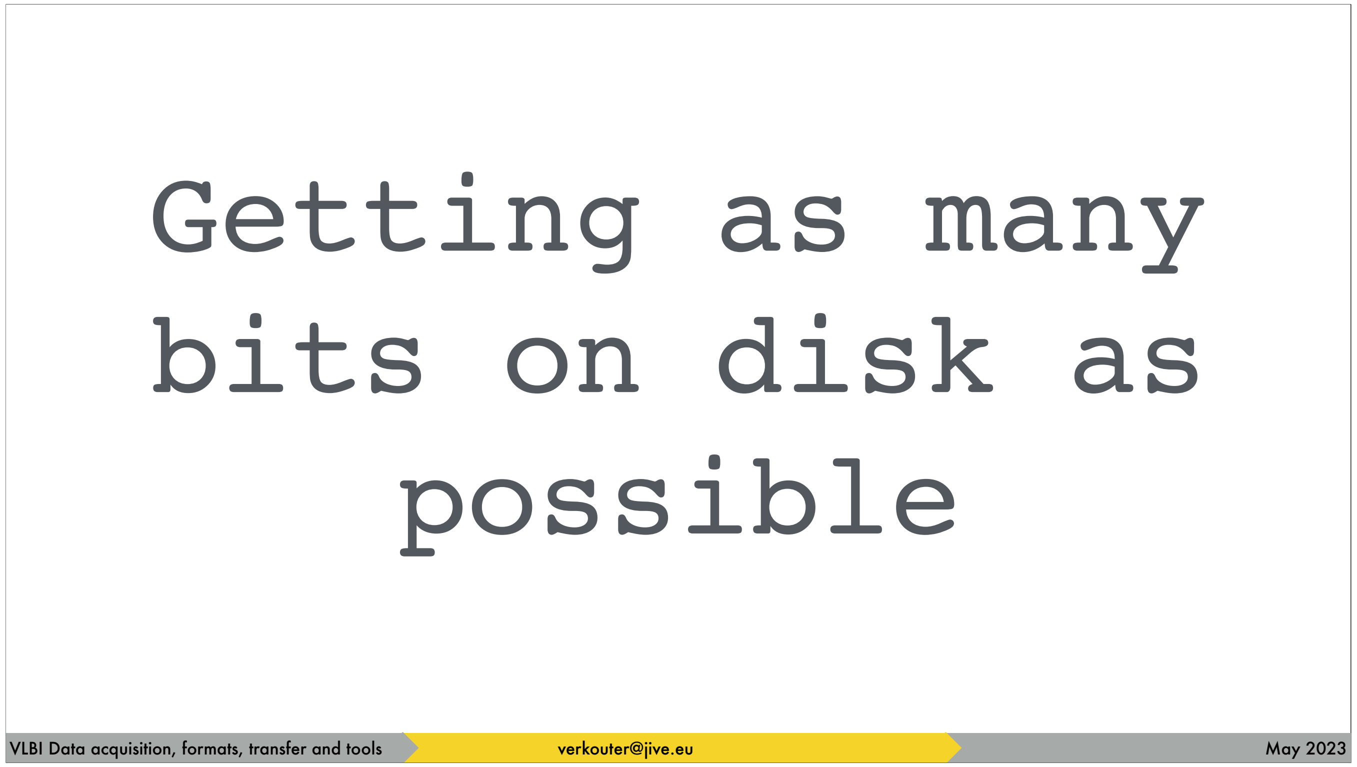
Digital Backend

n x SFP+/fibre
at 10 Gbps / link

Recorder

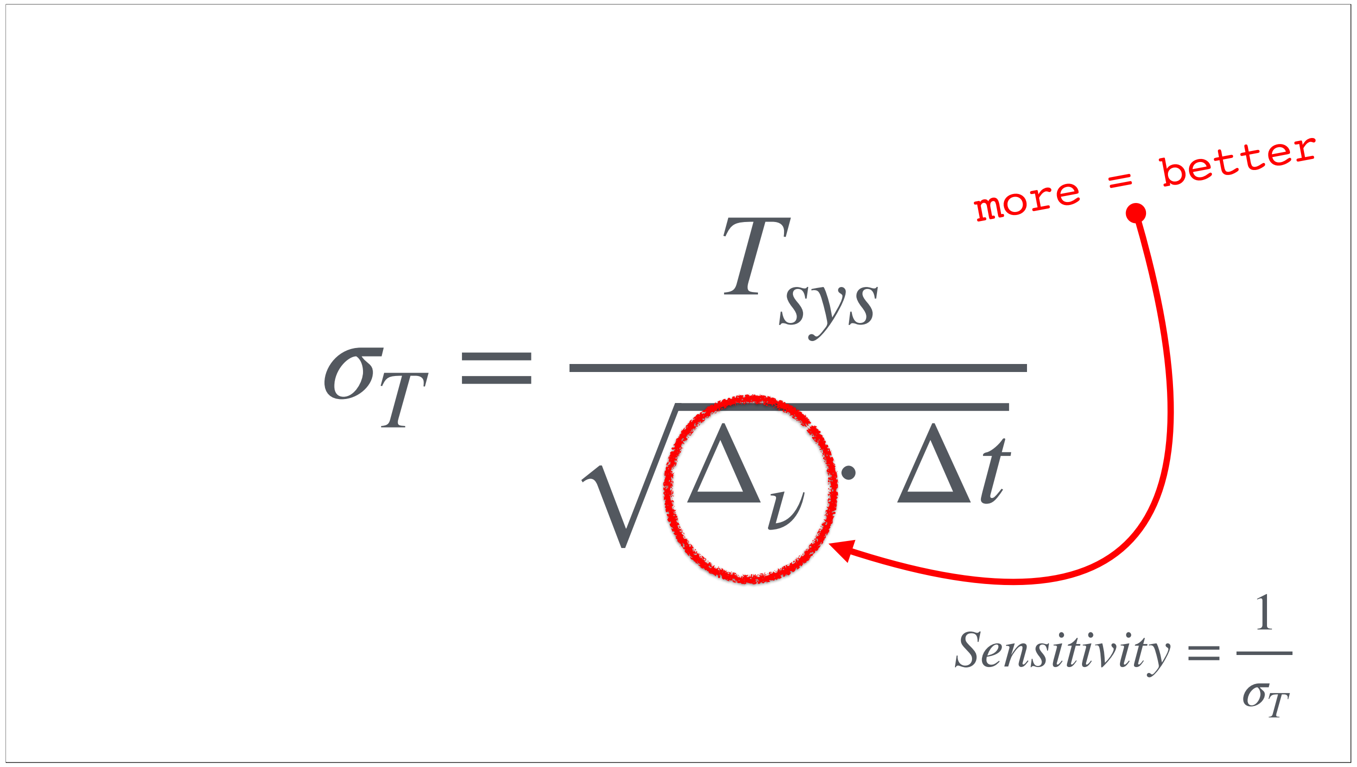
Recorder

or the DBBC3 sends its data to multiple FlexBuff recorders via a switch.



Getting as many bits on disk as possible

The driving force behind all this is always this, because of …

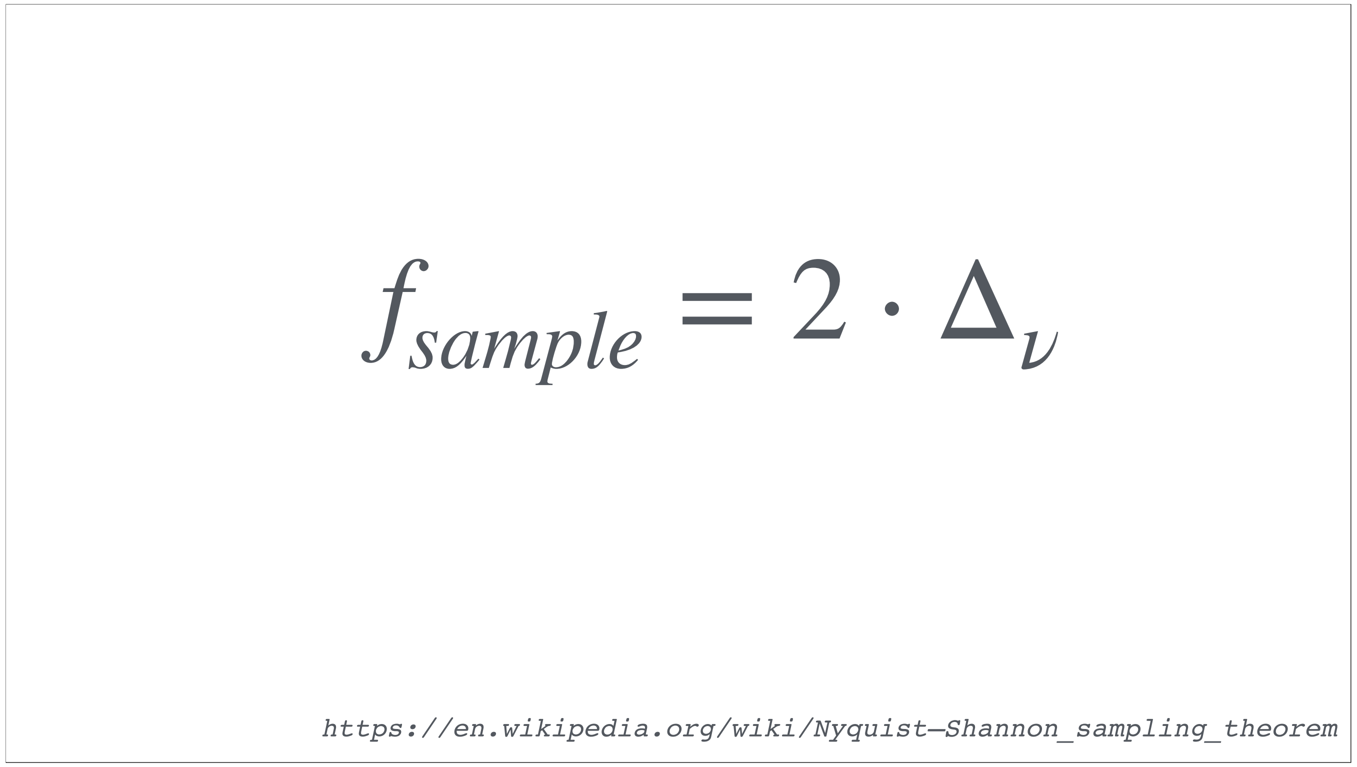


$$\sigma_T = \frac{T_{sys}}{\sqrt{\Delta_\nu \cdot \Delta t}}$$

more = better

$$Sensitivity = \frac{1}{\sigma_T}$$

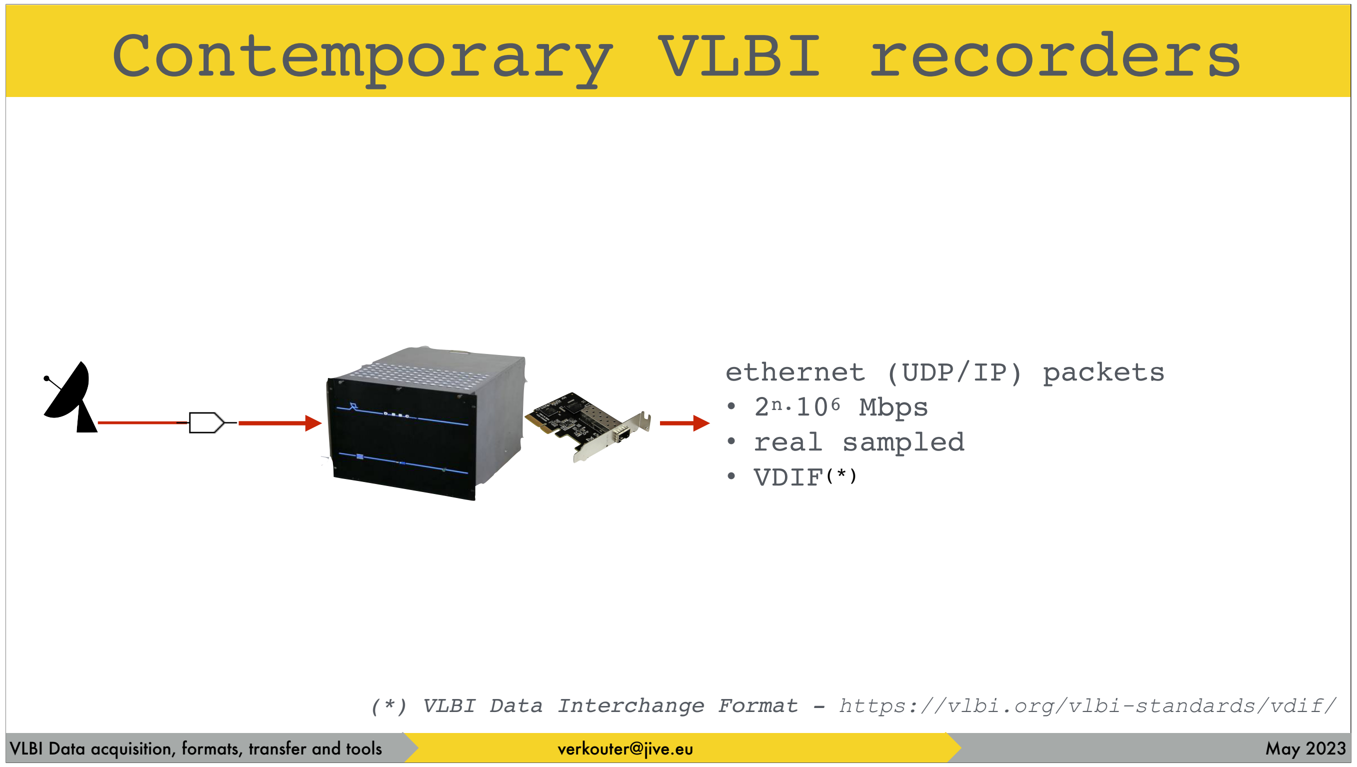
… the sensitivity. And this value depends on a number of things. The most efficient one is to [click] increase the recorded bandwidth.



$$f_{sample} = 2 \cdot \Delta_{\nu}$$

https://en.wikipedia.org/wiki/Nyquist–Shannon_sampling_theorem

and by the findings of certains misters Nyquist and Shannon, more bandwidth requires more samples, i.e. more bits to record.



Contemporary VLBI recorders

ethernet (UDP/IP) packets
- $2^n \cdot 10^6$ Mbps
- real sampled
- VDIF(*)

(*) VLBI Data Interchange Format - https://vlbi.org/vlbi-standards/vdif/

The producers of those bits are the contemporary digital backends. These produce [click] a stream - or streams - of ethernet packets with VDIF frames in them; we'll discuss VDIF later. It is up to the recorders

Contemporary VLBI recorders

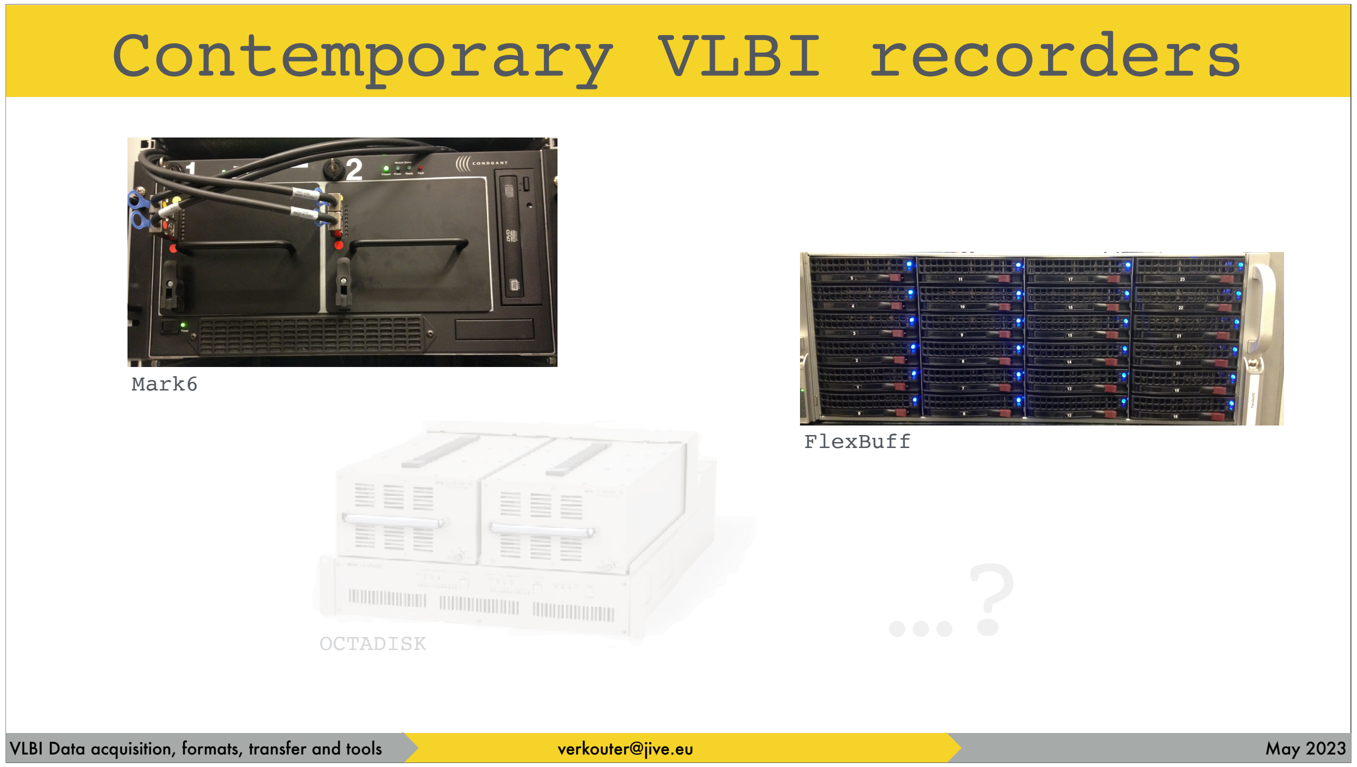
Mark6

FlexBuff

OCTADISK

...?

such as these to capture them.



Contemporary VLBI recorders

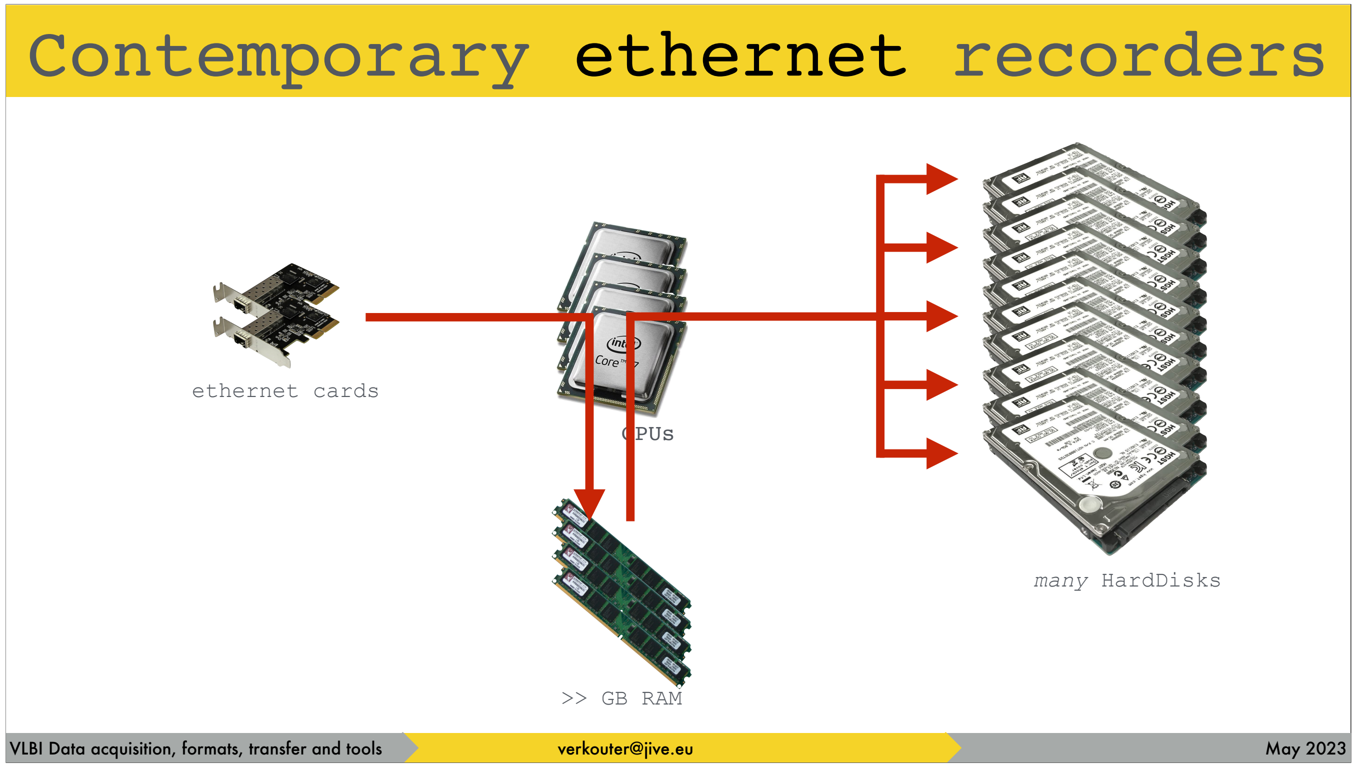
Mark6

FlexBuff

OCTADISK

...?

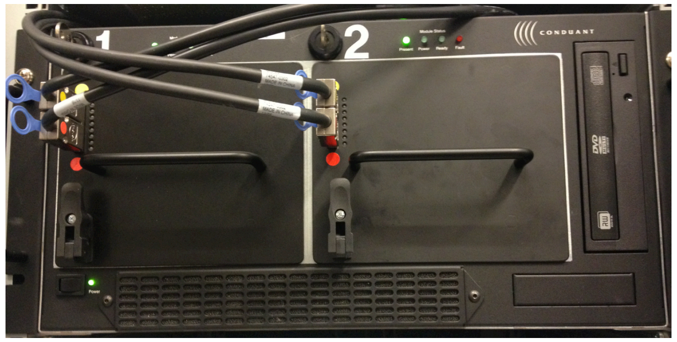
In this lecture we'll focus on Mark6 and FlexBuff



Contemporary ethernet recorders

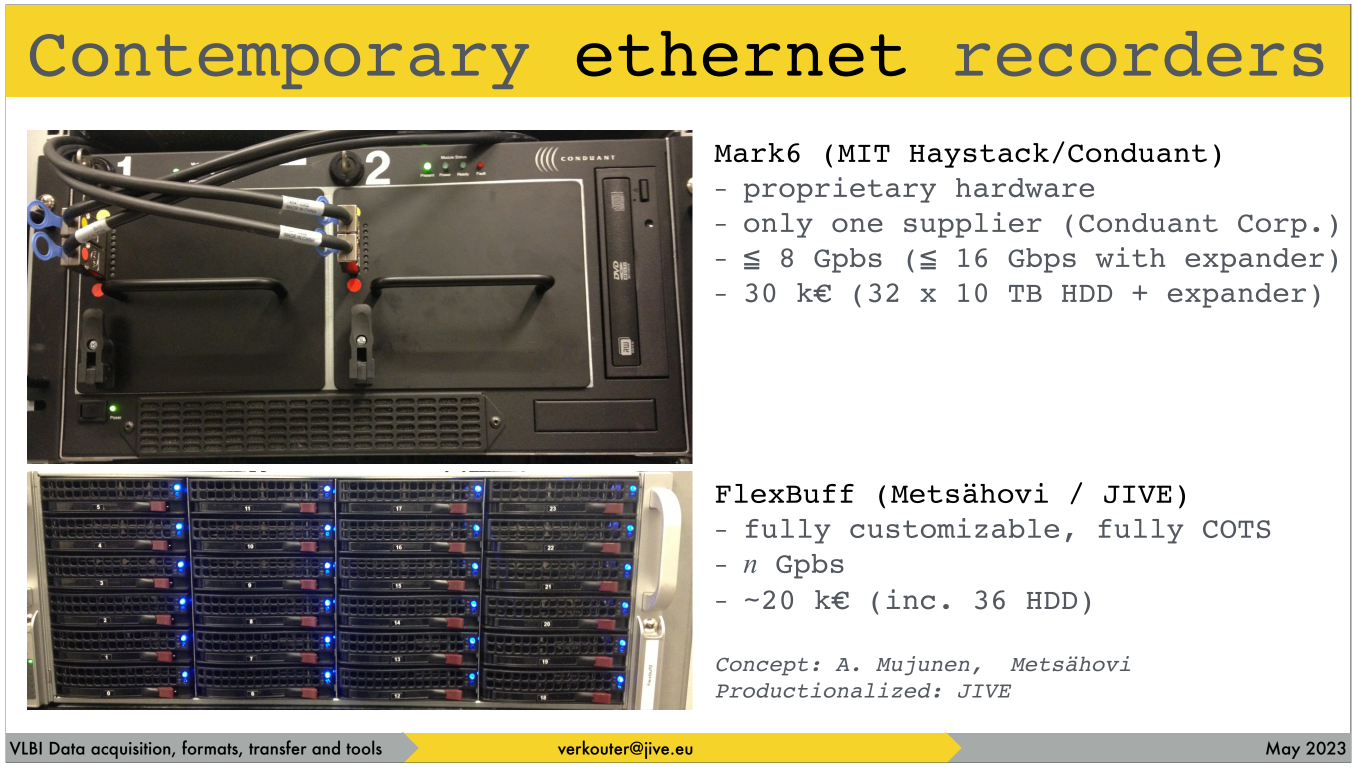
The thing to realize is that conceptually they are the same. They're all [click] _ethernet_ recorders. They consist of [click] a number of CPUs, [click] a large amount of RAM, and [click] a lot of disks. The operation is simple: [click] packets are captured from the network into memory and then [click] in the background scattered over the available disks.

Contemporary ethernet recorders



Mark6 (MIT Haystack/Conduant)
- proprietary hardware
- only one supplier (Conduant Corp.)
- ≦ 8 Gpbs
- 30 k€ (inc. 32 x 10 TB HDD)

the Mark6 is commercially available as a joint MIT Haystack/Conduant development



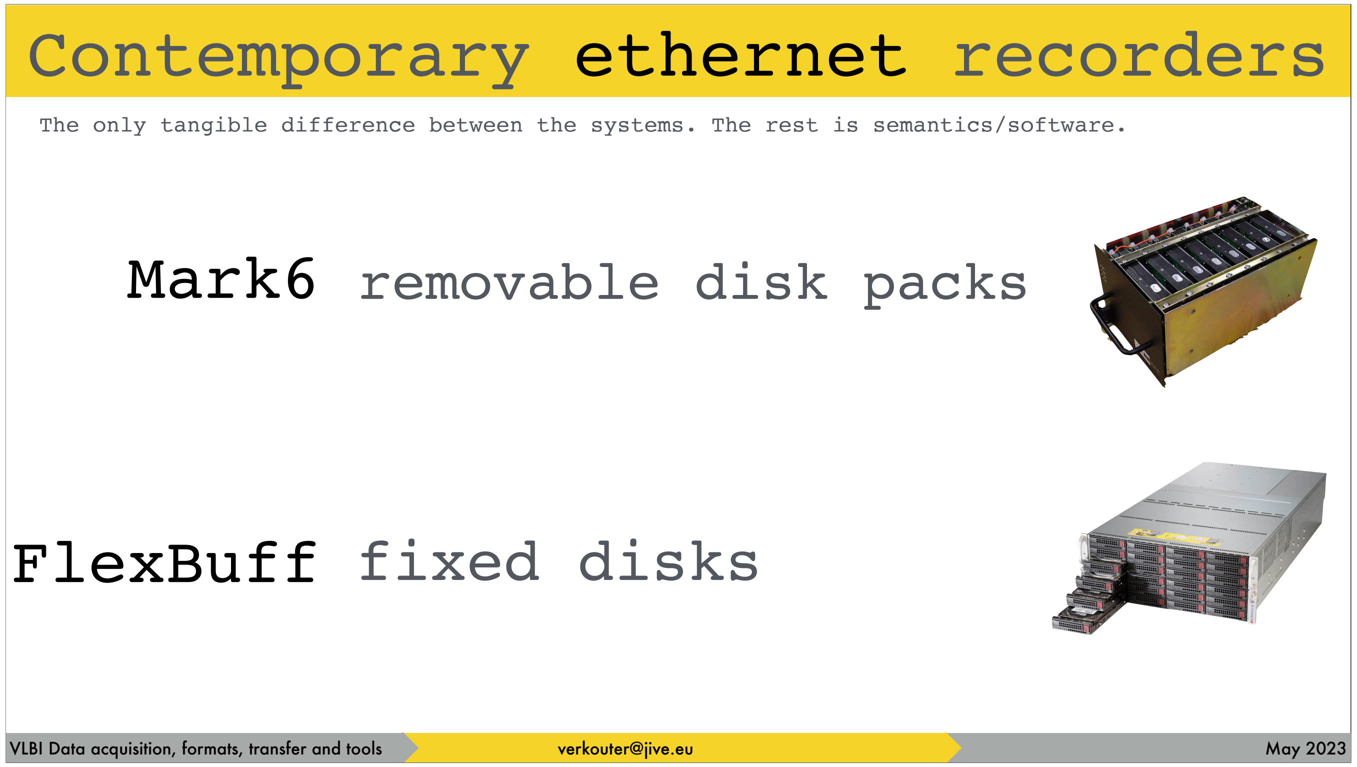
Contemporary ethernet recorders

Mark6 (MIT Haystack/Conduant)
- proprietary hardware
- only one supplier (Conduant Corp.)
- ≦ 8 Gpbs (≦ 16 Gbps with expander)
- 30 k€ (32 x 10 TB HDD + expander)

FlexBuff (Metsähovi / JIVE)
- fully customizable, fully COTS
- *n* Gpbs
- ~20 k€ (inc. 36 HDD)

Concept: A. Mujunen, Metsähovi
Productionalized: JIVE

whilst FlexBuff is a fully customizable off the shelf solution. A concept by Ari Mujunen and put into production by JIVE.



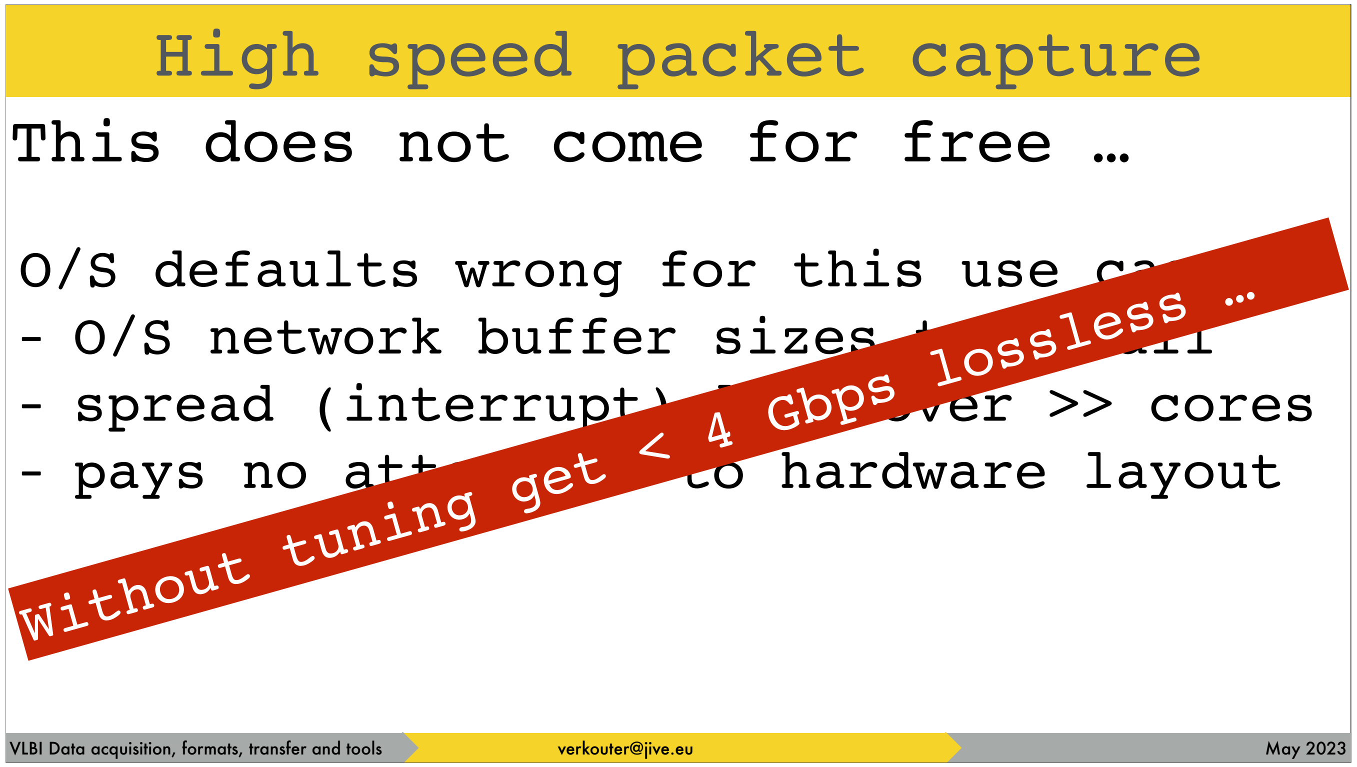
Contemporary ethernet recorders

The only tangible difference between the systems. The rest is semantics/software.

Mark6 removable disk packs

FlexBuff fixed disks

The only real difference between the two machines is the fact that Mark6 [click] has removable disk packs and the [click] flexbuff doesn’t. This has consequences for shipping the data, obviously, but we’ll get to that later.



High speed packet capture

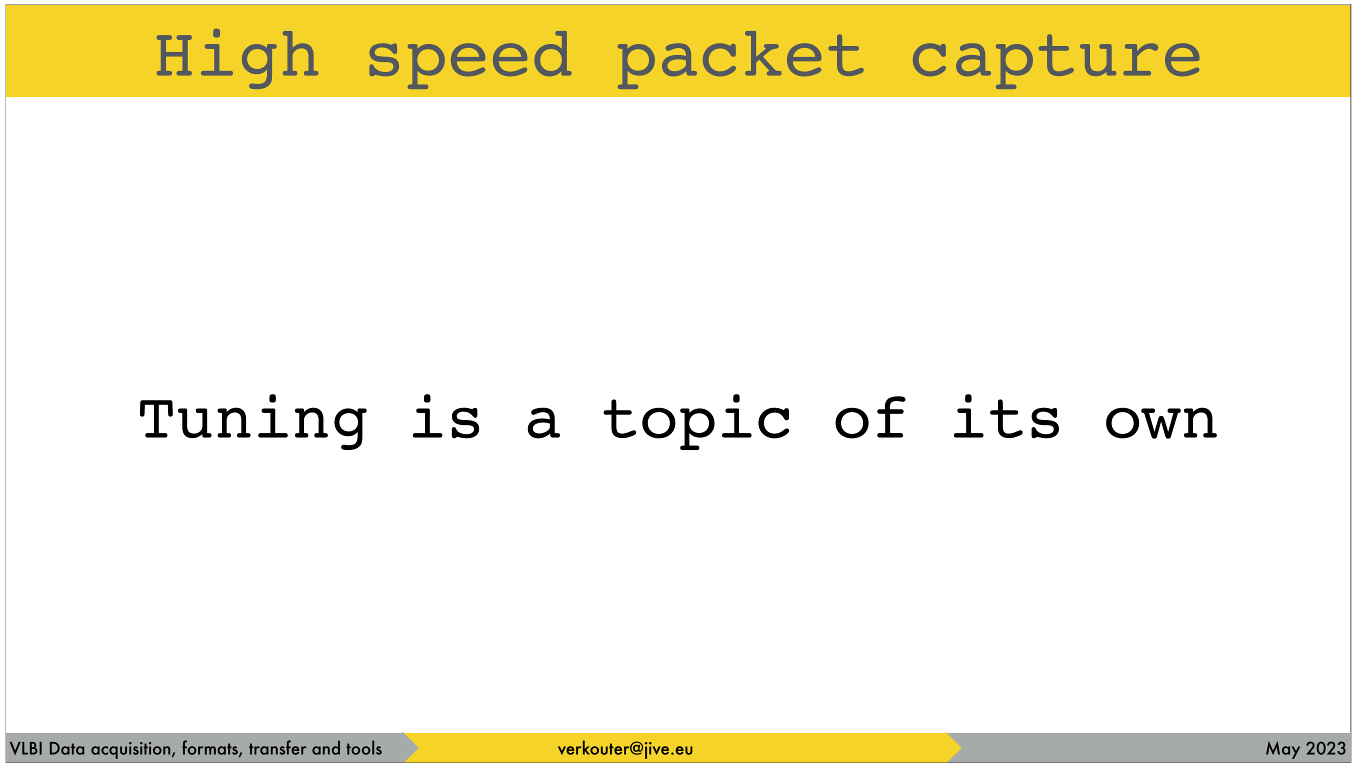
This does not come for free …

O/S defaults wrong for this use ca

- O/S network buffer sizes t all
- spread (interrupt) l ver >> cores
- pays no att to hardware layout

Without tuning get < 4 Gbps lossless …

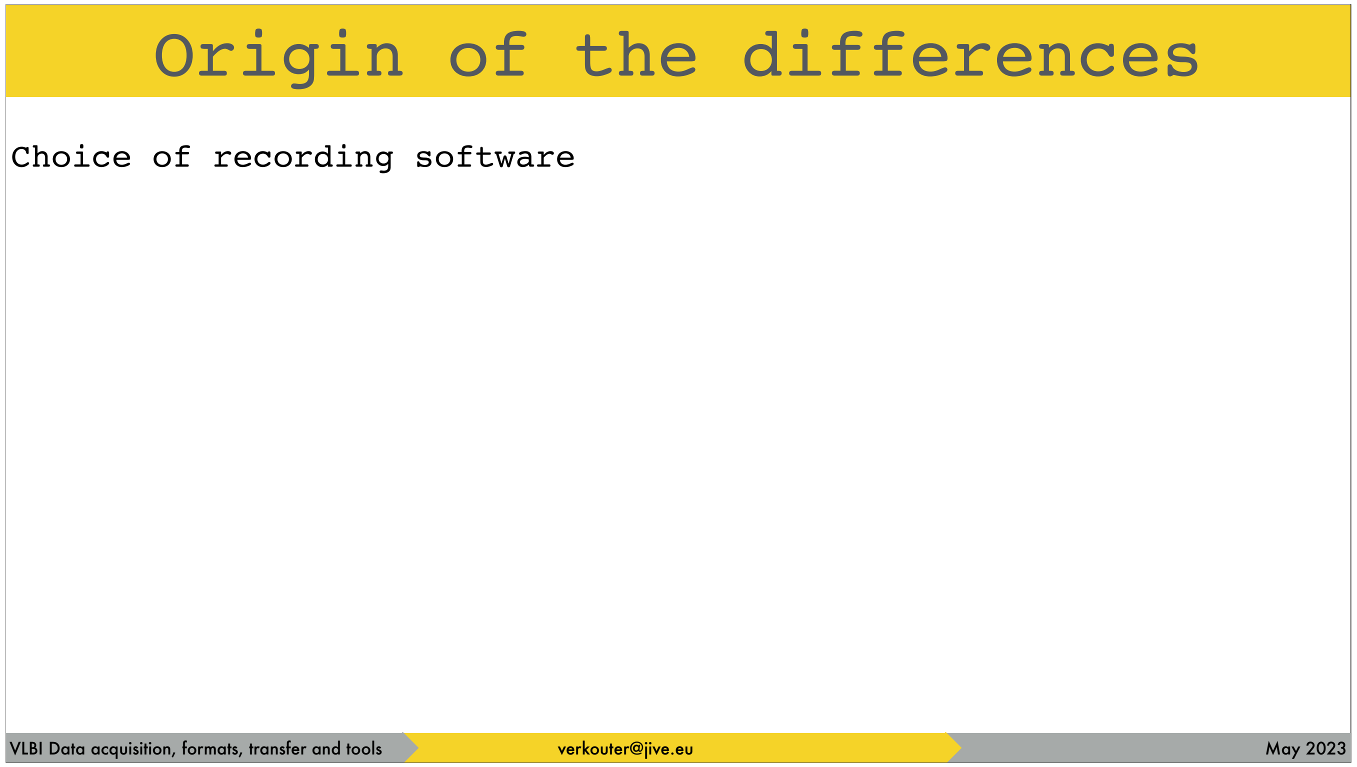
So, both systems try to solve the following problem: high speed packet capture from the network. [click] This does not come for free!!! And more specifically - even on a modern machine [click] you won't get beyond 4 Gpbs without tuning



High speed packet capture

Tuning is a topic of its own

We’ll get to more details about the tuning later on, after we’ve introduced the recorders a bit first.



Origin of the differences

Choice of recording software

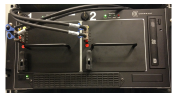
The biggest observable differences are caused by the actual choice of recording software. So what options exist?

Origin of the differences

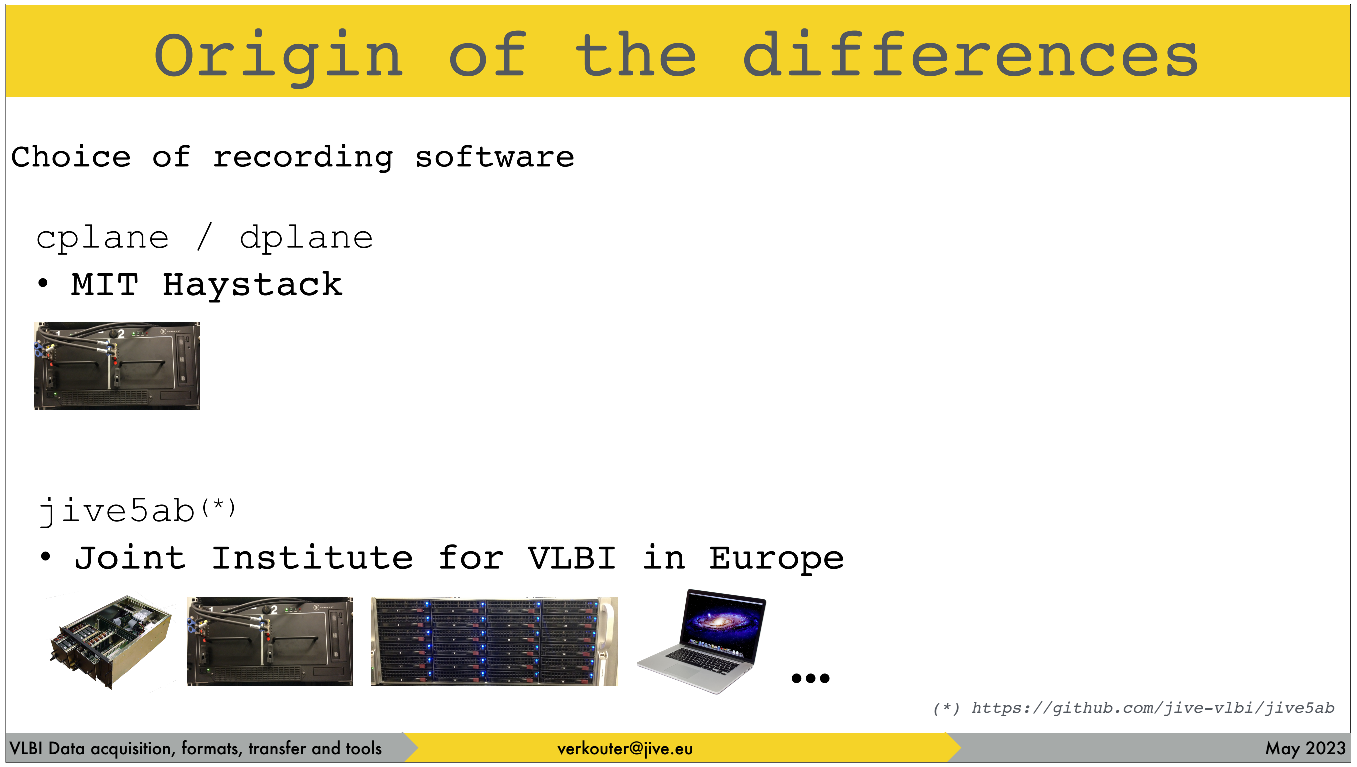
Choice of recording software

cplane / dplane

- MIT Haystack



The Mark6 comes preinstalled with the pair of programs called cplane and dplane



Origin of the differences

Choice of recording software

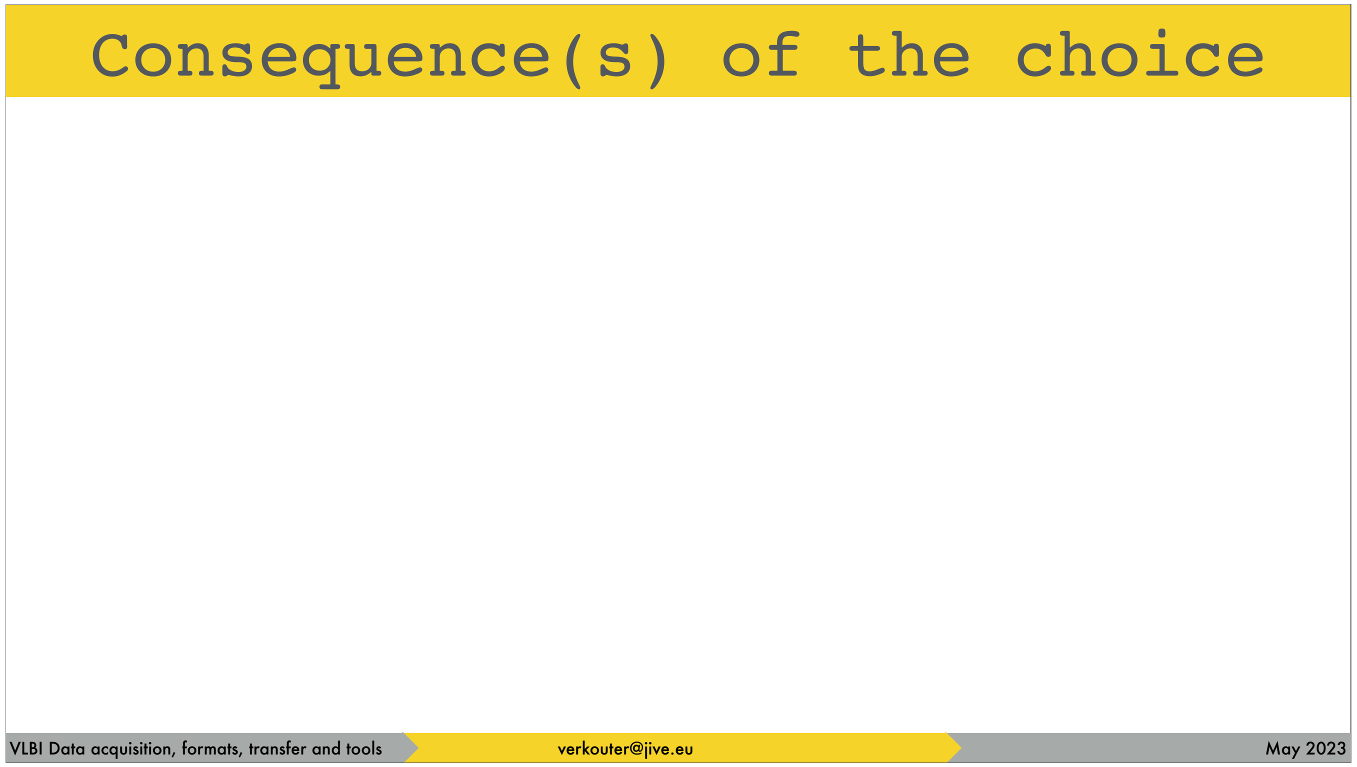
cplane / dplane
- MIT Haystack

jive5ab[*]
- Joint Institute for VLBI in Europe

…

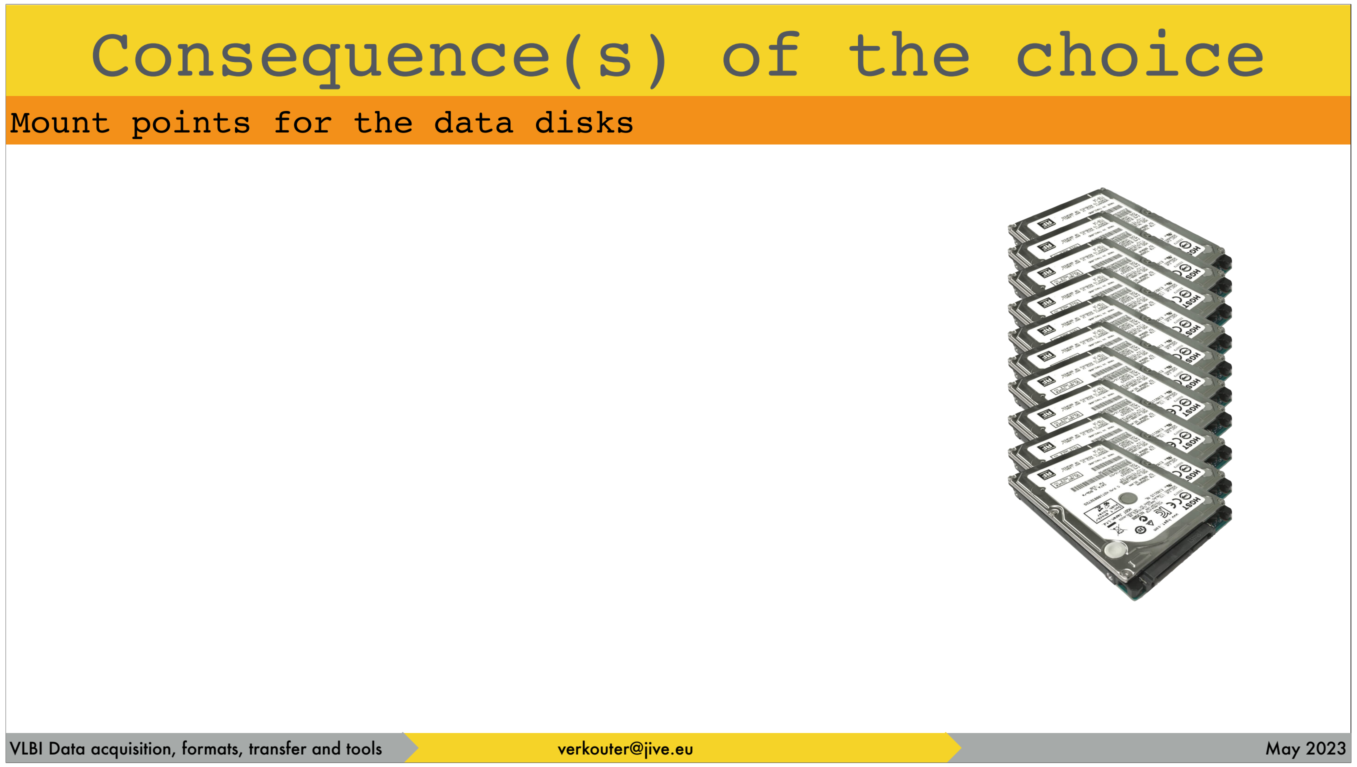
[*] https://github.com/jive-vlbi/jive5ab

The jive5ab software runs on all Mark5, Mark6, flexbuff and BSD style operating systems such as Mac OSX.



Consequence(s) of the choice

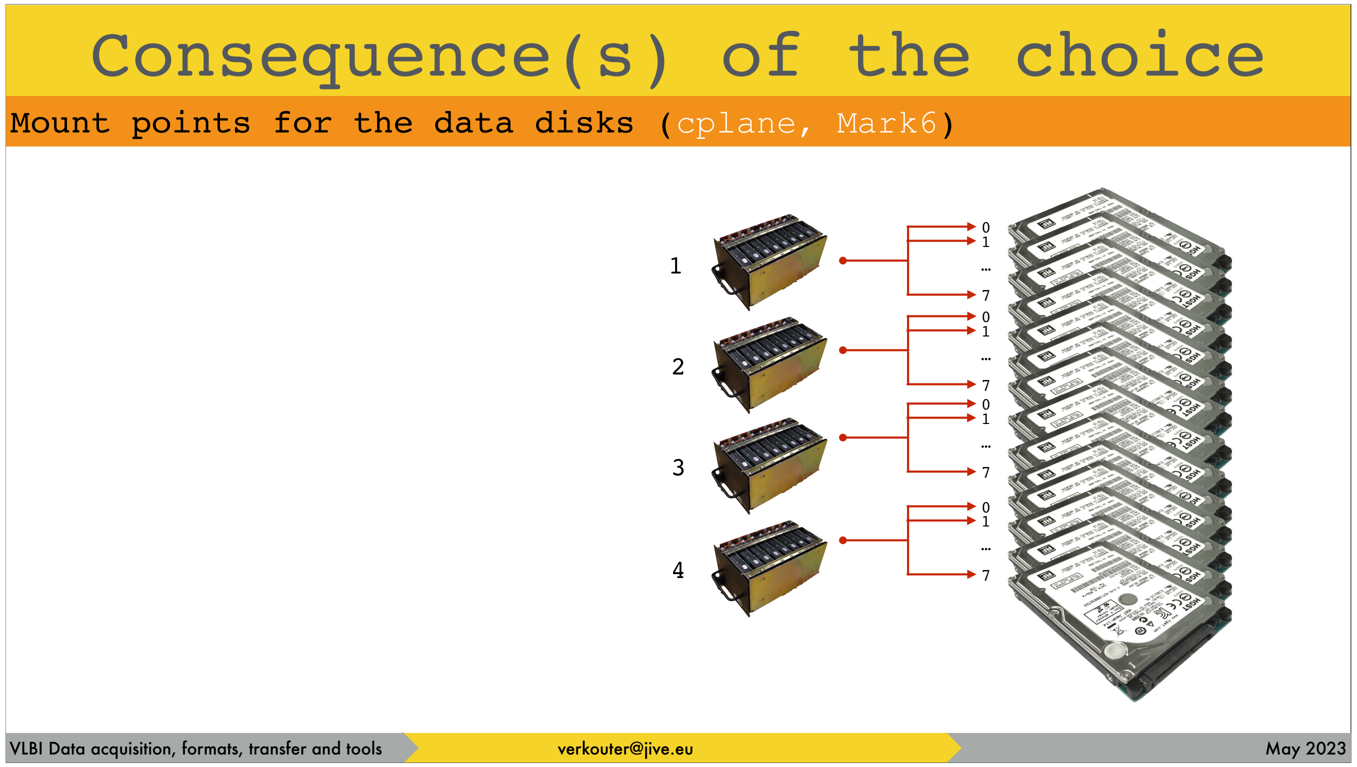
Choosing either has consequences and it is important to know what they are



Consequence(s) of the choice

Mount points for the data disks

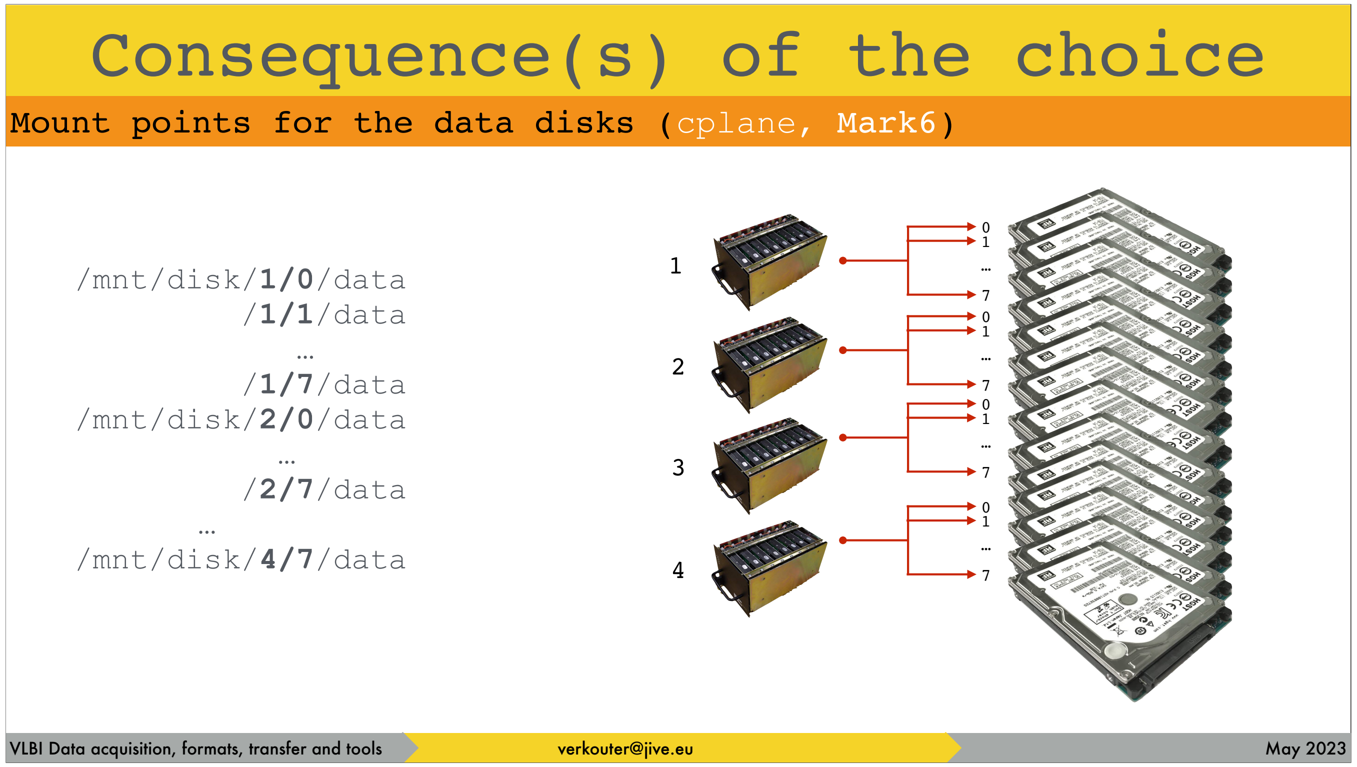
For example the storage. Since the recorders are just Linux or UNIX machines, the hard disks are made visible to the operating system under mount points.



Consequence(s) of the choice

Mount points for the data disks (cplane, Mark6)

On the Mark6 the cplane software expects the mount points to be organized per disk module

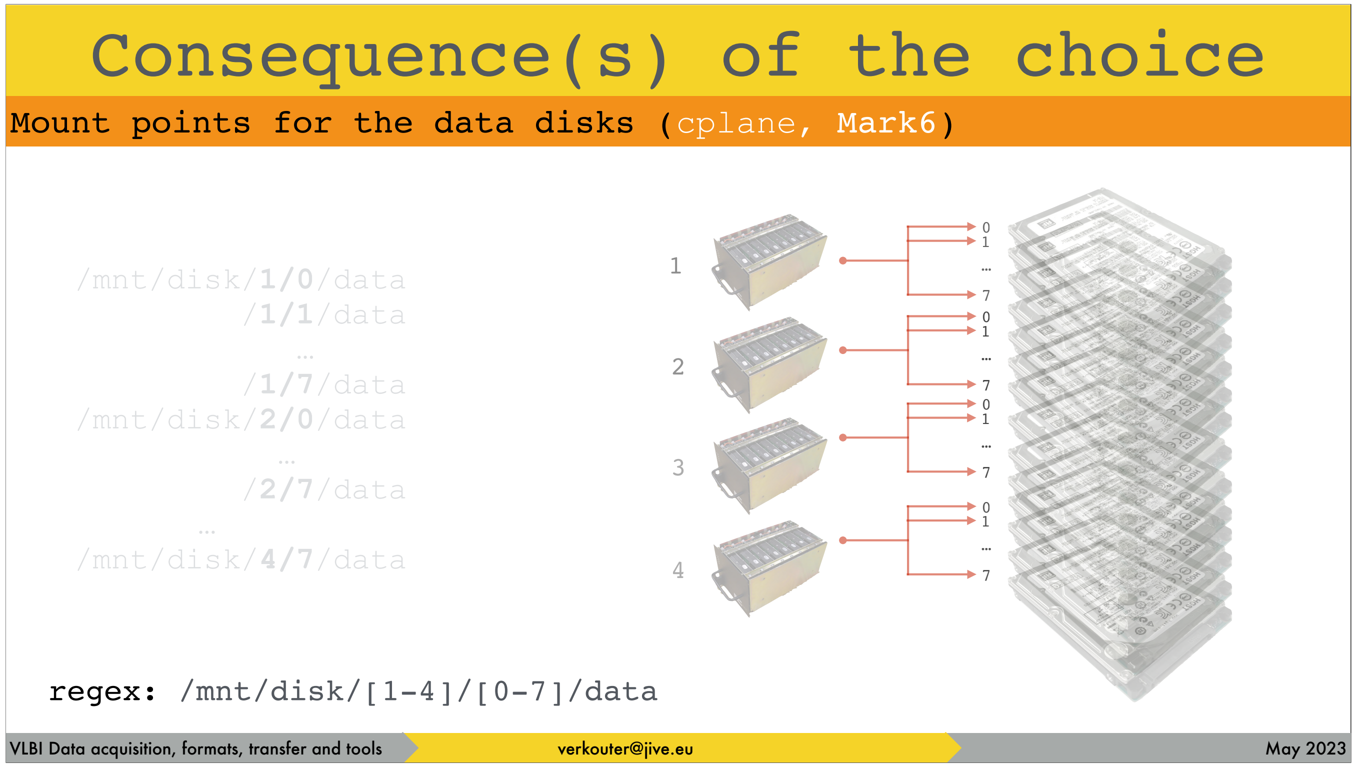


Consequence(s) of the choice

Mount points for the data disks (cplane, Mark6)

```
/mnt/disk/1/0/data
          /1/1/data
             …
          /1/7/data
/mnt/disk/2/0/data
             …
          /2/7/data
       …
/mnt/disk/4/7/data
```

and thus when cplane mounts modules they appear in the operating system as follows



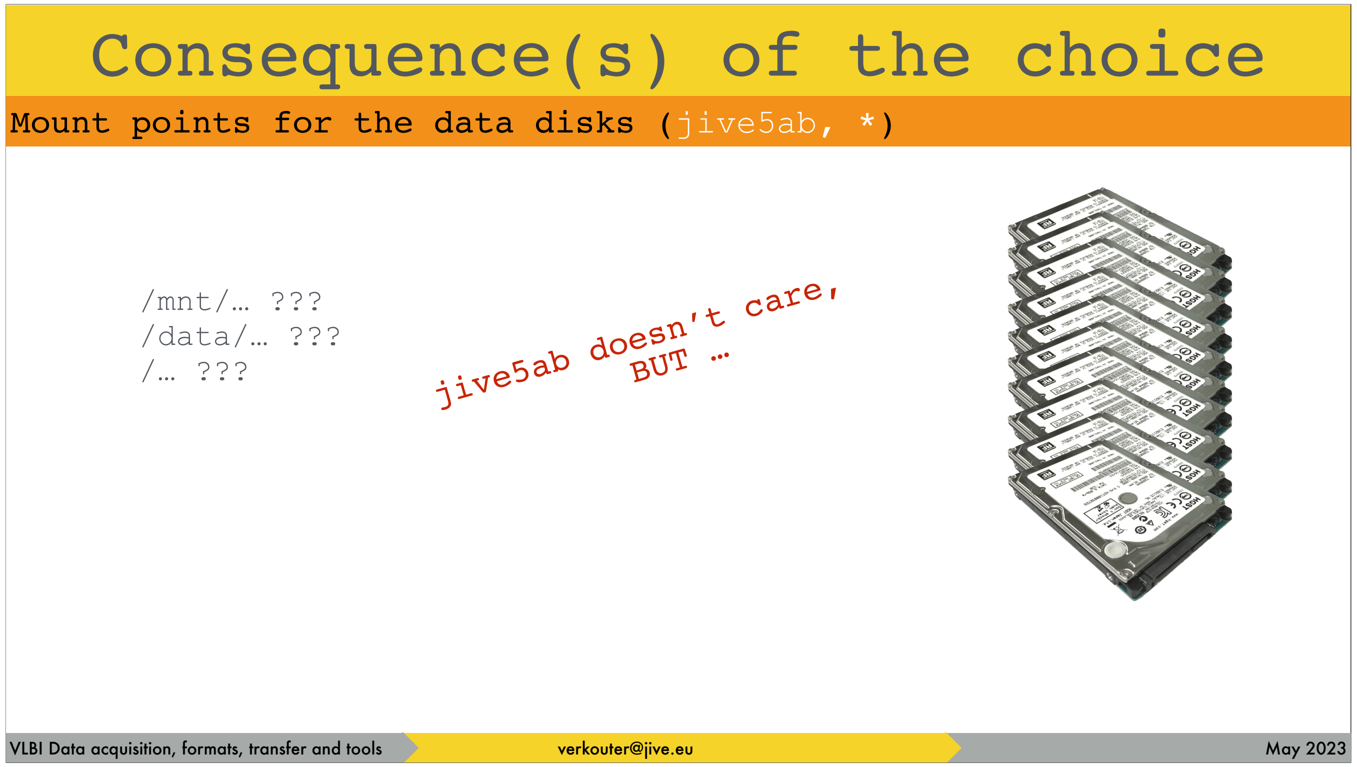
Consequence(s) of the choice

Mount points for the data disks (cplane, Mark6)

```
/mnt/disk/1/0/data
          /1/1/data
            …
          /1/7/data
/mnt/disk/2/0/data
            …
          /2/7/data
      …
/mnt/disk/4/7/data
```

regex: `/mnt/disk/[1-4]/[0-7]/data`

which can be summarized as this regular expression



Consequence(s) of the choice

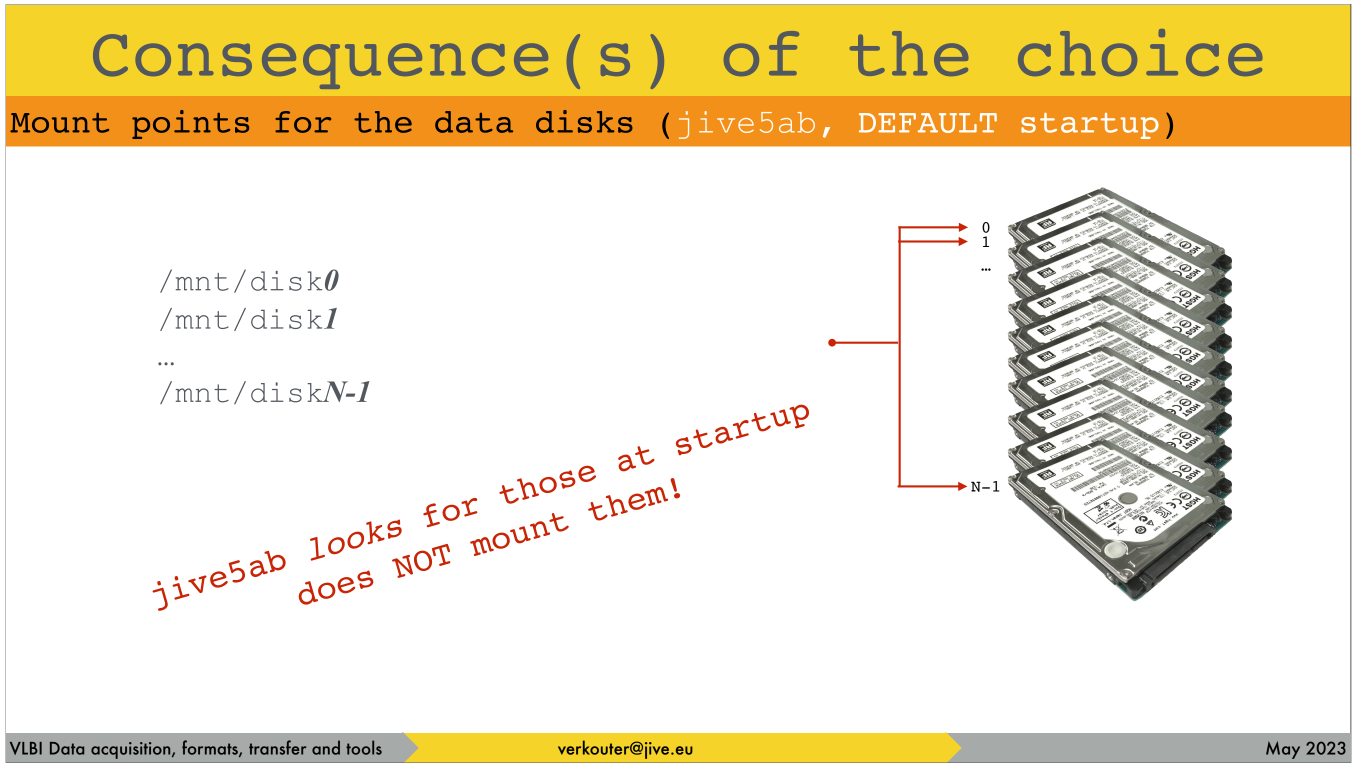
Mount points for the data disks (jive5ab, *)

```
/mnt/… ???
/data/… ???
/… ???
```

jive5ab doesn't care,
BUT …

jive5ab, frankly, doesn’t care where your [click] data disks are mounted.

HOWEVER, there are some compiled in defaults that you can use to make your life easier



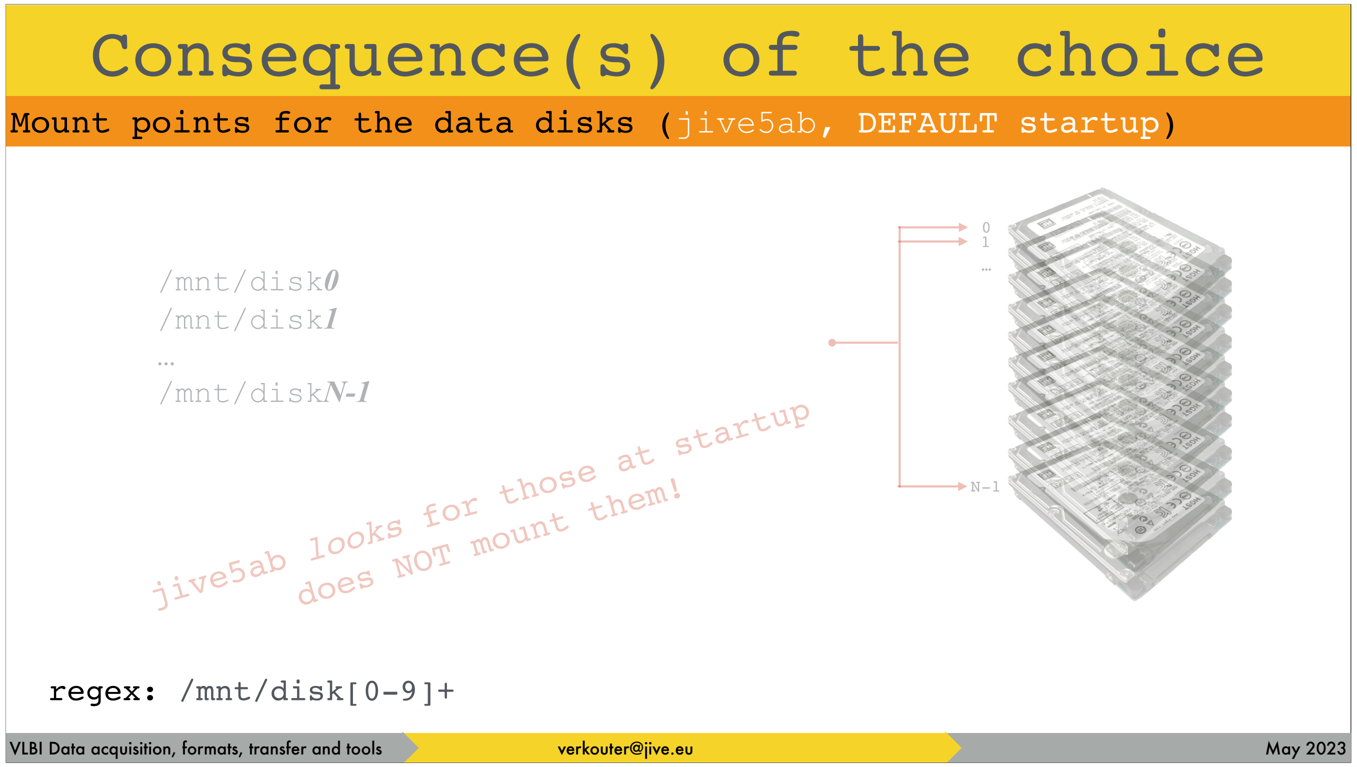
Consequence(s) of the choice

Mount points for the data disks (jive5ab, DEFAULT startup)

```
/mnt/disk0
/mnt/disk1
…
/mnt/diskN-1
```

jive5ab looks for those at startup
does NOT mount them!

By default jive5ab looks for mountpoints called /mnt/disk blah with blah being a number. [click] So if you mount your data disks as this regex



Consequence(s) of the choice

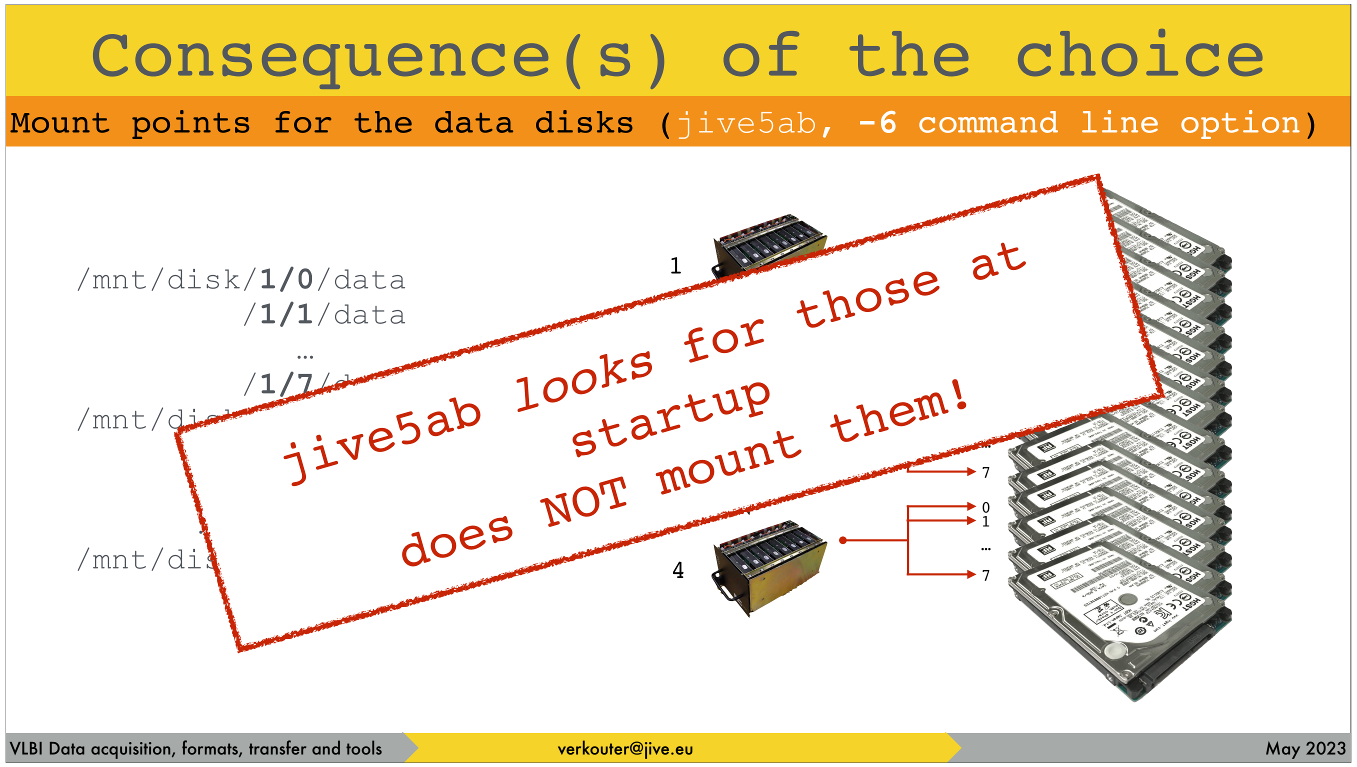
Mount points for the data disks (jive5ab, DEFAULT startup)

```
/mnt/disk0
/mnt/disk1
…
/mnt/diskN-1
```

jive5ab looks for those at startup
does NOT mount them!

regex: `/mnt/disk[0-9]+`

jive5ab will pick them up automatically at startup



Consequence(s) of the choice

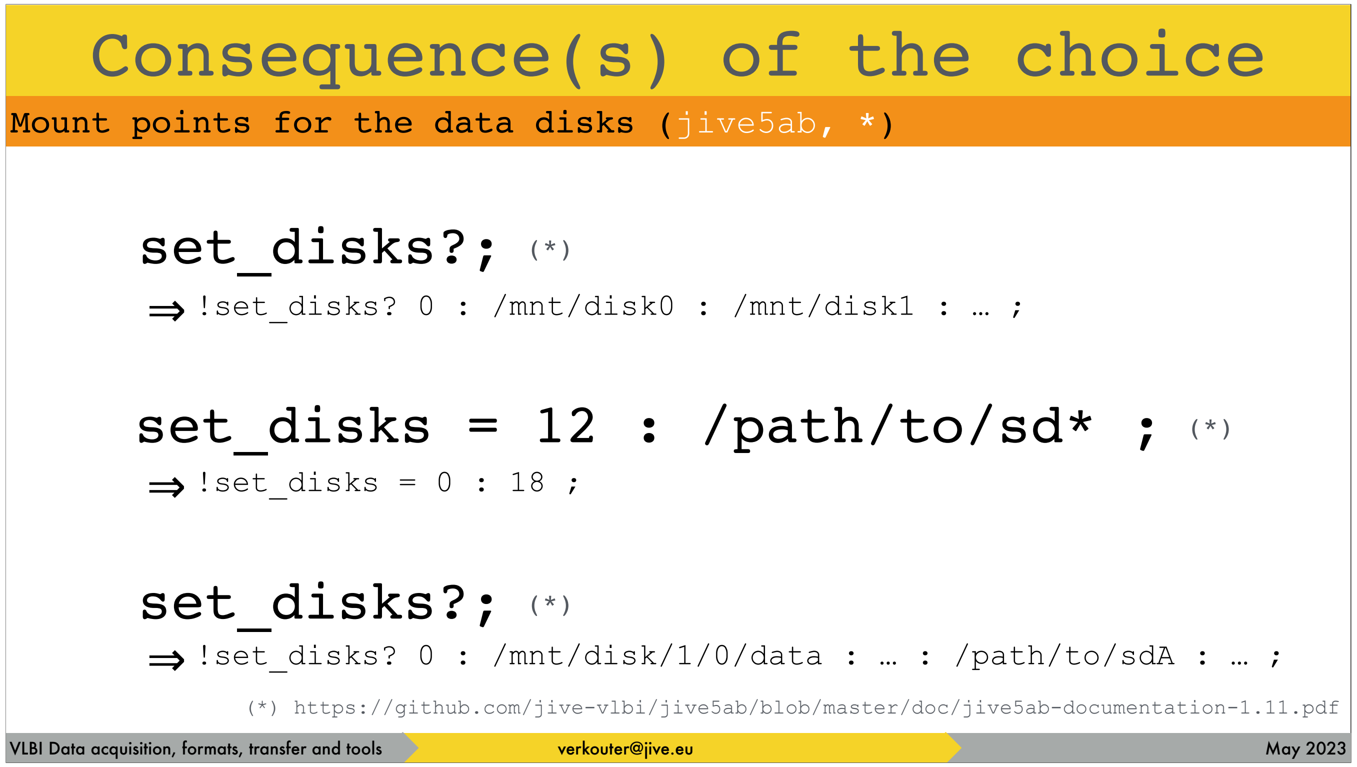
Mount points for the data disks (jive5ab, -6 command line option)

```
/mnt/disk/1/0/data
         /1/1/data
            …
         /1/7/d
/mnt/di

/mnt/di
```

jive5ab looks for those at startup

does NOT mount them!

However, if you pass the “-6” command line option, jive5ab will look for the mark6 modules instead. And I must stress again [click] - jive5ab does NOT mount harddisks, it expects to happen outside the program.



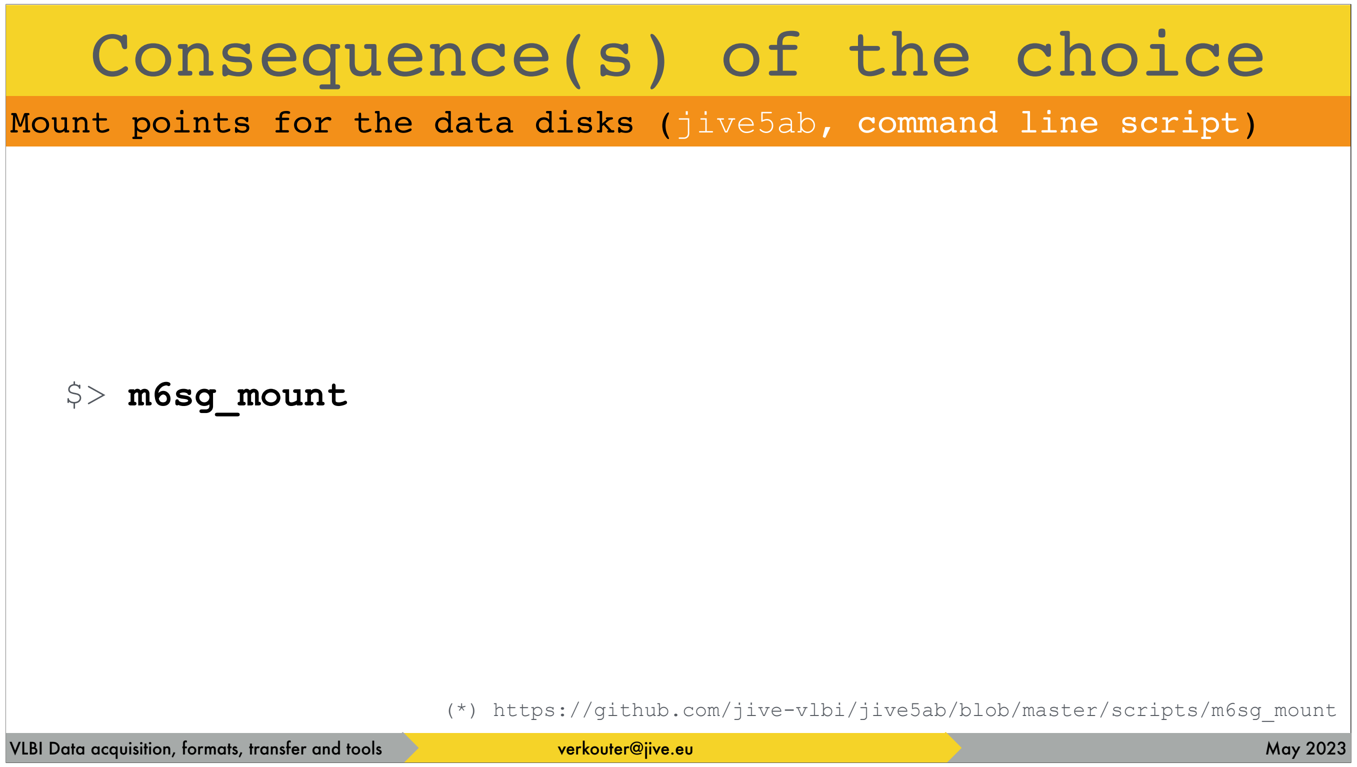
Consequence(s) of the choice

Mount points for the data disks (jive5ab, *)

```
set_disks?; (*)
⇒ !set_disks? 0 : /mnt/disk0 : /mnt/disk1 : … ;

set_disks = 12 : /path/to/sd* ; (*)
⇒ !set_disks = 0 : 18 ;

set_disks?; (*)
⇒ !set_disks? 0 : /mnt/disk/1/0/data : … : /path/to/sdA : … ;
```

(*) https://github.com/jive-vlbi/jive5ab/blob/master/doc/jive5ab-documentation-1.11.pdf

In jive5ab the disks to record on can be queried and changed [click] at runtime by issueing the `set_disk=…` command.



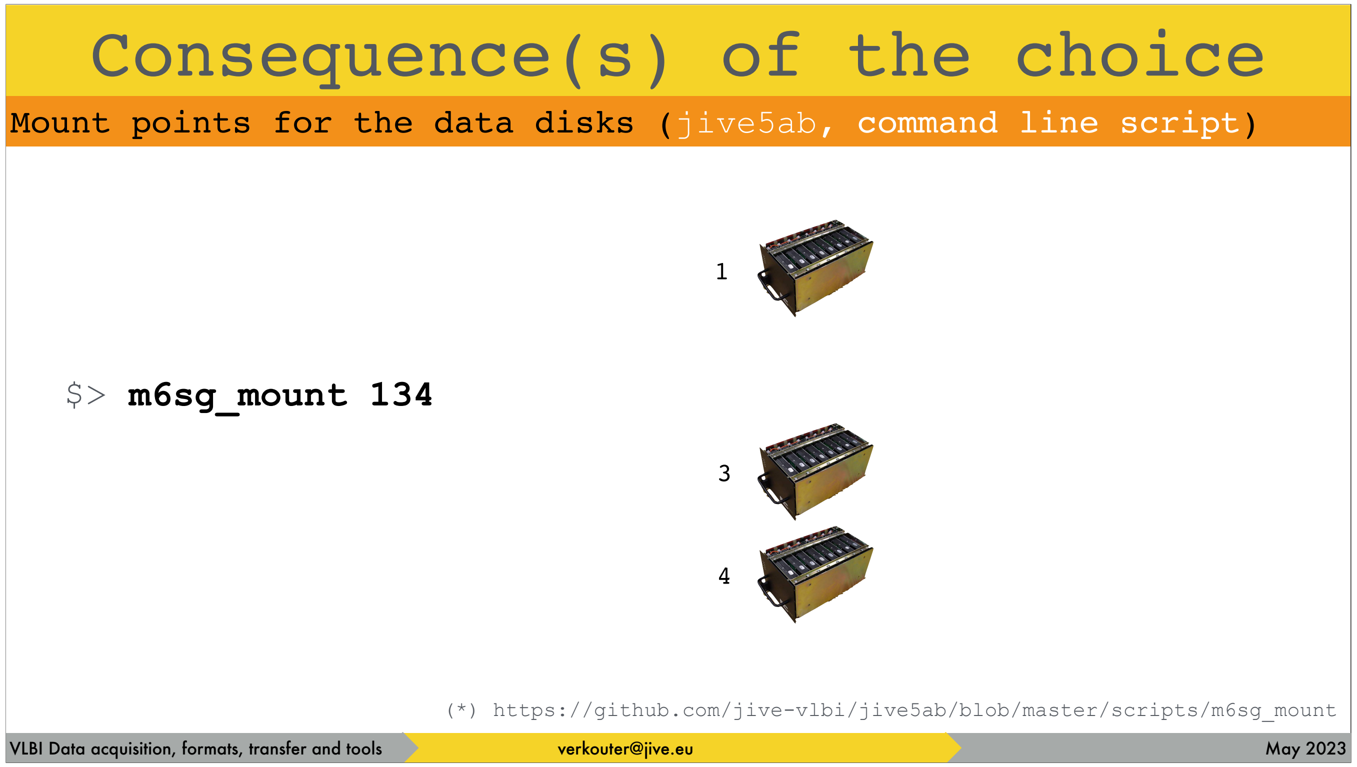
Consequence(s) of the choice

Mount points for the data disks (jive5ab, command line script)

```
$> m6sg_mount
```

(*) https://github.com/jive-vlbi/jive5ab/blob/master/scripts/m6sg_mount

In the jive5ab source code repository there is a command line script m6sg_mount



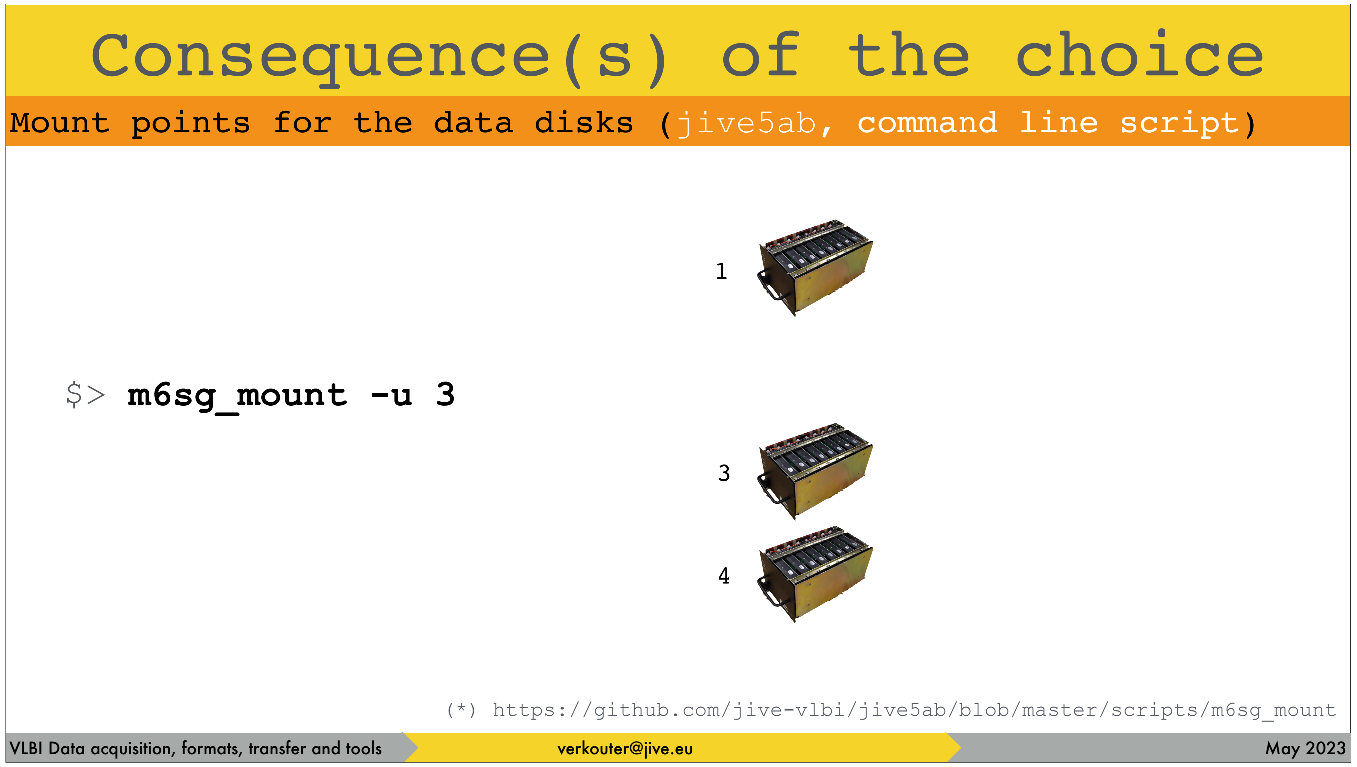
Consequence(s) of the choice

Mount points for the data disks (jive5ab, command line script)

```
$> m6sg_mount 134
```

(*) https://github.com/jive-vlbi/jive5ab/blob/master/scripts/m6sg_mount

that can be used to mount



Consequence(s) of the choice

Mount points for the data disks (jive5ab, command line script)

```
$> m6sg_mount -u 3
```

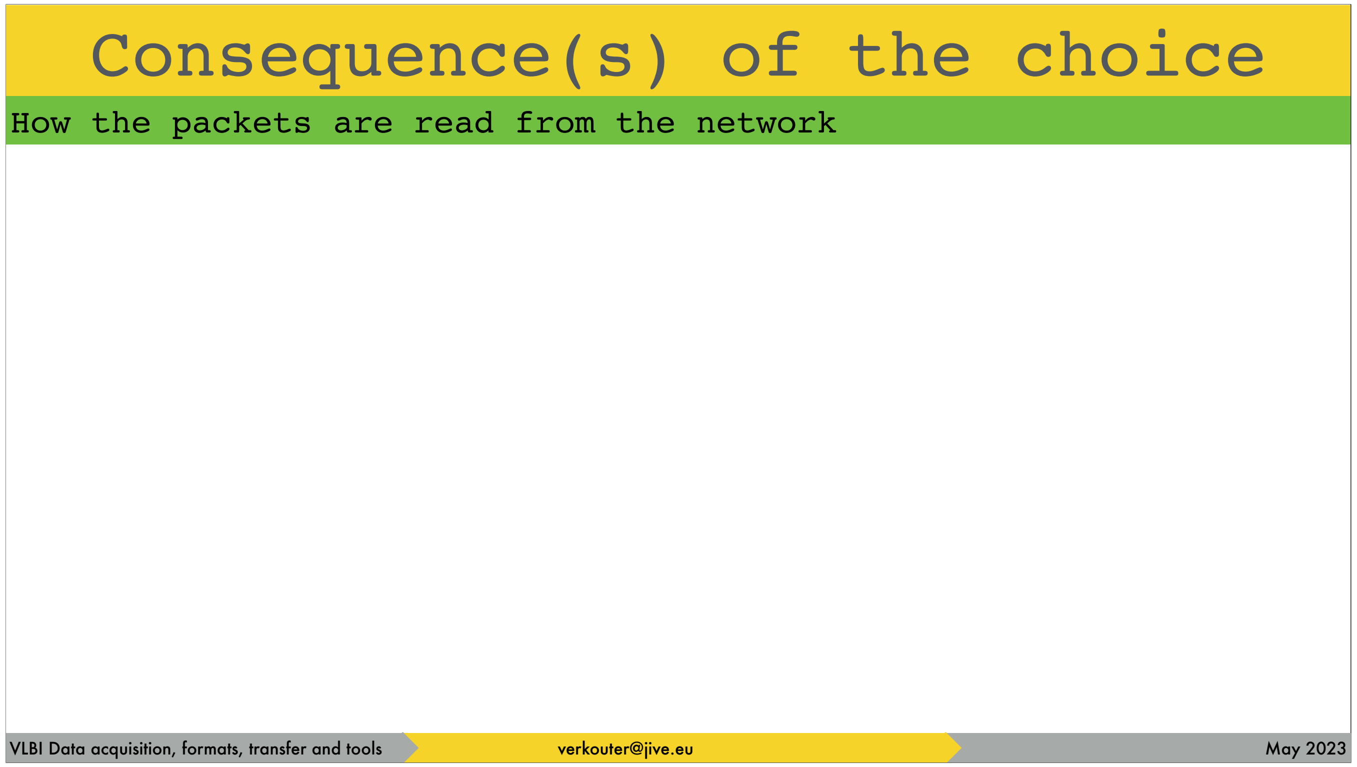
1

3

4

(*) https://github.com/jive-vlbi/jive5ab/blob/master/scripts/m6sg_mount

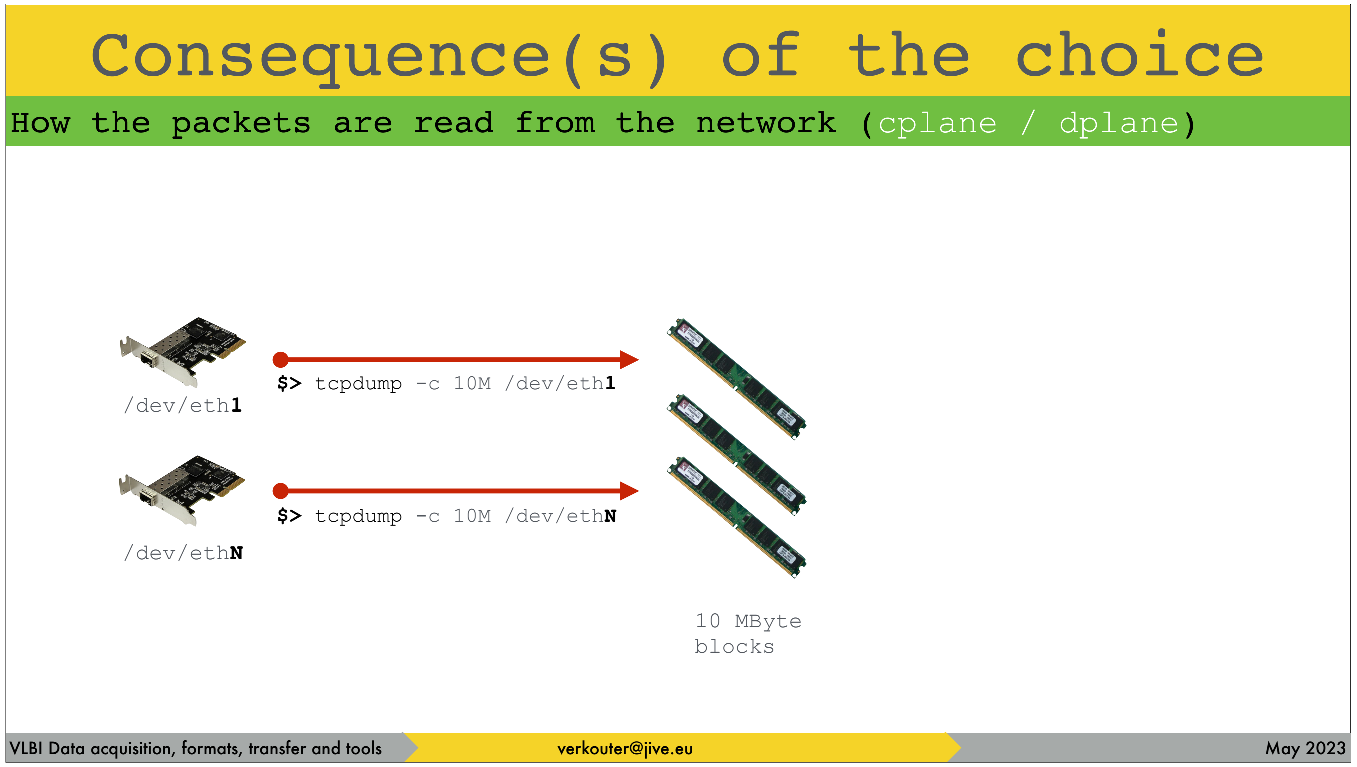
or unmount individual mark6 modules



Consequence(s) of the choice

How the packets are read from the network

Another big difference is how the packets are grabbed from the network



Consequence(s) of the choice

How the packets are read from the network (cplane / dplane)

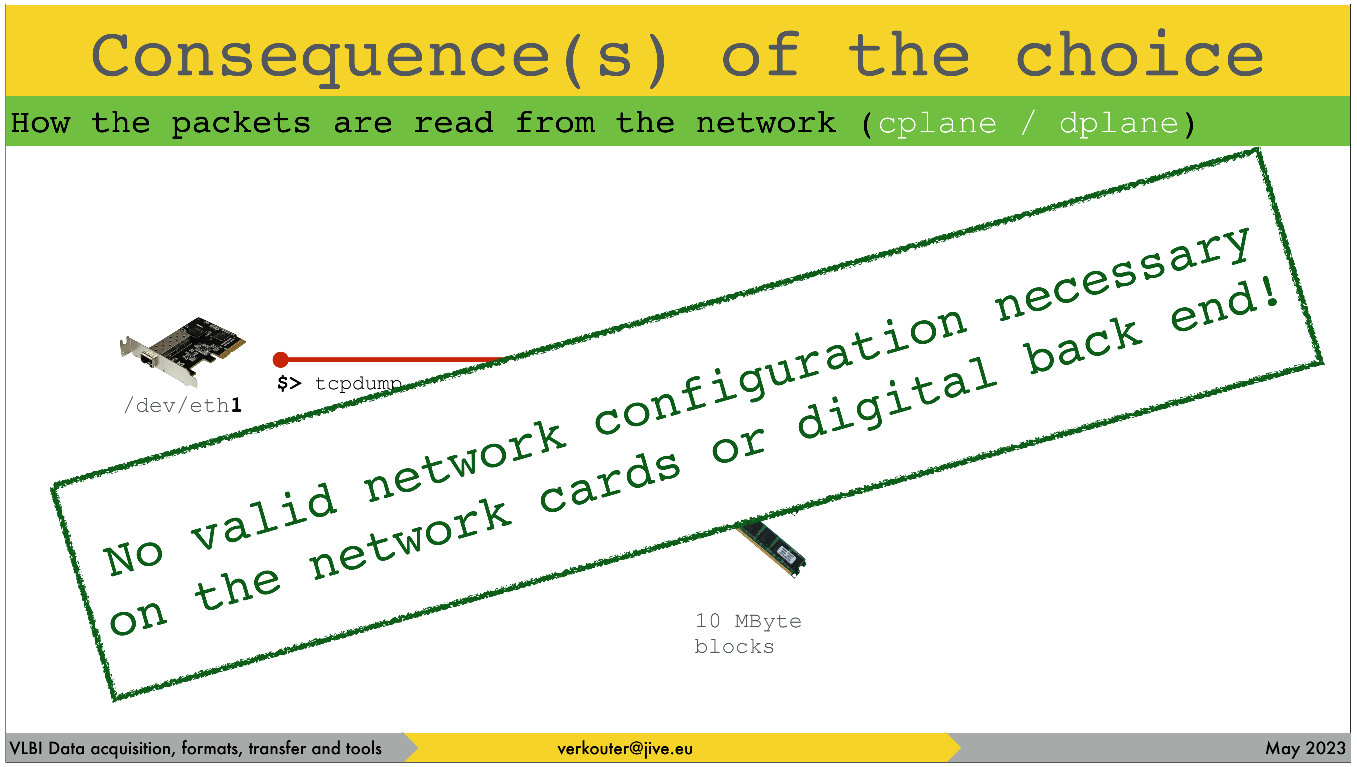
/dev/eth**1**

```
$> tcpdump -c 10M /dev/eth1
```

/dev/eth**N**

```
$> tcpdump -c 10M /dev/ethN
```

The dplane program accumulates frames from the ethernet devices directly into blocks in memory of about 10 MB. [click] It is not unlike running tcpdump on the interface.

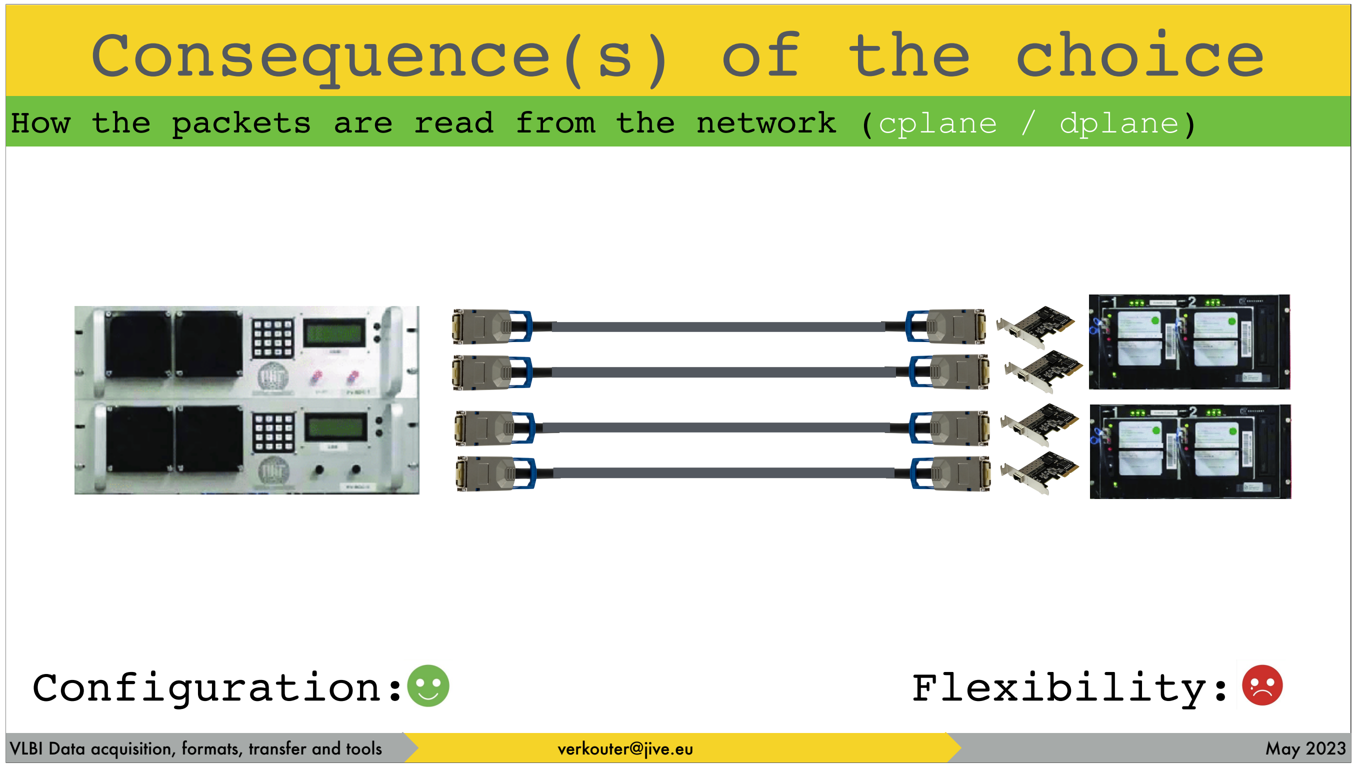


Consequence(s) of the choice

How the packets are read from the network (cplane / dplane)

No valid network configuration necessary on the network cards or digital back end!

The important property is that you don’t need a valid network configuration on ANY of the components in your system.



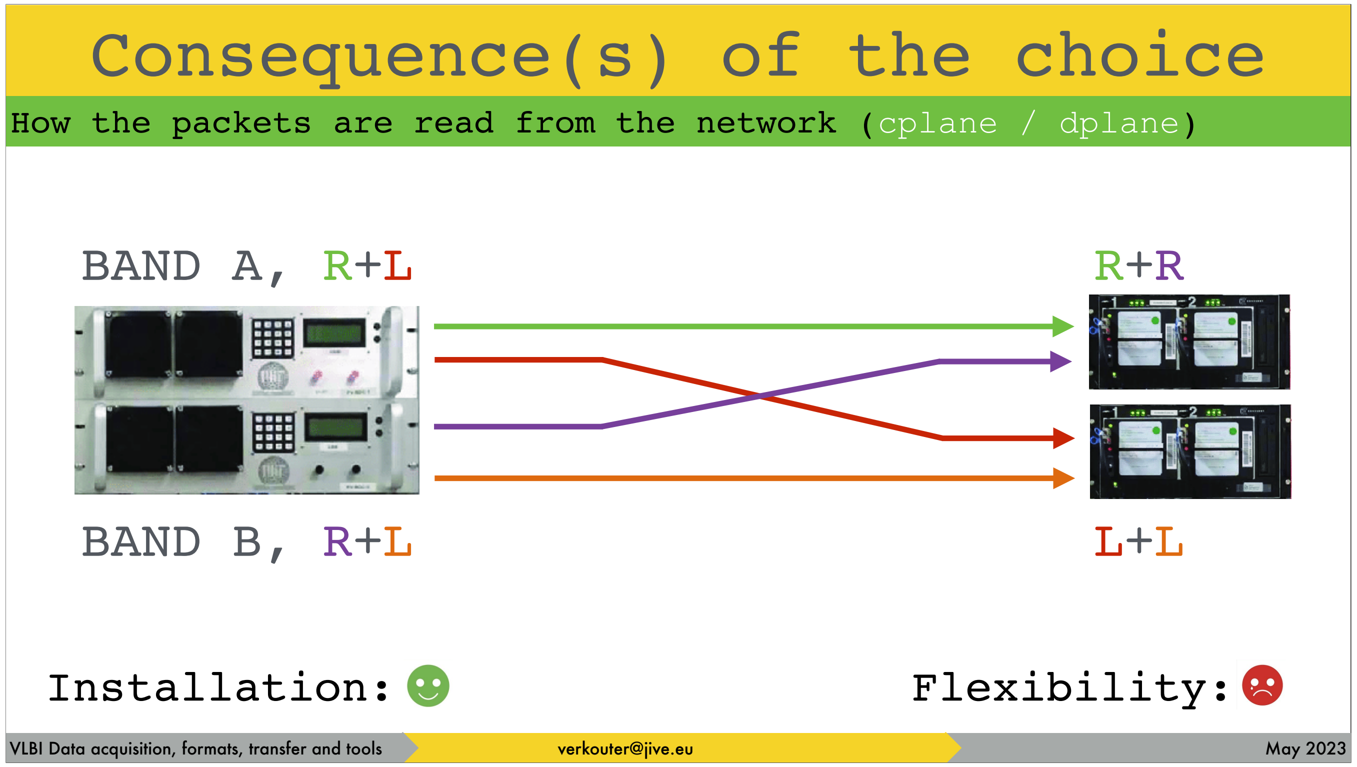
Consequence(s) of the choice

How the packets are read from the network (cplane / dplane)

Configuration: ☺

Flexibility: ☹

The model for cplane/dplane and the Mark6 is this: fixed point to point connections between digital receiver(s) and the network cards in the recorder. So this is VERY easy for installation and configuration



Consequence(s) of the choice

How the packets are read from the network (cplane / dplane)

BAND A, R+L

BAND B, R+L

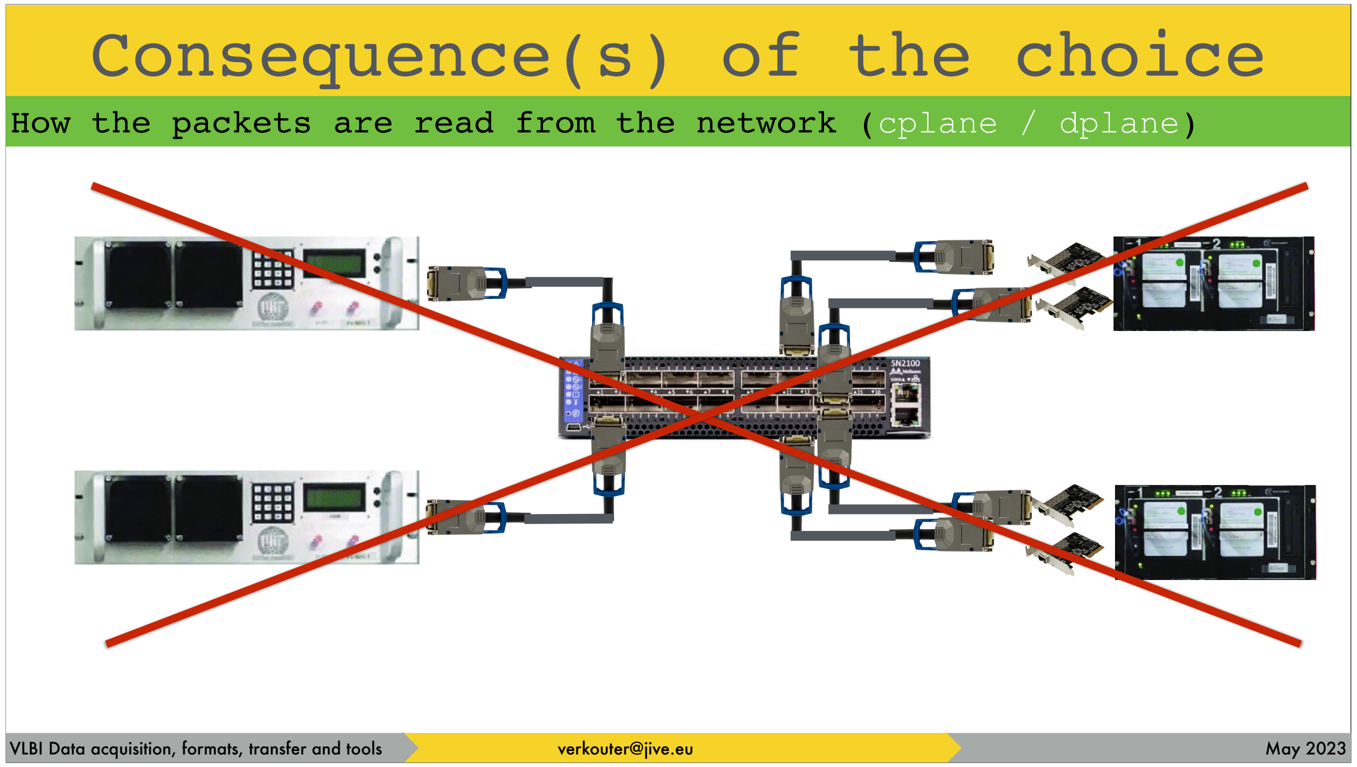
R+R

L+L

Installation: ☺

Flexibility: ☹

but e.g. collecting polarizations from two bands is impossible.

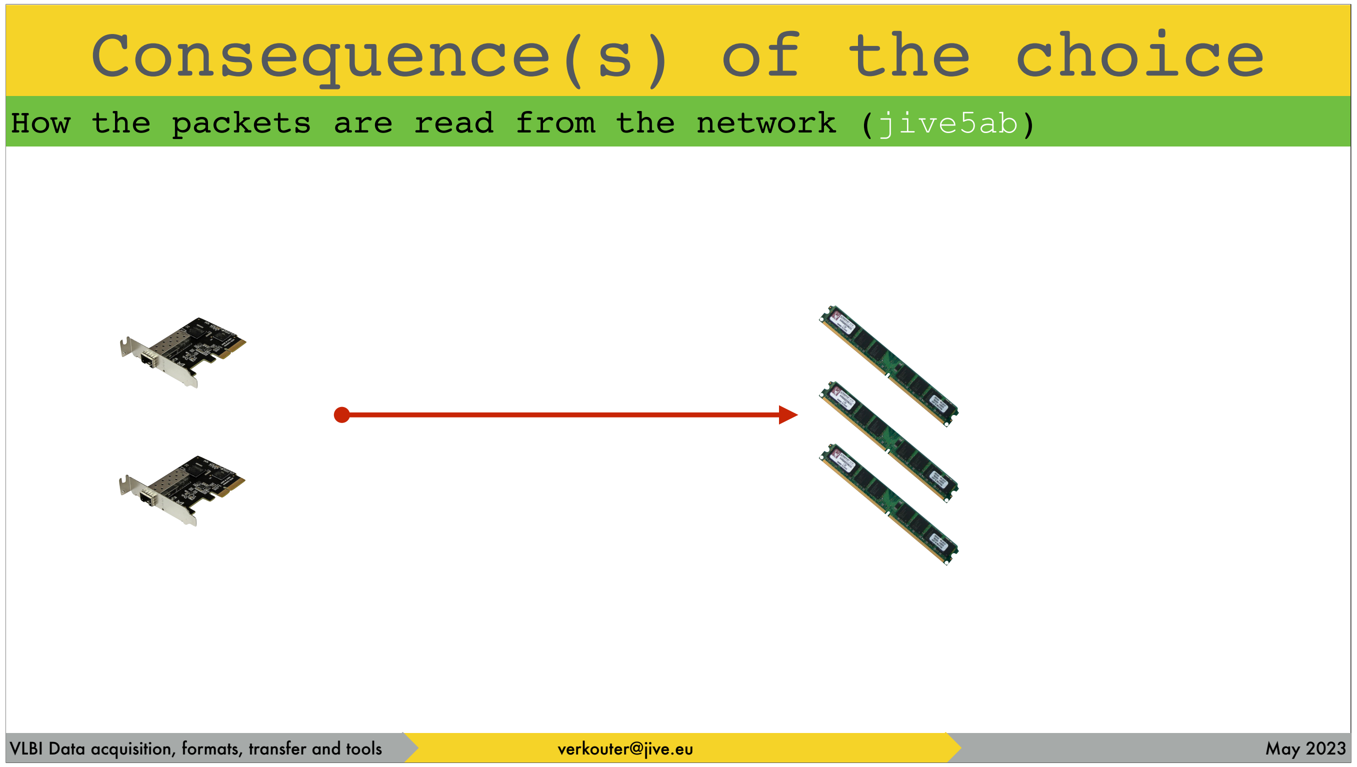


Consequence(s) of the choice

How the packets are read from the network (cplane / dplane)

VLBI Data acquisition, formats, transfer and tools verkouter@jive.eu May 2023

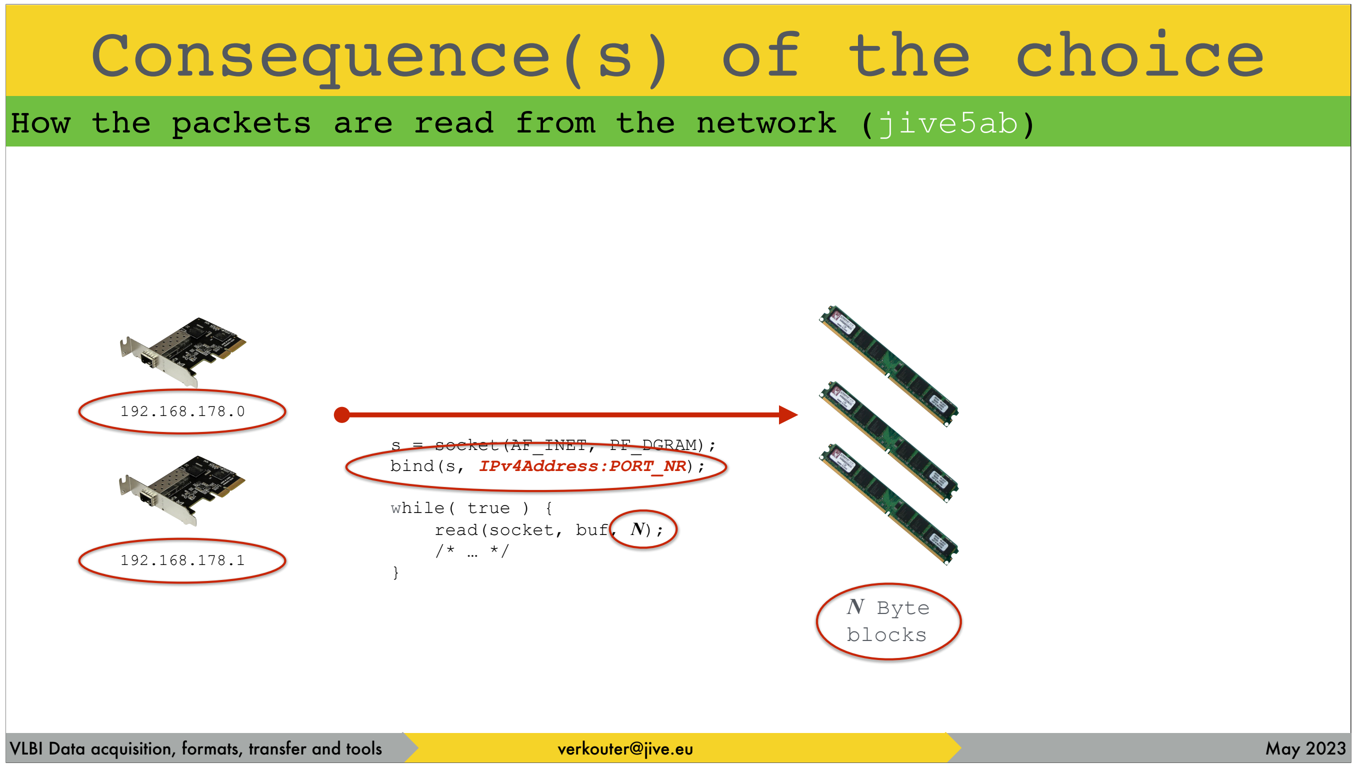
or putting a switch between your backends and recorders [click] is also not possible



Consequence(s) of the choice

How the packets are read from the network (jive5ab)

jive5ab on the other hand does things completely differently



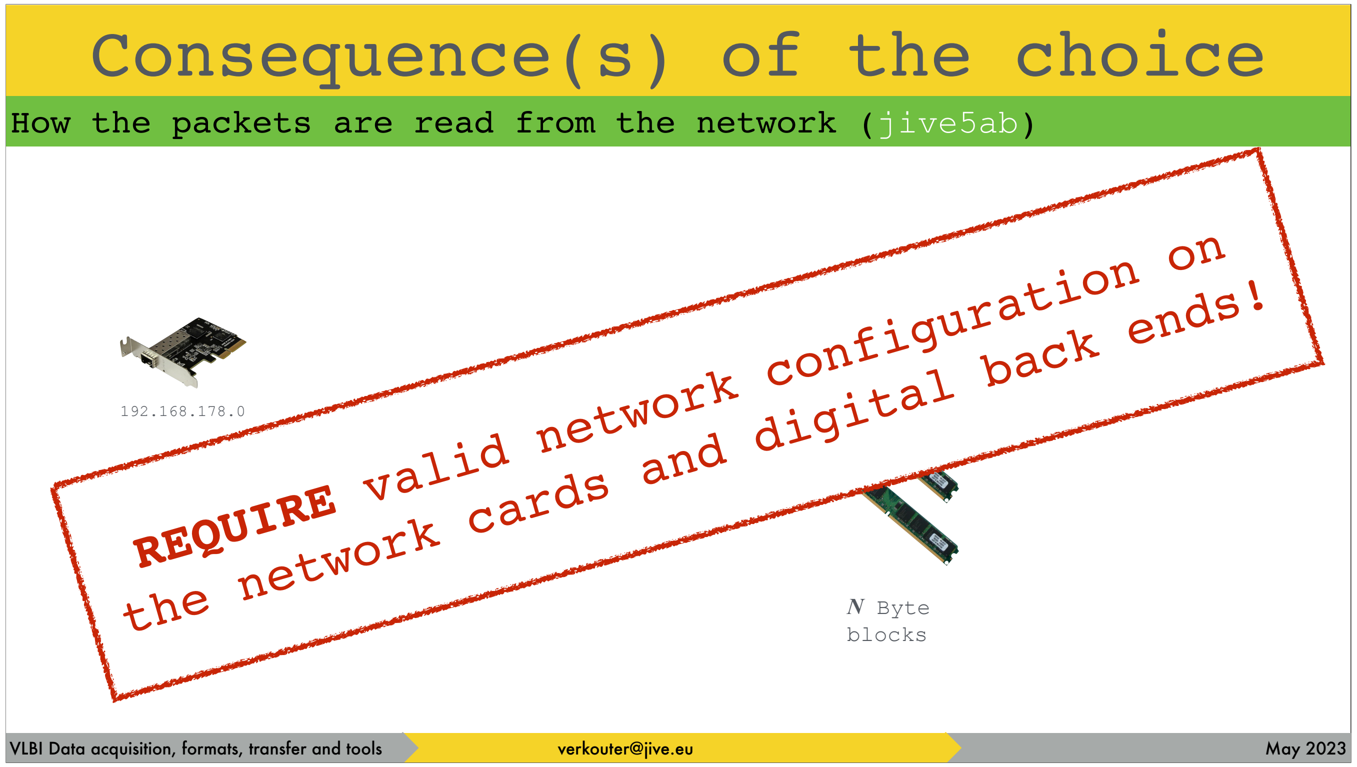
Consequence(s) of the choice

How the packets are read from the network (jive5ab)

```
s = socket(AF_INET, PF_DGRAM);
bind(s, IPv4Address:PORT_NR);

while( true ) {
    read(socket, buf, N);
    /* … */
}
```

It operates at the [click] IP address level.

And in the code it starts listening for data [click] on one (or all) IP addresses and a specified port number for incoming data, [click] and collects this in customizable size blocks.

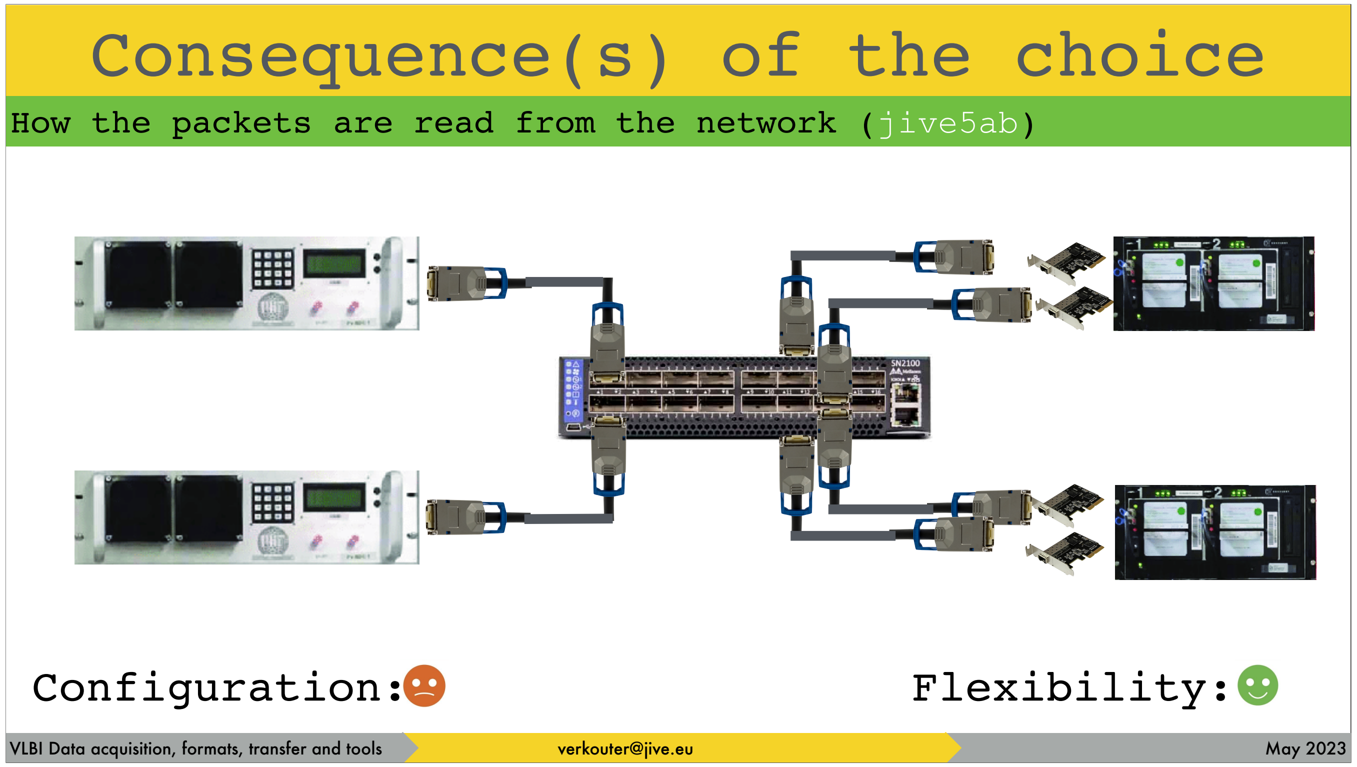


Consequence(s) of the choice

How the packets are read from the network (jive5ab)

REQUIRE valid network configuration on the network cards and digital back ends!

jive5ab requires you to have a valid network configuration on ALL components!



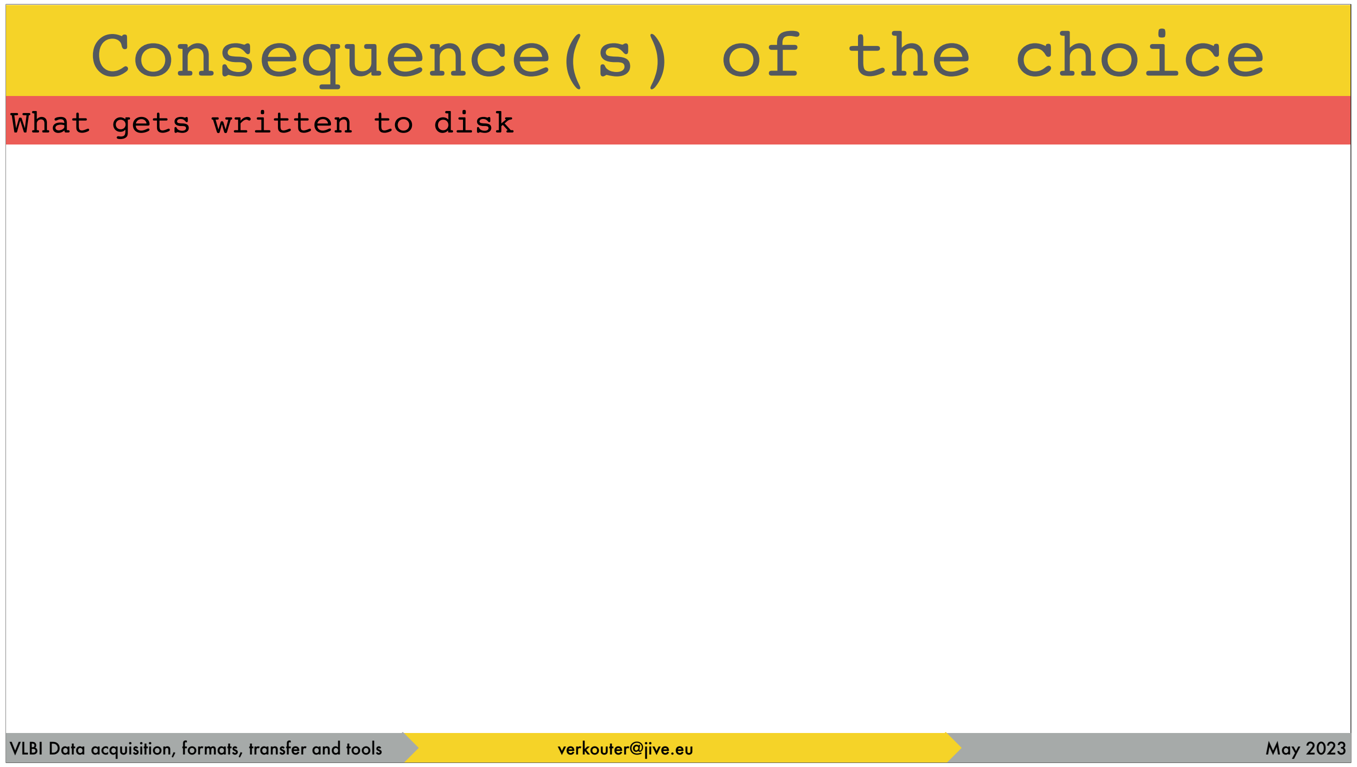
Consequence(s) of the choice

How the packets are read from the network (jive5ab)

Configuration:🙁

Flexibility:🙂

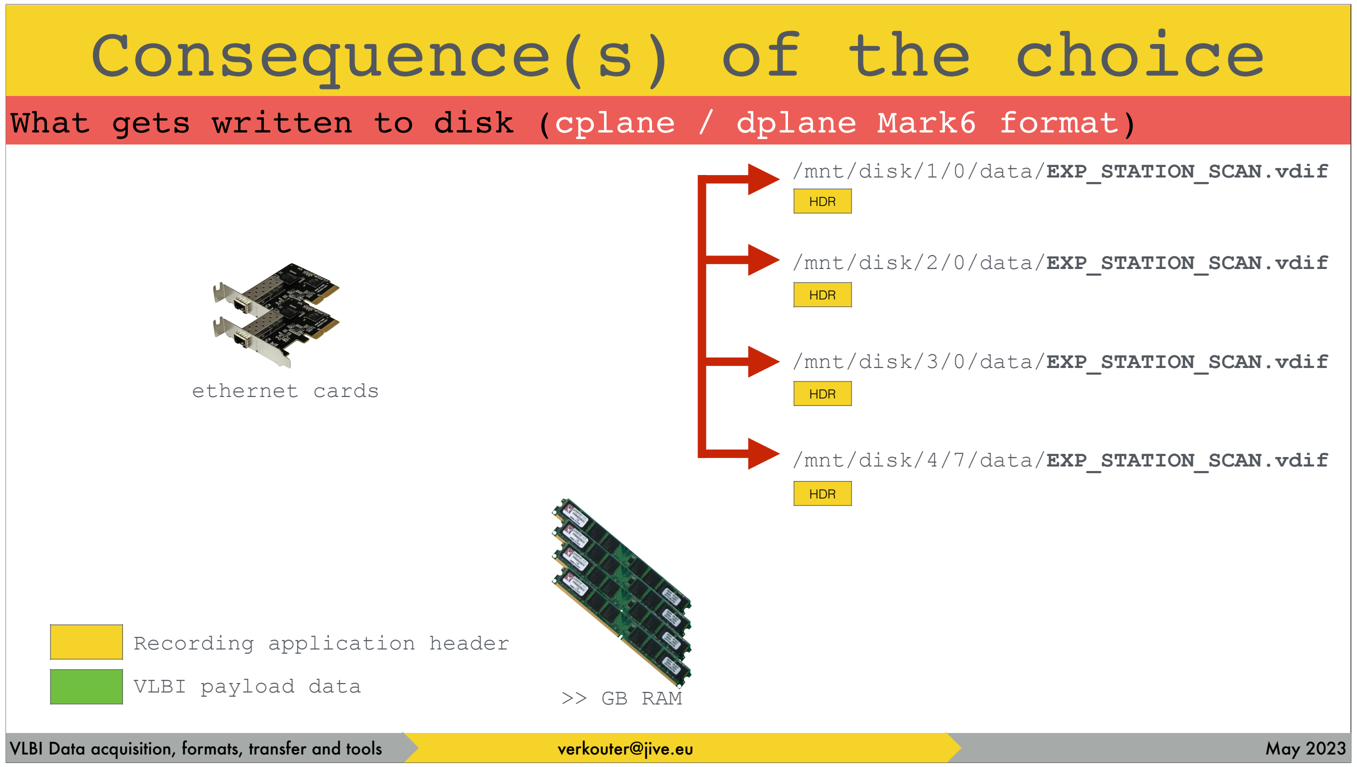
So configuration of this is a bit more difficult but flexibility is 100%



Consequence(s) of the choice

What gets written to disk

Another difference is WHAT gets recorded



Consequence(s) of the choice

What gets written to disk (cplane / dplane Mark6 format)

/mnt/disk/1/0/data/**EXP_STATION_SCAN.vdif**

HDR

/mnt/disk/2/0/data/**EXP_STATION_SCAN.vdif**

HDR

/mnt/disk/3/0/data/**EXP_STATION_SCAN.vdif**

HDR

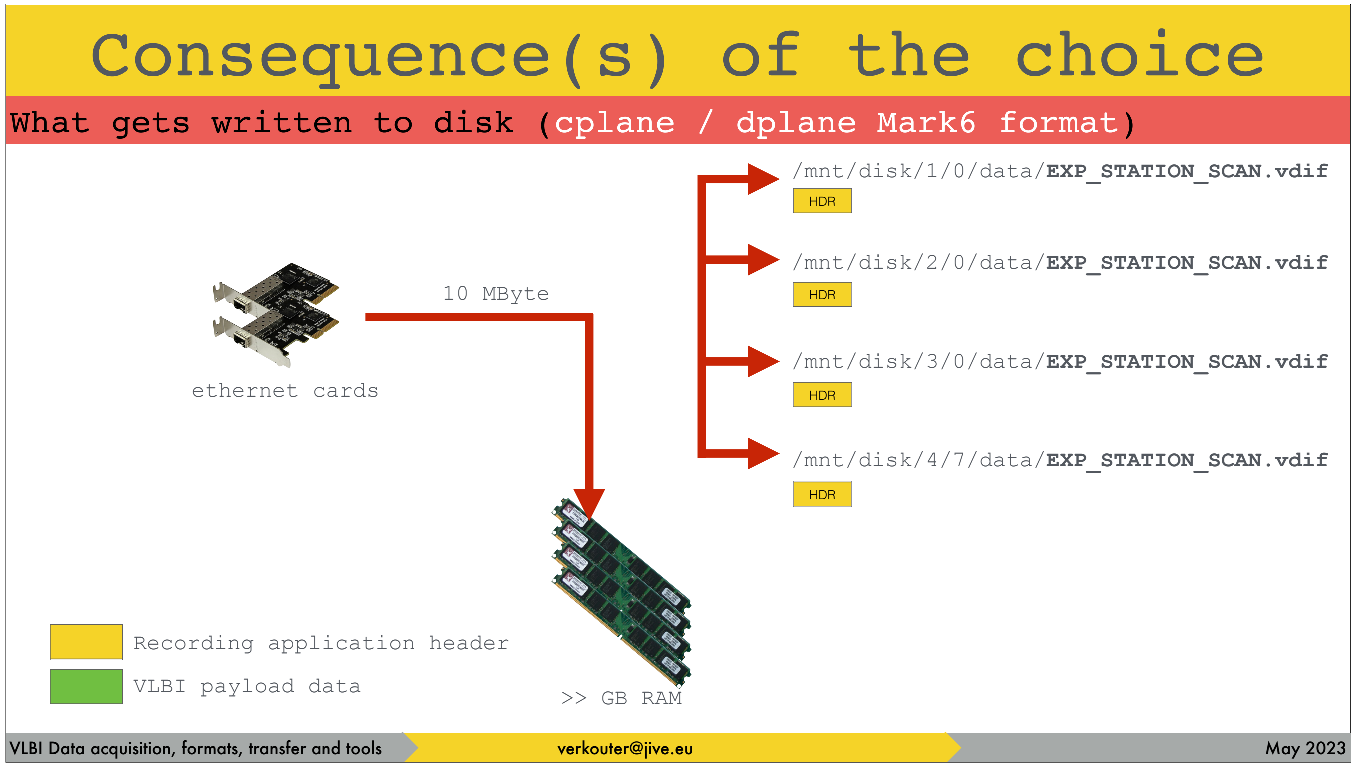
/mnt/disk/4/7/data/**EXP_STATION_SCAN.vdif**

HDR

🟨 Recording application header

🟩 VLBI payload data

The cplane/dplane software opens files on each disk, [click] writes a header identifying this as a Mark6 file.



Consequence(s) of the choice

What gets written to disk (cplane / dplane Mark6 format)

ethernet cards

10 MByte

>> GB RAM

/mnt/disk/1/0/data/**EXP_STATION_SCAN.vdif**

HDR

/mnt/disk/2/0/data/**EXP_STATION_SCAN.vdif**

HDR

/mnt/disk/3/0/data/**EXP_STATION_SCAN.vdif**

HDR

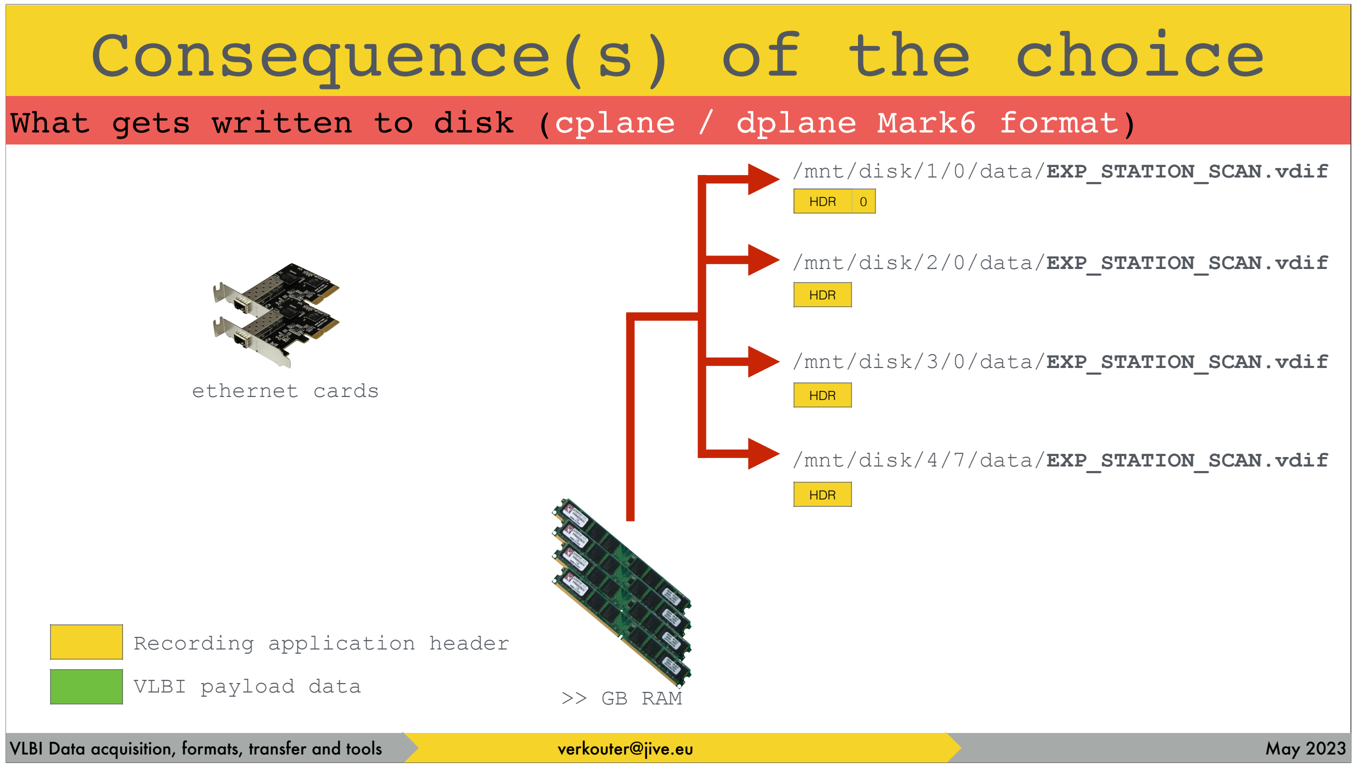
/mnt/disk/4/7/data/**EXP_STATION_SCAN.vdif**

HDR

🟨 Recording application header

🟩 VLBI payload data

When a 10 MB block has been read



Consequence(s) of the choice

What gets written to disk (cplane / dplane Mark6 format)

/mnt/disk/1/0/data/**EXP_STATION_SCAN.vdif**

HDR 0

/mnt/disk/2/0/data/**EXP_STATION_SCAN.vdif**

HDR

/mnt/disk/3/0/data/**EXP_STATION_SCAN.vdif**

HDR

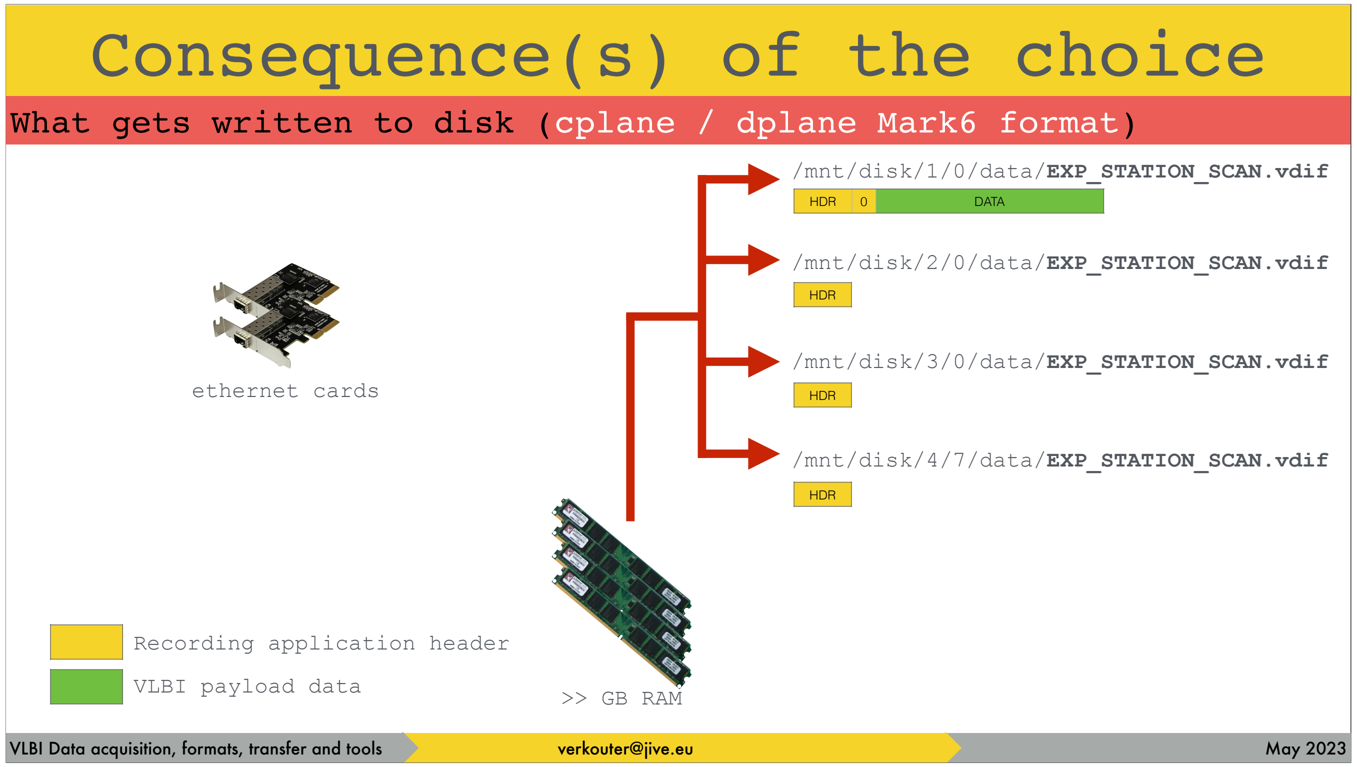
/mnt/disk/4/7/data/**EXP_STATION_SCAN.vdif**

HDR

🟨 Recording application header

🟩 VLBI payload data

the first available file is found and a block header is written in the file,



Consequence(s) of the choice

What gets written to disk (cplane / dplane Mark6 format)

ethernet cards

/mnt/disk/1/0/data/**EXP_STATION_SCAN.vdif**

/mnt/disk/2/0/data/**EXP_STATION_SCAN.vdif**

/mnt/disk/3/0/data/**EXP_STATION_SCAN.vdif**

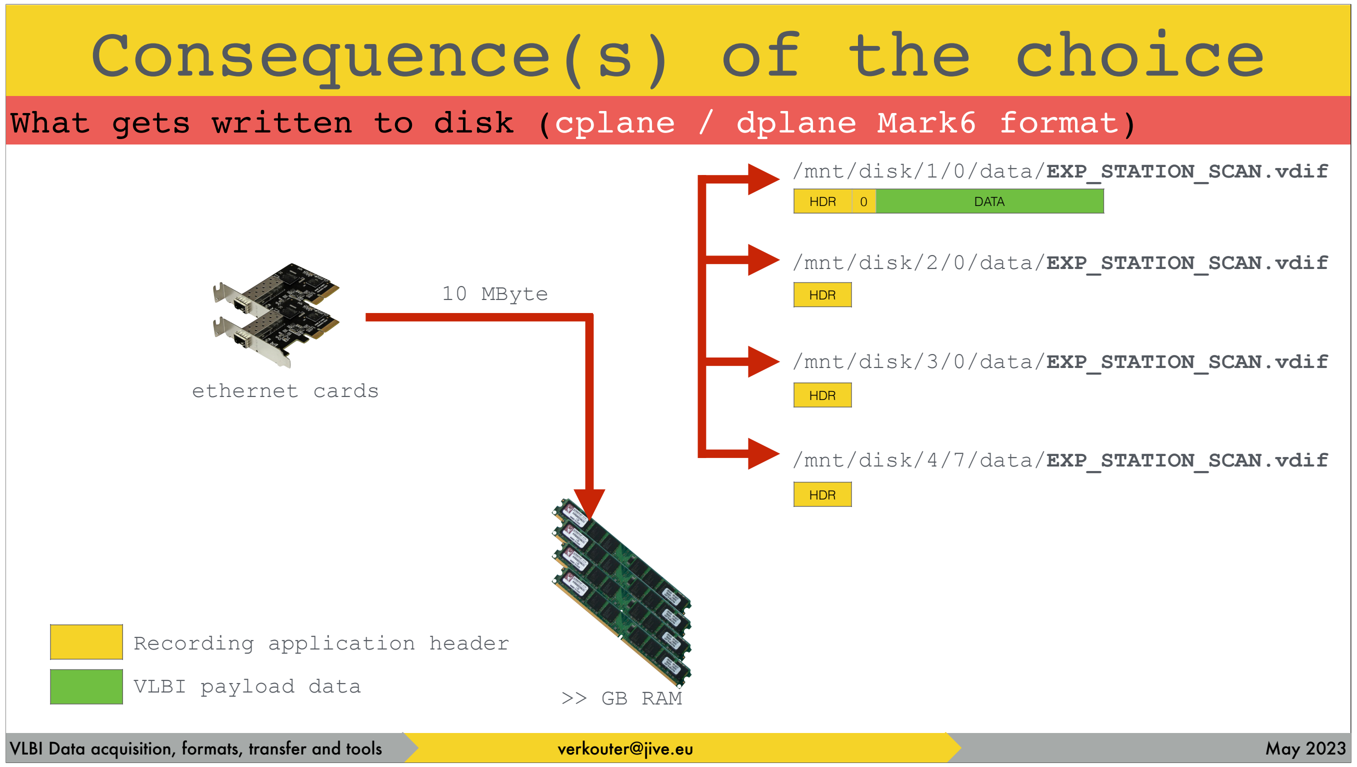
/mnt/disk/4/7/data/**EXP_STATION_SCAN.vdif**

>> GB RAM

🟨 Recording application header

🟩 VLBI payload data

and after that the actual block of data



Consequence(s) of the choice

What gets written to disk (cplane / dplane Mark6 format)

ethernet cards

10 MByte

>> GB RAM

/mnt/disk/1/0/data/**EXP_STATION_SCAN.vdif**

HDR | 0 | DATA

/mnt/disk/2/0/data/**EXP_STATION_SCAN.vdif**

HDR

/mnt/disk/3/0/data/**EXP_STATION_SCAN.vdif**

HDR

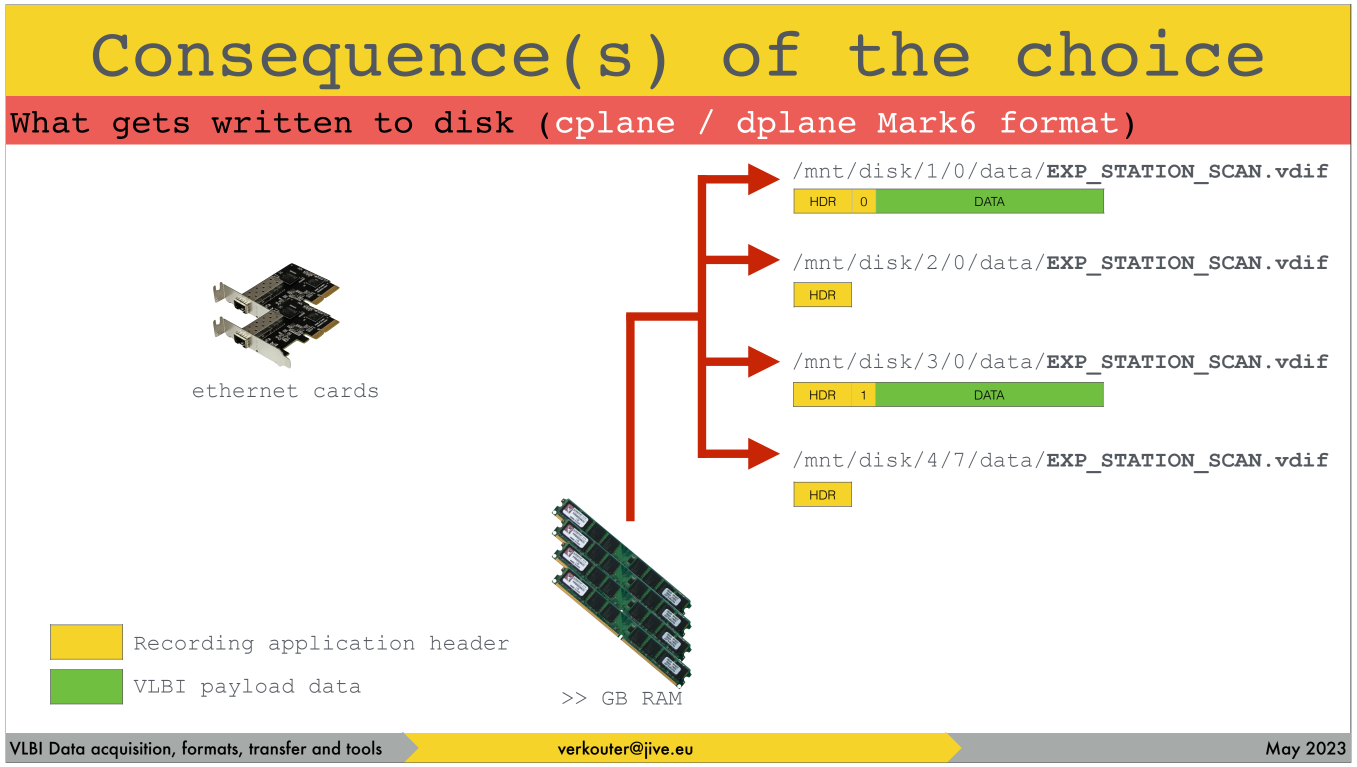
/mnt/disk/4/7/data/**EXP_STATION_SCAN.vdif**

HDR

🟨 Recording application header

🟩 VLBI payload data

The next block comes in



Consequence(s) of the choice

What gets written to disk (cplane / dplane Mark6 format)

ethernet cards

>> GB RAM

/mnt/disk/1/0/data/**EXP_STATION_SCAN.vdif**

HDR | 0 | DATA

/mnt/disk/2/0/data/**EXP_STATION_SCAN.vdif**

HDR

/mnt/disk/3/0/data/**EXP_STATION_SCAN.vdif**

HDR | 1 | DATA

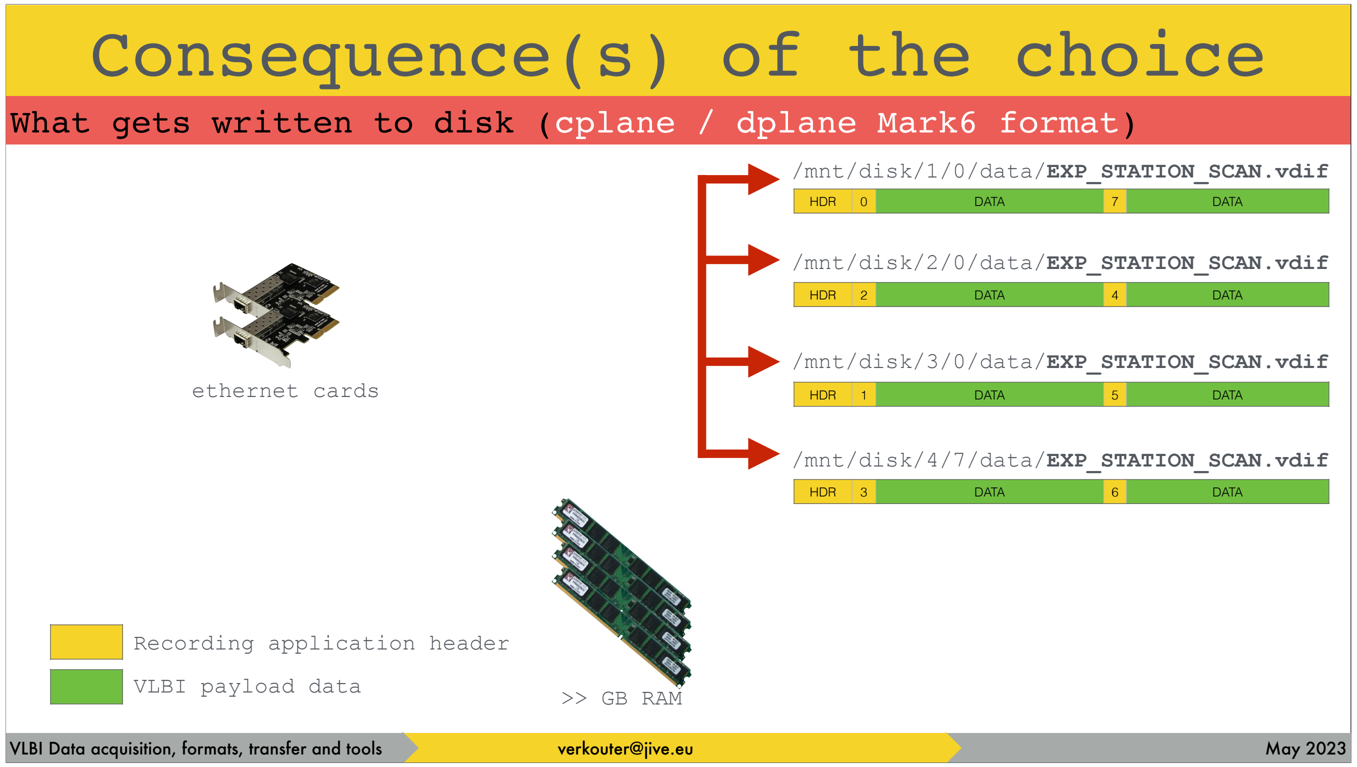
/mnt/disk/4/7/data/**EXP_STATION_SCAN.vdif**

HDR

🟨 Recording application header

🟩 VLBI payload data

and another file gets written to



Consequence(s) of the choice

What gets written to disk (cplane / dplane Mark6 format)

/mnt/disk/1/0/data/**EXP_STATION_SCAN.vdif**

HDR	0	DATA	7	DATA

/mnt/disk/2/0/data/**EXP_STATION_SCAN.vdif**

HDR	2	DATA	4	DATA

/mnt/disk/3/0/data/**EXP_STATION_SCAN.vdif**

HDR	1	DATA	5	DATA

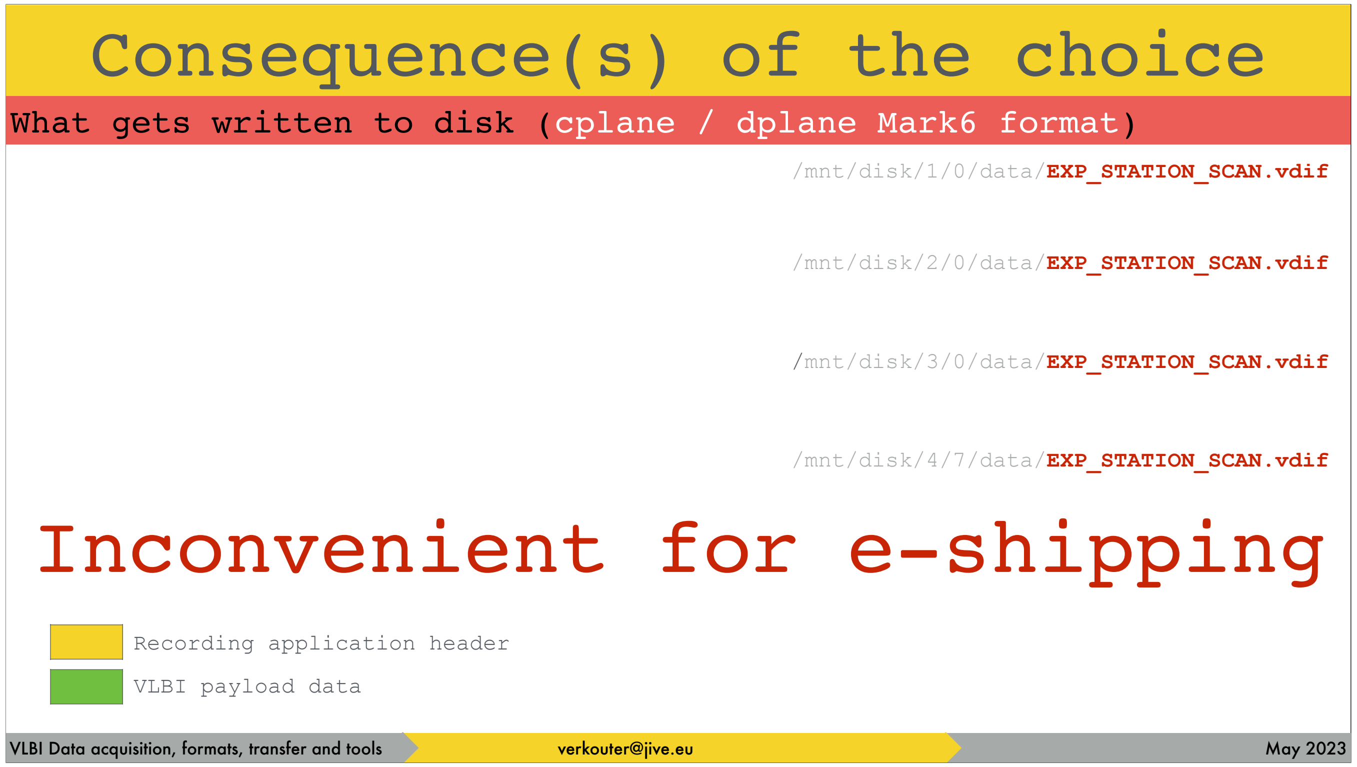
/mnt/disk/4/7/data/**EXP_STATION_SCAN.vdif**

HDR	3	DATA	6	DATA

🟨 Recording application header

🟩 VLBI payload data

That way data is scattered across the files.



Consequence(s) of the choice

What gets written to disk (cplane / dplane Mark6 format)

/mnt/disk/1/0/data/**EXP_STATION_SCAN.vdif**

/mnt/disk/2/0/data/**EXP_STATION_SCAN.vdif**

/mnt/disk/3/0/data/**EXP_STATION_SCAN.vdif**

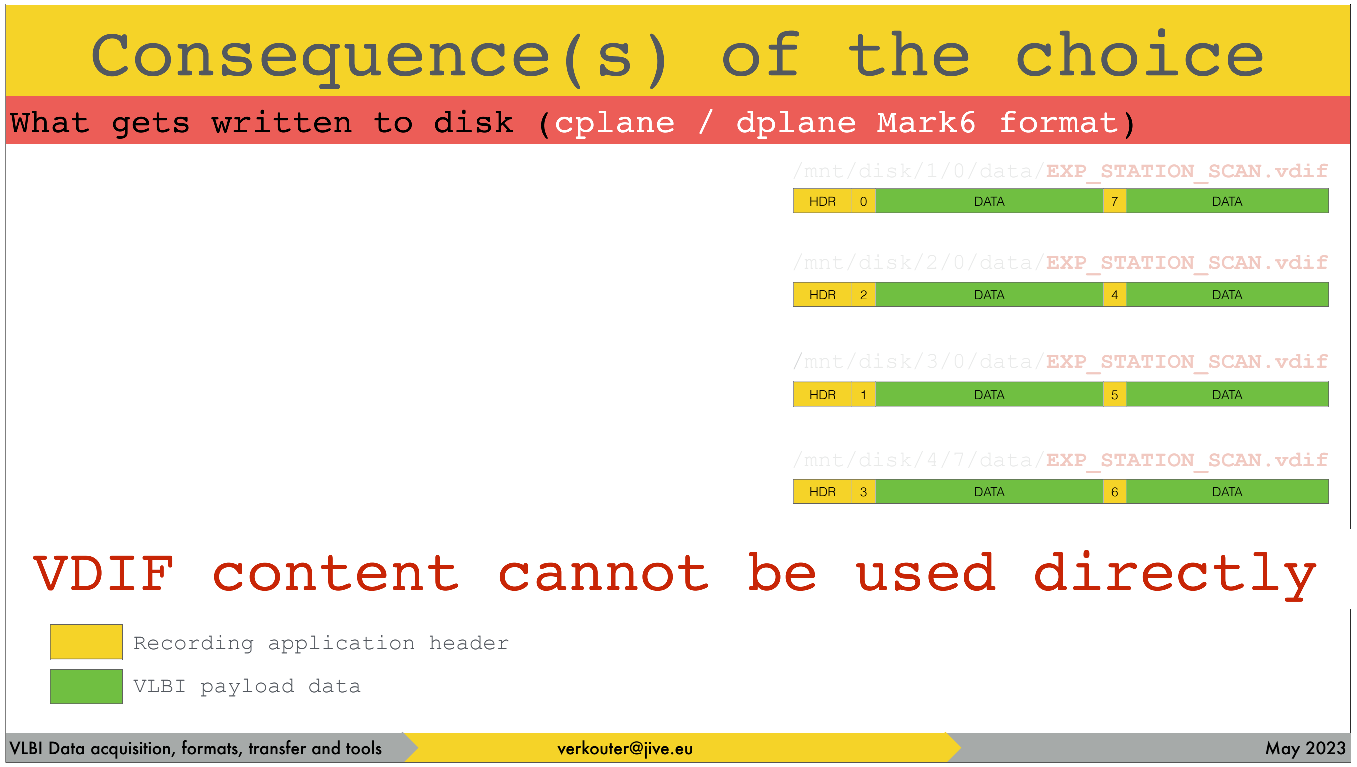
/mnt/disk/4/7/data/**EXP_STATION_SCAN.vdif**

Inconvenient for e-shipping

🟨 Recording application header

🟩 VLBI payload data

As you can see, the files are named all the same, [click] which is not convenient for electronic transfer!



/mnt/disk/1/0/data/EXP_STATION_SCAN.vdif
HDR
0
DATA
7
DATA
/mnt/disk/2/0/data/EXP_STATION_SCAN.vdif
HDR
2
DATA
4
DATA
/mnt/disk/3/0/data/EXP_STATION_SCAN.vdif
HDR
1
DATA
5
DATA
/mnt/disk/4/7/data/EXP_STATION_SCAN.vdif
HDR
3
DATA
6
DATA

Consequence(s) of the choice

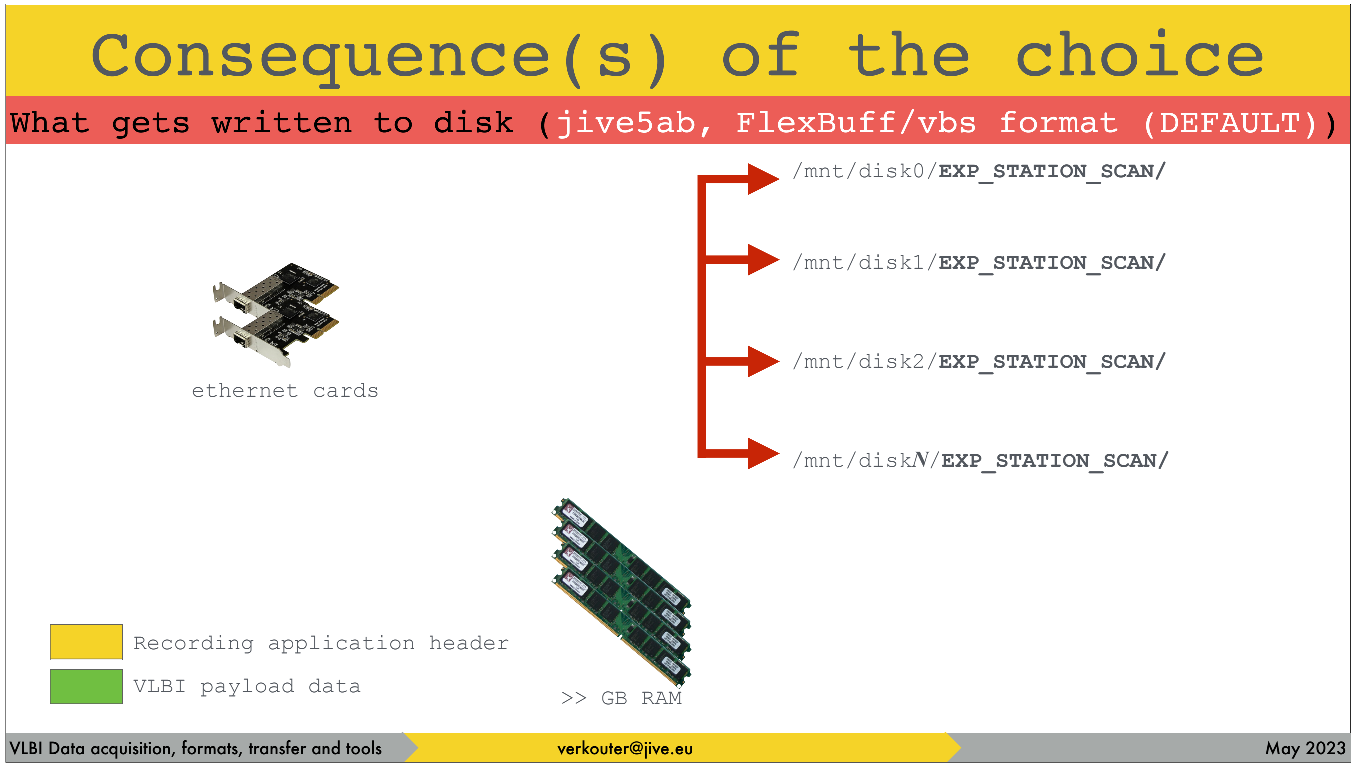
What gets written to disk (cplane / dplane Mark6 format)

VDIF content cannot be used directly

🟨 Recording application header

🟩 VLBI payload data

Another thing is that the VLBI data is mixed with application headers. [click] Which means that the files cannot be easily processed by an arbitrary VDIF tool



Consequence(s) of the choice

What gets written to disk (jive5ab, FlexBuff/vbs format (DEFAULT))

/mnt/disk0/**EXP_STATION_SCAN/**

/mnt/disk1/**EXP_STATION_SCAN/**

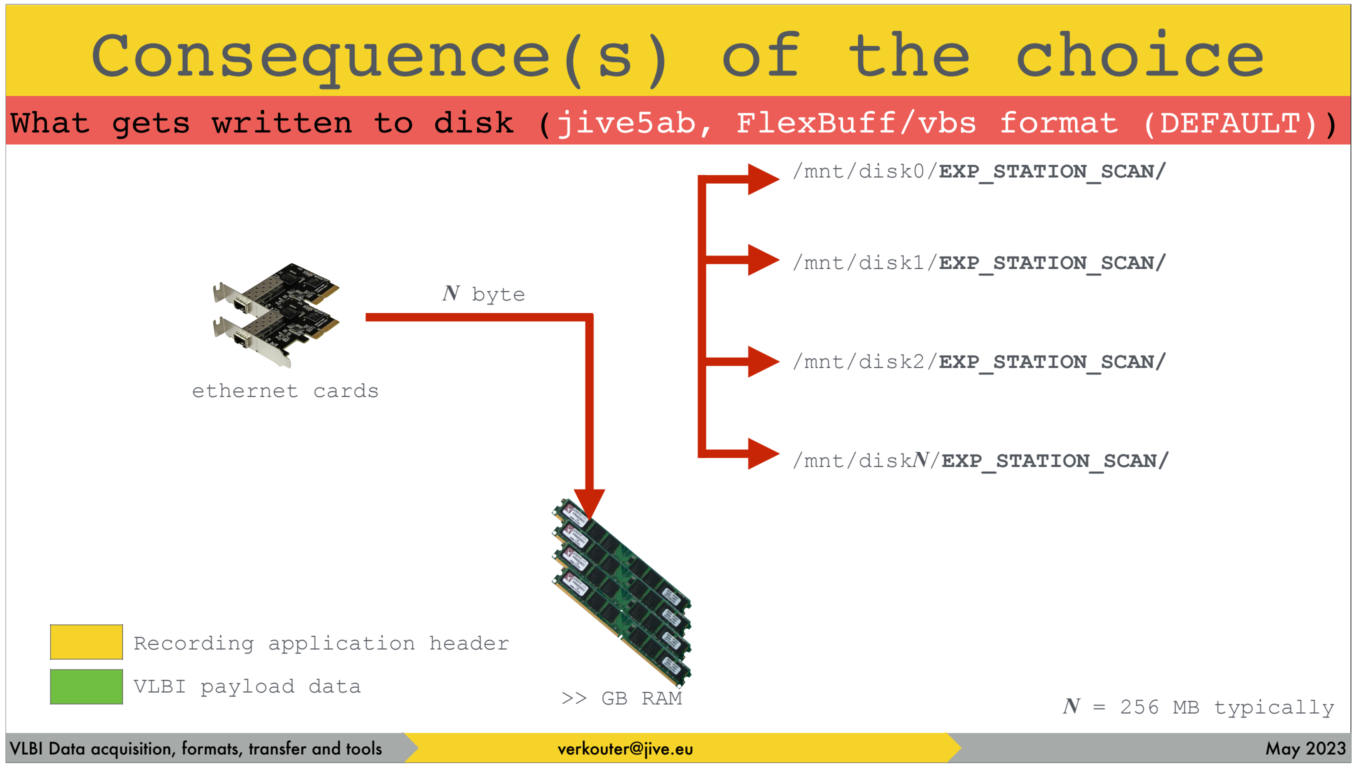
/mnt/disk2/**EXP_STATION_SCAN/**

/mnt/disk*N*/**EXP_STATION_SCAN/**

🟨 Recording application header

🟩 VLBI payload data

The default format that jive5ab writes is the FlexBuff “vbs” format. [click] jive5ab creates *directories* with the recording name on all disks



Consequence(s) of the choice

What gets written to disk (jive5ab, FlexBuff/vbs format (DEFAULT))

/mnt/disk0/**EXP_STATION_SCAN/**

/mnt/disk1/**EXP_STATION_SCAN/**

/mnt/disk2/**EXP_STATION_SCAN/**

/mnt/disk*N*/**EXP_STATION_SCAN/**

🟨 Recording application header

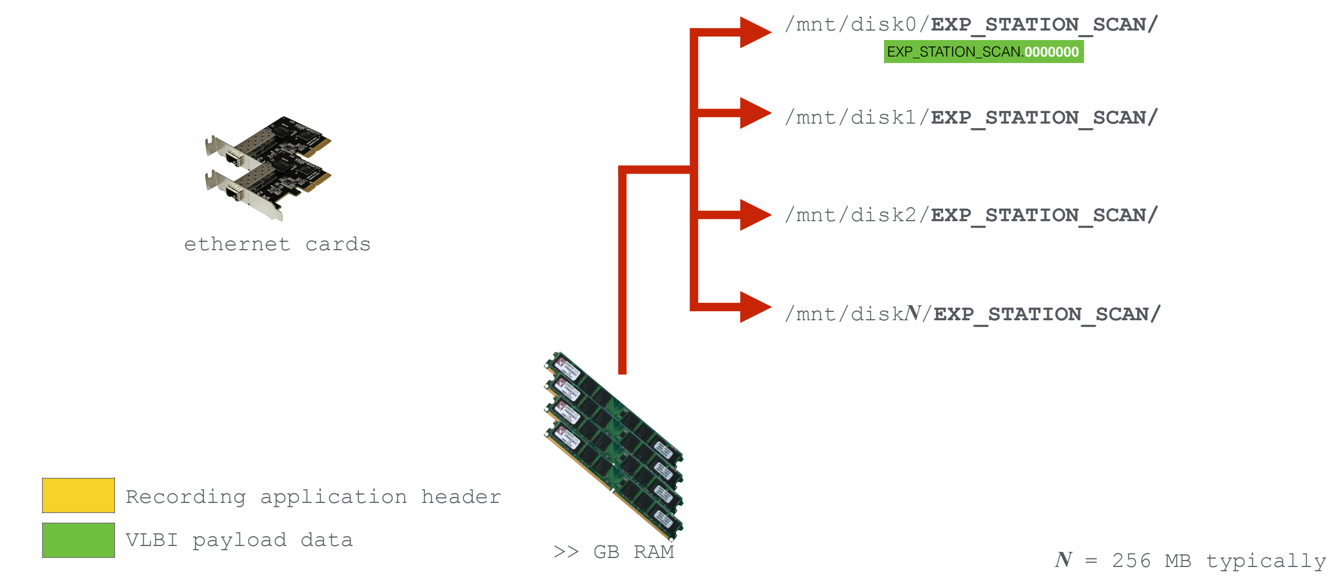
🟩 VLBI payload data

N = 256 MB typically

After reading a block of typically 256 MByte

Consequence(s) of the choice

What gets written to disk (jive5ab, FlexBuff/vbs format (DEFAULT))



/mnt/disk0/**EXP_STATION_SCAN/**

EXP_STATION_SCAN.0000000

/mnt/disk1/**EXP_STATION_SCAN/**

/mnt/disk2/**EXP_STATION_SCAN/**

/mnt/disk*N*/**EXP_STATION_SCAN/**

ethernet cards

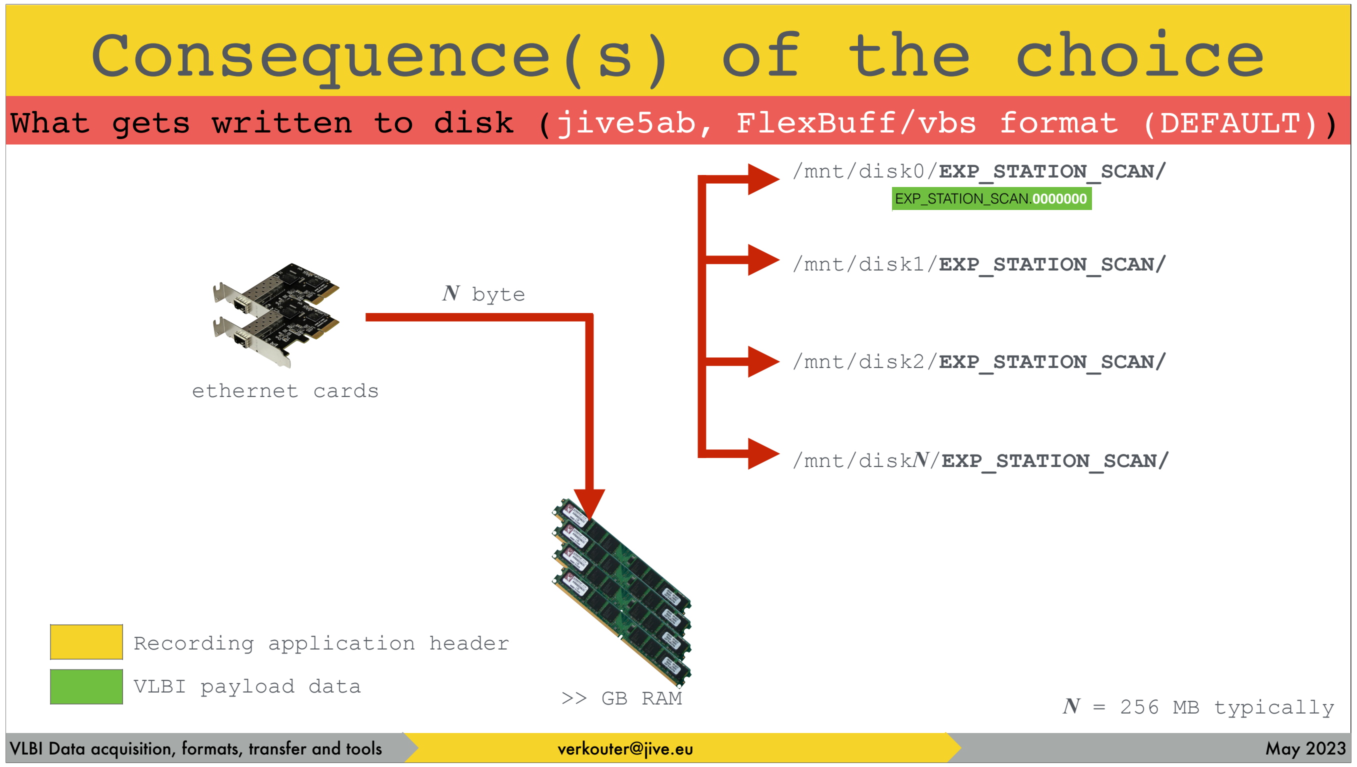
🟨 Recording application header

🟩 VLBI payload data

>> GB RAM

N = 256 MB typically

The first available disk is selected and the block is written to a single file with the block sequence number as the extension



Consequence(s) of the choice

What gets written to disk (jive5ab, FlexBuff/vbs format (DEFAULT))

ethernet cards

N byte

>> GB RAM

/mnt/disk0/**EXP_STATION_SCAN/**

EXP_STATION_SCAN.0000000

/mnt/disk1/**EXP_STATION_SCAN/**

/mnt/disk2/**EXP_STATION_SCAN/**

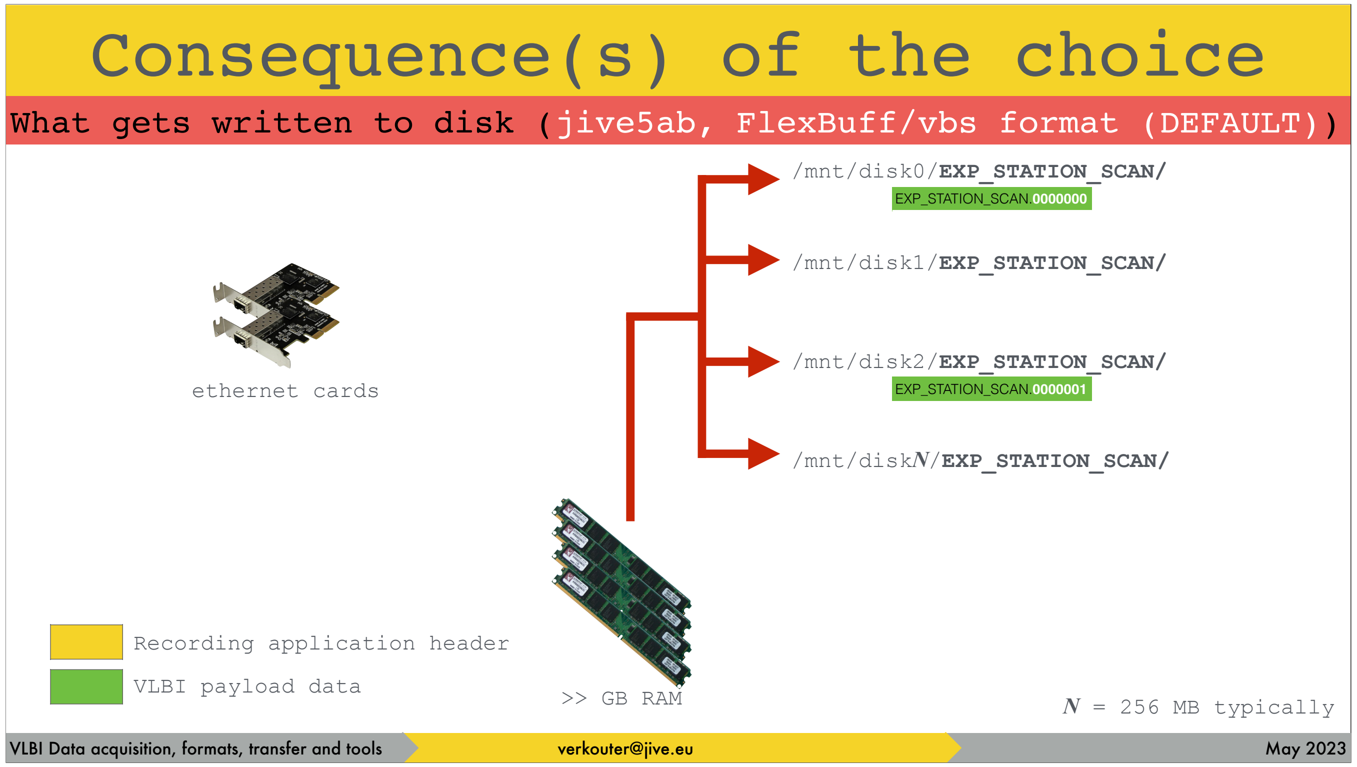
/mnt/disk*N*/**EXP_STATION_SCAN/**

🟨 Recording application header

🟩 VLBI payload data

N = 256 MB typically

the next block is captured



Consequence(s) of the choice

What gets written to disk (jive5ab, FlexBuff/vbs format (DEFAULT))

ethernet cards

/mnt/disk0/**EXP_STATION_SCAN/**

EXP_STATION_SCAN.0000000

/mnt/disk1/**EXP_STATION_SCAN/**

/mnt/disk2/**EXP_STATION_SCAN/**

EXP_STATION_SCAN.0000001

/mnt/disk*N*/**EXP_STATION_SCAN/**

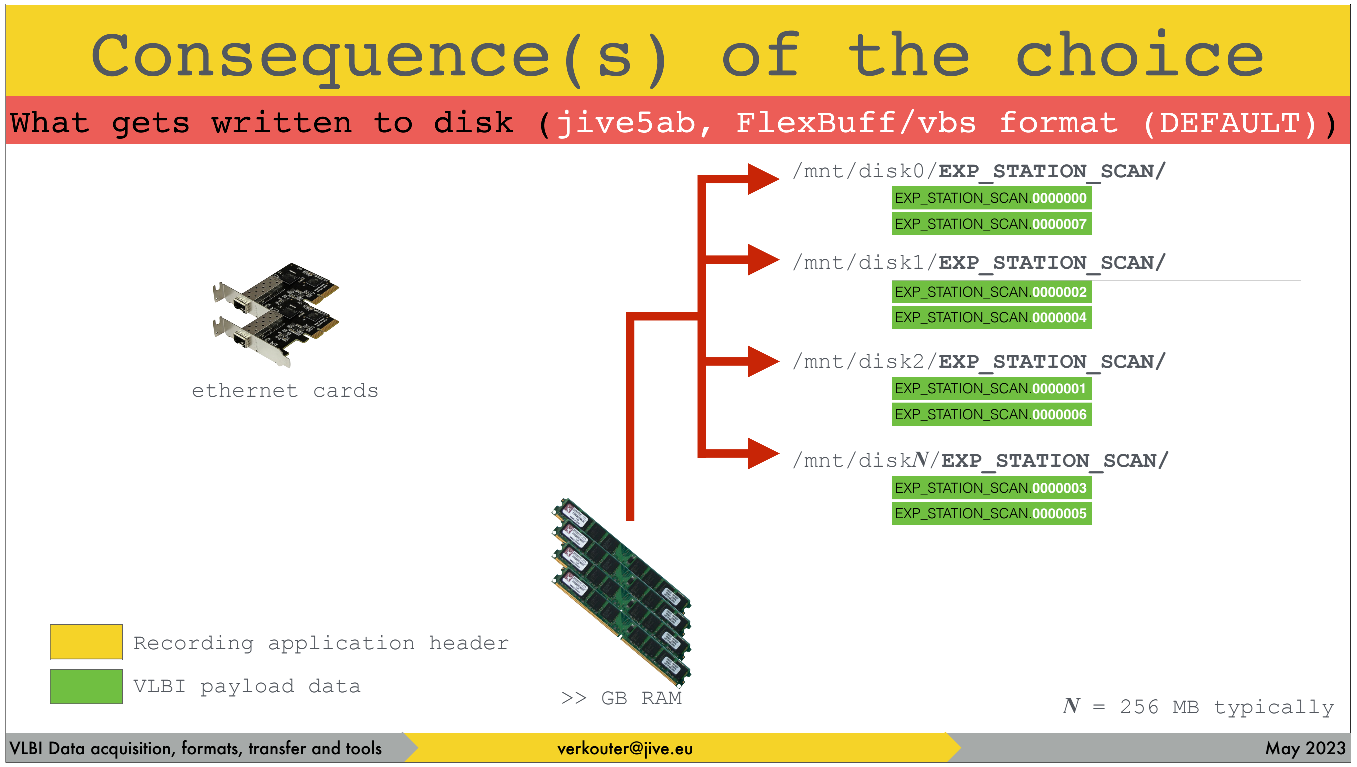
🟨 Recording application header

🟩 VLBI payload data

>> GB RAM

N = 256 MB typically

and is written to the next available disk



Consequence(s) of the choice

What gets written to disk (jive5ab, FlexBuff/vbs format (DEFAULT))

/mnt/disk0/**EXP_STATION_SCAN/**

- EXP_STATION_SCAN.0000000
- EXP_STATION_SCAN.0000007

/mnt/disk1/**EXP_STATION_SCAN/**

- EXP_STATION_SCAN.0000002
- EXP_STATION_SCAN.0000004

/mnt/disk2/**EXP_STATION_SCAN/**

- EXP_STATION_SCAN.0000001
- EXP_STATION_SCAN.0000006

/mnt/disk*N*/**EXP_STATION_SCAN/**

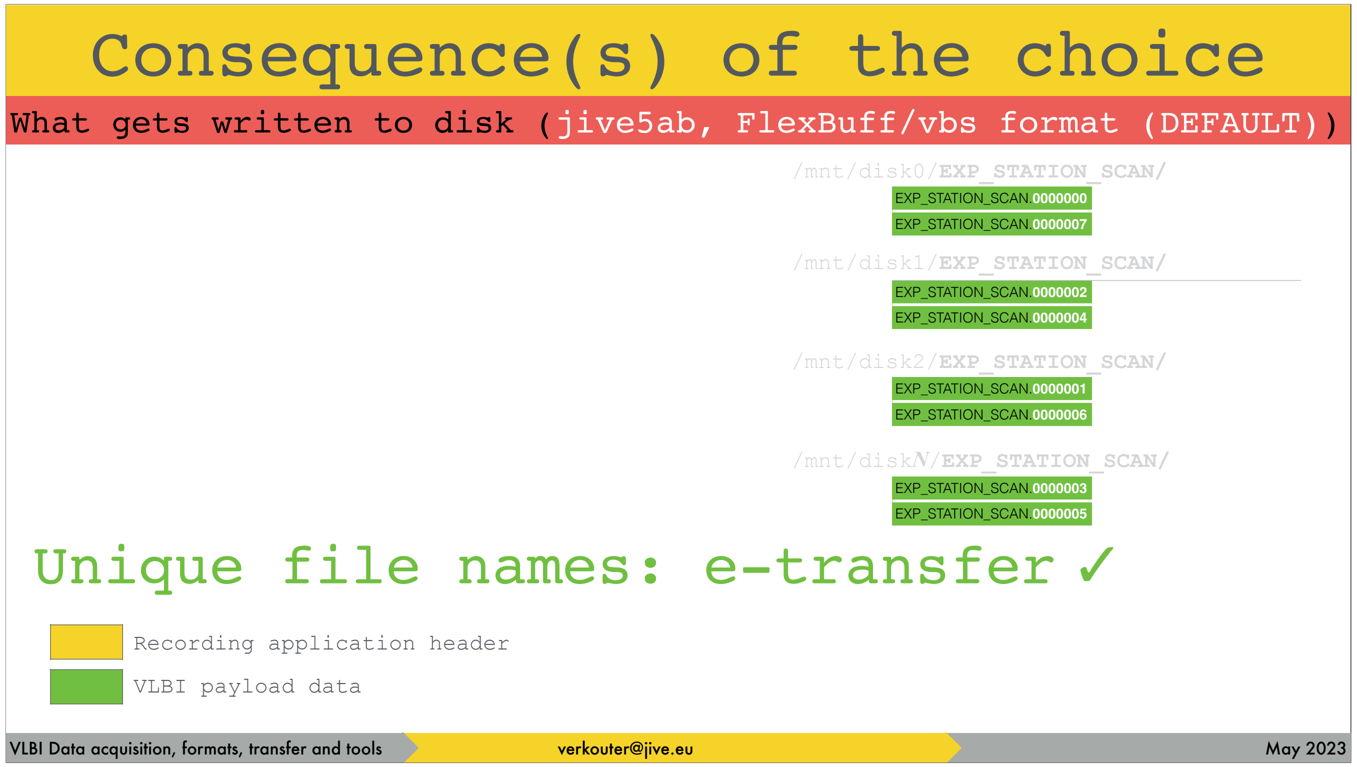
- EXP_STATION_SCAN.0000003
- EXP_STATION_SCAN.0000005

🟨 Recording application header

🟩 VLBI payload data

N = 256 MB typically

that way the directories are populated with chunks of VLBI data



Consequence(s) of the choice

What gets written to disk (jive5ab, FlexBuff/vbs format (DEFAULT))

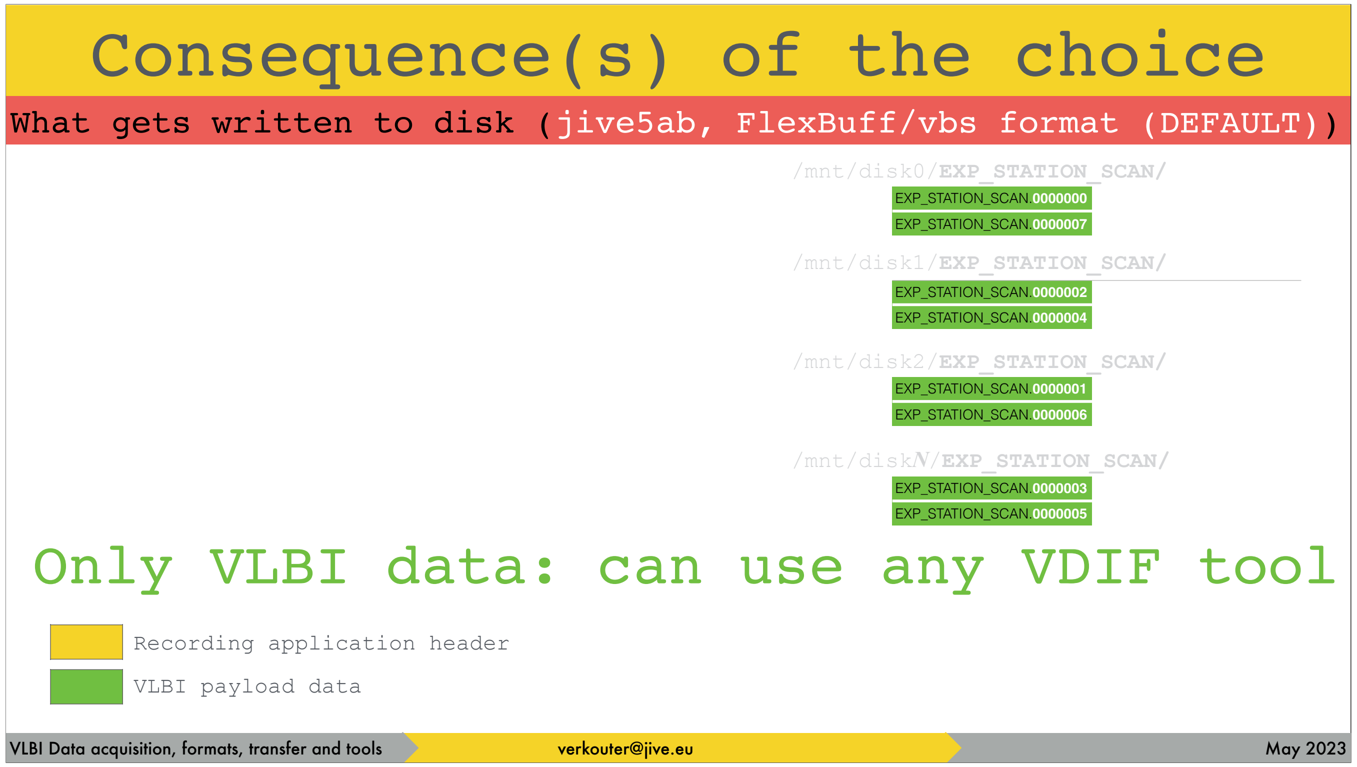
```
/mnt/disk0/EXP_STATION_SCAN/
    EXP_STATION_SCAN.0000000
    EXP_STATION_SCAN.0000007
/mnt/disk1/EXP_STATION_SCAN/
    EXP_STATION_SCAN.0000002
    EXP_STATION_SCAN.0000004
/mnt/disk2/EXP_STATION_SCAN/
    EXP_STATION_SCAN.0000001
    EXP_STATION_SCAN.0000006
/mnt/diskN/EXP_STATION_SCAN/
    EXP_STATION_SCAN.0000003
    EXP_STATION_SCAN.0000005
```

Unique file names: e-transfer ✓

🟨 Recording application header

🟩 VLBI payload data

The file names are unique which is extremely handy for e-transfer



Consequence(s) of the choice

What gets written to disk (jive5ab, FlexBuff/vbs format (DEFAULT))

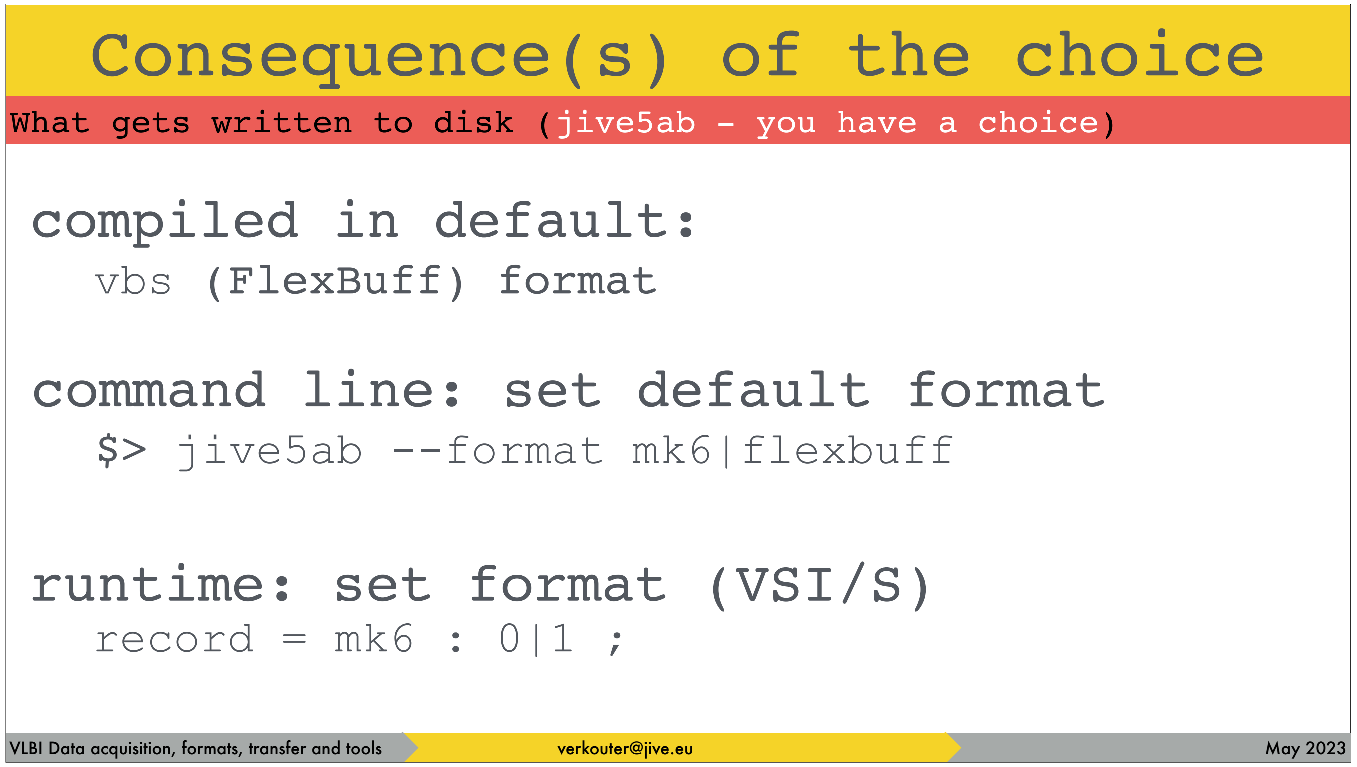
```
/mnt/disk0/EXP_STATION_SCAN/
    EXP_STATION_SCAN.0000000
    EXP_STATION_SCAN.0000007
/mnt/disk1/EXP_STATION_SCAN/
    EXP_STATION_SCAN.0000002
    EXP_STATION_SCAN.0000004
/mnt/disk2/EXP_STATION_SCAN/
    EXP_STATION_SCAN.0000001
    EXP_STATION_SCAN.0000006
/mnt/diskN/EXP_STATION_SCAN/
    EXP_STATION_SCAN.0000003
    EXP_STATION_SCAN.0000005
```

Only VLBI data: can use any VDIF tool

🟨 Recording application header

🟩 VLBI payload data

and there are no headers inbetween the useful data so any VDIF tool can use the snippets directly!



Consequence(s) of the choice

What gets written to disk (jive5ab - you have a choice)

compiled in default:

vbs (FlexBuff) format

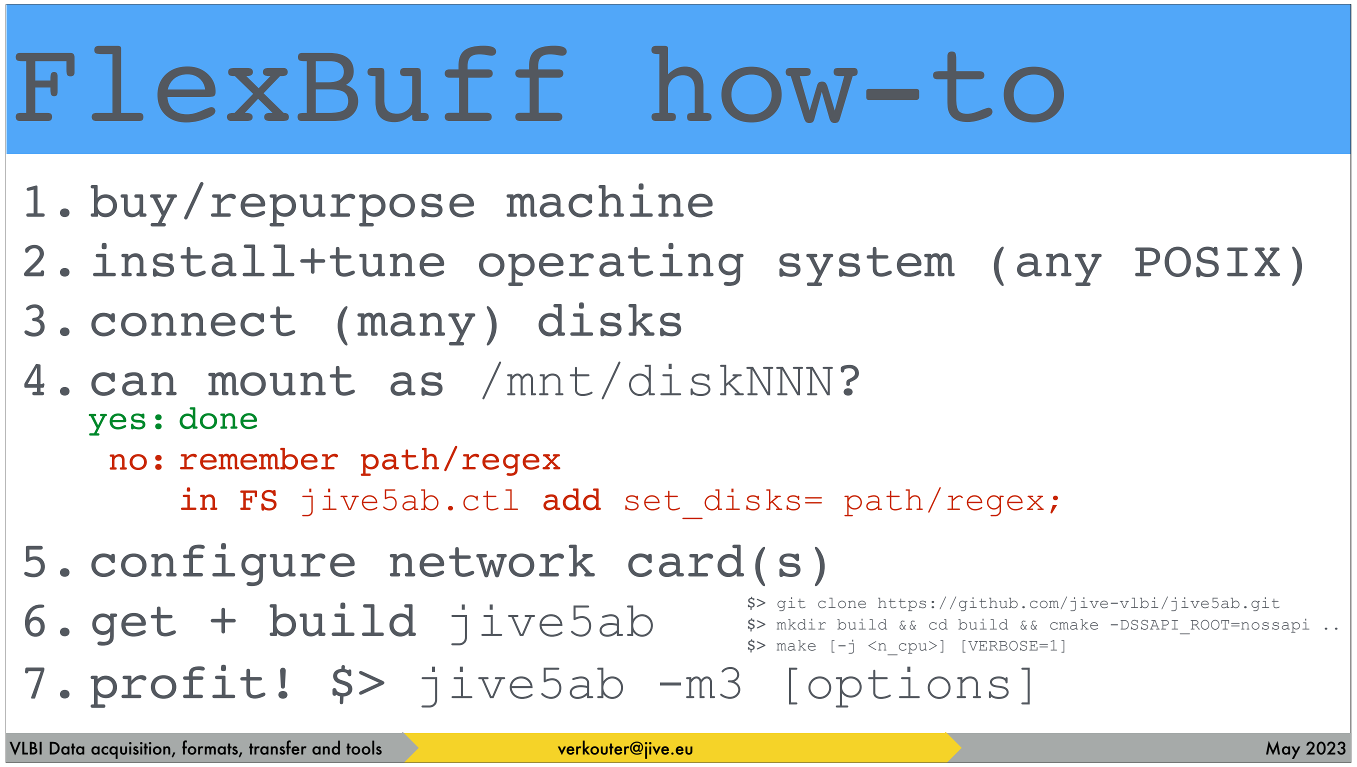
command line: set default format

```
$> jive5ab --format mk6|flexbuff
```

runtime: set format (VSI/S)

```
record = mk6 : 0|1 ;
```

jive5ab can record in both formats and there are several ways to change the recording format, [click] e.g. from the command line [click] or at runtime.



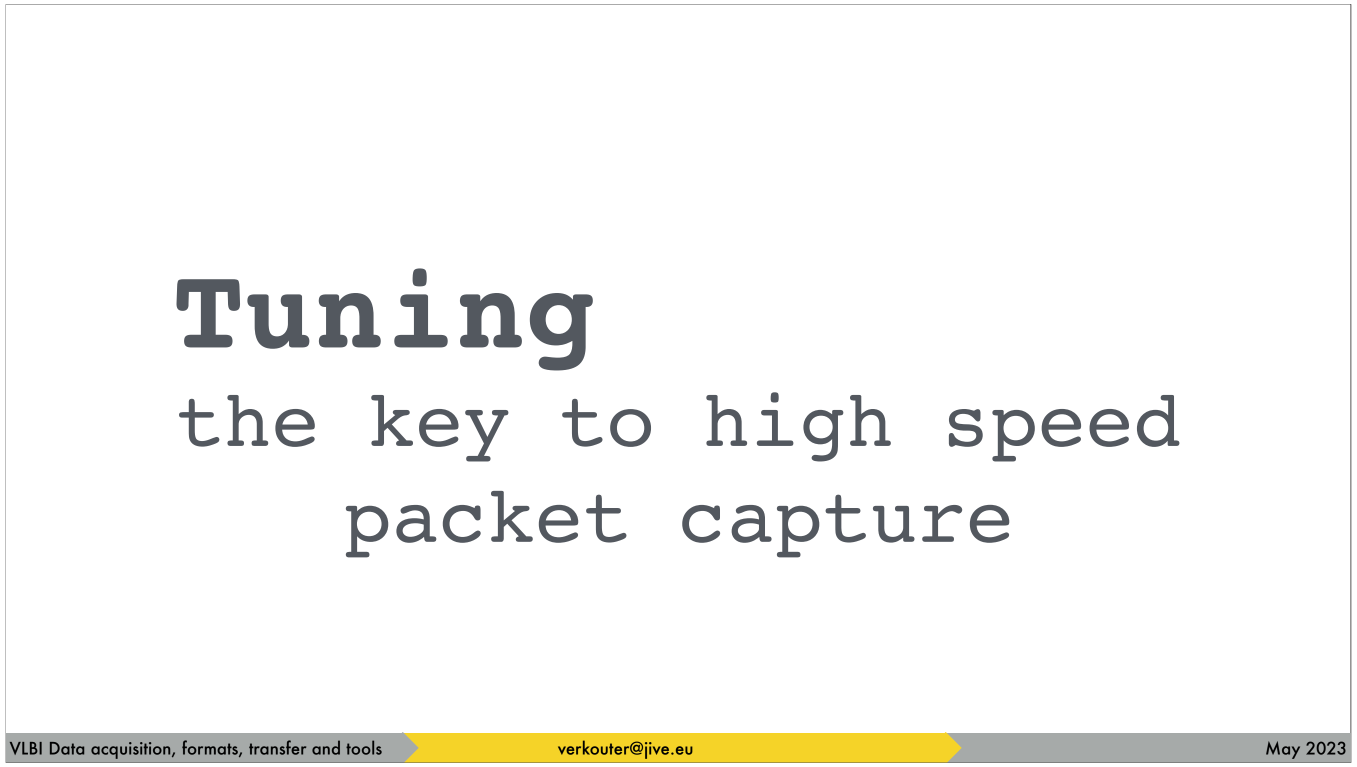
FlexBuff how-to

1. buy/repurpose machine
2. install+tune operating system (any POSIX)
3. connect (many) disks
4. can mount as /mnt/diskNNN?
   yes: done
   no: remember path/regex
   in FS jive5ab.ctl add set_disks= path/regex;
5. configure network card(s)
6. get + build jive5ab

```
$> git clone https://github.com/jive-vlbi/jive5ab.git
$> mkdir build && cd build && cmake -DSSAPI_ROOT=nossapi ..
$> make [-j <n_cpu>] [VERBOSE=1]
```

7. profit! $> jive5ab -m3 [options]

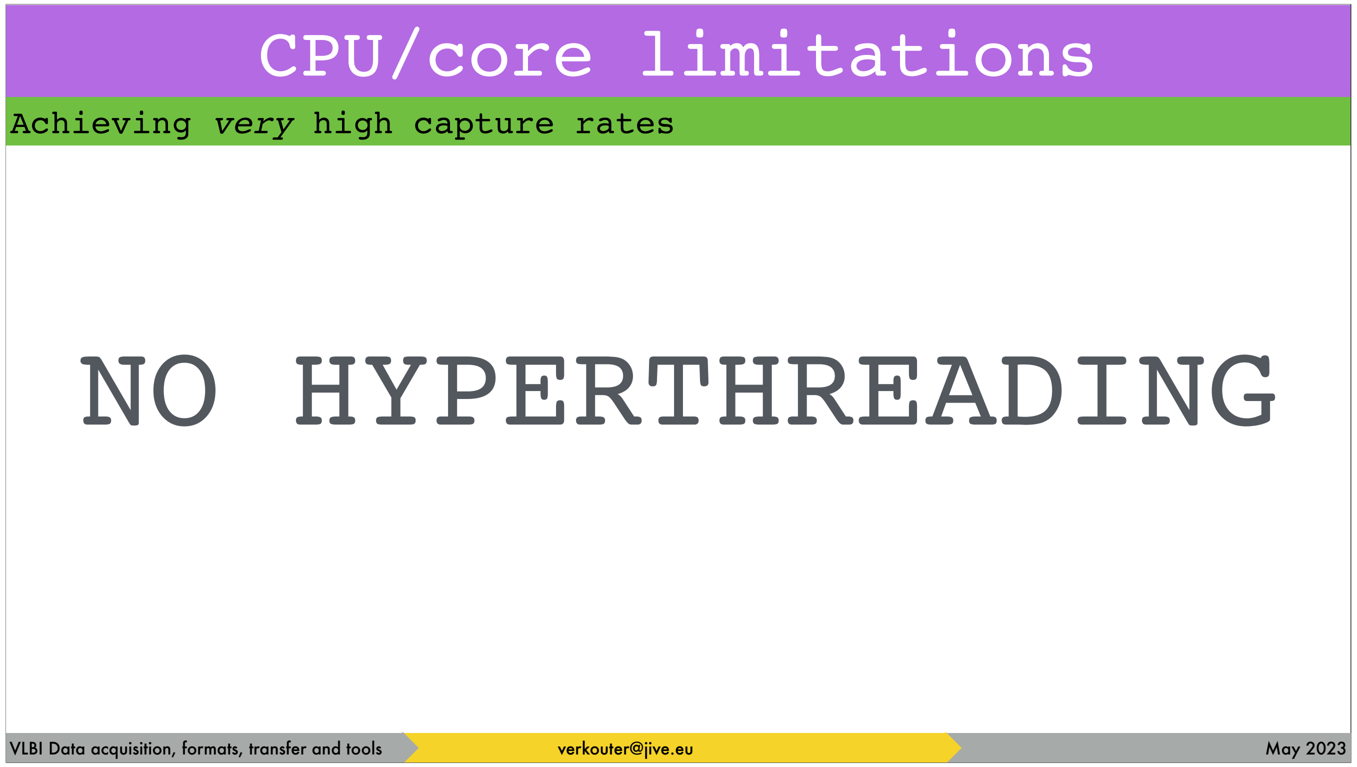
For quick reference, this is a one-slide recipe to building a flexbuff. It is really simple; have a machine with storage and run jive5ab.



Tuning

the key to high speed packet capture

As promised, let’s take a look into tuning

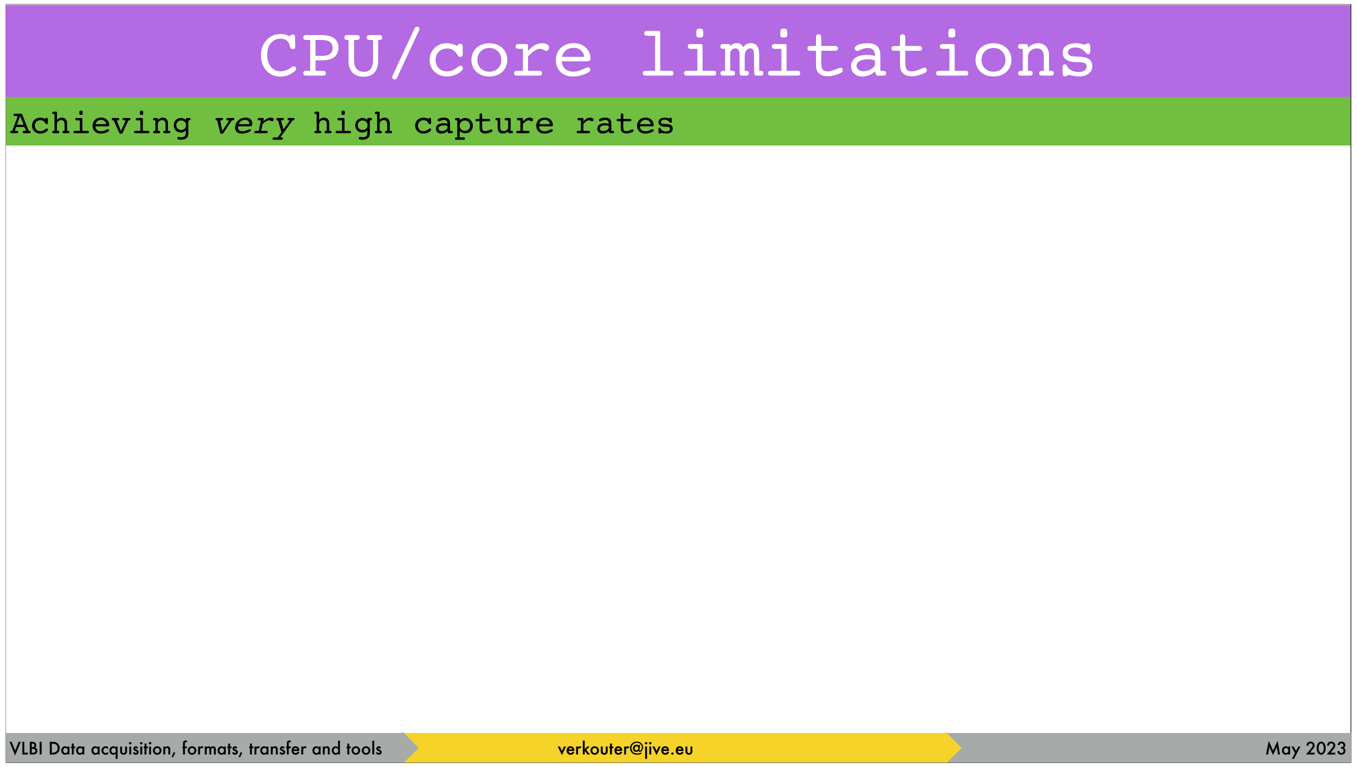


CPU/core limitations

Achieving *very* high capture rates

NO HYPERTHREADING

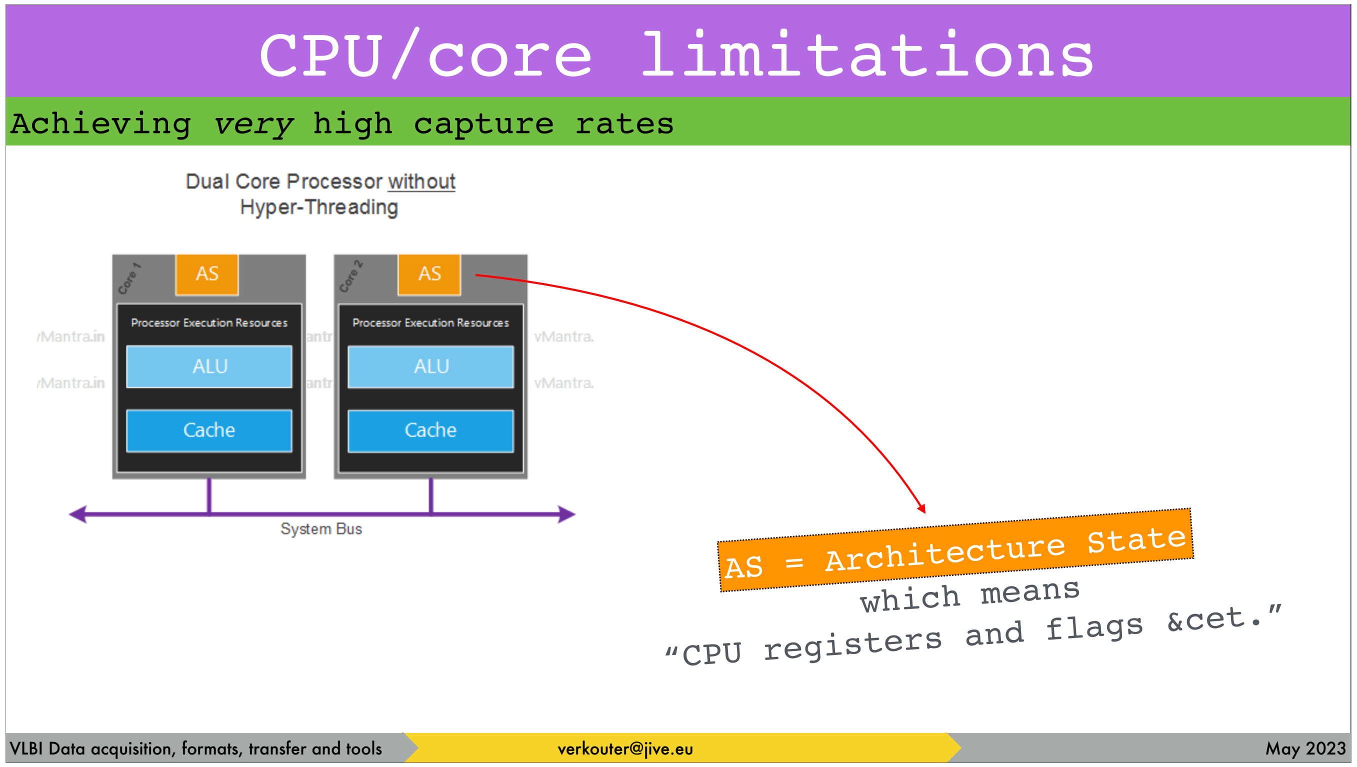
THE most important tuning is about this



CPU/core limitations

Achieving *very* high capture rates

Let’s compare two situations ..



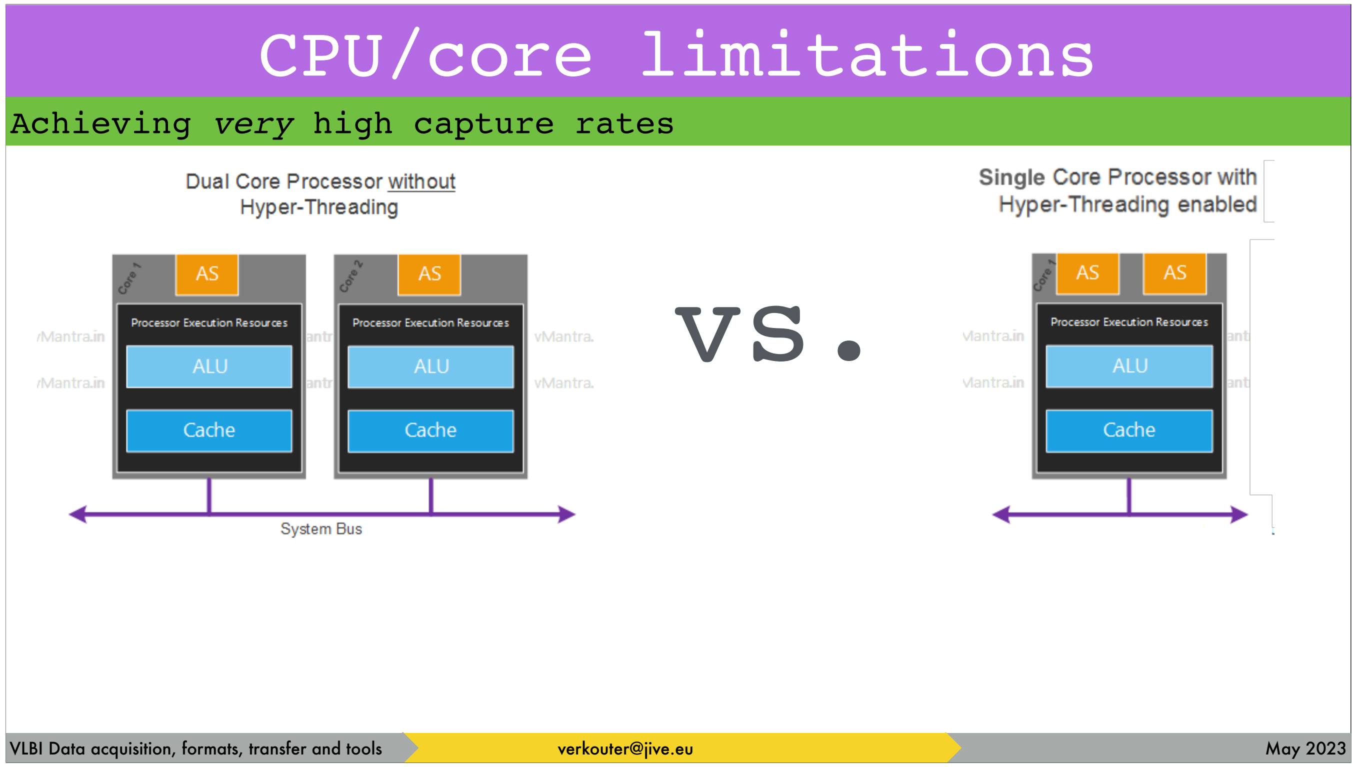
CPU/core limitations

Achieving *very* high capture rates

Dual Core Processor without Hyper-Threading

AS = Architecture State
which means
"CPU registers and flags &cet."

A dual core processor without hyperthreading versus …

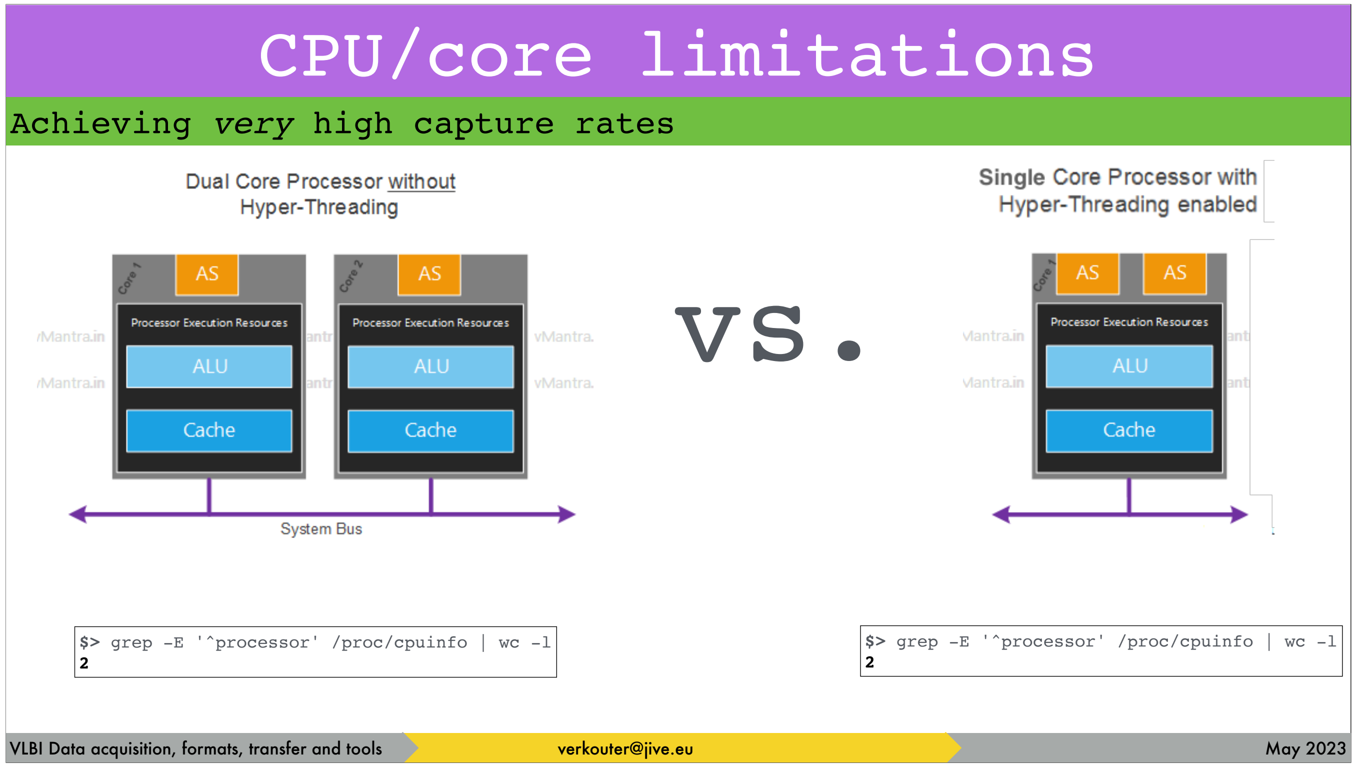


CPU/core limitations

Achieving *very* high capture rates

Dual Core Processor without Hyper-Threading
Core 1
AS
Processor Execution Resources
ALU
Cache
Core 2
AS
Processor Execution Resources
ALU
Cache
System Bus
vs.
Single Core Processor with Hyper-Threading enabled
Core 1
AS
AS
Processor Execution Resources
ALU
Cache
vMantra.in

a single core processor WITH hyper threading



CPU/core limitations

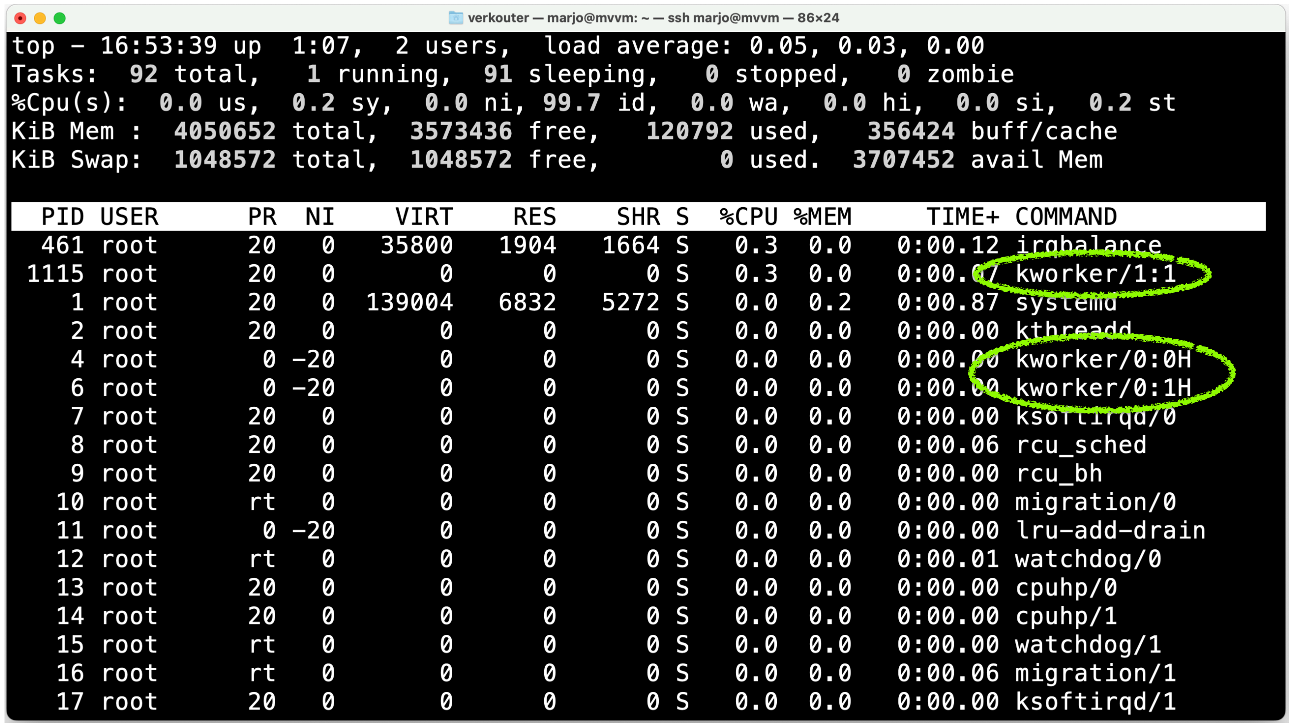
Achieving *very* high capture rates

Dual Core Processor without Hyper-Threading
Core 1
Core 2
AS
AS
Processor Execution Resources
Processor Execution Resources
ALU
ALU
Cache
Cache
System Bus
vs.
Single Core Processor with Hyper-Threading enabled
Core 1
AS
AS
Processor Execution Resources
ALU
Cache

```
$> grep -E '^processor' /proc/cpuinfo | wc -l
2
```

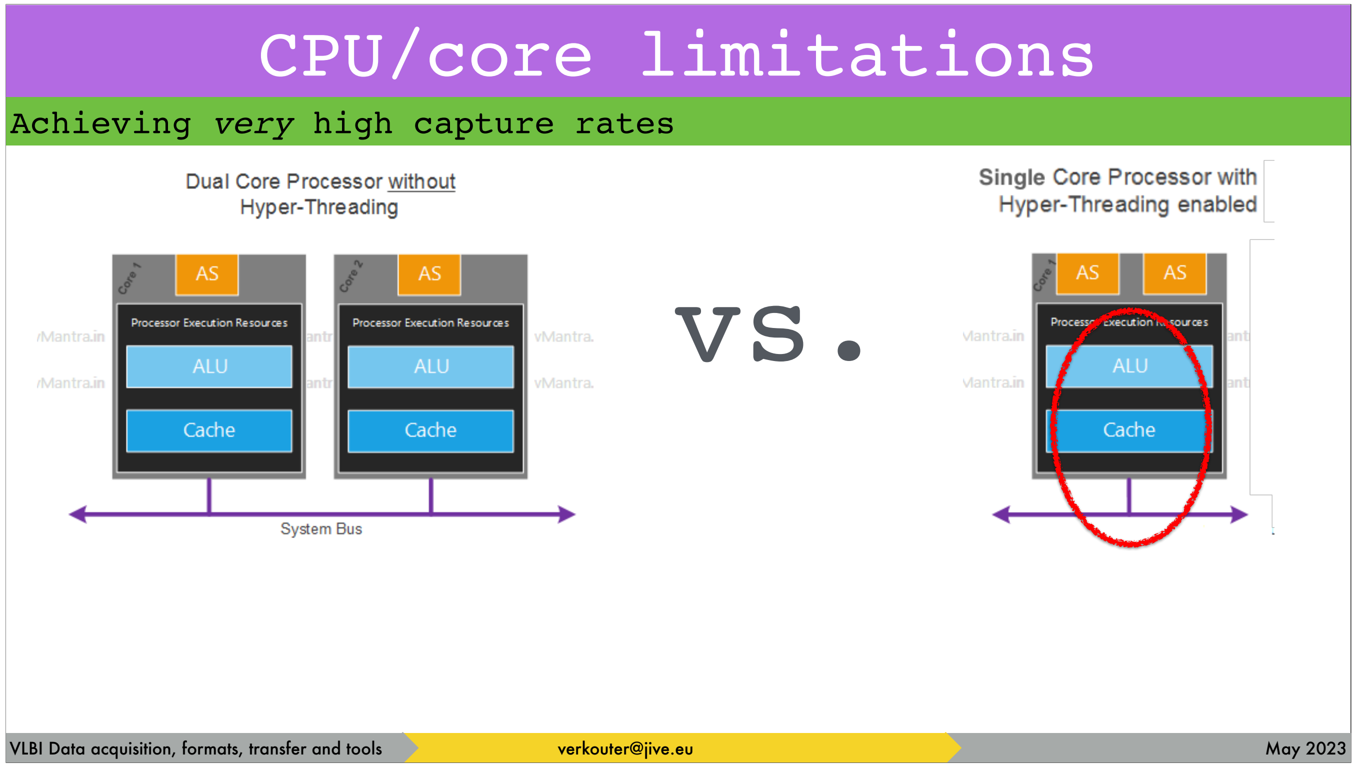
```
$> grep -E '^processor' /proc/cpuinfo | wc -l
2
```

in both cases the operating system will report TWO cpu’s



```
top - 16:53:39 up  1:07,  2 users,  load average: 0.05, 0.03, 0.00
Tasks:  92 total,   1 running,  91 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.2 sy,  0.0 ni, 99.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.2 st
KiB Mem :  4050652 total,  3573436 free,   120792 used,   356424 buff/cache
KiB Swap:  1048572 total,  1048572 free,        0 used.  3707452 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
  461 root      20   0   35800   1904   1664 S   0.3  0.0   0:00.12 irqbalance
 1115 root      20   0       0      0      0 S   0.3  0.0   0:00.07 kworker/1:1
    1 root      20   0  139004   6832   5272 S   0.0  0.2   0:00.87 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.00 kthreadd
    4 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/0:0H
    6 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/0:1H
    7 root      20   0       0      0      0 S   0.0  0.0   0:00.00 ksoftirqd/0
    8 root      20   0       0      0      0 S   0.0  0.0   0:00.06 rcu_sched
    9 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcu_bh
   10 root      rt   0       0      0      0 S   0.0  0.0   0:00.00 migration/0
   11 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 lru-add-drain
   12 root      rt   0       0      0      0 S   0.0  0.0   0:00.01 watchdog/0
   13 root      20   0       0      0      0 S   0.0  0.0   0:00.00 cpuhp/0
   14 root      20   0       0      0      0 S   0.0  0.0   0:00.00 cpuhp/1
   15 root      rt   0       0      0      0 S   0.0  0.0   0:00.00 watchdog/1
   16 root      rt   0       0      0      0 S   0.0  0.0   0:00.06 migration/1
   17 root      20   0       0      0      0 S   0.0  0.0   0:00.00 ksoftirqd/1
```

this means the O/S will schedule input/output tasks on BOTH cpu’s

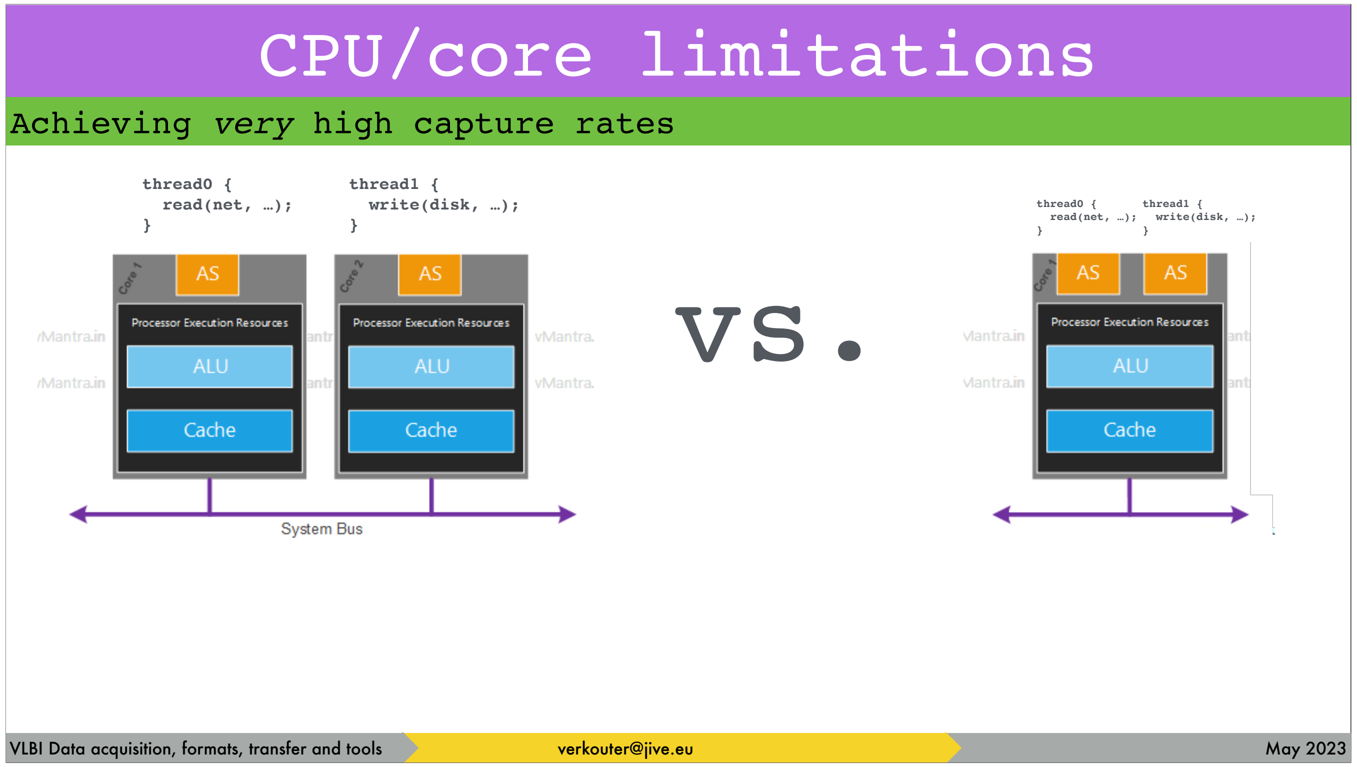


CPU/core limitations

Achieving *very* high capture rates

Dual Core Processor without Hyper-Threading
Core 1
AS
Processor Execution Resources
ALU
Cache
Core 2
AS
Processor Execution Resources
ALU
Cache
vMantra.in
System Bus
vs.
Single Core Processor with Hyper-Threading enabled
Core 1
AS
AS
Processor Execution Resources
ALU
Cache

but what you can see here is that in the hyperthreading case the "cpu's" share the cache, arithmetic unit and the connection to the systembus!



CPU/core limitations

Achieving *very* high capture rates

```
thread0 {
  read(net, …);
}
```

```
thread1 {
  write(disk, …);
}
```

Core 1
AS
Processor Execution Resources
ALU
Cache
Core 2
AS
Processor Execution Resources
ALU
Cache
vMantra.in
System Bus

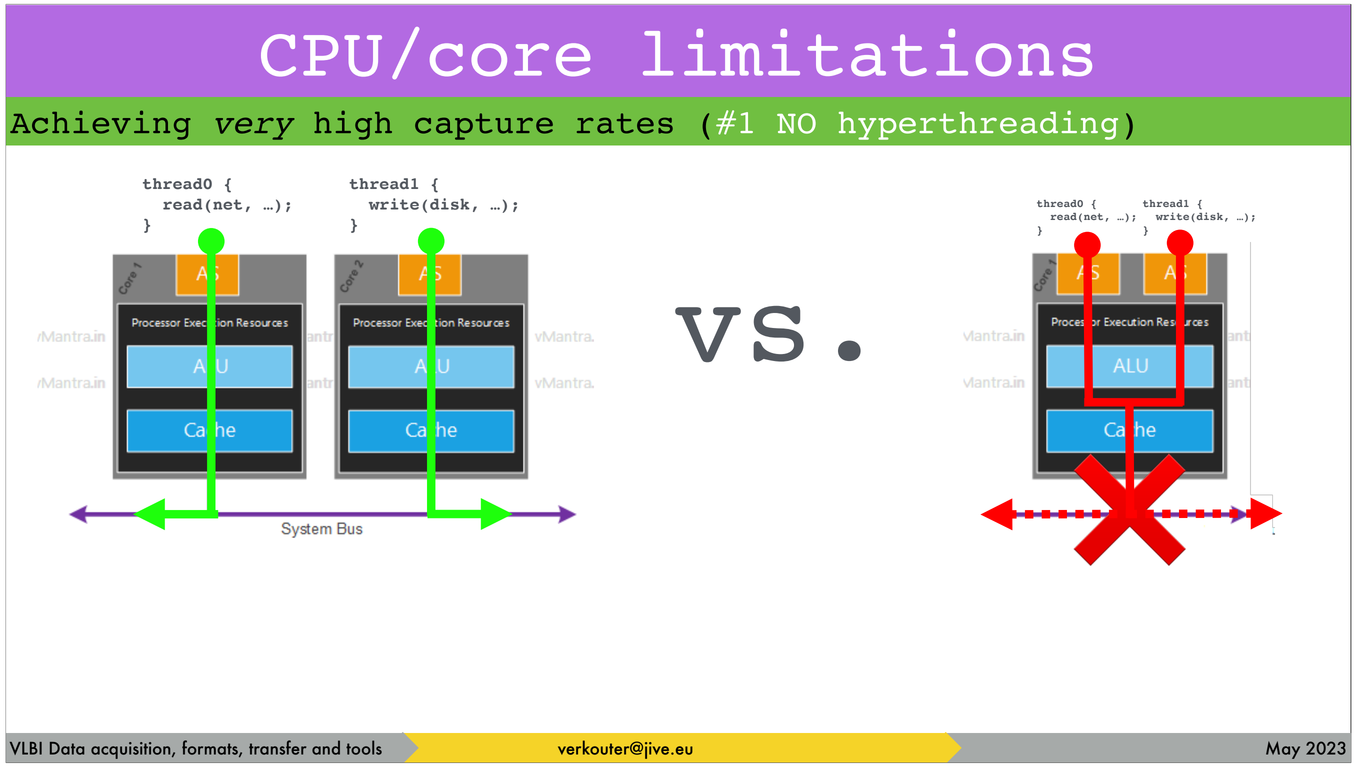
vs.

```
thread0 {
  read(net, …);
}
```

```
thread1 {
  write(disk, …);
}
```

Core 1
AS
AS
Processor Execution Resources
ALU
Cache
vMantra.in

so when multiple threads try to do I/O then



CPU/core limitations

Achieving *very* high capture rates (#1 NO hyperthreading)

```
thread0 {
  read(net, …);
}
```

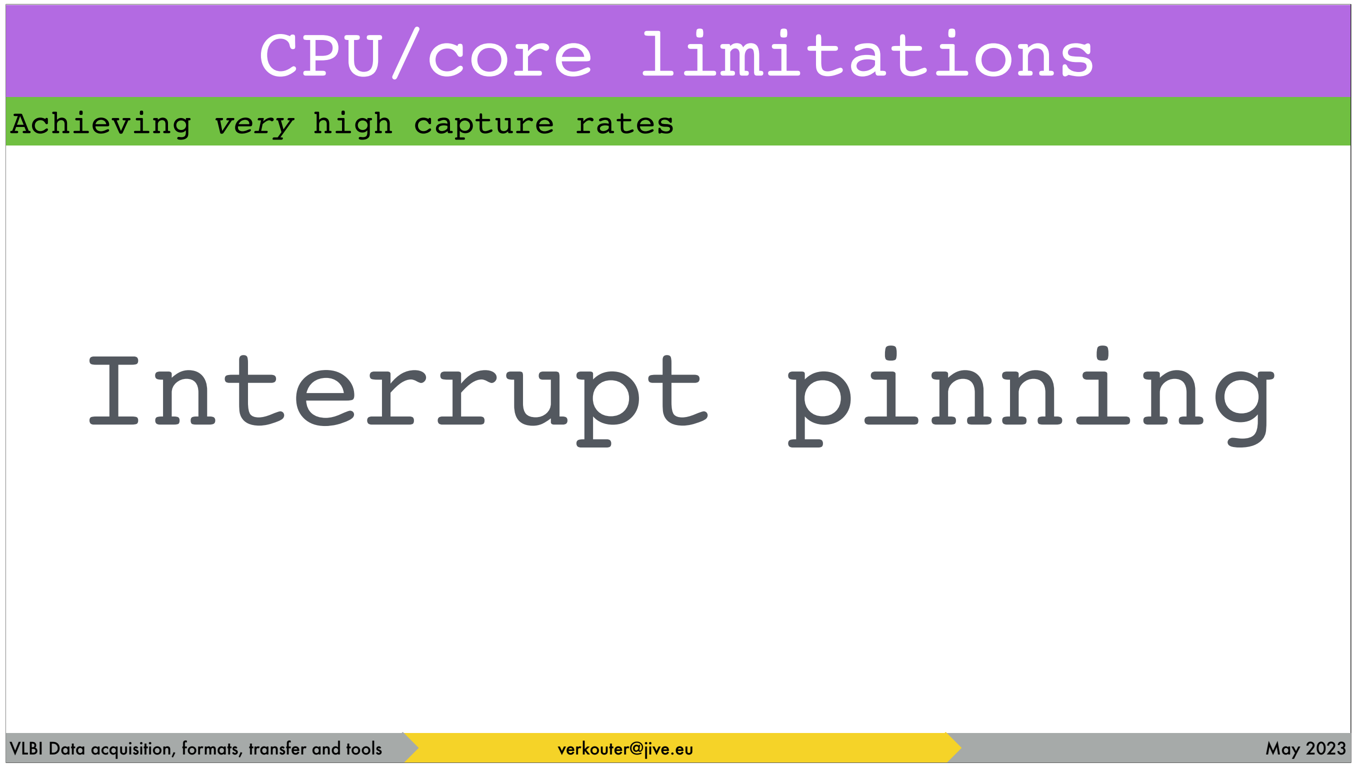
```
thread1 {
  write(disk, …);
}
```

vs.

```
thread0 {
  read(net, …);
}
```

```
thread1 {
  write(disk, …);
}
```

the threads on the hyperthreaded CPU will actually fight for the bus and end up with LESS performance!

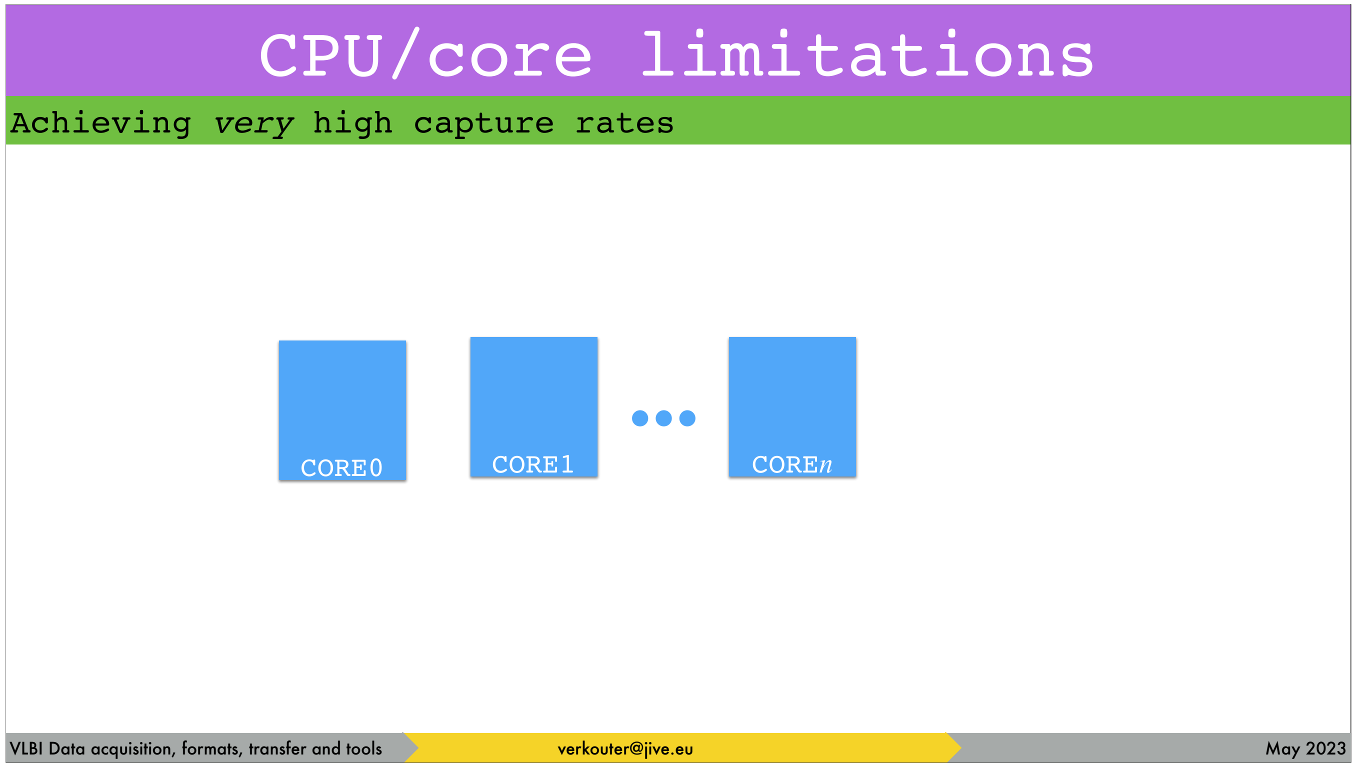


CPU/core limitations

Achieving *very* high capture rates

Interrupt pinning

Once you’re sure that no hyperthreading is enabled, THIS IS THE KILLER that HAS to be done.



CPU/core limitations

Achieving *very* high capture rates

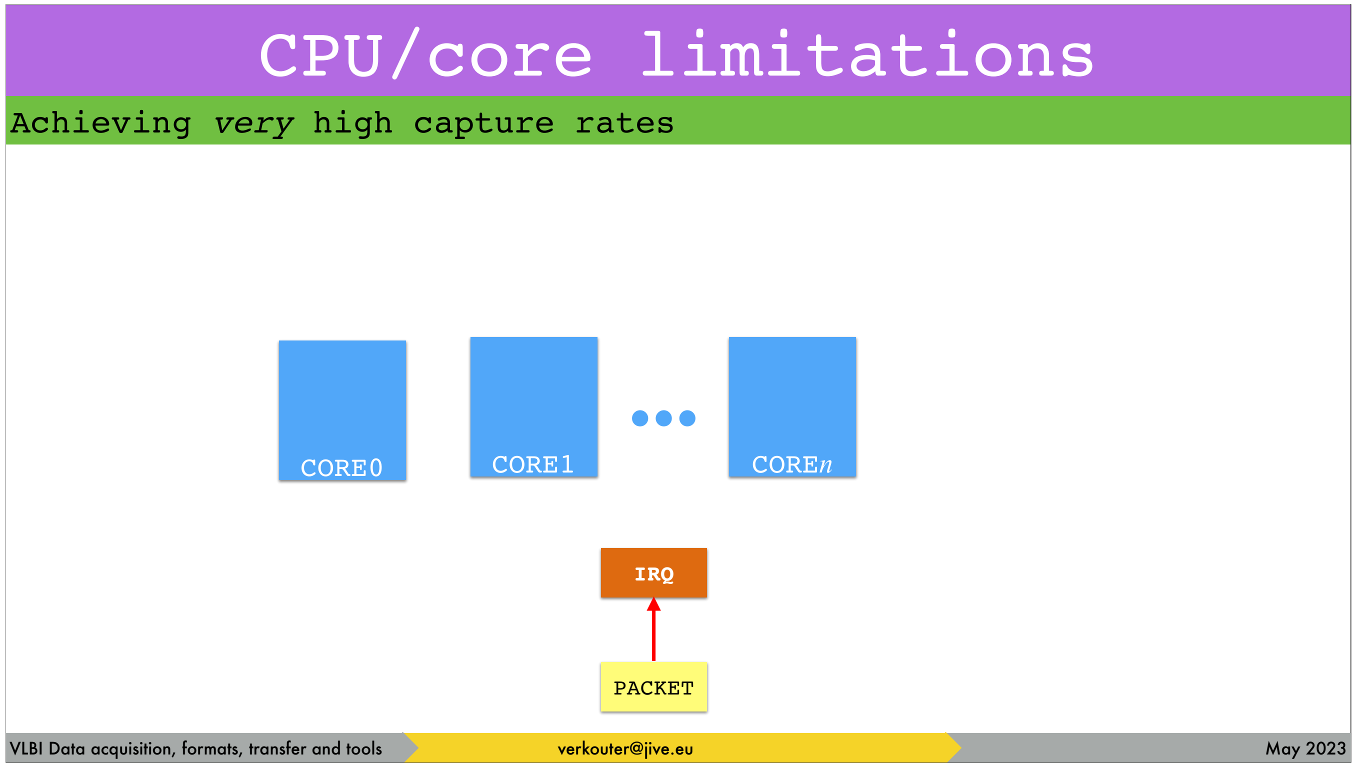
CORE0

CORE1

•••

CORE*n*

let’s look at what the linux kernel does by default on our multi-core machine.



CPU/core limitations

Achieving *very* high capture rates

CORE0

CORE1

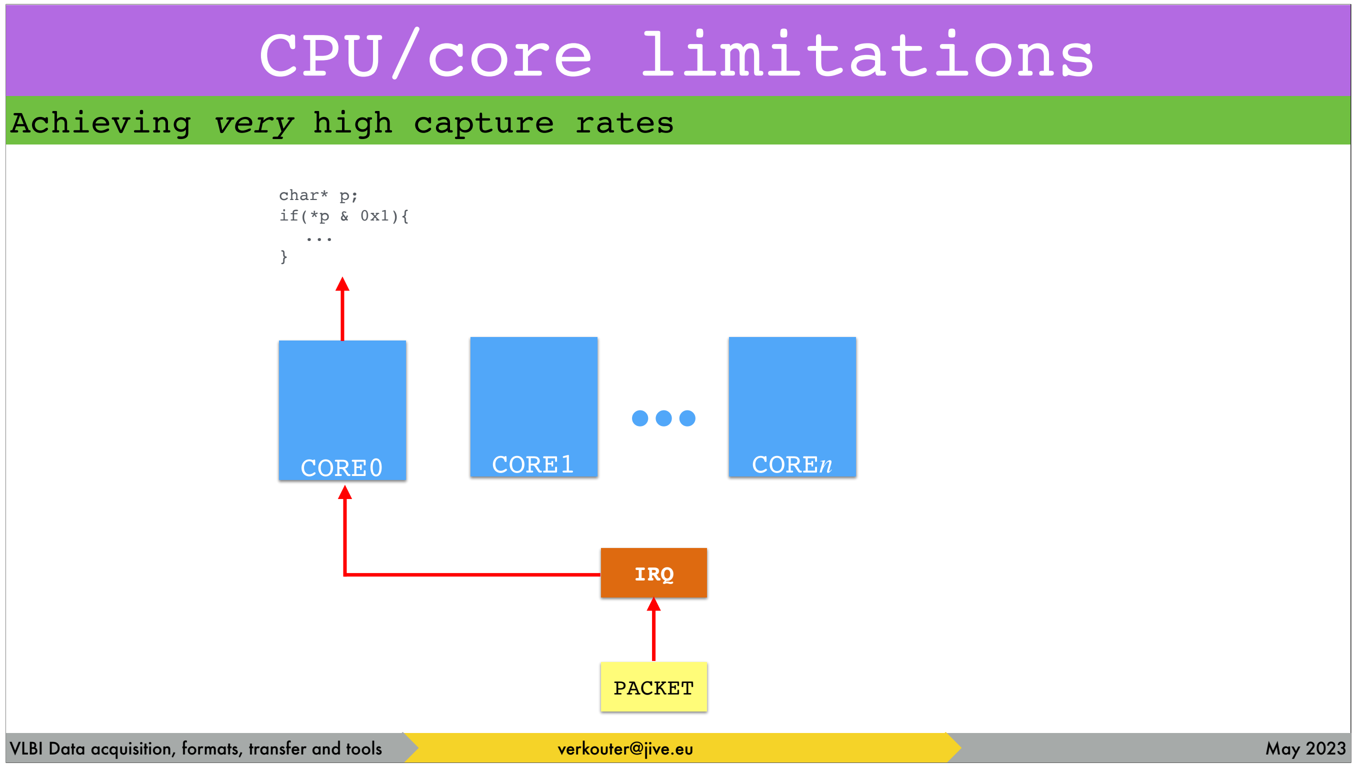
•••

CORE*n*

IRQ

PACKET

If a packet arrives [click] it generates an interrupt

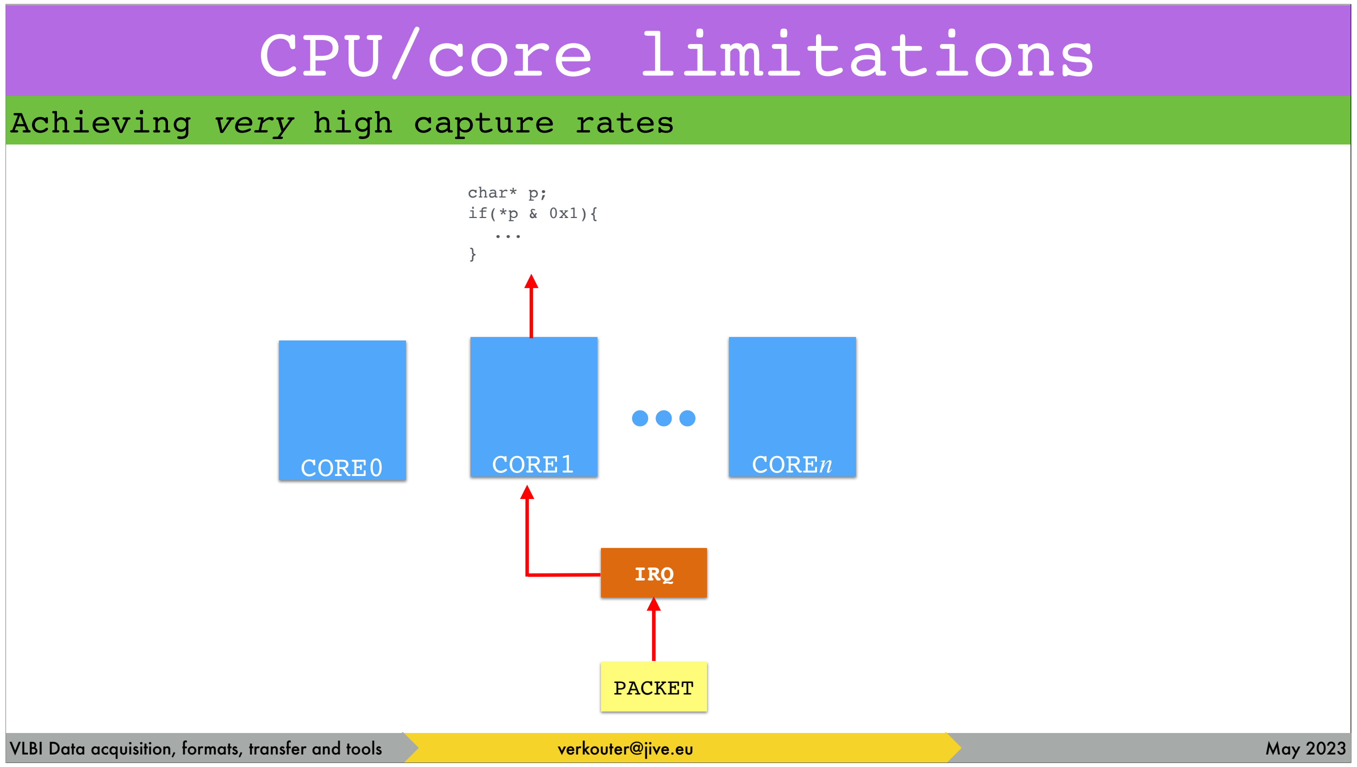


CPU/core limitations

Achieving *very* high capture rates

```
char* p;
if(*p & 0x1){
   ...
}
```

Linux chooses an available core [click] to handle the interrupt and [click] triggers the interrupt handler code to run.

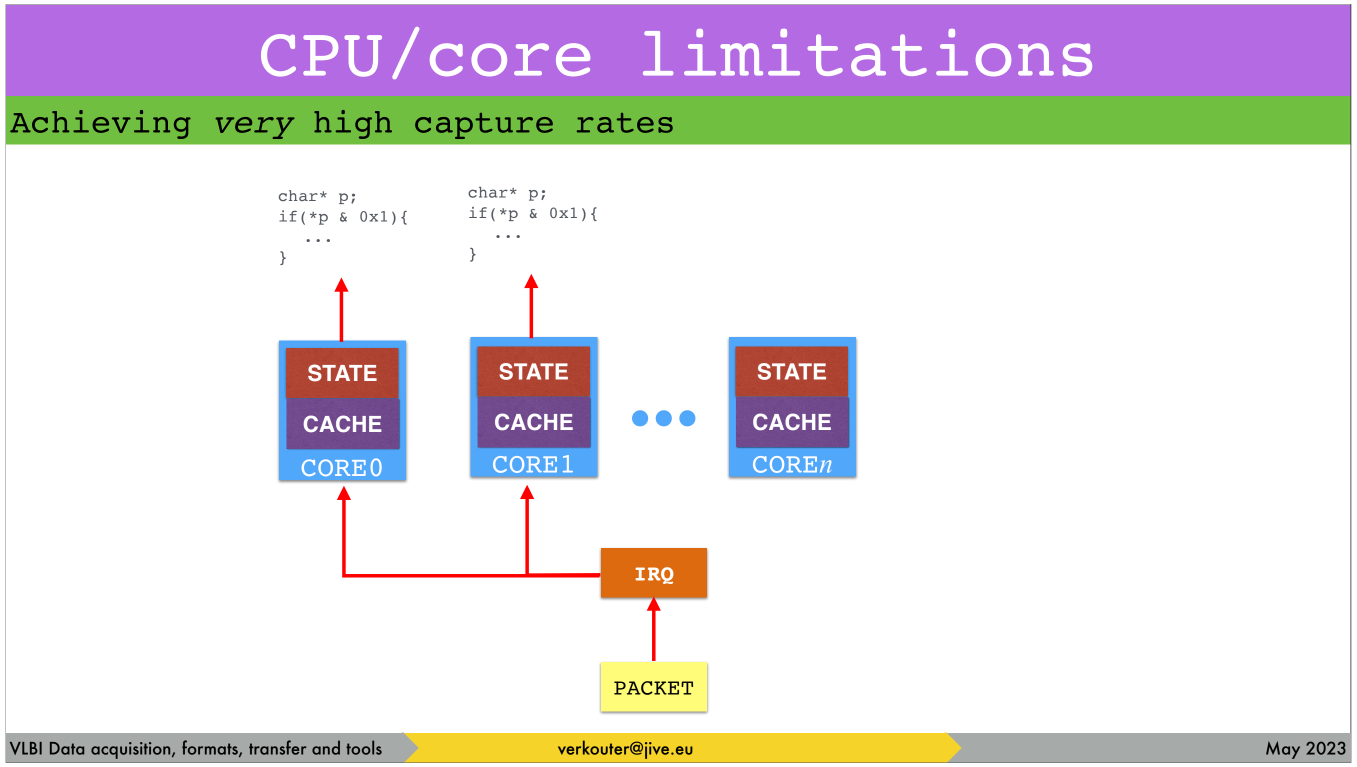


CPU/core limitations

Achieving *very* high capture rates

```
char* p;
if(*p & 0x1){
   ...
}
```

When a next packet arrives Linux chooses another available core to handle the interrupt. This sounds OK, it gives a really good performance on a diverse work load … except it’s not the whole story.



CPU/core limitations

Achieving *very* high capture rates

```
char* p;
if(*p & 0x1){
   ...
}
```

```
char* p;
if(*p & 0x1){
   ...
}
```

STATE

CACHE

CORE0

STATE

CACHE

CORE1

●●●

STATE

CACHE

CORE*n*

IRQ

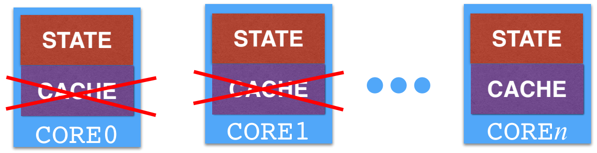
PACKET

VLBI Data acquisition, formats, transfer and tools verkouter@jive.eu May 2023

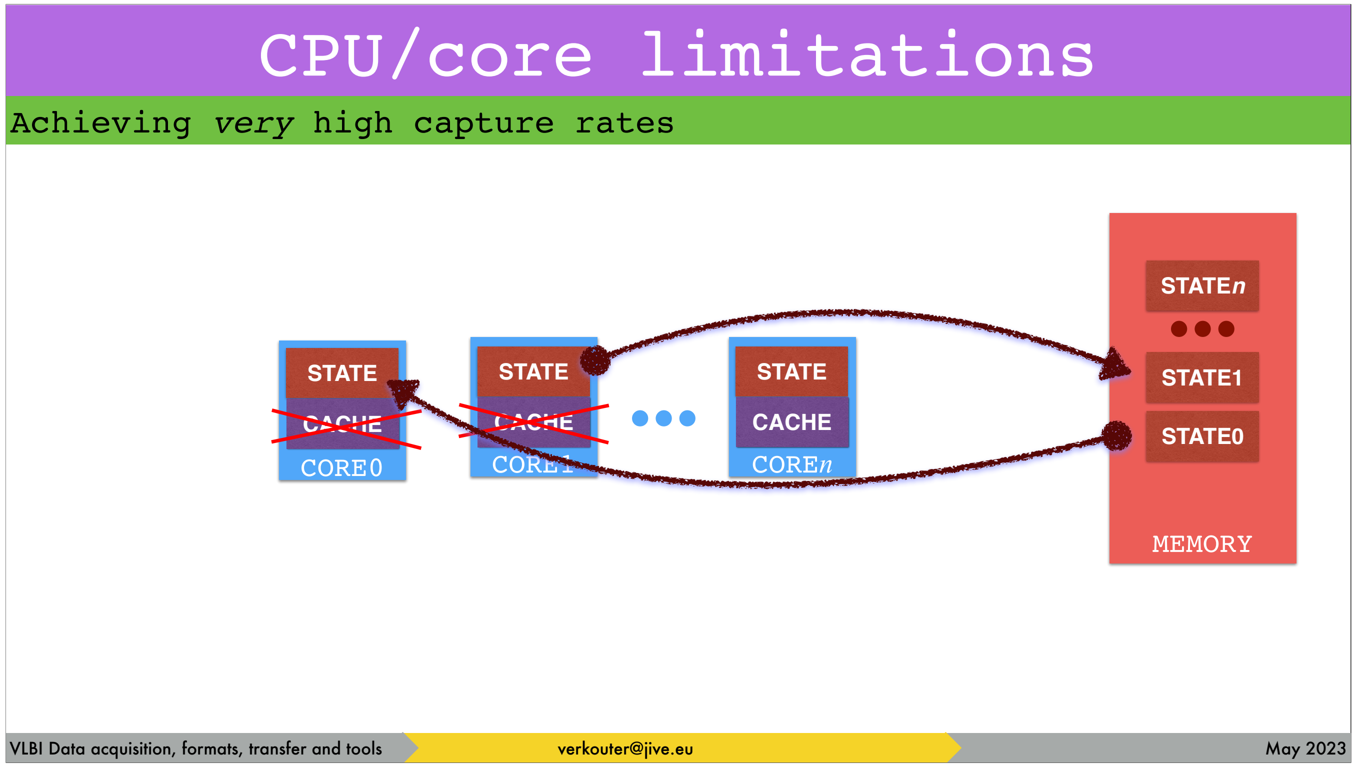
CPU cores have caches and state. What happens if a bit of code is migrated [click] from one core to the next, such as our interrupt handler,

CPU/core limitations

Achieving *very* high capture rates



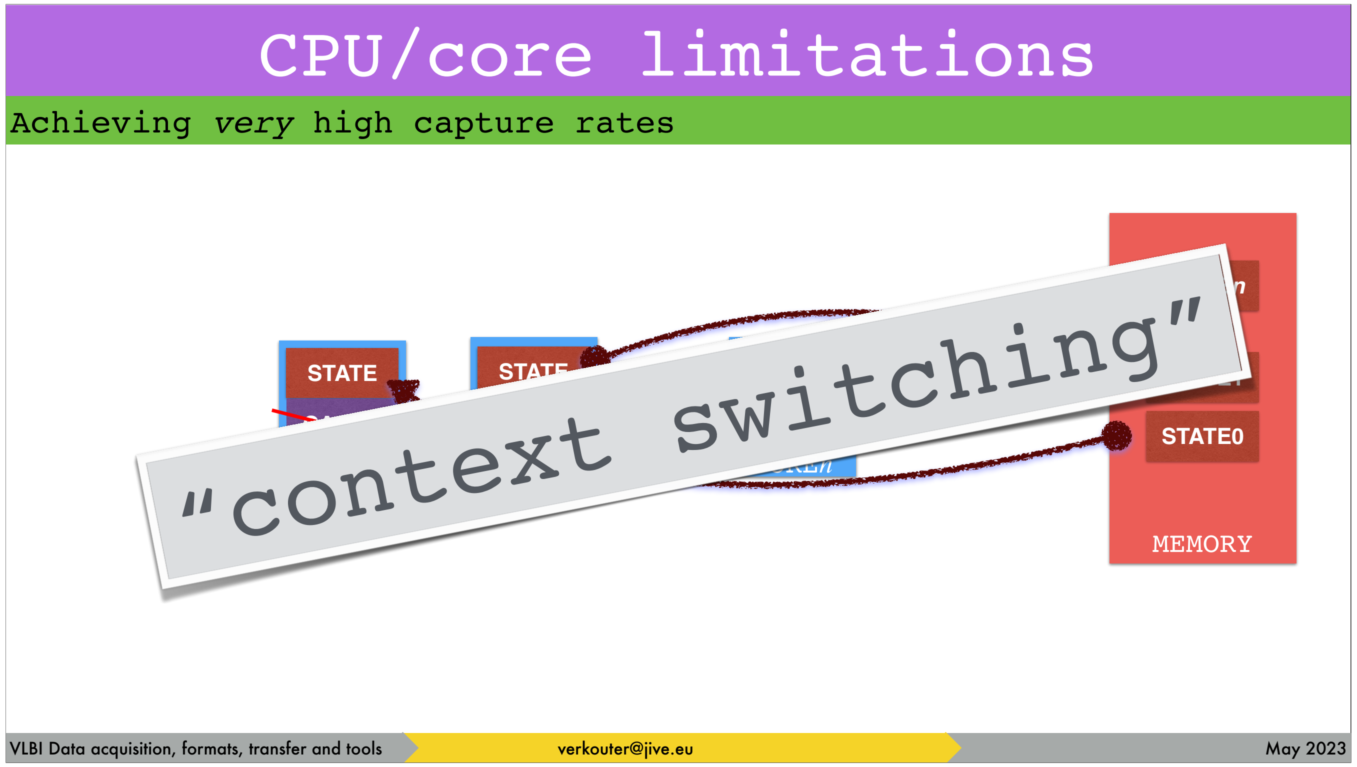
the cache of both cores gets invalidated



CPU/core limitations

Achieving *very* high capture rates

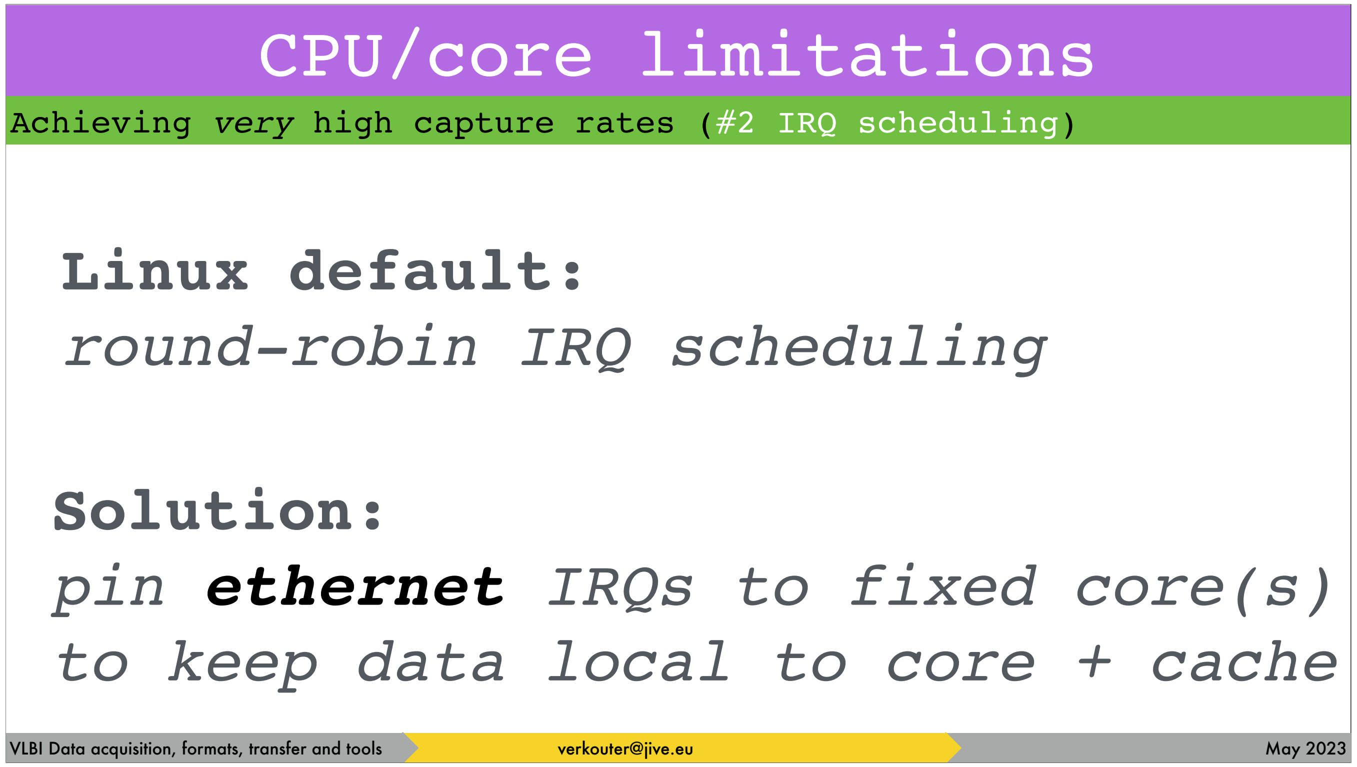
… and the CPU states have to be stored and loaded.



CPU/core limitations

Achieving *very* high capture rates

This is called context switching and it’s a very expensive operation. So expensive that this inhibits capturing more than a few gigabits per second if not addressed.



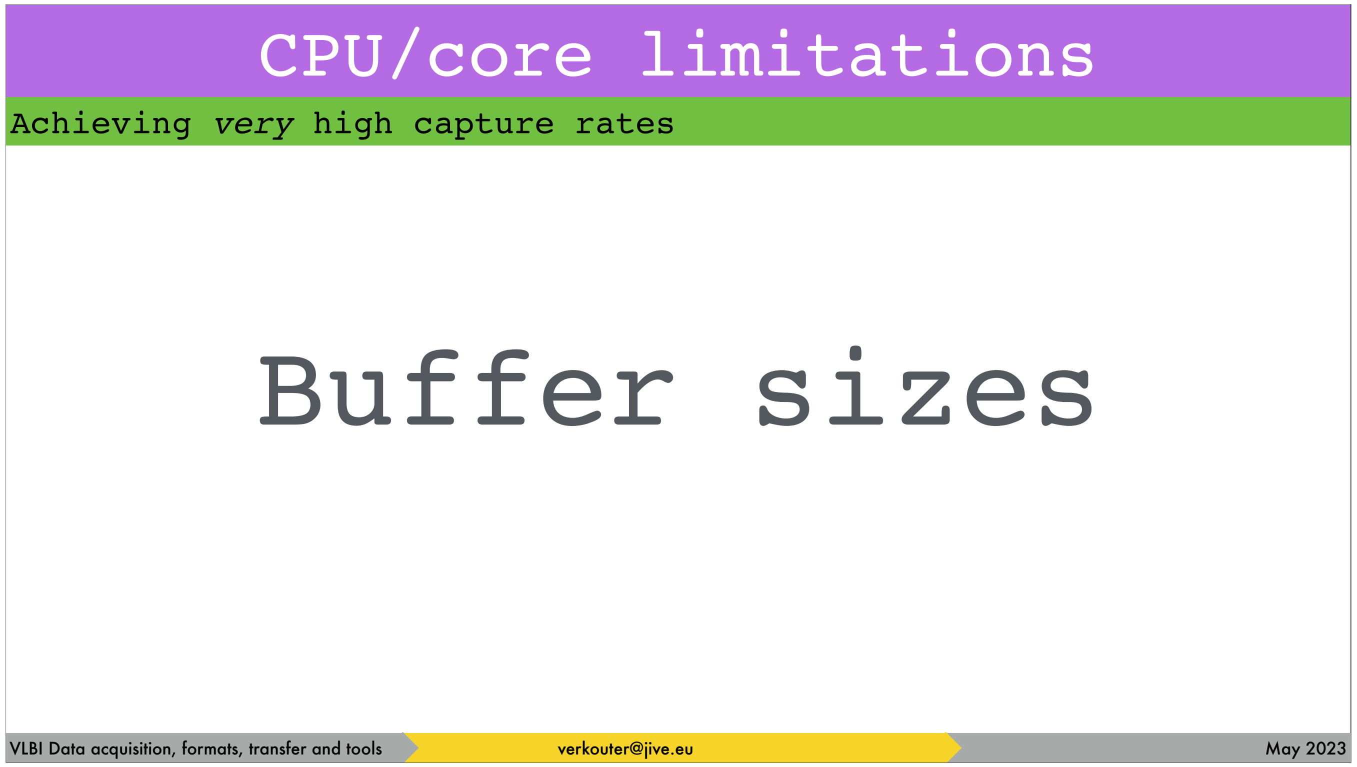
CPU/core limitations

Achieving *very* high capture rates (#2 IRQ scheduling)

Linux default:
round-robin IRQ scheduling

Solution:
pin ***ethernet*** *IRQs to fixed core(s)*
to keep data local to core + cache

It is most important to fix this for the ethernet card interrupts

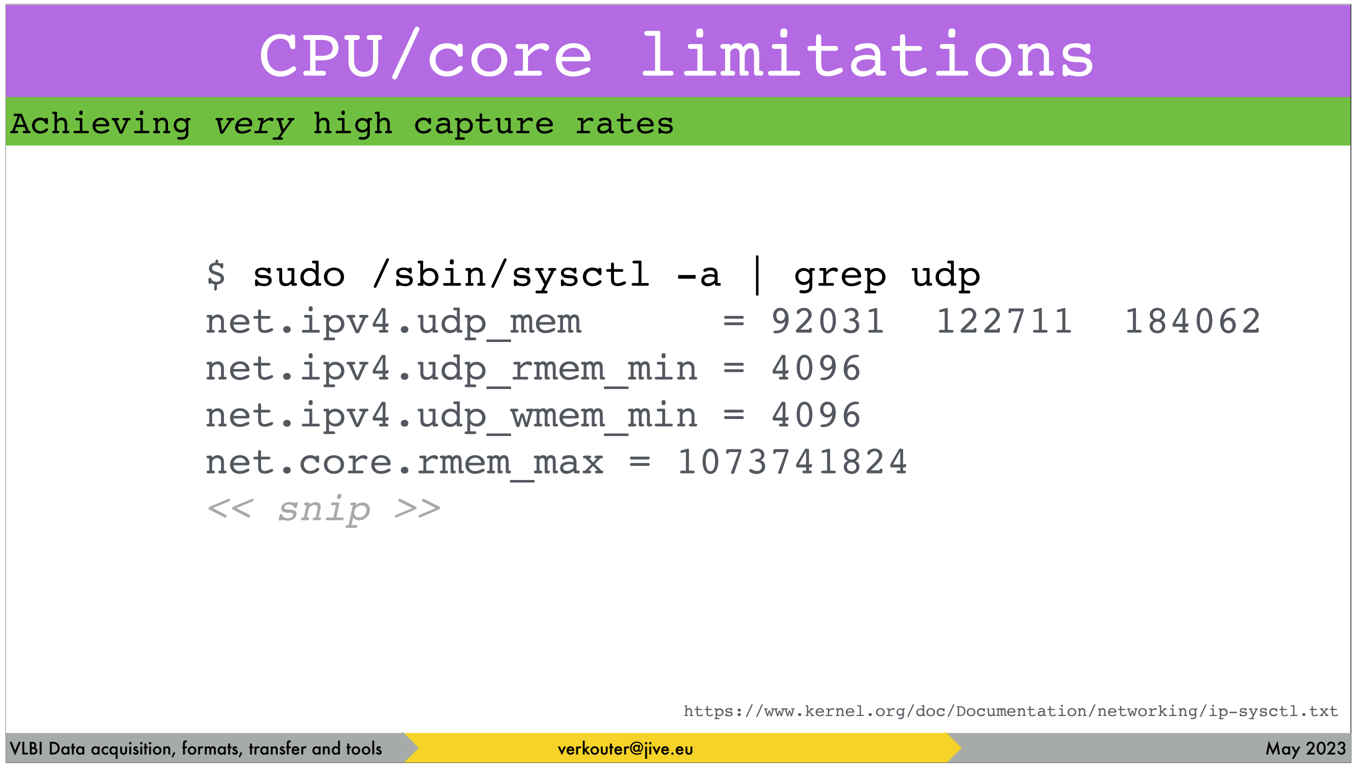


CPU/core limitations

Achieving *very* high capture rates

Buffer sizes

Another VERY important issue is tuning buffer sizes



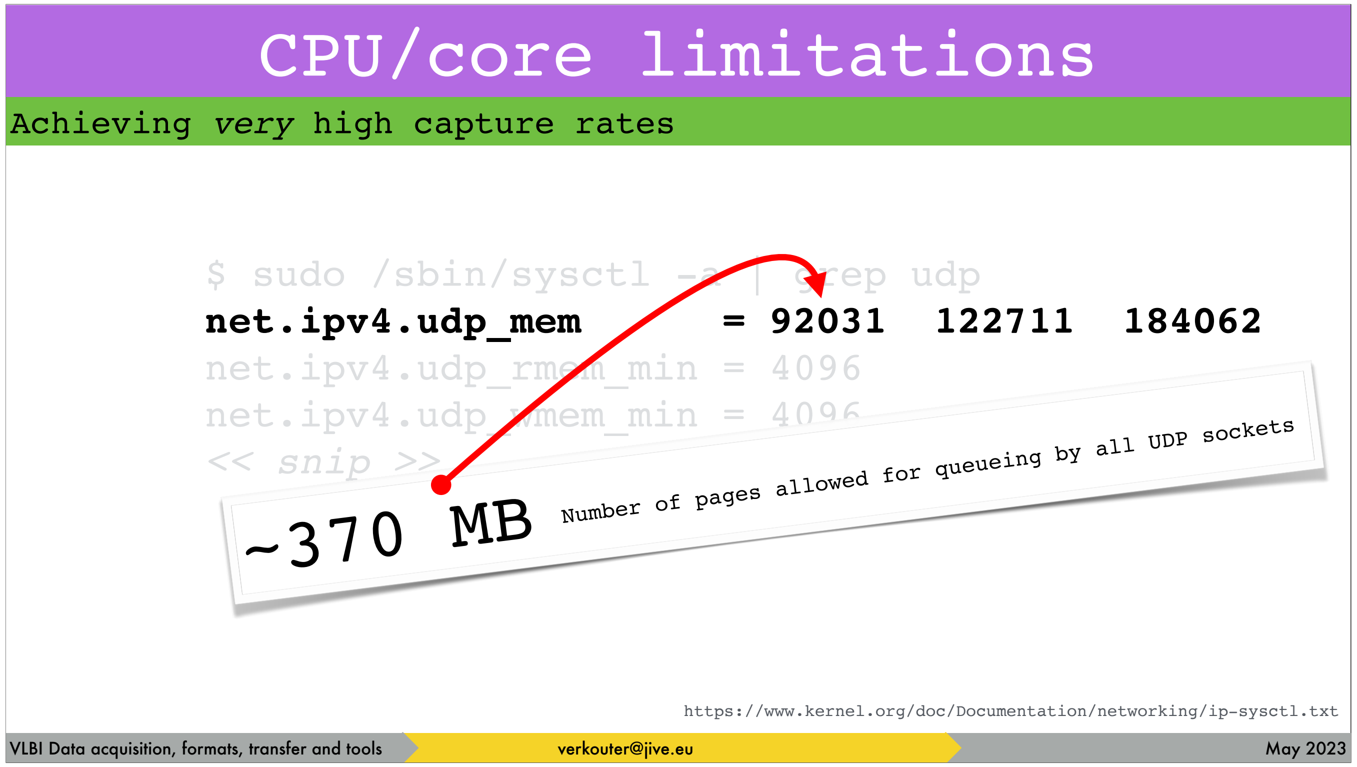
CPU/core limitations

Achieving *very* high capture rates

```
$ sudo /sbin/sysctl -a | grep udp
net.ipv4.udp_mem       = 92031  122711  184062
net.ipv4.udp_rmem_min = 4096
net.ipv4.udp_wmem_min = 4096
net.core.rmem_max = 1073741824
<< snip >>
```

https://www.kernel.org/doc/Documentation/networking/ip-sysctl.txt

Linux defaults are _pathetic_ for high speed UDP packet capturing!



CPU/core limitations

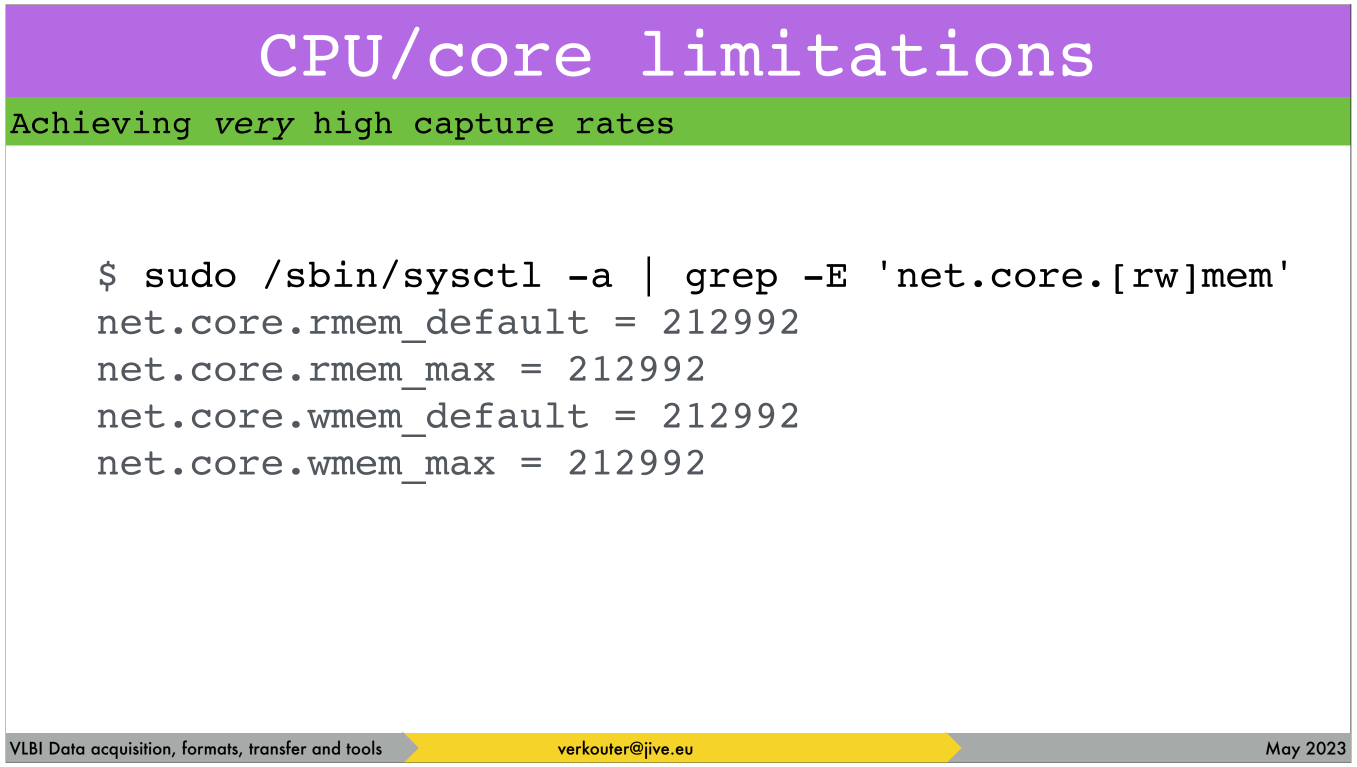
Achieving *very* high capture rates

```
$ sudo /sbin/sysctl -a | grep udp
net.ipv4.udp_mem      = 92031   122711   184062
net.ipv4.udp_rmem_min = 4096
net.ipv4.udp_wmem_min = 4096
<< snip >>
```

~370 MB Number of pages allowed for queueing by all UDP sockets

https://www.kernel.org/doc/Documentation/networking/ip-sysctl.txt

This one needs to be cranked up a lot to hundreds of megabytes even gigabytes. 300 MB is about half a second at 4 Gbps

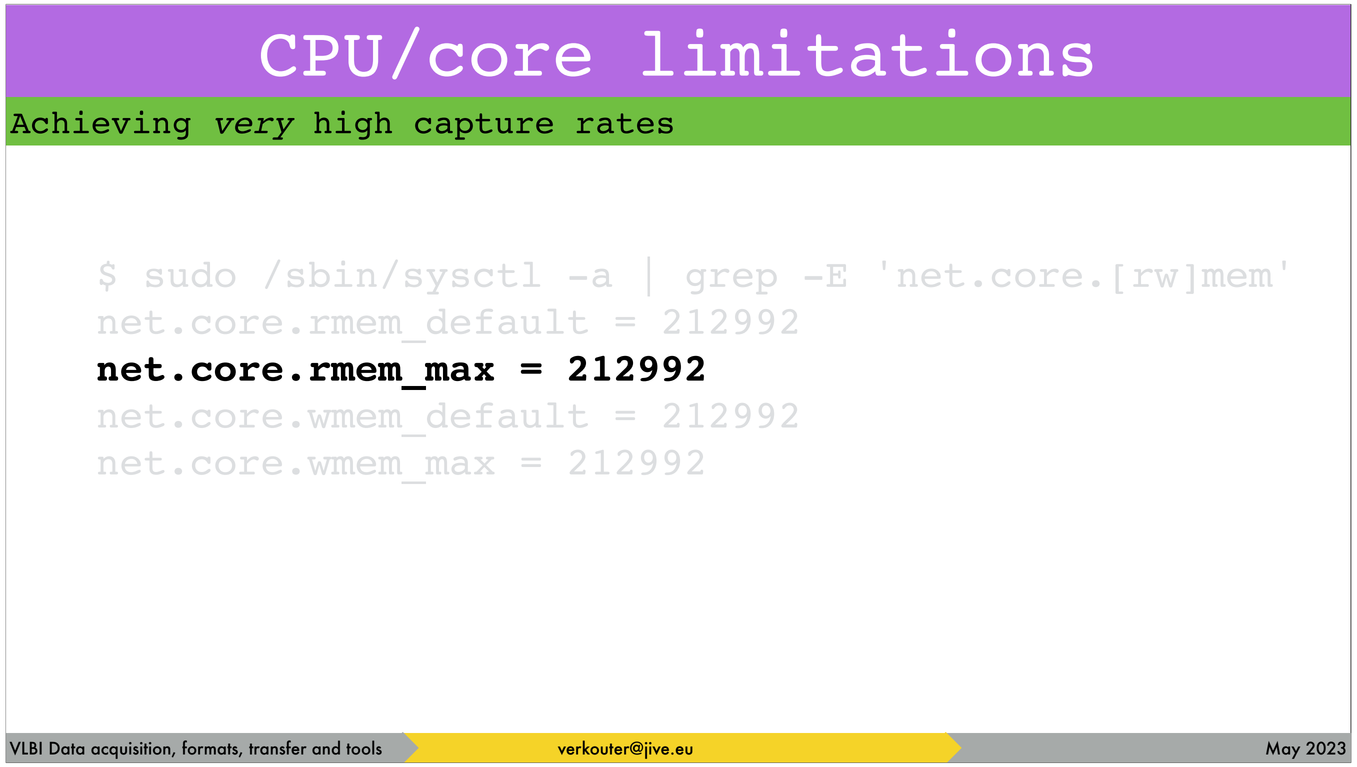


CPU/core limitations

Achieving *very* high capture rates

```
$ sudo /sbin/sysctl -a | grep -E 'net.core.[rw]mem'
net.core.rmem_default = 212992
net.core.rmem_max = 212992
net.core.wmem_default = 212992
net.core.wmem_max = 212992
```

The previous buffer was UDP specific. There are also network-wide buffer sizings

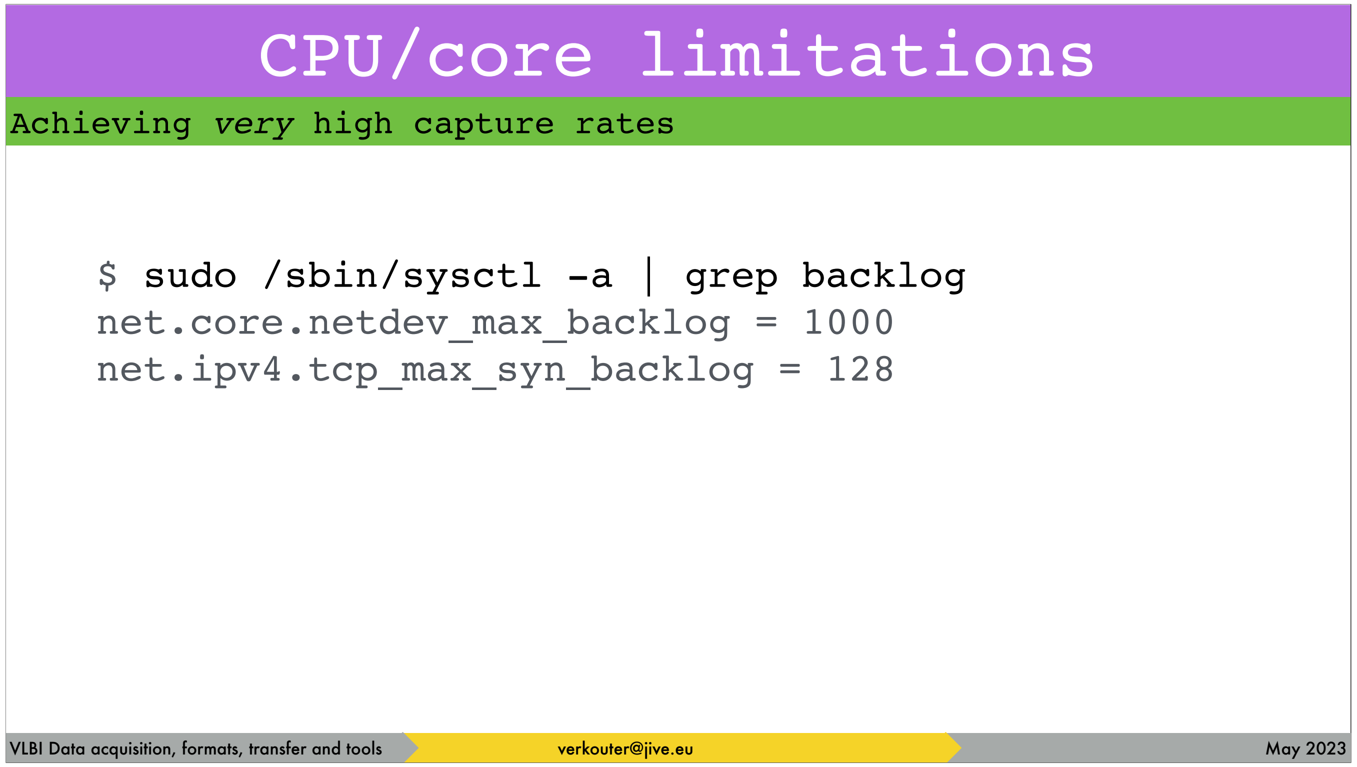


CPU/core limitations

Achieving *very* high capture rates

```
$ sudo /sbin/sysctl -a | grep -E 'net.core.[rw]mem'
net.core.rmem_default = 212992
net.core.rmem_max = 212992
net.core.wmem_default = 212992
net.core.wmem_max = 212992
```

for receiving this parameter is VERY important to increase

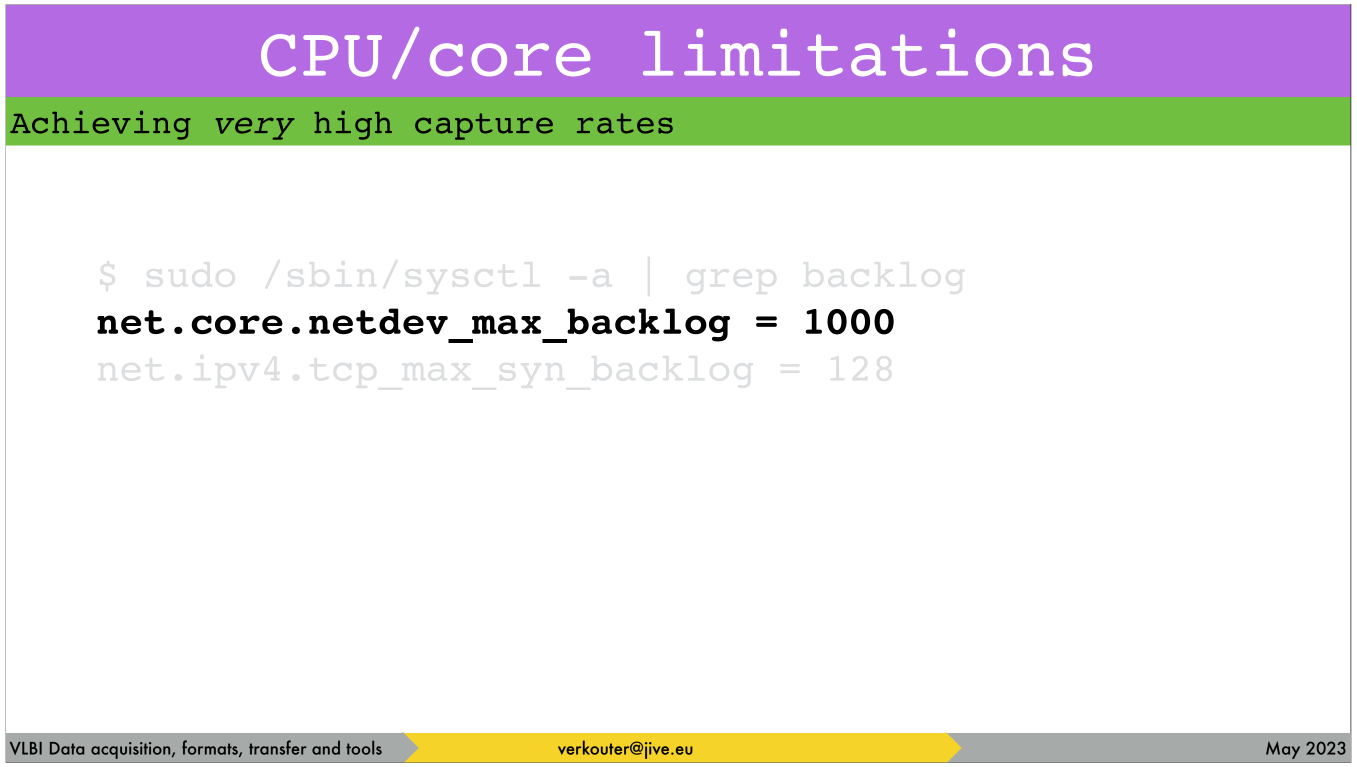


CPU/core limitations

Achieving *very* high capture rates

```
$ sudo /sbin/sysctl -a | grep backlog
net.core.netdev_max_backlog = 1000
net.ipv4.tcp_max_syn_backlog = 128
```

The kernel also keeps a backlog of packets …

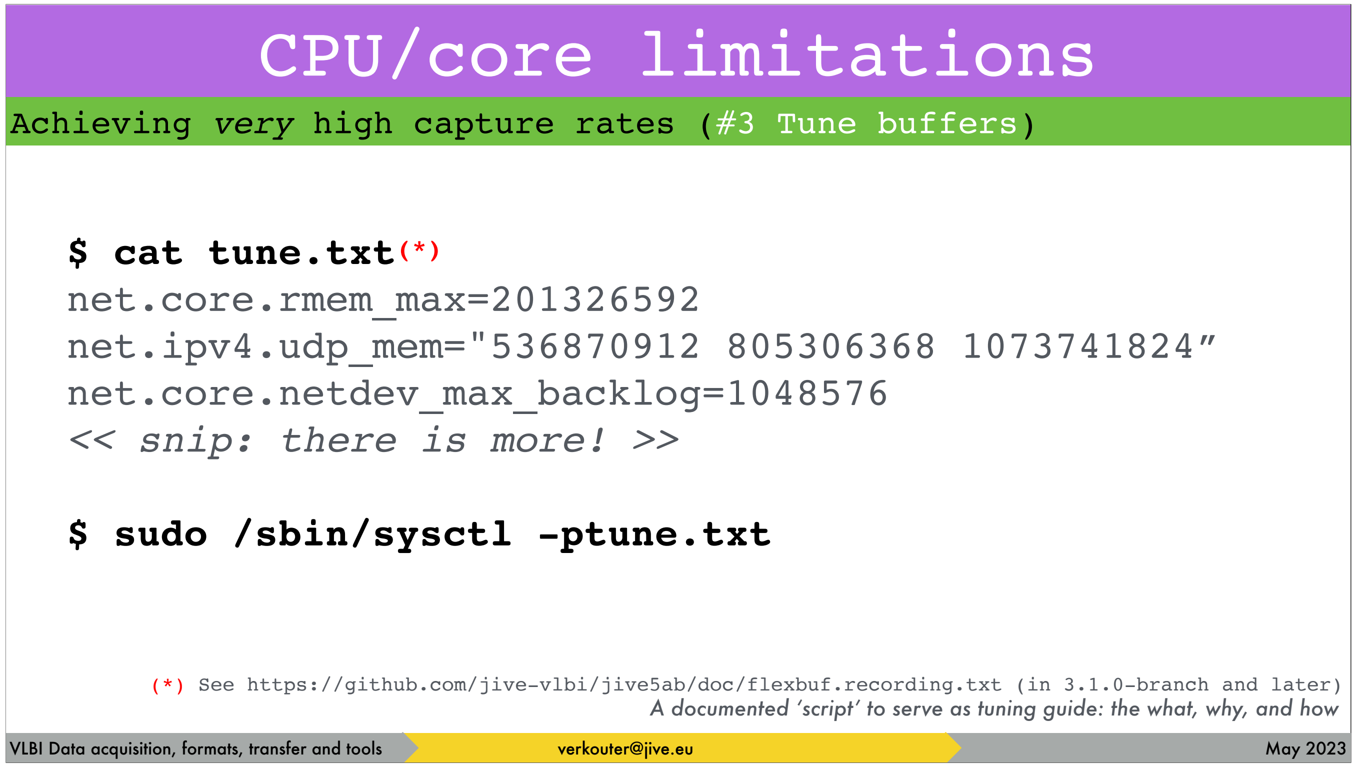


CPU/core limitations

Achieving *very* high capture rates

```
$ sudo /sbin/sysctl -a | grep backlog
net.core.netdev_max_backlog = 1000
net.ipv4.tcp_max_syn_backlog = 128
```

… which is extreeeeemely small!!!



CPU/core limitations

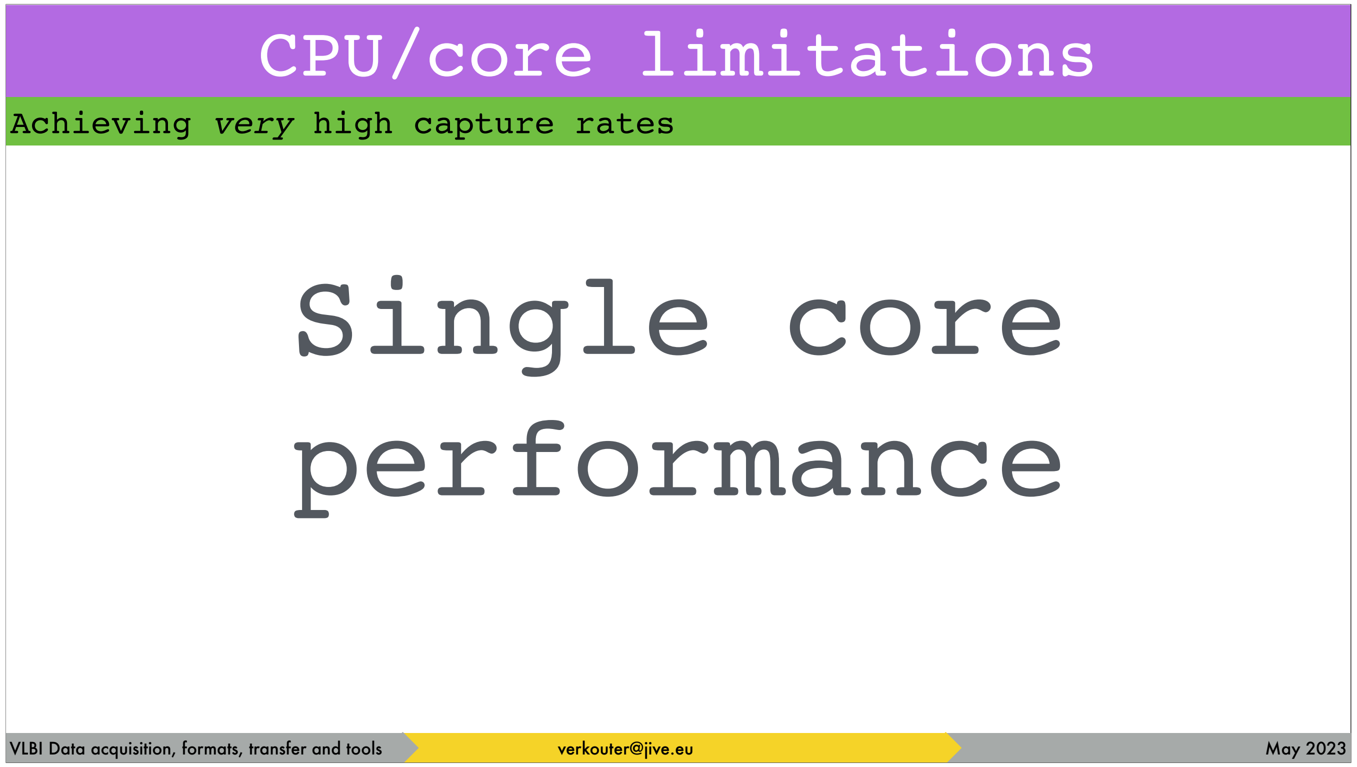
Achieving *very* high capture rates (#3 Tune buffers)

```
$ cat tune.txt(*)
net.core.rmem_max=201326592
net.ipv4.udp_mem="536870912 805306368 1073741824”
net.core.netdev_max_backlog=1048576
<< snip: there is more! >>

$ sudo /sbin/sysctl -ptune.txt
```

(*) See https://github.com/jive-vlbi/jive5ab/doc/flexbuf.recording.txt (in 3.1.0-branch and later)

A documented 'script' to serve as tuning guide: the what, why, and how

So it is important to tune up your system. Be sure to check the tuning document in jive5ab’s github repository

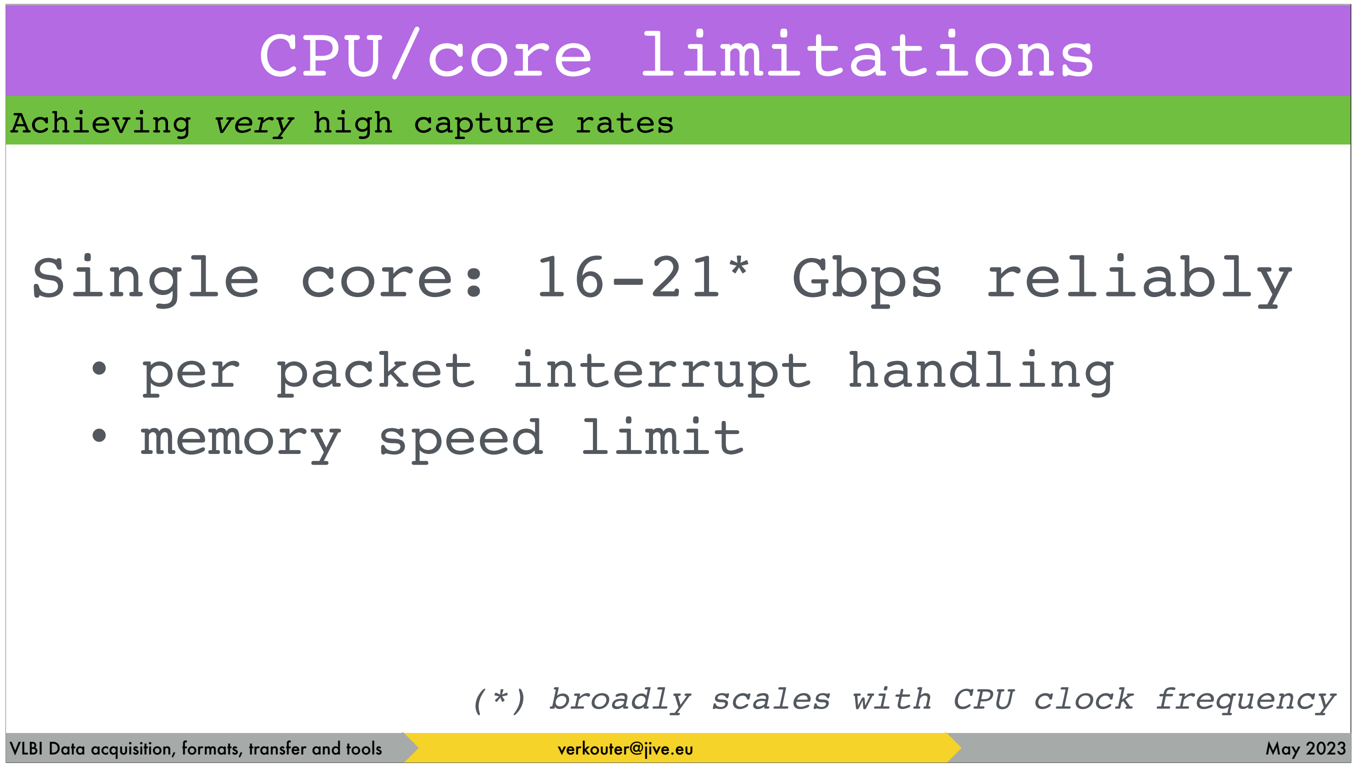


CPU/core limitations

Achieving *very* high capture rates

Single core performance

Believe it or not - eventually you’ll run into *this* bottleneck



CPU/core limitations

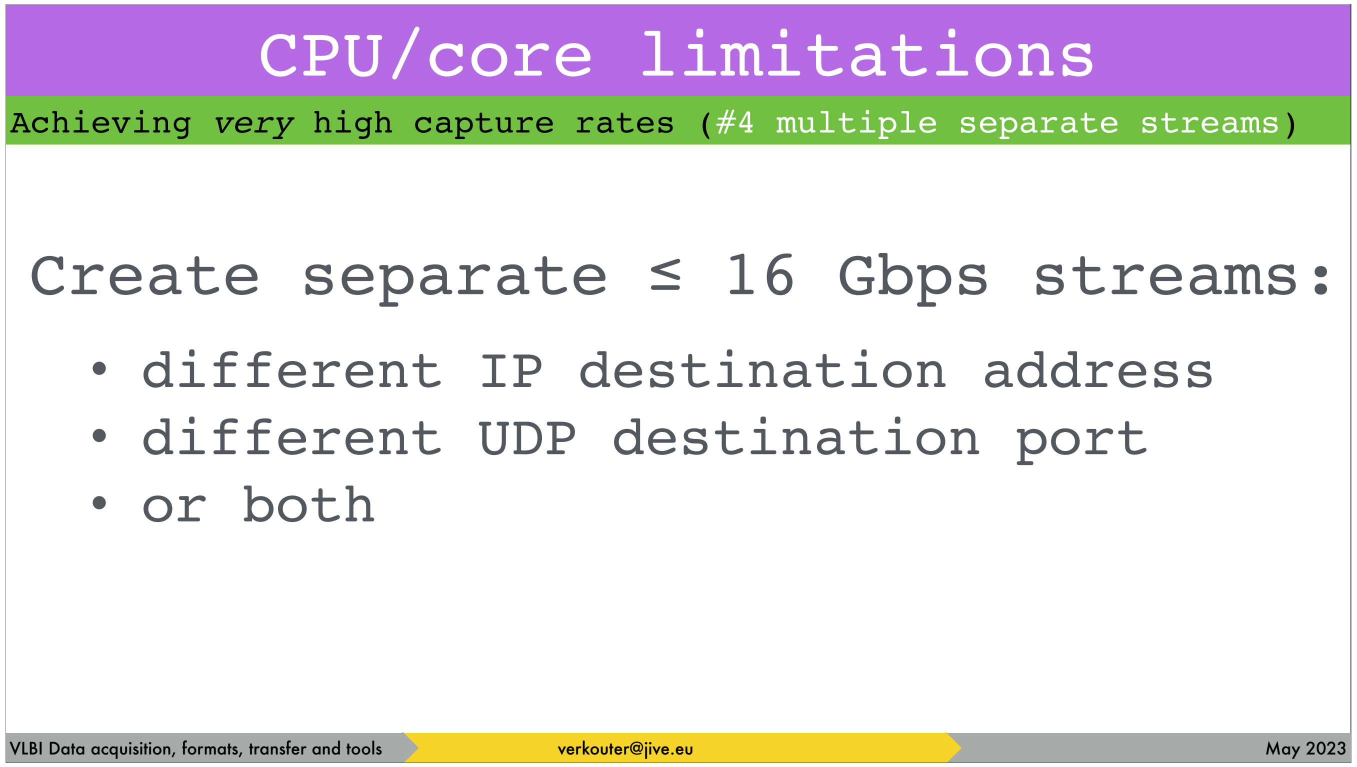
Achieving *very* high capture rates

Single core: 16-21* Gbps reliably

- per packet interrupt handling
- memory speed limit

(*) broadly scales with CPU clock frequency

Experience has shown that a single CPU core can handle on the order of 16 Gbps maximum - at some point either or both of these limits become the bottleneck



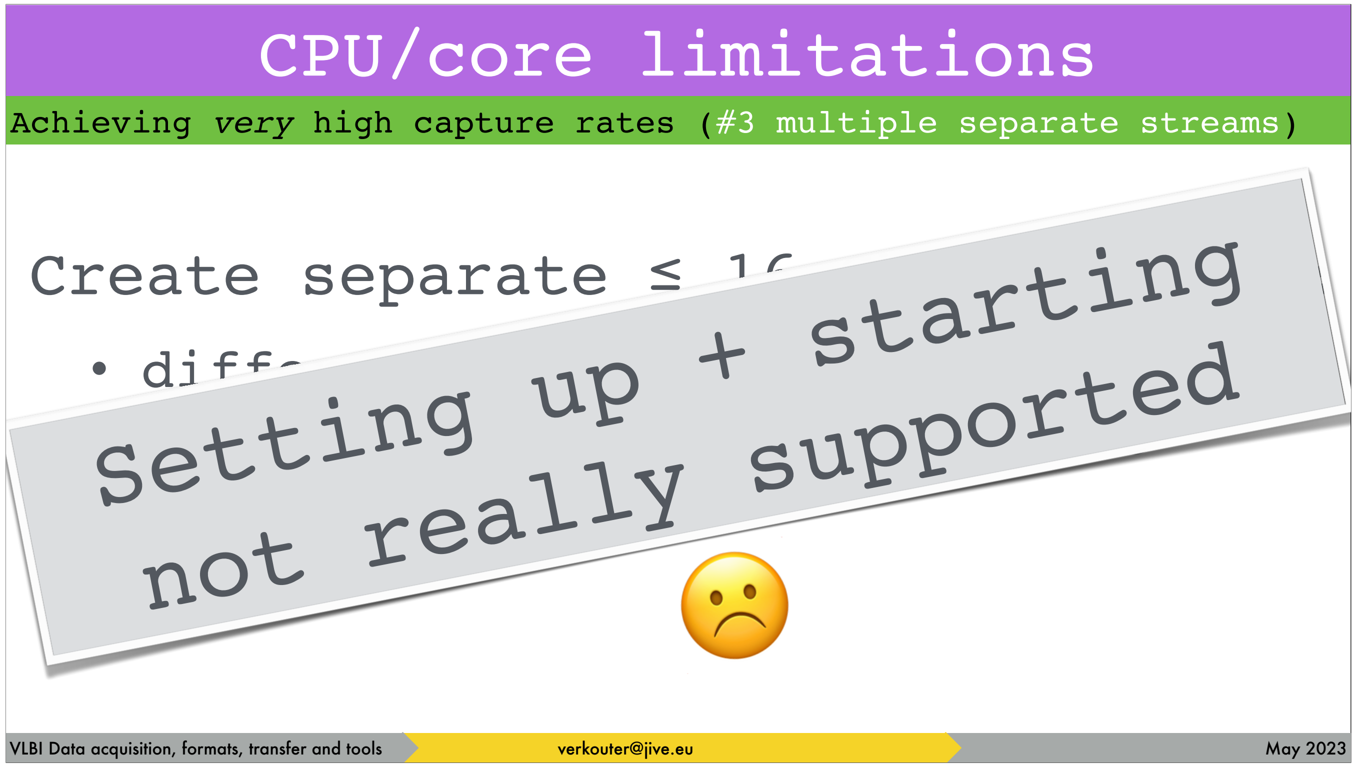
CPU/core limitations

Achieving *very* high capture rates (#4 multiple separate streams)

Create separate ≤ 16 Gbps streams:

- different IP destination address
- different UDP destination port
- or both

The solution is parallelization. You need to create multiple data streams, separating the traffic by destination address, port or both



CPU/core limitations

Achieving *very* high capture rates (#3 multiple separate streams)

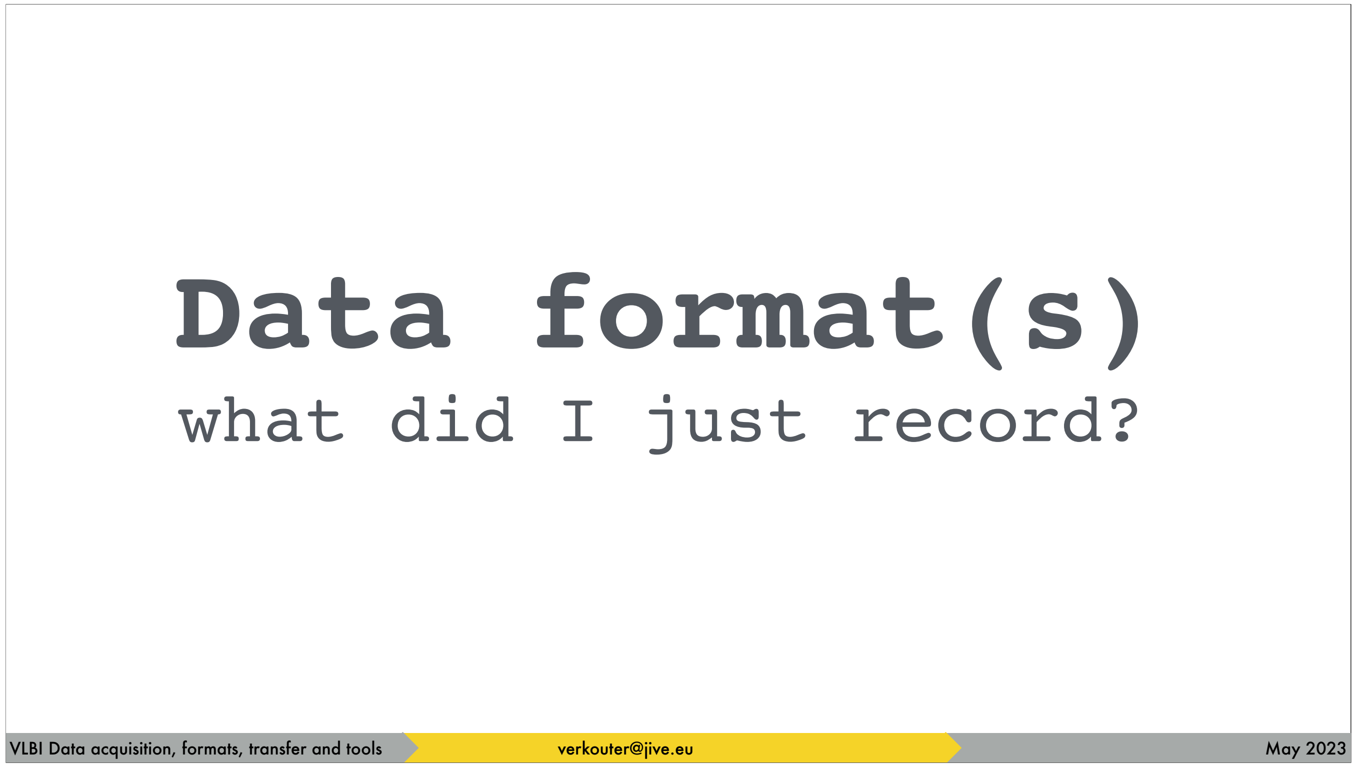
Create separate ≤ 1

- diff

Setting up + starting not really supported

☹️

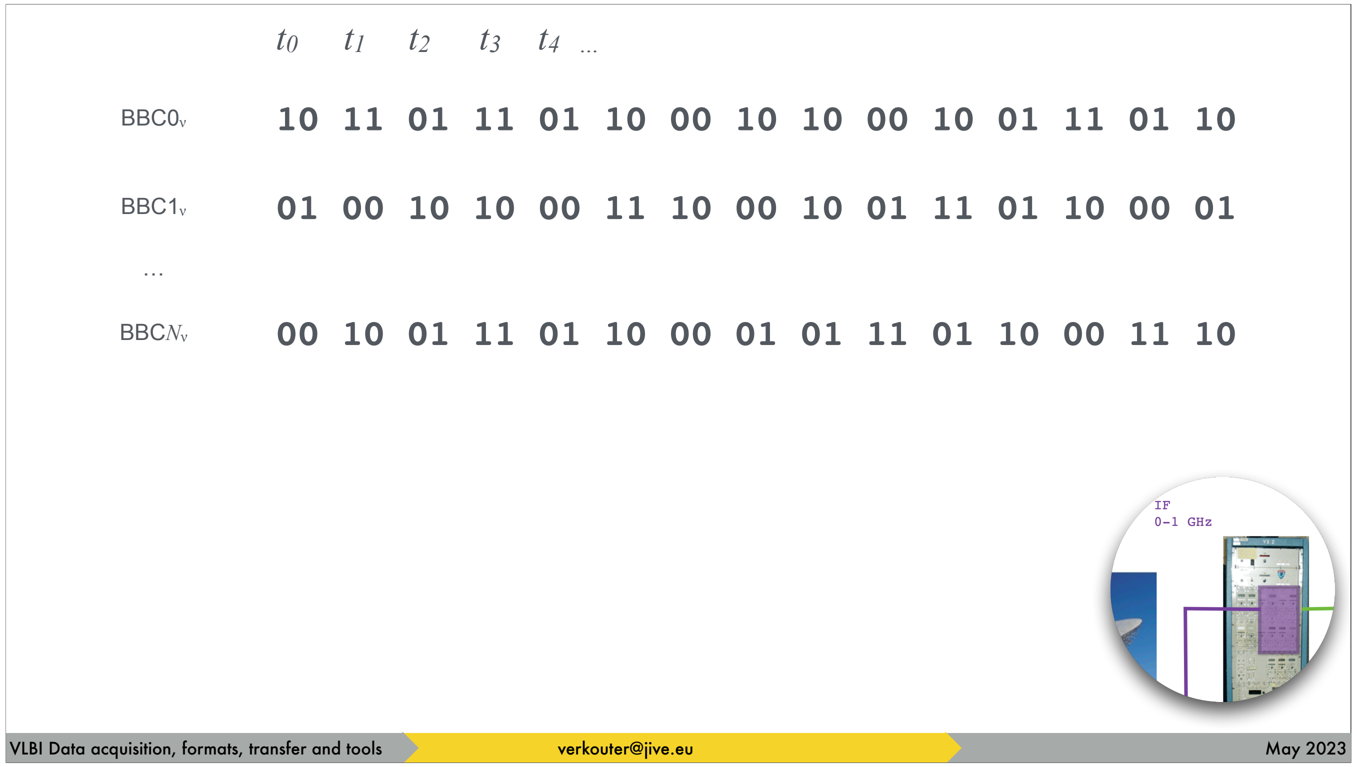
however, starting those is not quite well-supported



Data format(s)

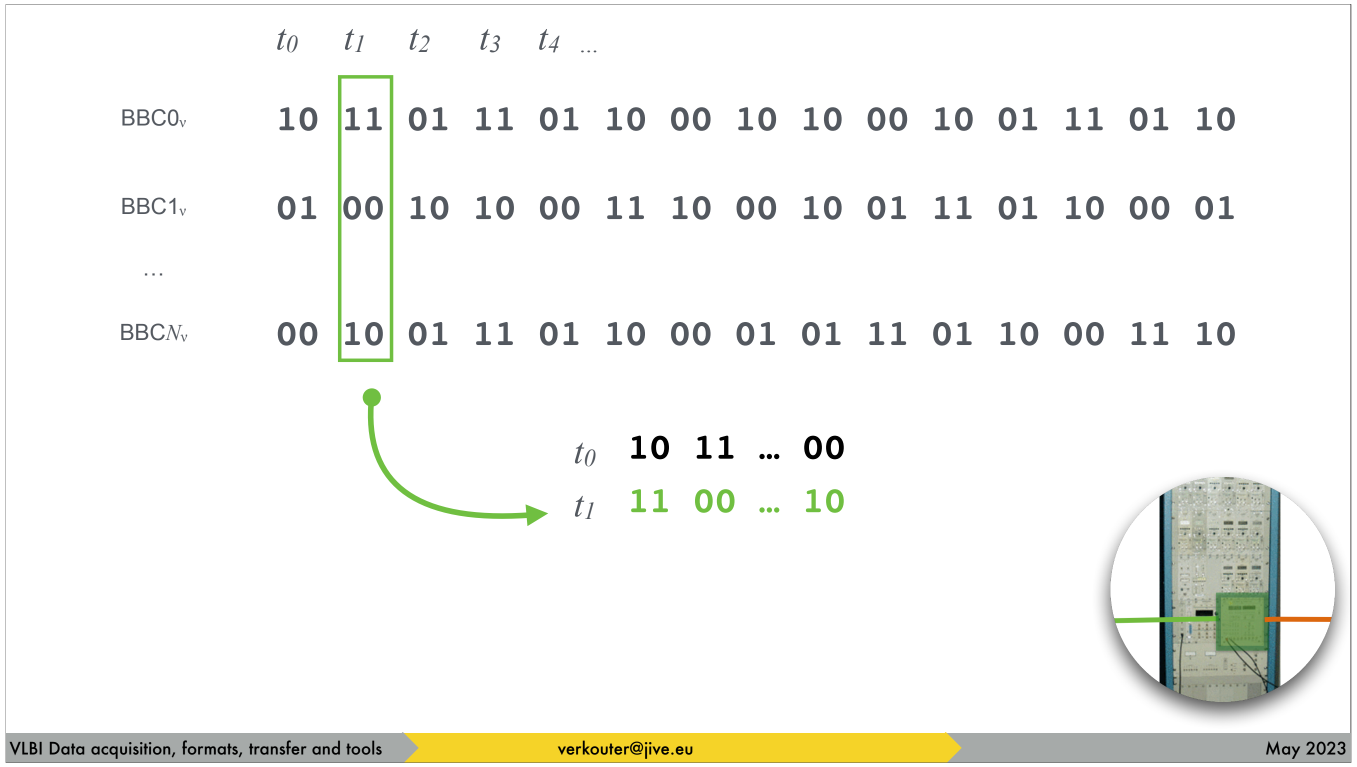
what did I just record?

As introduced in the beginning …



	t_0	t_1	t_2	t_3	t_4	…									
BBC0_v	10	11	01	11	01	10	00	10	10	00	10	01	11	01	10
BBC1_v	01	00	10	10	00	11	10	00	10	01	11	01	10	00	01
…															
BBCN_v	00	10	01	11	01	10	00	01	01	11	01	10	00	11	10

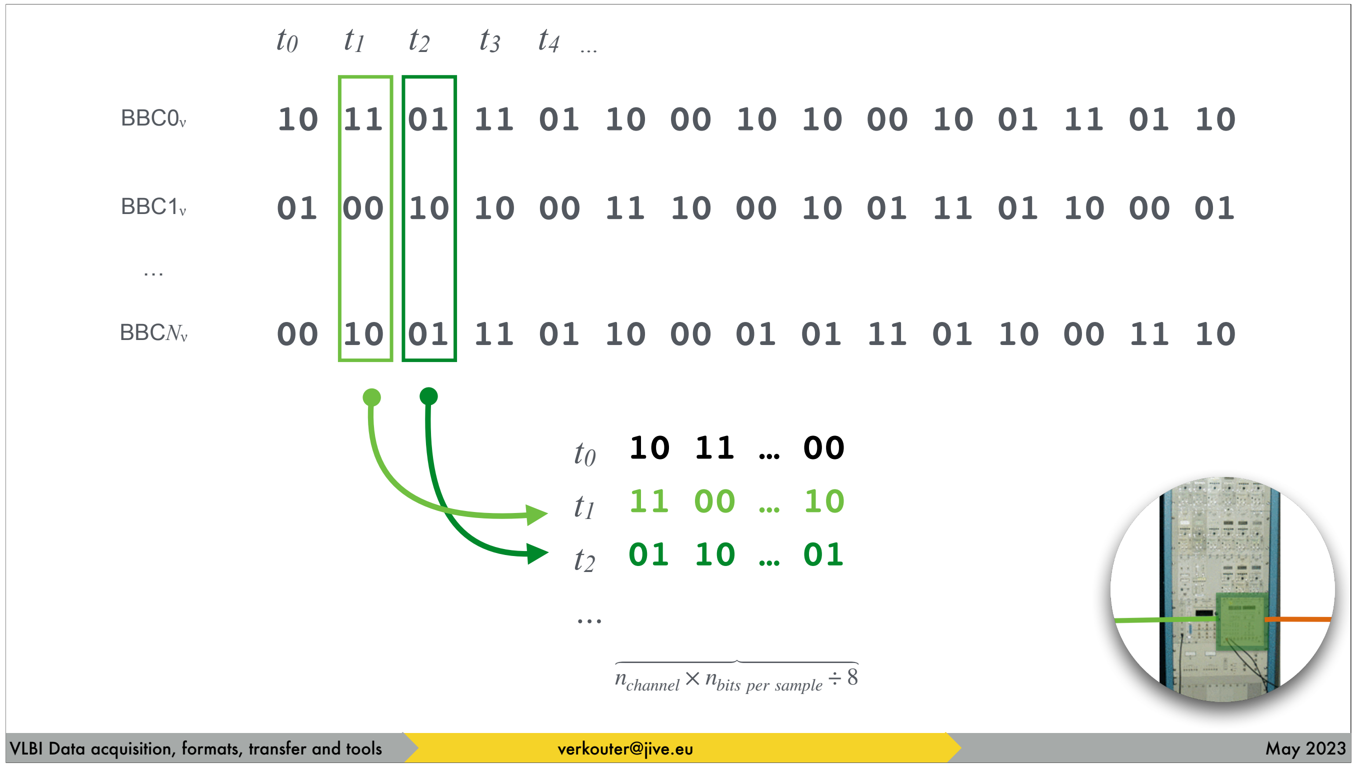
each baseband converter provides a stream of bits for a channel



	t_0	t_1	t_2	t_3	t_4 ...										
BBC0_ν	10	11	01	11	01	10	00	10	10	00	10	01	11	01	10
BBC1_ν	01	00	10	10	00	11	10	00	10	01	11	01	10	00	01
…															
$\text{BBC}N_\nu$	00	10	01	11	01	10	00	01	01	11	01	10	00	11	10

t_0	10	11	…	00
t_1	11	00	…	10

and the formatter rearranges them

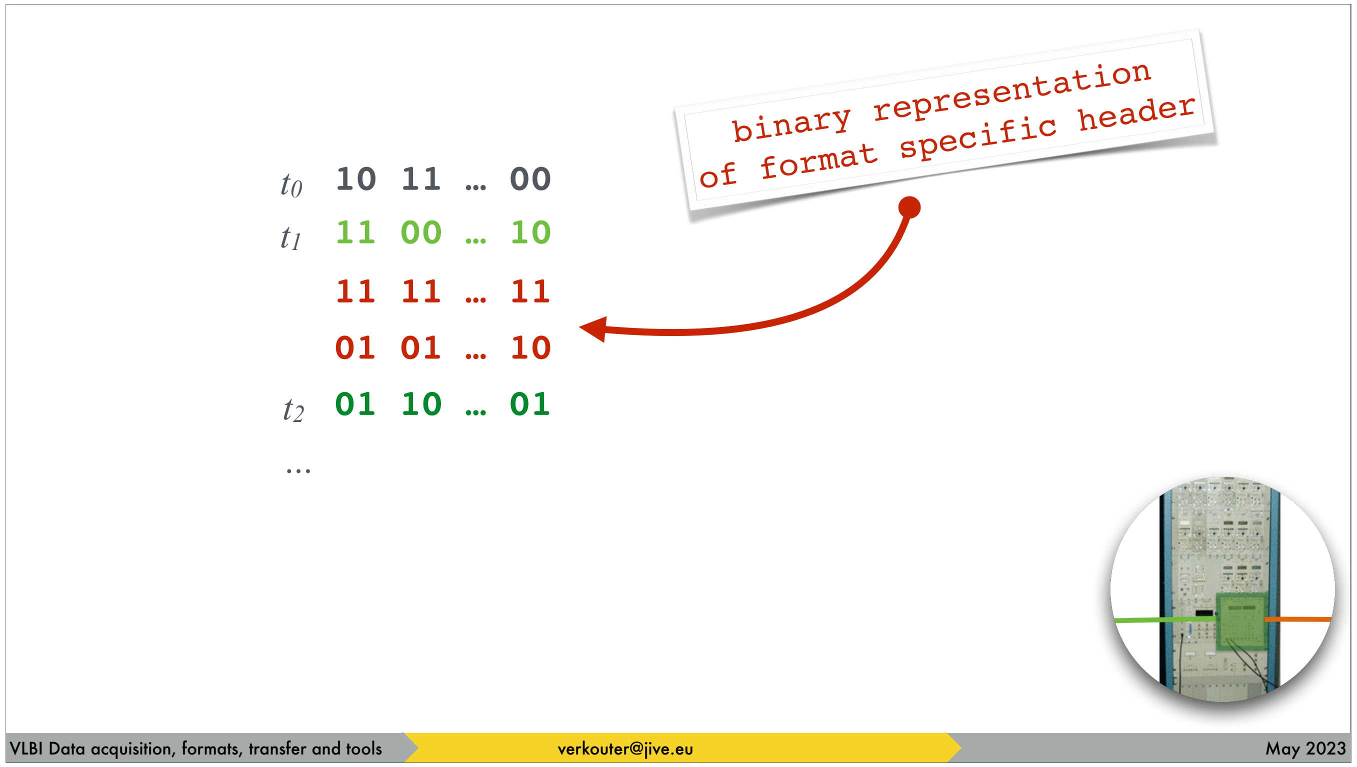


	t_0	t_1	t_2	t_3	t_4	…									
BBC0$_v$	10	11	01	11	01	10	00	10	10	00	10	01	11	01	10
BBC1$_v$	01	00	10	10	00	11	10	00	10	01	11	01	10	00	01
…															
BBCN_v	00	10	01	11	01	10	00	01	01	11	01	10	00	11	10

t_0	10	11	…	00
t_1	11	00	…	10
t_2	01	10	…	01
…				

$\overbrace{n_{channel} \times n_{bits\ per\ sample} \div 8}$

to have all samples of the same time stamp grouped together.



t_0 10 11 … 00

t_1 11 00 … 10

11 11 … 11

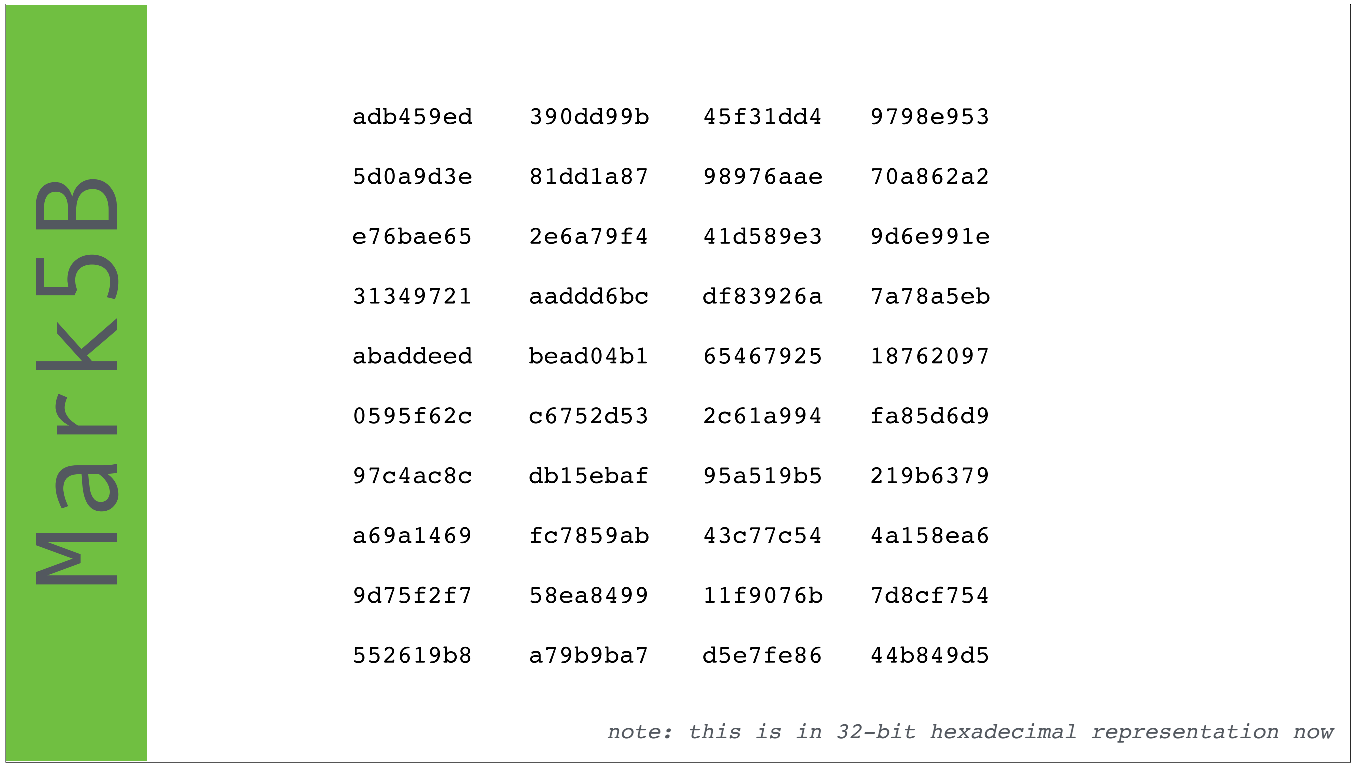
01 01 … 10

t_2 01 10 … 01

…

binary representation of format specific header

and at regular intervals inserts a header between the data.

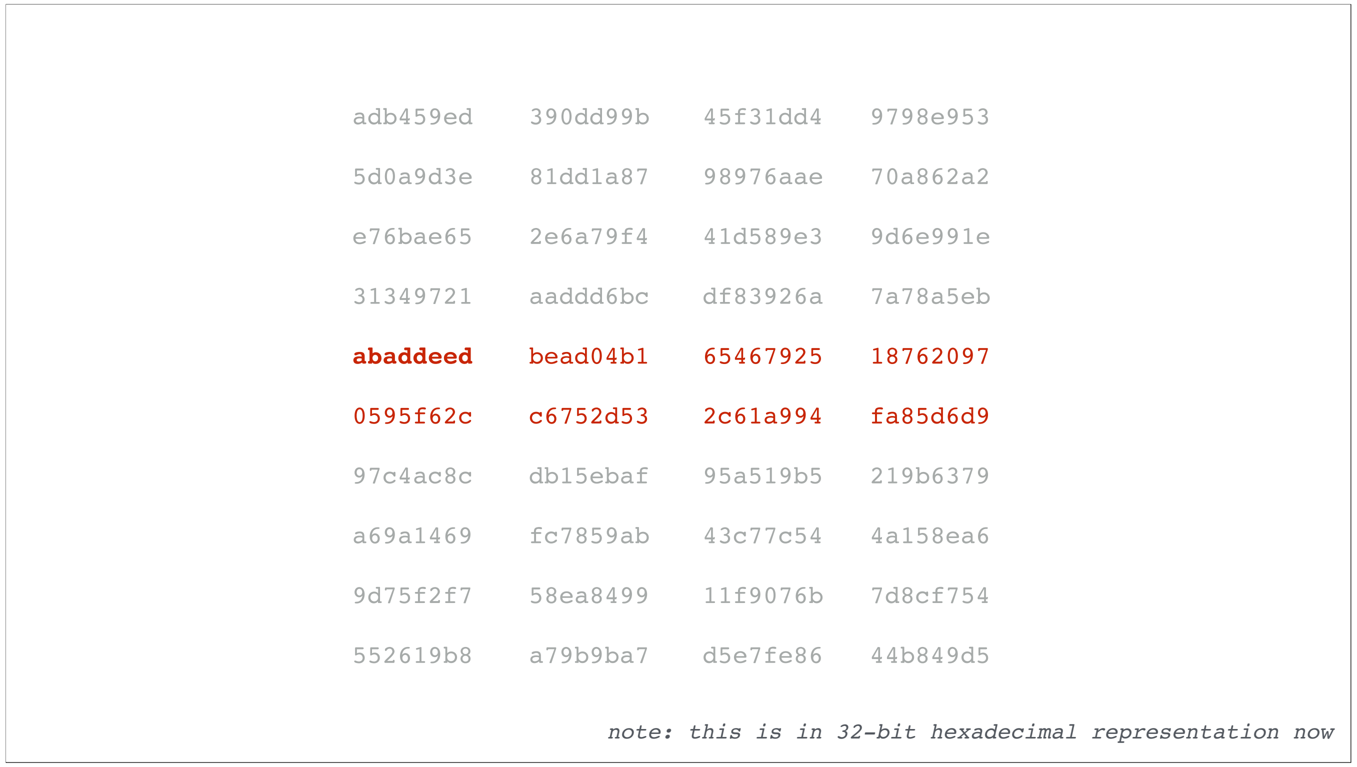


Mark5B

```
adb459ed    390dd99b    45f31dd4    9798e953

5d0a9d3e    81dd1a87    98976aae    70a862a2

e76bae65    2e6a79f4    41d589e3    9d6e991e

31349721    aaddd6bc    df83926a    7a78a5eb

abaddeed    bead04b1    65467925    18762097

0595f62c    c6752d53    2c61a994    fa85d6d9

97c4ac8c    db15ebaf    95a519b5    219b6379

a69a1469    fc7859ab    43c77c54    4a158ea6

9d75f2f7    58ea8499    11f9076b    7d8cf754

552619b8    a79b9ba7    d5e7fe86    44b849d5
```

note: this is in 32-bit hexadecimal representation now

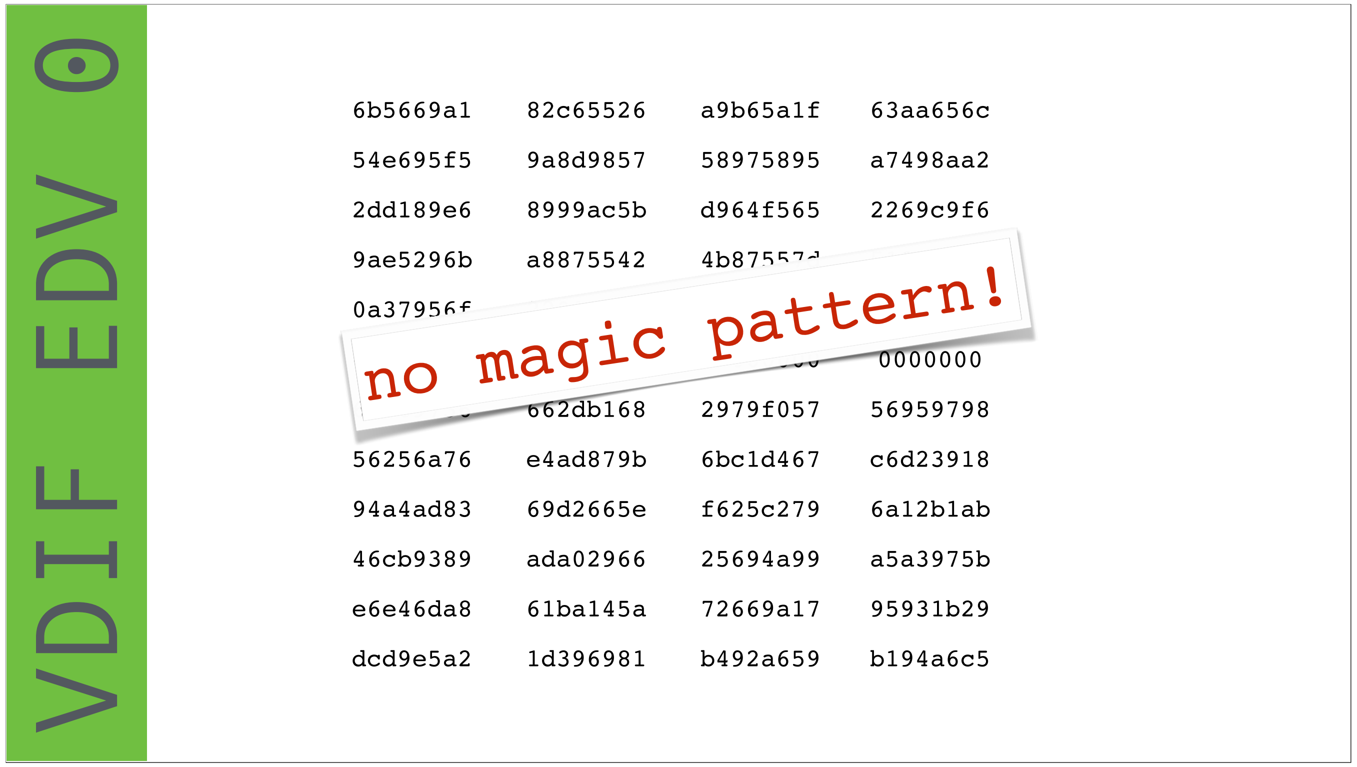
here we have an example snippet of Mark5B data



```
adb459ed    390dd99b    45f31dd4    9798e953

5d0a9d3e    81dd1a87    98976aae    70a862a2

e76bae65    2e6a79f4    41d589e3    9d6e991e

31349721    aaddd6bc    df83926a    7a78a5eb

abaddeed    bead04b1    65467925    18762097

0595f62c    c6752d53    2c61a994    fa85d6d9

97c4ac8c    db15ebaf    95a519b5    219b6379

a69a1469    fc7859ab    43c77c54    4a158ea6

9d75f2f7    58ea8499    11f9076b    7d8cf754

552619b8    a79b9ba7    d5e7fe86    44b849d5
```

note: this is in 32-bit hexadecimal representation now

where the header can be recognised because it starts with a magic pattern abaddeed

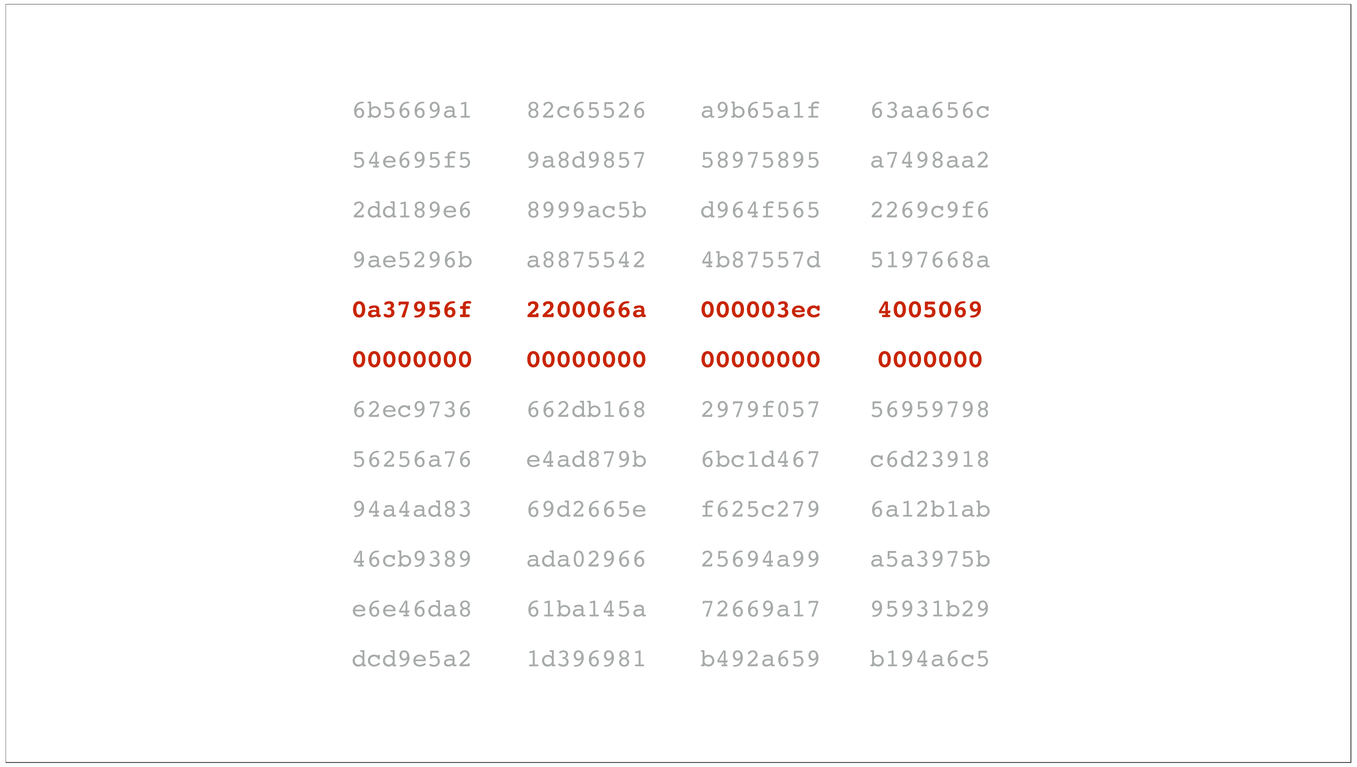


VDIF EDV 0

```
6b5669a1    82c65526    a9b65a1f    63aa656c
54e695f5    9a8d9857    58975895    a7498aa2
2dd189e6    8999ac5b    d964f565    2269c9f6
9ae5296b    a8875542    4b87557d
0a37956f
                                     0000000
            662db168    2979f057    56959798
56256a76    e4ad879b    6bc1d467    c6d23918
94a4ad83    69d2665e    f625c279    6a12b1ab
46cb9389    ada02966    25694a99    a5a3975b
e6e46da8    61ba145a    72669a17    95931b29
dcd9e5a2    1d396981    b492a659    b194a6c5
```

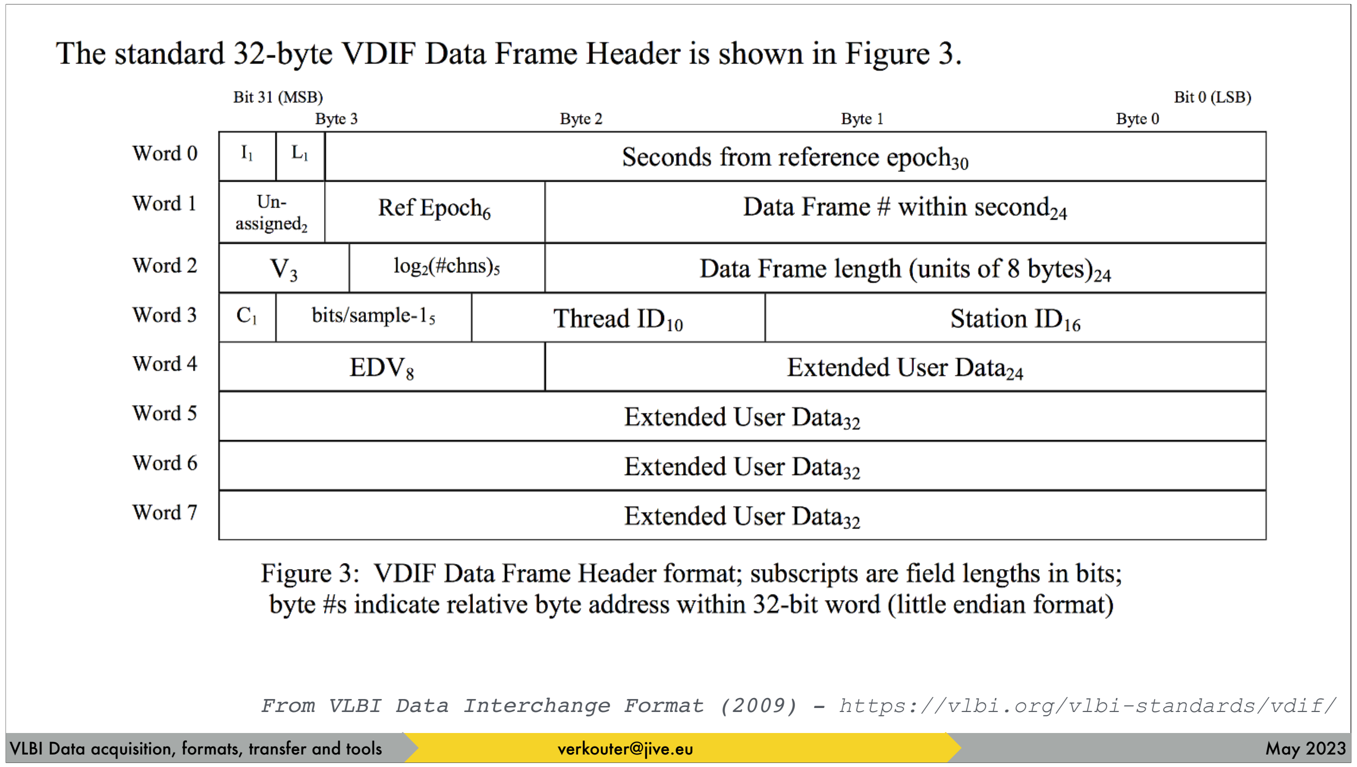
no magic pattern!

and here we are looking at one specific sub-flavour of VDIF, extended data version number zero. [click] in VDIF there is no standardised magic pattern so you have to count bytes …



```
6b5669a1    82c65526    a9b65a1f    63aa656c
54e695f5    9a8d9857    58975895    a7498aa2
2dd189e6    8999ac5b    d964f565    2269c9f6
9ae5296b    a8875542    4b87557d    5197668a
0a37956f    2200066a    000003ec     4005069
00000000    00000000    00000000     0000000
62ec9736    662db168    2979f057    56959798
56256a76    e4ad879b    6bc1d467    c6d23918
94a4ad83    69d2665e    f625c279    6a12b1ab
46cb9389    ada02966    25694a99    a5a3975b
e6e46da8    61ba145a    72669a17    95931b29
dcd9e5a2    1d396981    b492a659    b194a6c5
```

… and hope no bytes are lost to KNOW that the header is here.



The standard 32-byte VDIF Data Frame Header is shown in Figure 3.

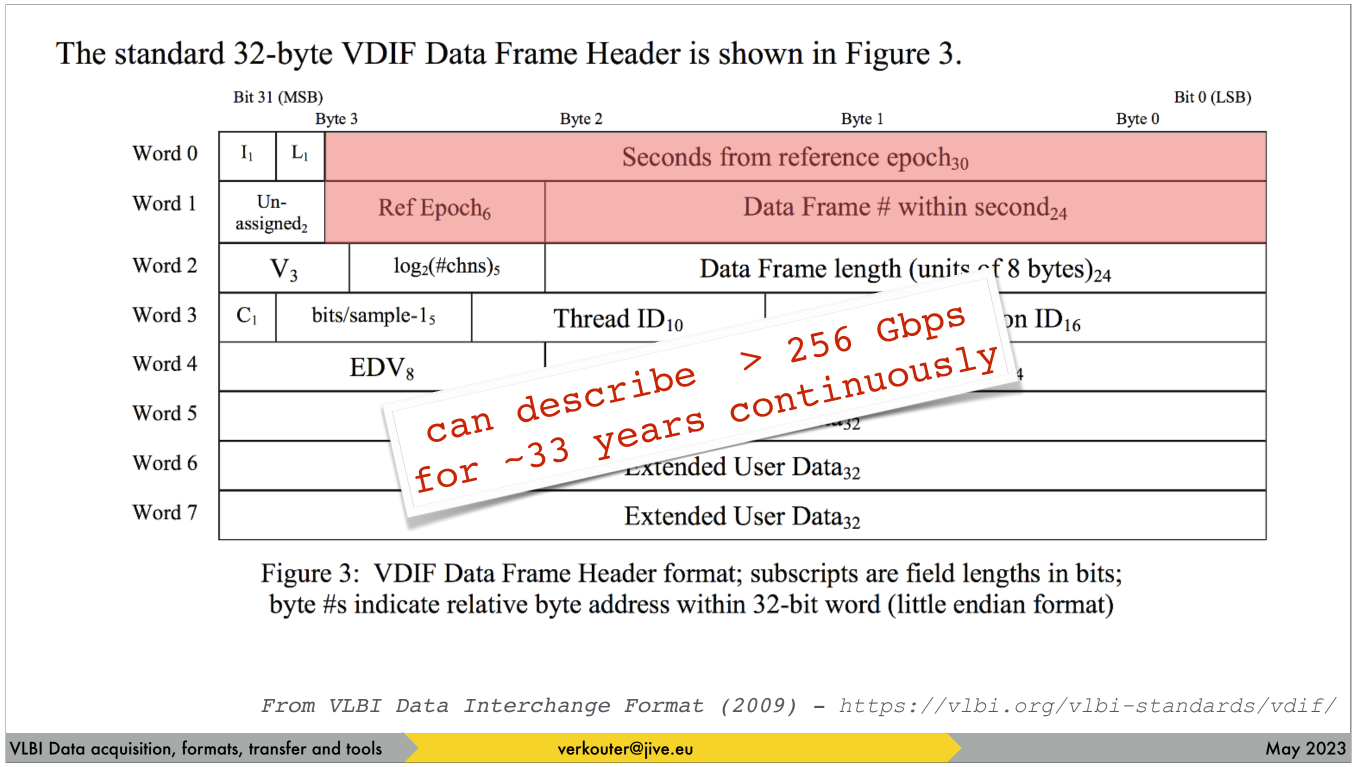
Bit 31 (MSB) Byte 3 Byte 2 Byte 1 Byte 0 Bit 0 (LSB)

Word 0	I_1	L_1	Seconds from reference epoch$_{30}$		
Word 1	Un-assigned$_2$	Ref Epoch$_6$	Data Frame # within second$_{24}$		
Word 2	V_3	log$_2$(#chns)$_5$	Data Frame length (units of 8 bytes)$_{24}$		
Word 3	C_1	bits/sample-1$_5$	Thread ID$_{10}$	Station ID$_{16}$	
Word 4	EDV$_8$	Extended User Data$_{24}$			
Word 5	Extended User Data$_{32}$				
Word 6	Extended User Data$_{32}$				
Word 7	Extended User Data$_{32}$				

Figure 3: VDIF Data Frame Header format; subscripts are field lengths in bits; byte #s indicate relative byte address within 32-bit word (little endian format)

From VLBI Data Interchange Format (2009) - https://vlbi.org/vlbi-standards/vdif/

Still, VDIF is the format of choice of new equipment since 2009. Its header contains a lot of useful information!



The standard 32-byte VDIF Data Frame Header is shown in Figure 3.

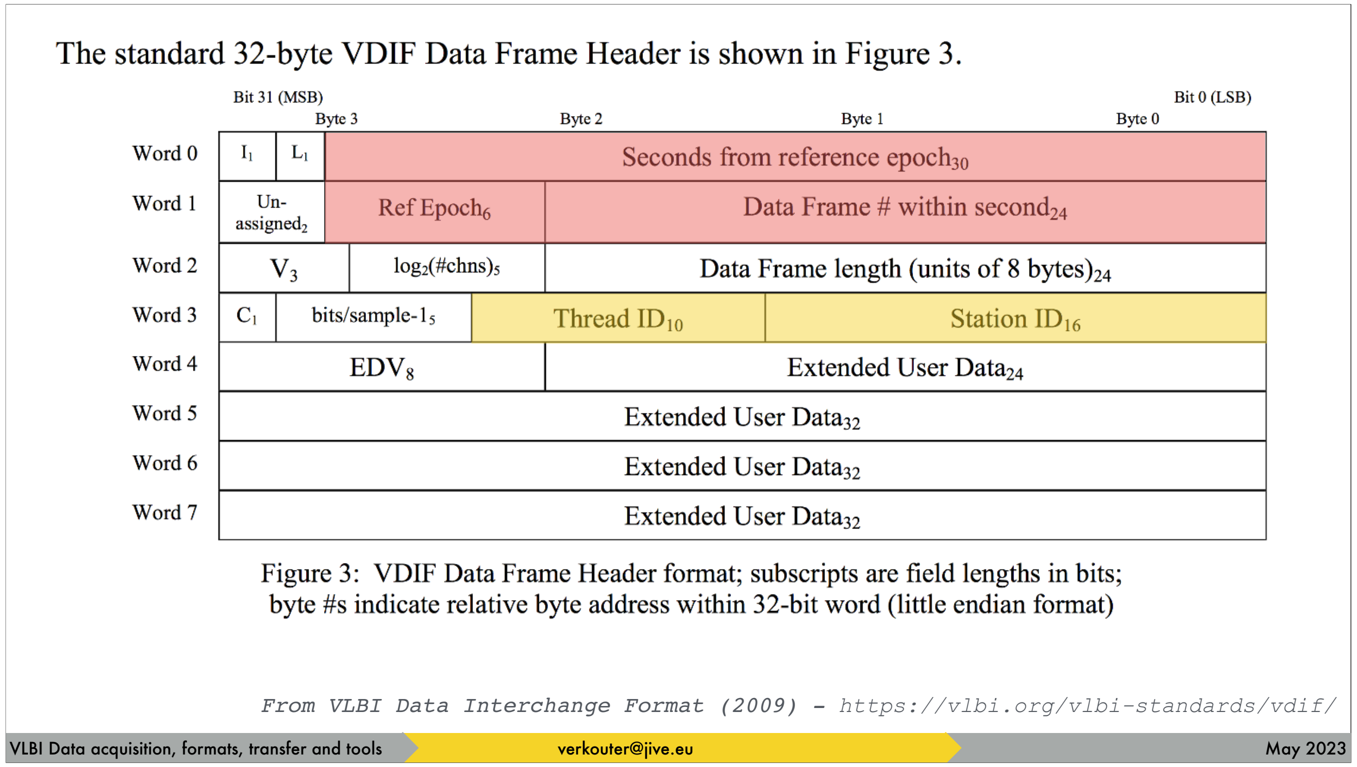
	Bit 31 (MSB) Byte 3		Byte 2	Byte 1	Byte 0 Bit 0 (LSB)
Word 0	I_1	L_1	Seconds from reference epoch$_{30}$		
Word 1	Un-assigned$_2$	Ref Epoch$_6$	Data Frame # within second$_{24}$		
Word 2	V_3	$\log_2$(#chns)$_5$	Data Frame length (units of 8 bytes)$_{24}$		
Word 3	C_1	bits/sample-1$_5$	Thread ID$_{10}$	on ID$_{16}$	
Word 4	EDV$_8$		$_4$		
Word 5	$_{32}$				
Word 6	xtended User Data$_{32}$				
Word 7	Extended User Data$_{32}$				

can describe > 256 Gbps for ~33 years continuously

Figure 3: VDIF Data Frame Header format; subscripts are field lengths in bits; byte #s indicate relative byte address within 32-bit word (little endian format)

From VLBI Data Interchange Format (2009) - https://vlbi.org/vlbi-standards/vdif/

There are 60 bits reserved for the time stamp. To indicate how much that is, [click] it allows describing data rates of 256 Gbits per second for 33 years, continuously



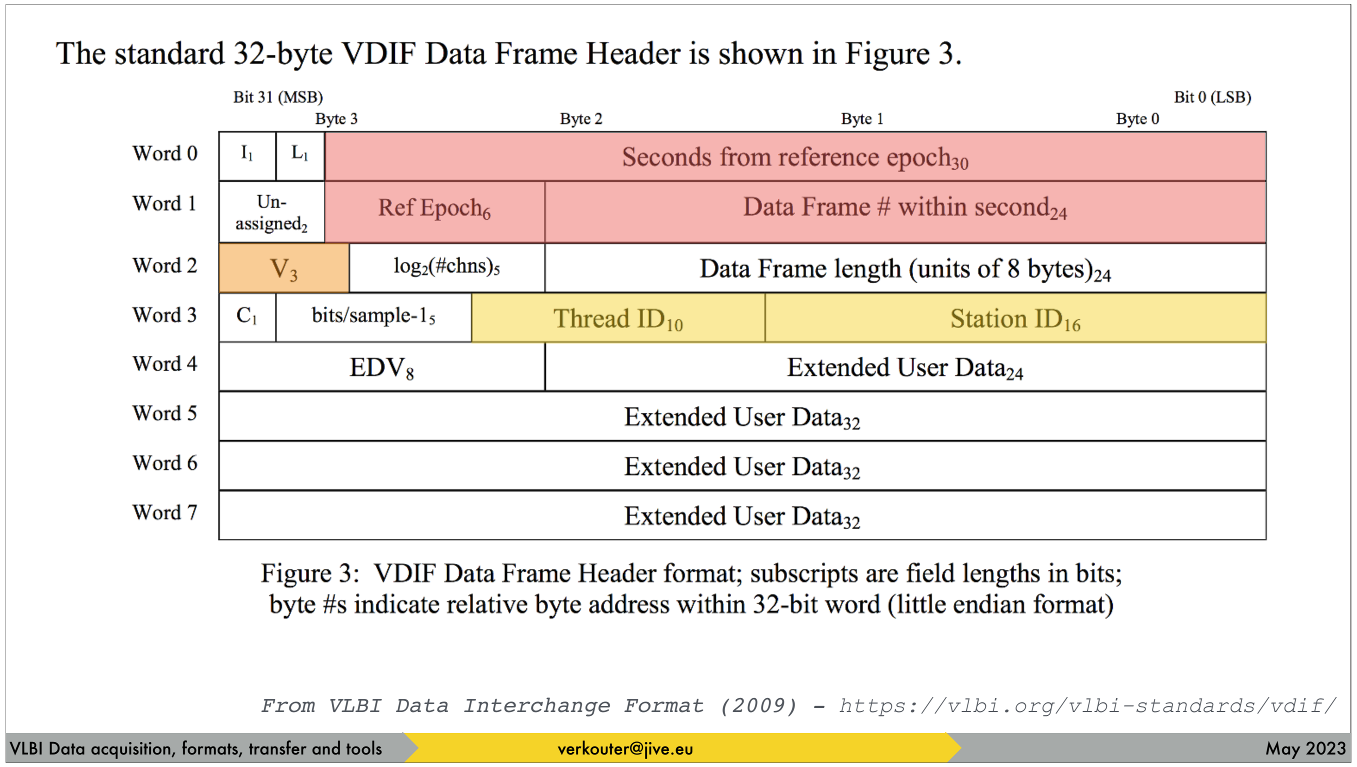
The standard 32-byte VDIF Data Frame Header is shown in Figure 3.

	Bit 31 (MSB) Byte 3			Byte 2	Byte 1	Byte 0 Bit 0 (LSB)
Word 0	I_1	L_1	Seconds from reference epoch$_{30}$			
Word 1	Un-assigned$_2$		Ref Epoch$_6$	Data Frame # within second$_{24}$		
Word 2	V_3		$\log_2$(#chns)$_5$	Data Frame length (units of 8 bytes)$_{24}$		
Word 3	C_1	bits/sample-1$_5$		Thread ID$_{10}$	Station ID$_{16}$	
Word 4	EDV$_8$			Extended User Data$_{24}$		
Word 5	Extended User Data$_{32}$					
Word 6	Extended User Data$_{32}$					
Word 7	Extended User Data$_{32}$					

Figure 3: VDIF Data Frame Header format; subscripts are field lengths in bits; byte #s indicate relative byte address within 32-bit word (little endian format)

From VLBI Data Interchange Format (2009) – https://vlbi.org/vlbi-standards/vdif/

then there are two fields available for identification of the data a station id and a thread id



The standard 32-byte VDIF Data Frame Header is shown in Figure 3.

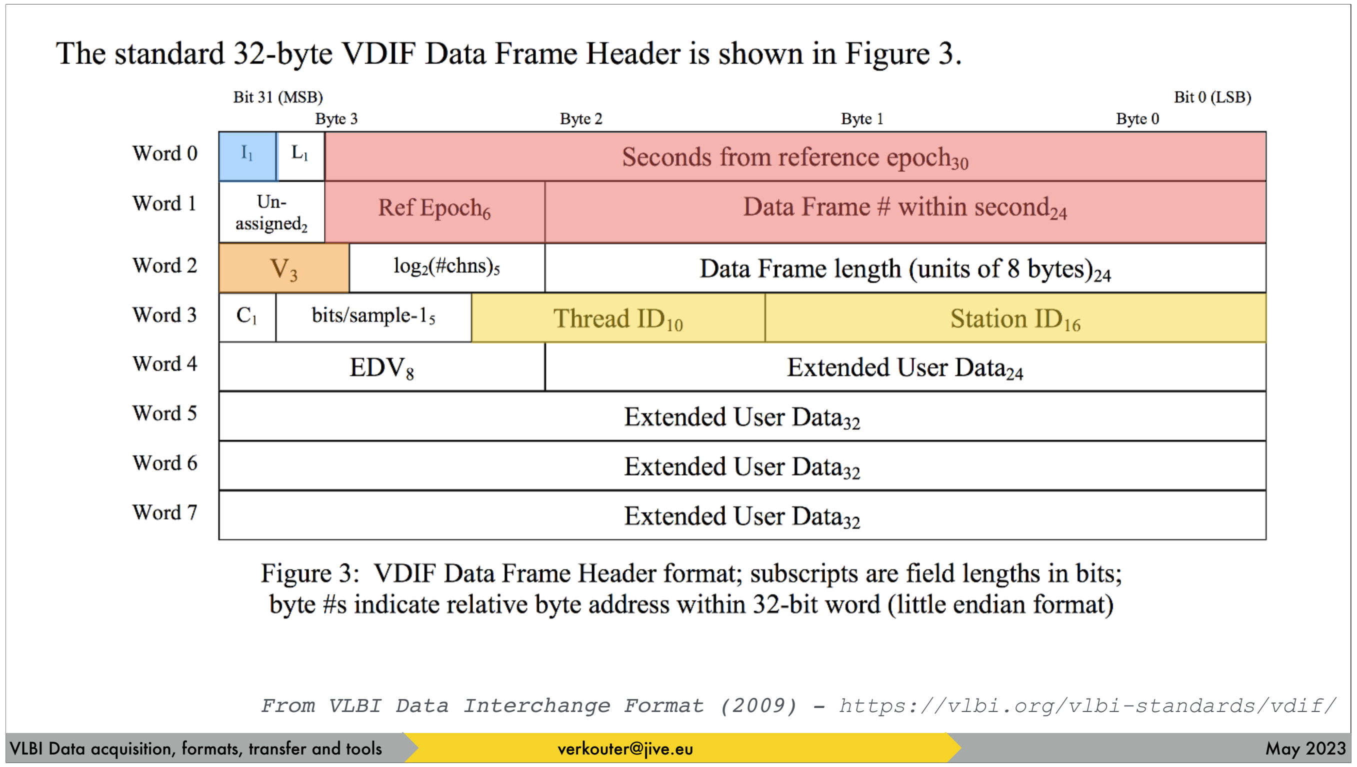
Bit 31 (MSB) · Byte 3 · Byte 2 · Byte 1 · Byte 0 · Bit 0 (LSB)

Word 0	I_1	L_1	Seconds from reference epoch$_{30}$	
Word 1	Un-assigned$_2$	Ref Epoch$_6$	Data Frame # within second$_{24}$	
Word 2	V_3	log$_2$(#chns)$_5$	Data Frame length (units of 8 bytes)$_{24}$	
Word 3	C_1	bits/sample-1$_5$	Thread ID$_{10}$	Station ID$_{16}$
Word 4	EDV$_8$	Extended User Data$_{24}$		
Word 5	Extended User Data$_{32}$			
Word 6	Extended User Data$_{32}$			
Word 7	Extended User Data$_{32}$			

Figure 3: VDIF Data Frame Header format; subscripts are field lengths in bits; byte #s indicate relative byte address within 32-bit word (little endian format)

From VLBI Data Interchange Format (2009) – https://vlbi.org/vlbi-standards/vdif/

there's a format version number in the header



The standard 32-byte VDIF Data Frame Header is shown in Figure 3.

Bit 31 (MSB) Bit 0 (LSB)

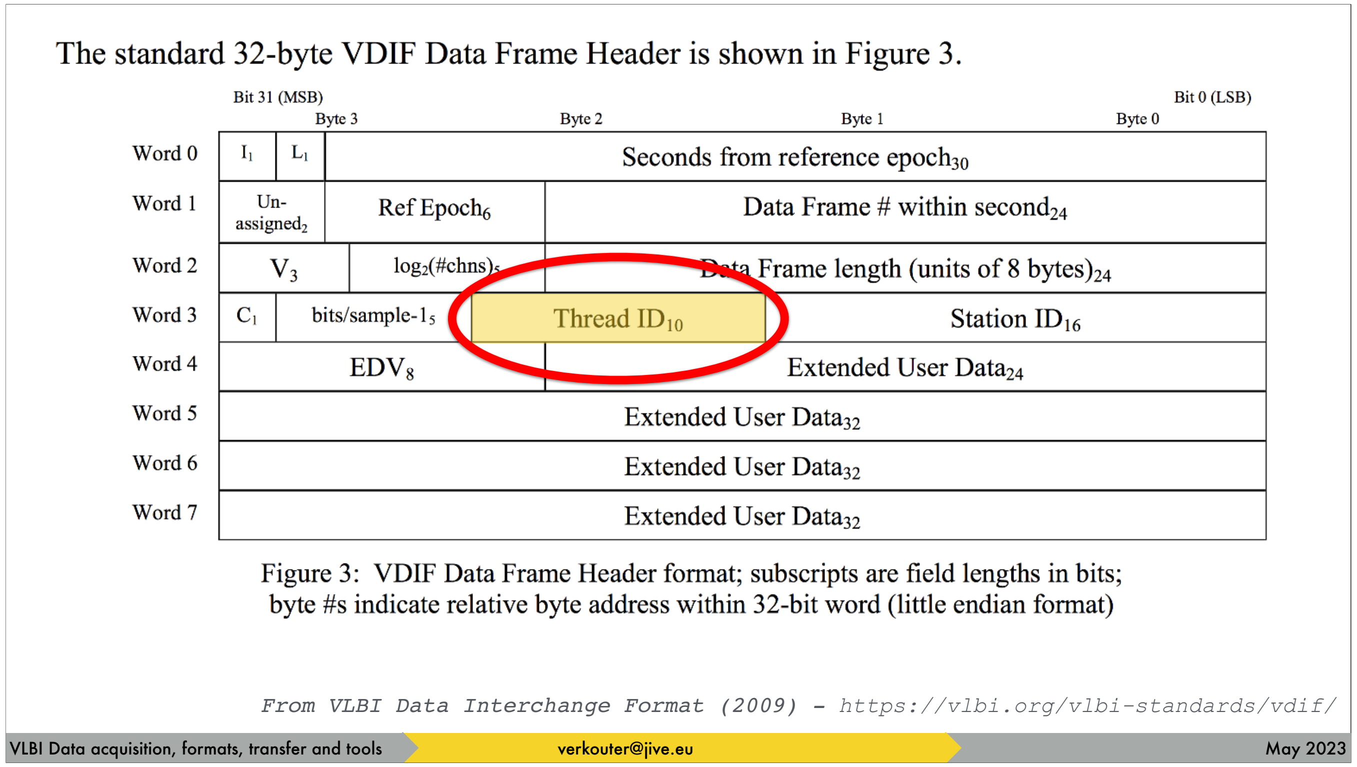
Byte 3 Byte 2 Byte 1 Byte 0

Word 0	I_1	L_1	Seconds from reference epoch$_{30}$	
Word 1	Un-assigned$_2$	Ref Epoch$_6$	Data Frame # within second$_{24}$	
Word 2	V_3	$\log_2(\#chns)_5$	Data Frame length (units of 8 bytes)$_{24}$	
Word 3	C_1	bits/sample-1$_5$	Thread ID$_{10}$	Station ID$_{16}$
Word 4	EDV$_8$	Extended User Data$_{24}$		
Word 5	Extended User Data$_{32}$			
Word 6	Extended User Data$_{32}$			
Word 7	Extended User Data$_{32}$			

Figure 3: VDIF Data Frame Header format; subscripts are field lengths in bits; byte #s indicate relative byte address within 32-bit word (little endian format)

From VLBI Data Interchange Format (2009) – https://vlbi.org/vlbi-standards/vdif/

and you can send data but mark it invalid



The standard 32-byte VDIF Data Frame Header is shown in Figure 3.

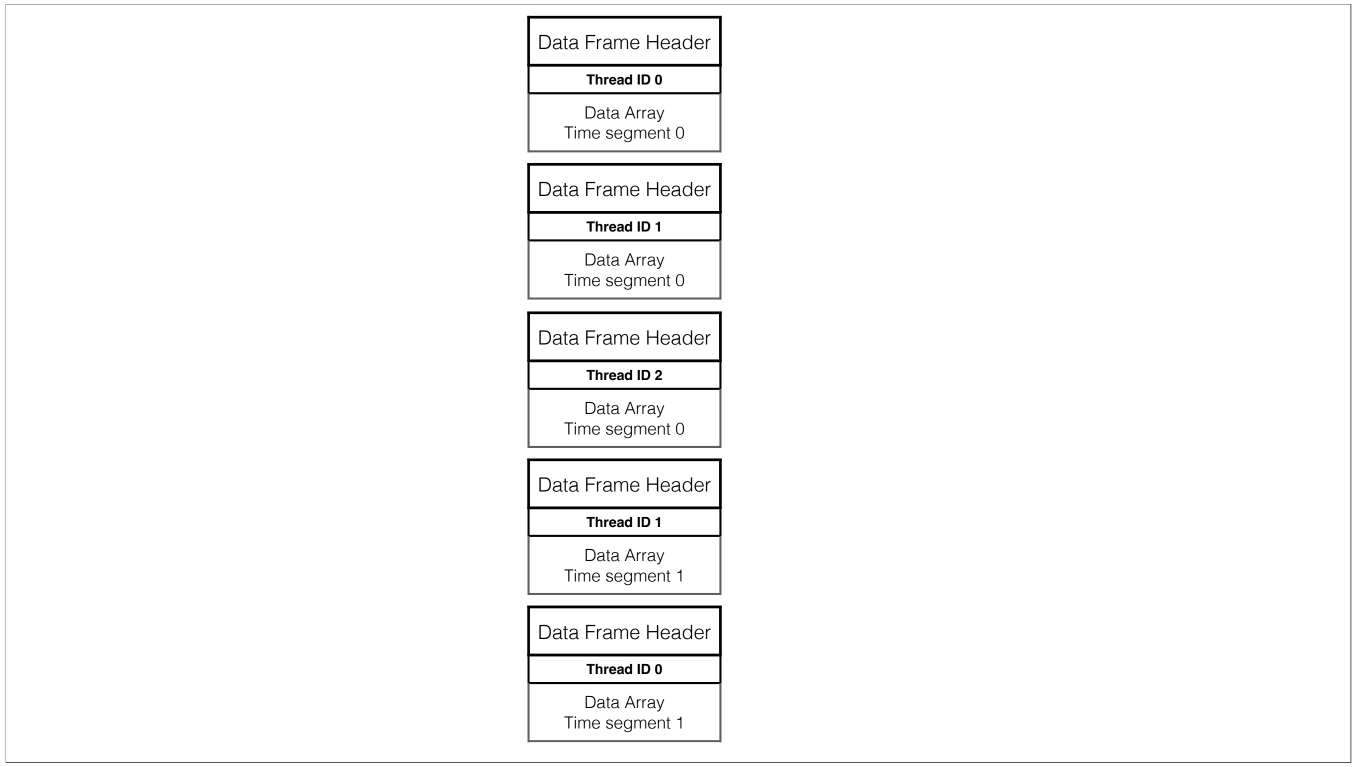
Bit 31 (MSB) · Byte 3 · Byte 2 · Byte 1 · Byte 0 · Bit 0 (LSB)

Word 0	I$_1$	L$_1$	Seconds from reference epoch$_{30}$	
Word 1	Un-assigned$_2$	Ref Epoch$_6$	Data Frame # within second$_{24}$	
Word 2	V$_3$	log$_2$(#chns)$_5$	Data Frame length (units of 8 bytes)$_{24}$	
Word 3	C$_1$	bits/sample-1$_5$	Thread ID$_{10}$	Station ID$_{16}$
Word 4	EDV$_8$	Extended User Data$_{24}$		
Word 5	Extended User Data$_{32}$			
Word 6	Extended User Data$_{32}$			
Word 7	Extended User Data$_{32}$			

Figure 3: VDIF Data Frame Header format; subscripts are field lengths in bits; byte #s indicate relative byte address within 32-bit word (little endian format)

From VLBI Data Interchange Format (2009) - https://vlbi.org/vlbi-standards/vdif/

This is all nice but we must spend some time on THIS field, the thread id



Data Frame Header
Thread ID 0
Data Array
Time segment 0

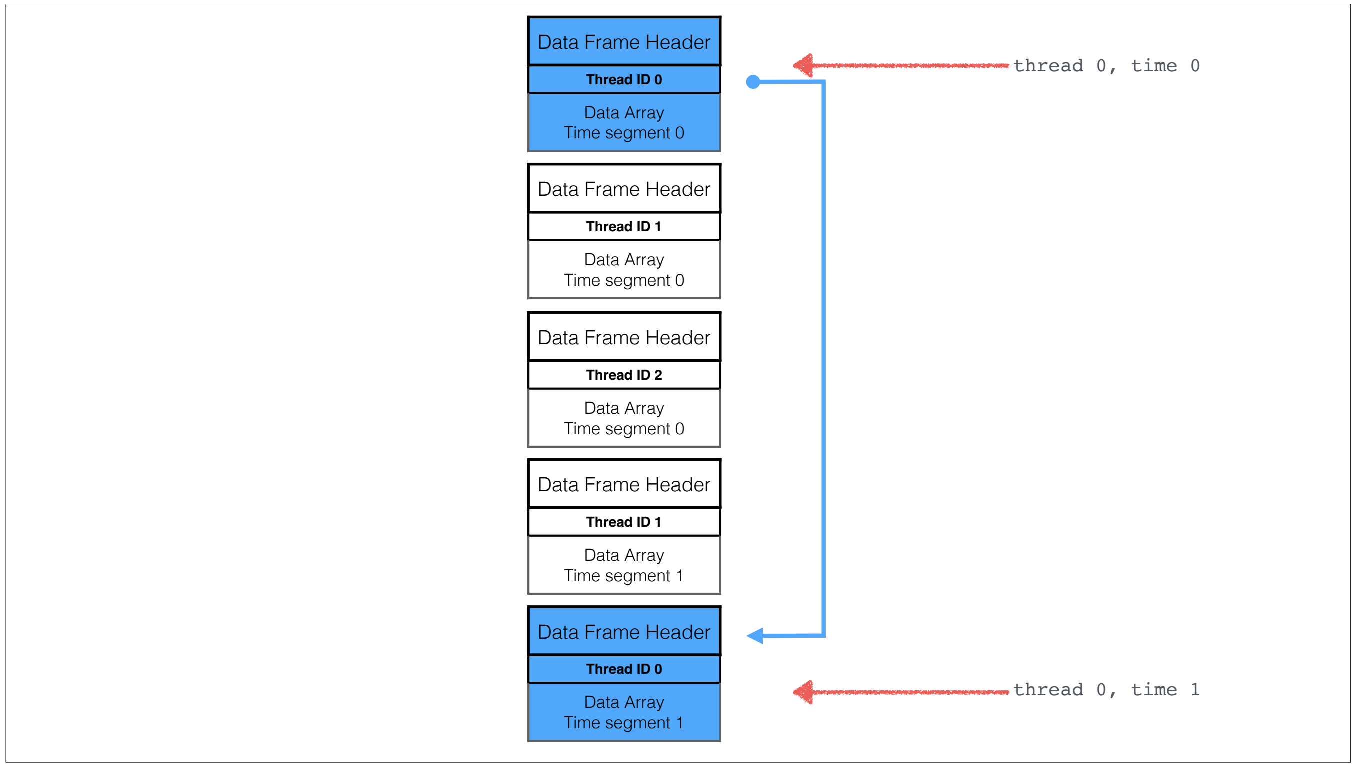
Data Frame Header
Thread ID 1
Data Array
Time segment 0

Data Frame Header
Thread ID 2
Data Array
Time segment 0

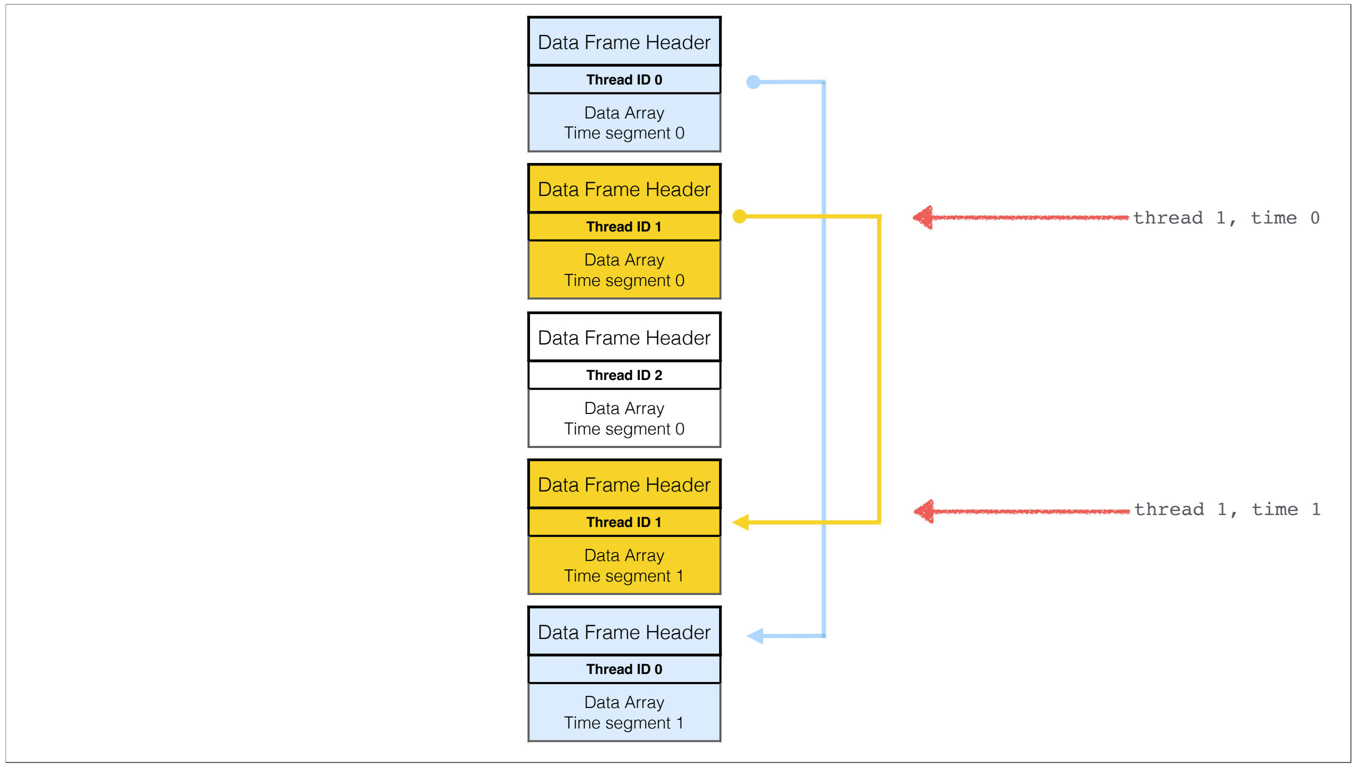
Data Frame Header
Thread ID 1
Data Array
Time segment 1

Data Frame Header
Thread ID 0
Data Array
Time segment 1

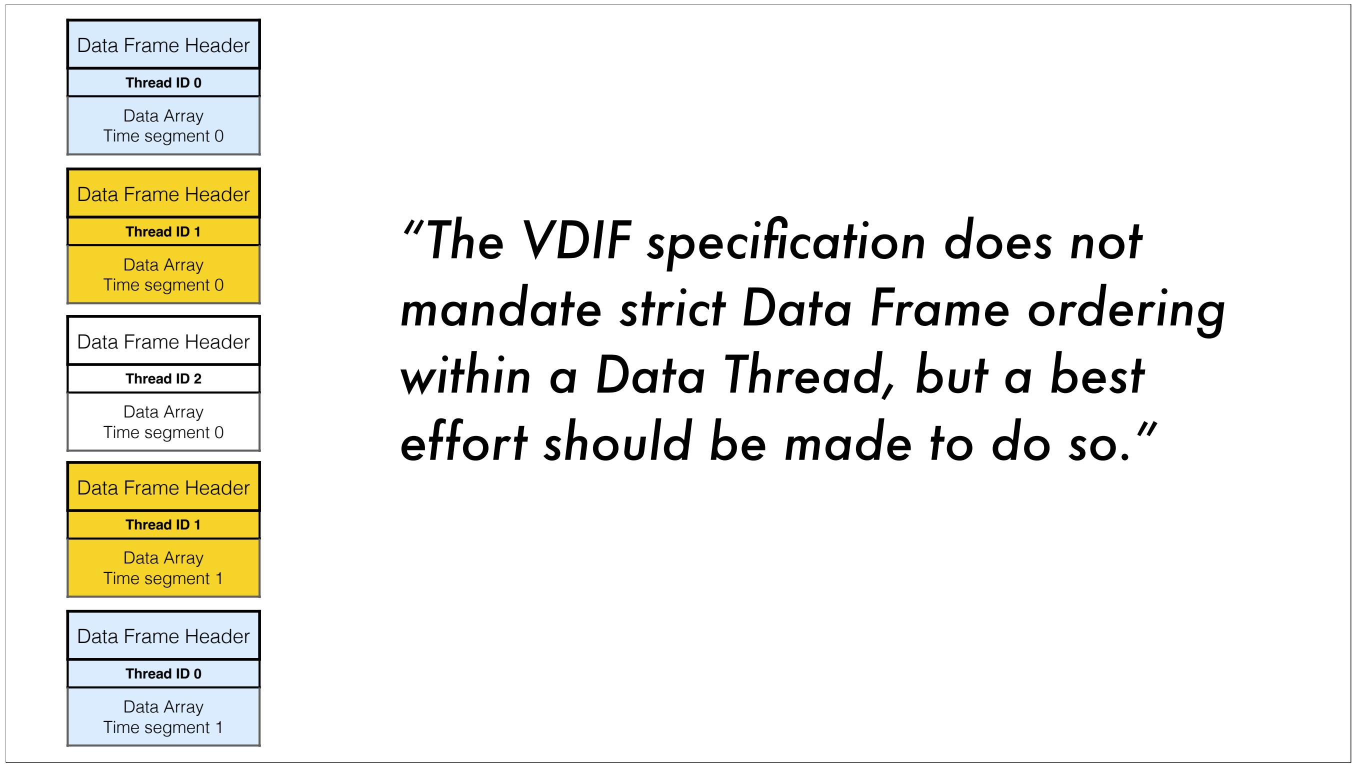
threads play a specific part in VDIF. A VDIF stream is a sequence of frames such as this.



All frames of a particular thread id form the time series for the same set of channels



but the order in which the frames are stored in the stream is NOT GUARANTEED by the standard



Data Frame Header
Thread ID 0
Data Array
Time segment 0

Data Frame Header
Thread ID 1
Data Array
Time segment 0

Data Frame Header
Thread ID 2
Data Array
Time segment 0

Data Frame Header
Thread ID 1
Data Array
Time segment 1

Data Frame Header
Thread ID 0
Data Array
Time segment 1

"The VDIF specification does not mandate strict Data Frame ordering within a Data Thread, but a best effort should be made to do so."

in fact it has this to say: all equipment and software should make a *best effort* to make time increase monotonically.

So VDIF is like a box of chocolates, you never know what you're going to get.

Data is recorded

…

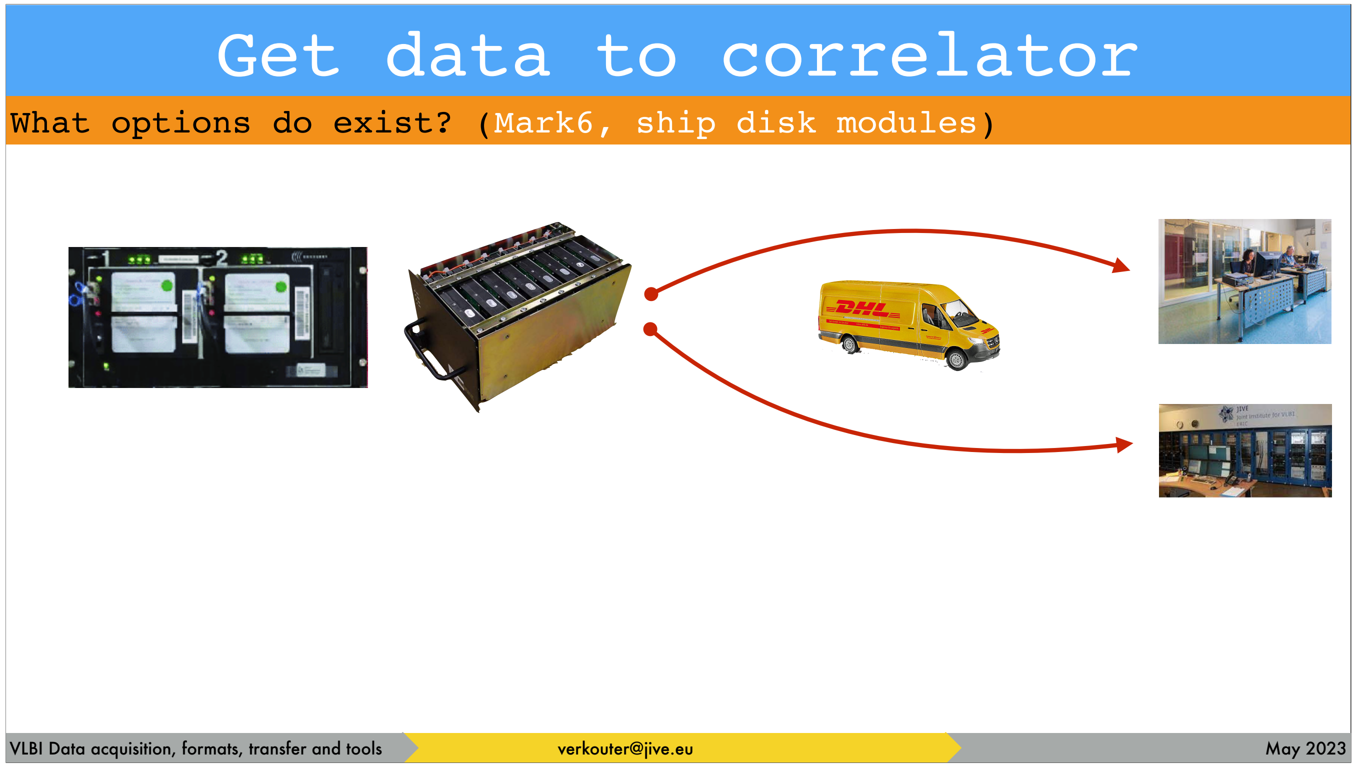
now what?

So, once the data has been recorded, then what?

Get data to correlator

What options do exist?

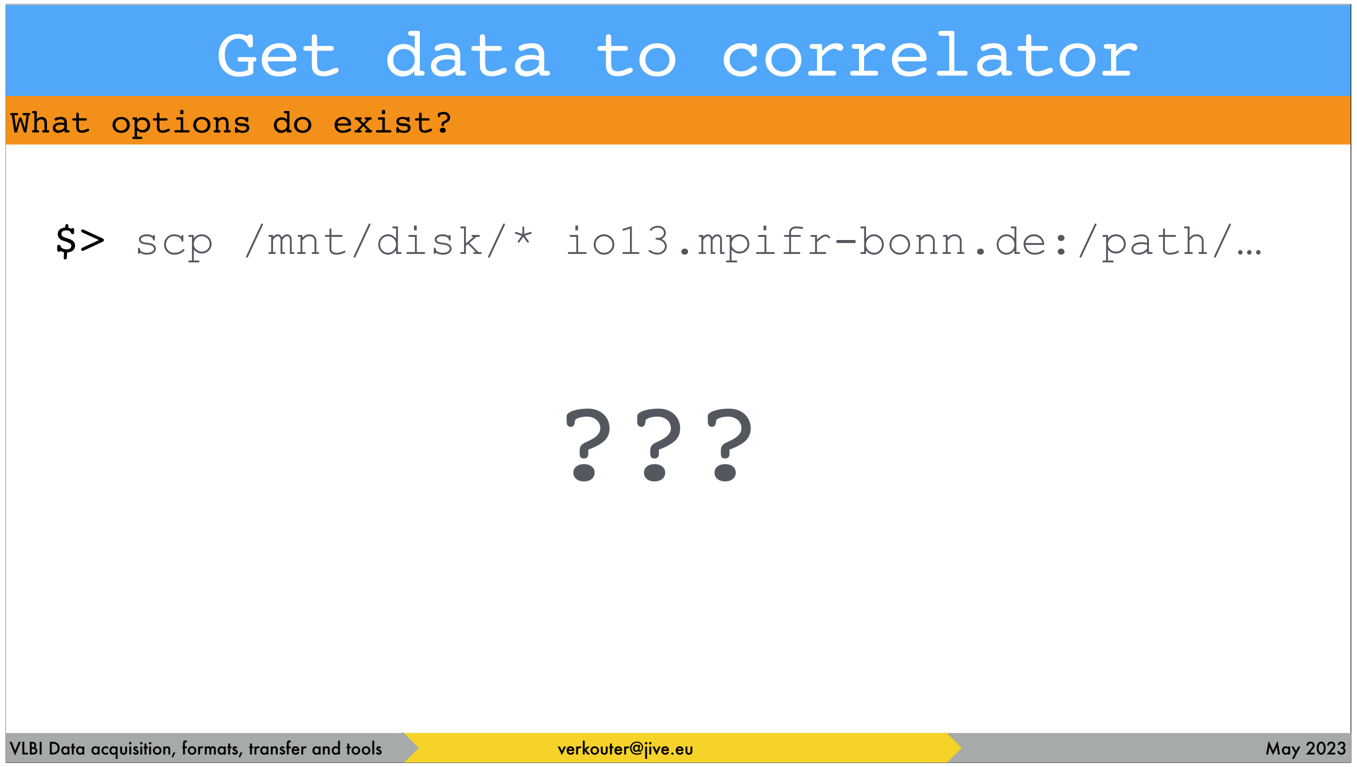
it needs to be transferred to the correlator. So what options do exist?



Get data to correlator

What options do exist? (Mark6, ship disk modules)

on the Mark6 you have removable disk packs so shipping those to [click] ONE correlator site is an option.



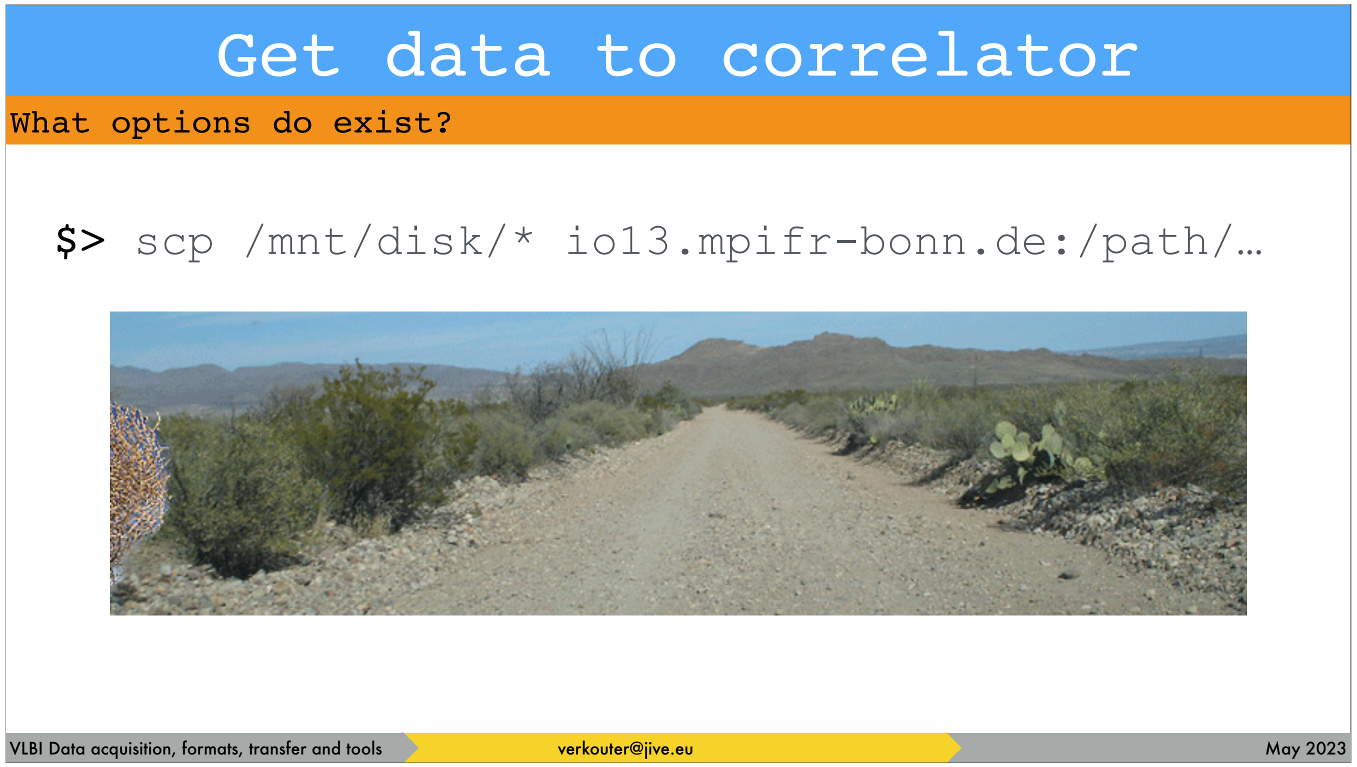
Get data to correlator

What options do exist?

```
$> scp /mnt/disk/* io13.mpifr-bonn.de:/path/…
```

???

but what about e-transfer? If you actually TRY this command …

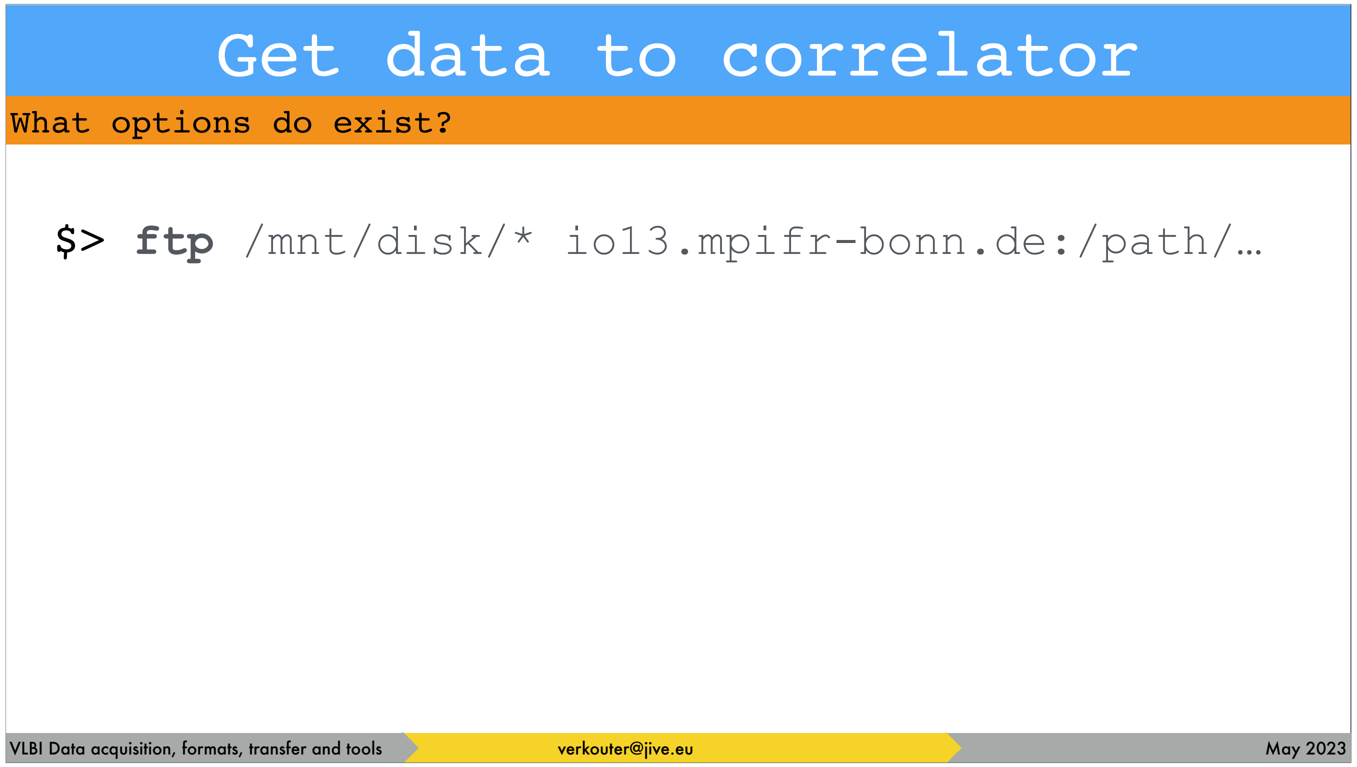


Get data to correlator

What options do exist?

```
$> scp /mnt/disk/* io13.mpifr-bonn.de:/path/…
```

… what likely will happen is this

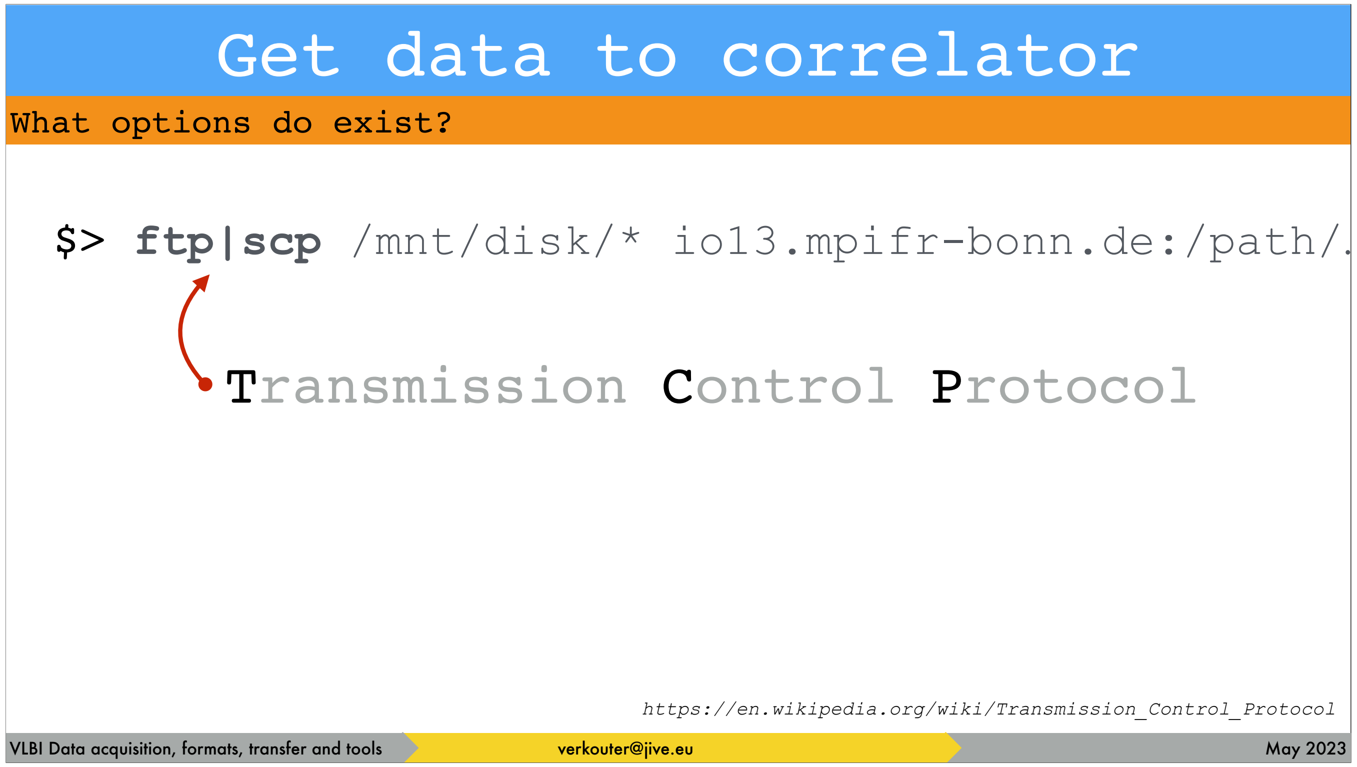


Get data to correlator

What options do exist?

```
$> ftp /mnt/disk/* io13.mpifr-bonn.de:/path/…
```

And even if you try THIS, the same will happen.



Get data to correlator

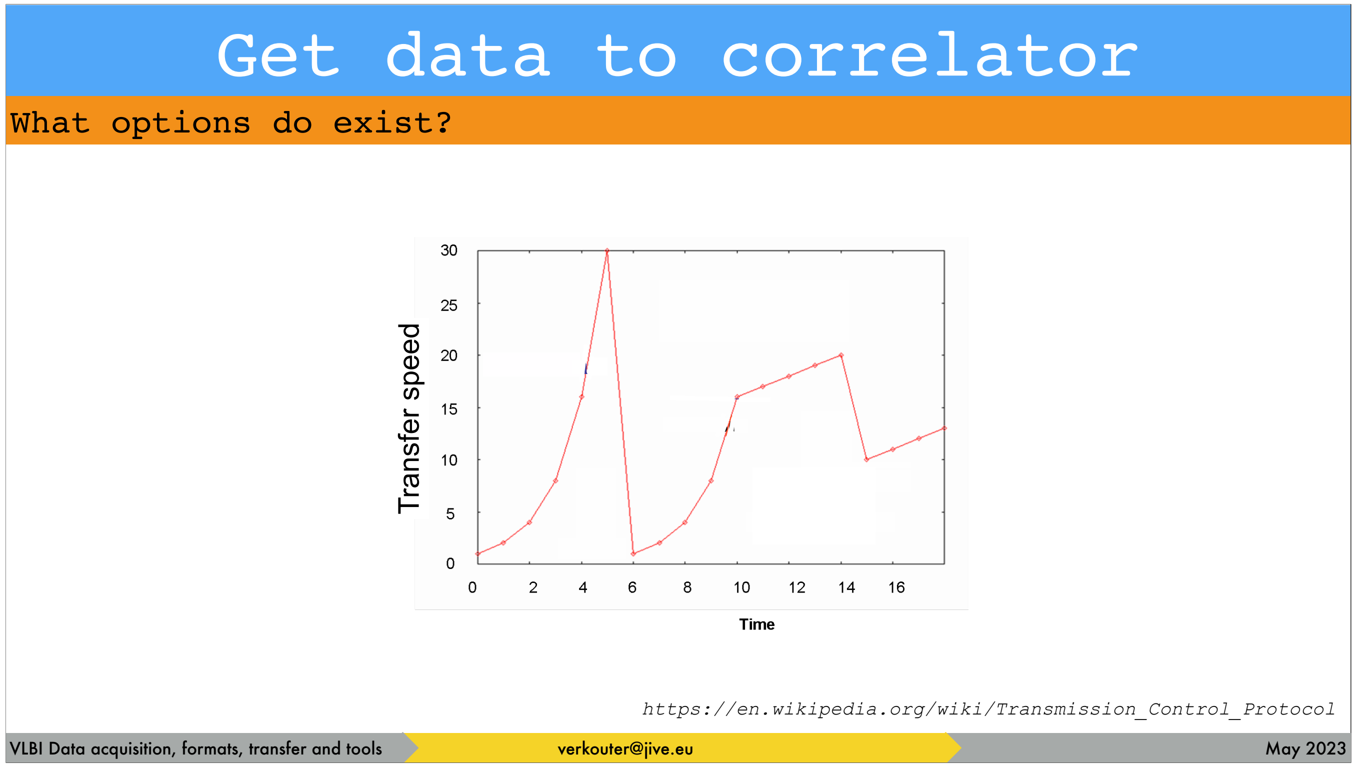
What options do exist?

```
$> ftp|scp /mnt/disk/* io13.mpifr-bonn.de:/path/.
```

Transmission Control Protocol

https://en.wikipedia.org/wiki/Transmission_Control_Protocol

The problem is that these tools work with the TCP protocol

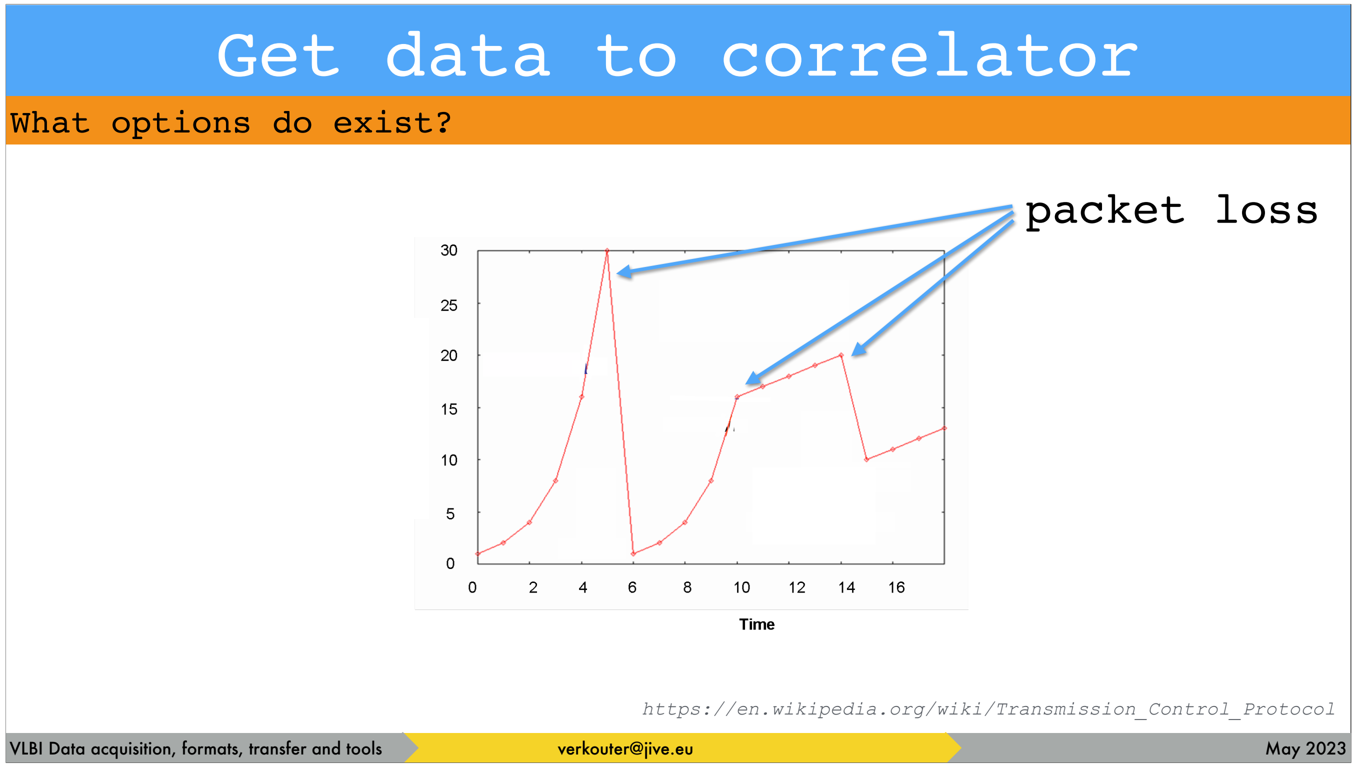


Get data to correlator

What options do exist?

https://en.wikipedia.org/wiki/Transmission_Control_Protocol

And that doesn’t work very well on the long fat international links, the speed is very erratic.



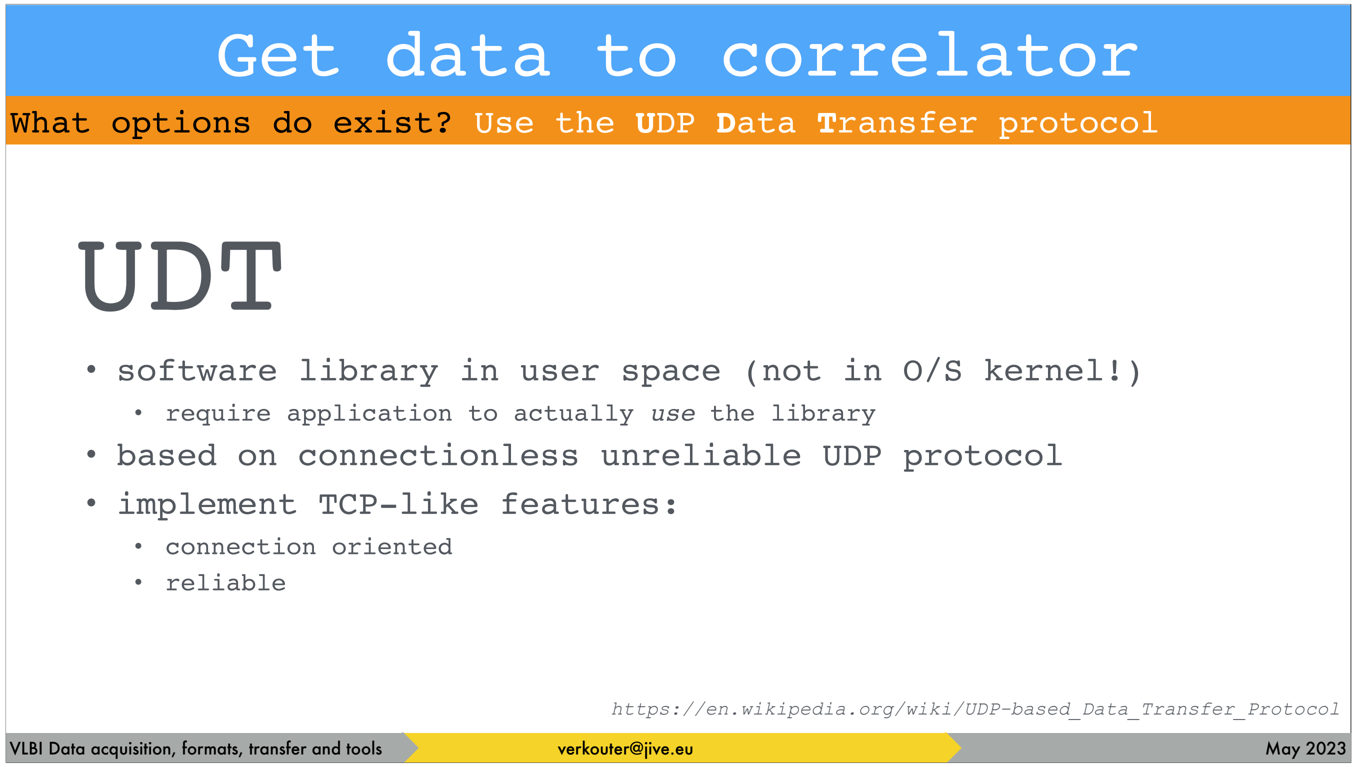
Get data to correlator

What options do exist?

packet loss

https://en.wikipedia.org/wiki/Transmission_Control_Protocol

the protocol is VERY sensitive to packet loss



Get data to correlator

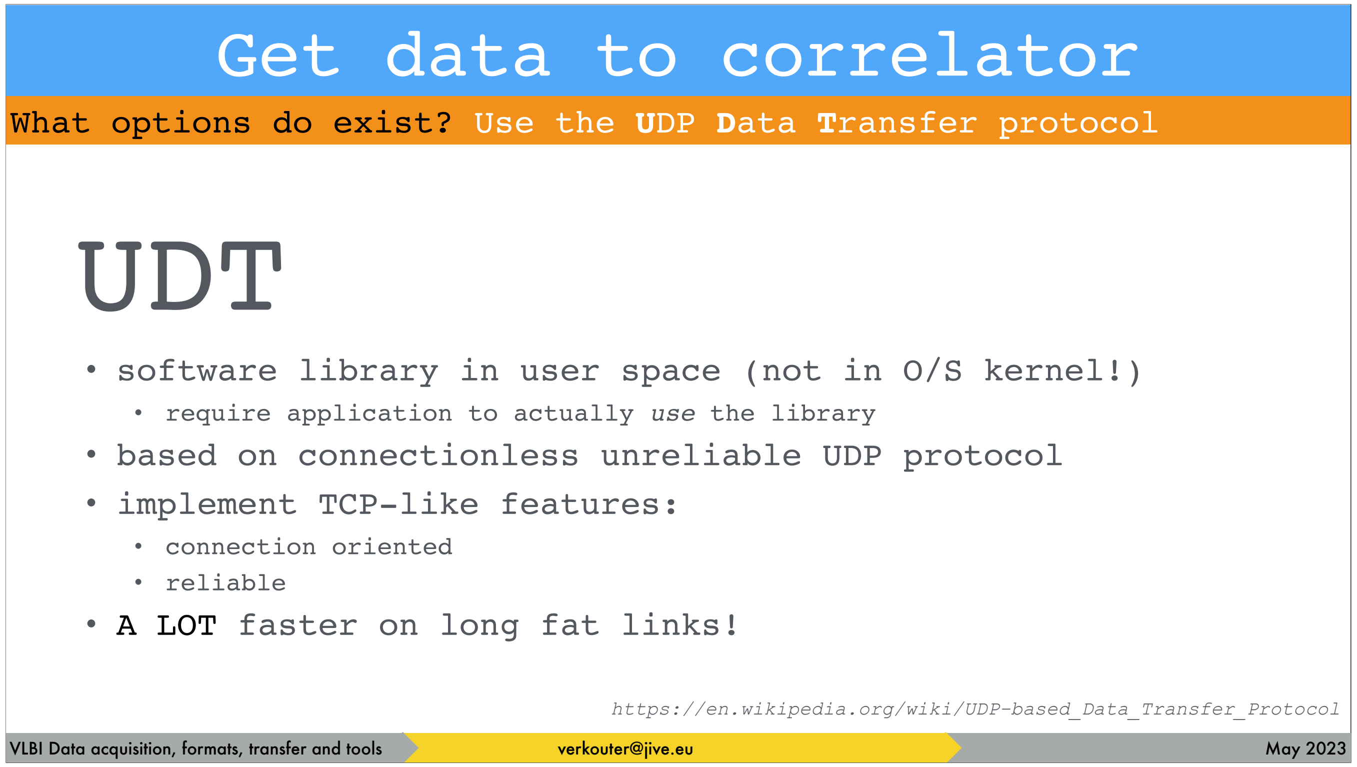
What options do exist? Use the **U**DP **D**ata **T**ransfer protocol

UDT

- software library in user space (not in O/S kernel!)
  - require application to actually *use* the library
- based on connectionless unreliable UDP protocol
- implement TCP-like features:
  - connection oriented
  - reliable

https://en.wikipedia.org/wiki/UDP-based_Data_Transfer_Protocol

fortunately there is a solution. Someone invented the udt data transfer protocol. Which is implemented as a software library simulating tcp on top of udp. So it’s not supported by the operating system but an application must use the library.



Get data to correlator

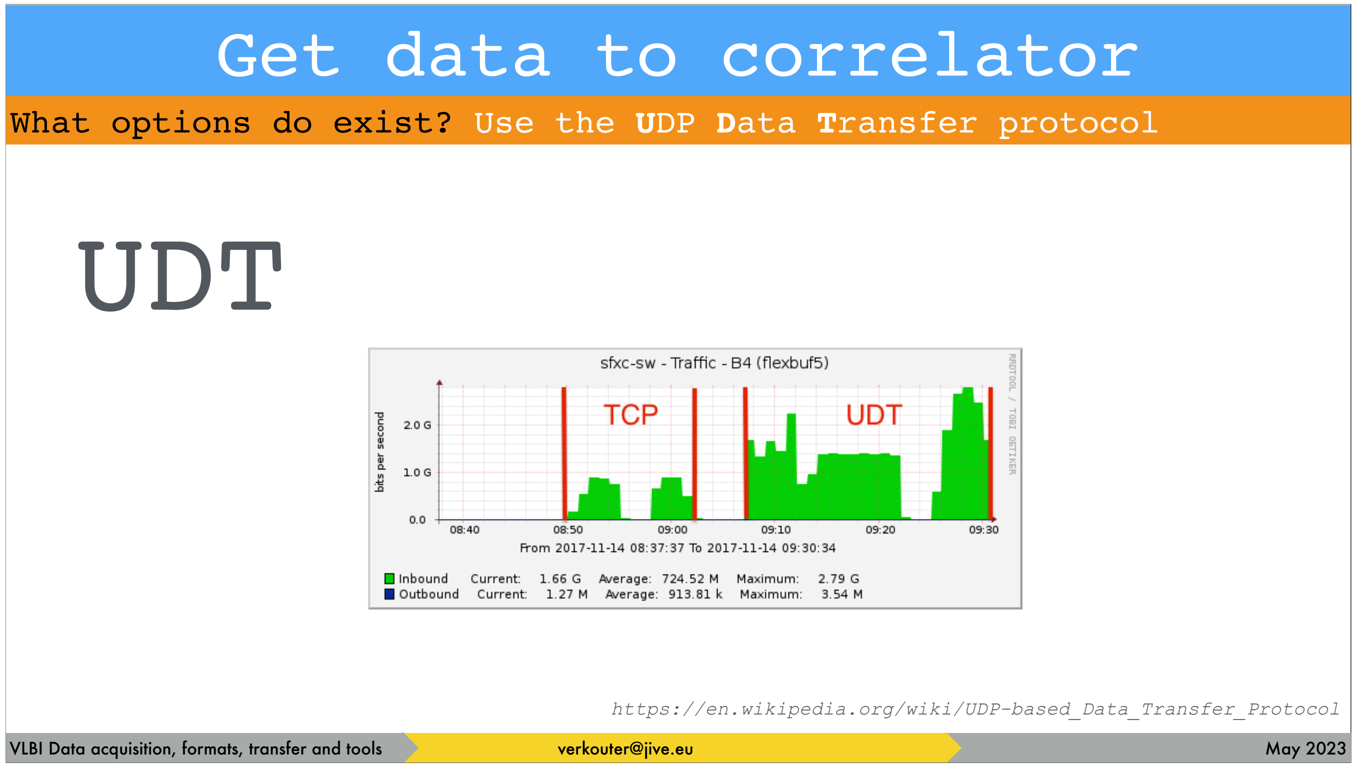
What options do exist? Use the **U**DP **D**ata **T**ransfer protocol

UDT

- software library in user space (not in O/S kernel!)
  - require application to actually *use* the library
- based on connectionless unreliable UDP protocol
- implement TCP-like features:
  - connection oriented
  - reliable
- A LOT faster on long fat links!

https://en.wikipedia.org/wiki/UDP-based_Data_Transfer_Protocol

The main feature is that it is really A LOT faster than TCP.



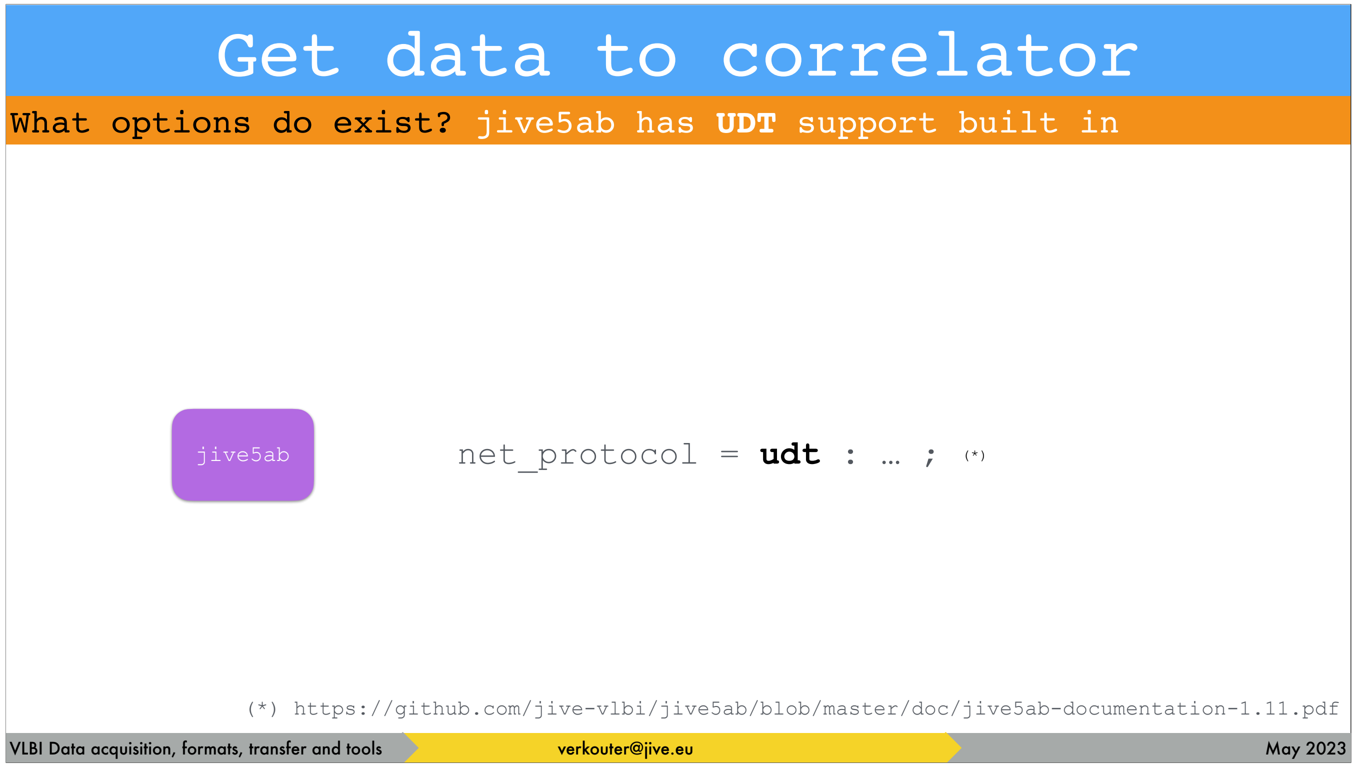
Get data to correlator

What options do exist? Use the **U**DP **D**ata **T**ransfer protocol

UDT

https://en.wikipedia.org/wiki/UDP-based_Data_Transfer_Protocol

As can be seen in these network graphs. Note that the TCP transfers are misleading because they were cancelled - the file transfer just froze and we ^C'ed the program before retrying again.



Get data to correlator

What options do exist? jive5ab has UDT support built in

```
net_protocol = udt : … ; (*)
```

(*) https://github.com/jive-vlbi/jive5ab/blob/master/doc/jive5ab-documentation-1.11.pdf

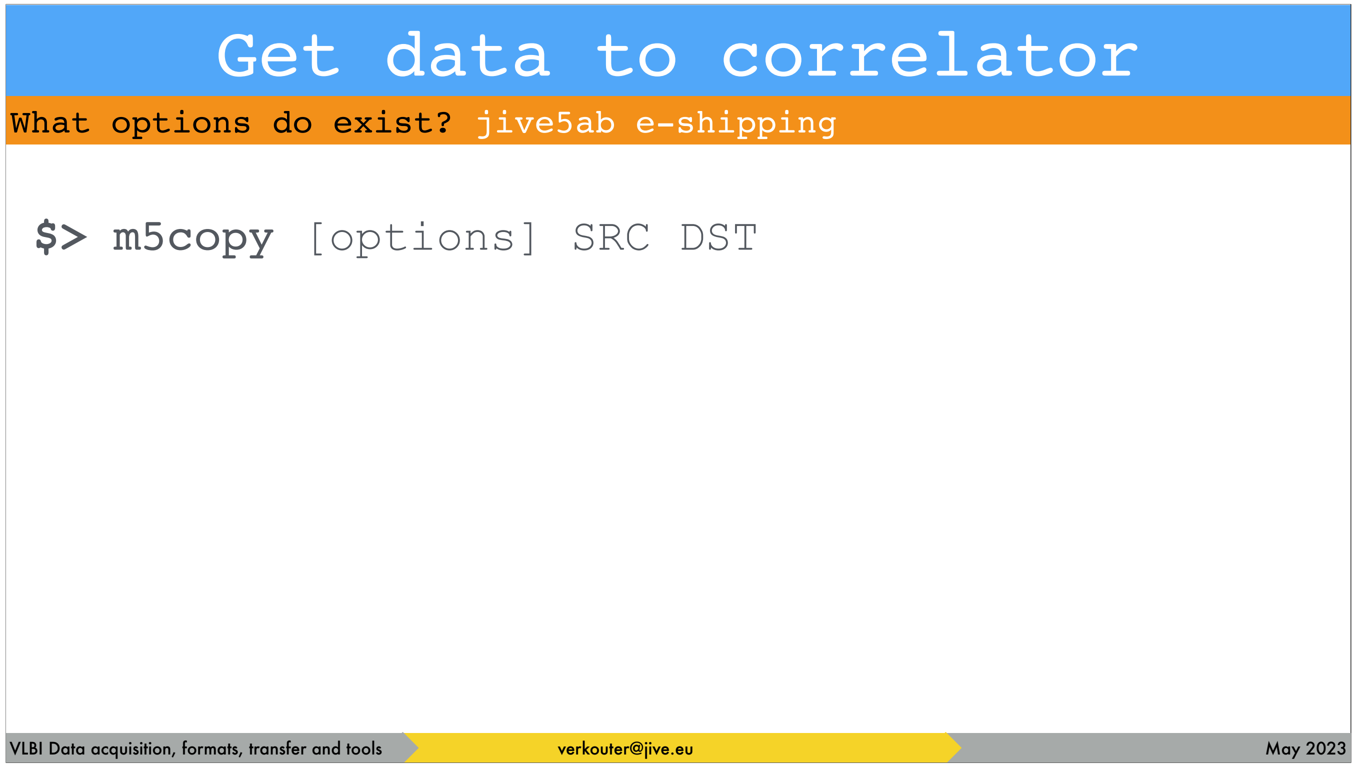
So you need an application to use this fast protocol. [click] jive5ab is such an application.

Get data to correlator

What options do exist? jive5ab e-shipping

https://github.com/jive-vlbi/jive5ab/blob/master/scripts/m5copy

If you get the following ingredients, [click] 2 servers running jive5ab and the [click] m5copy python script from the jive5ab sources

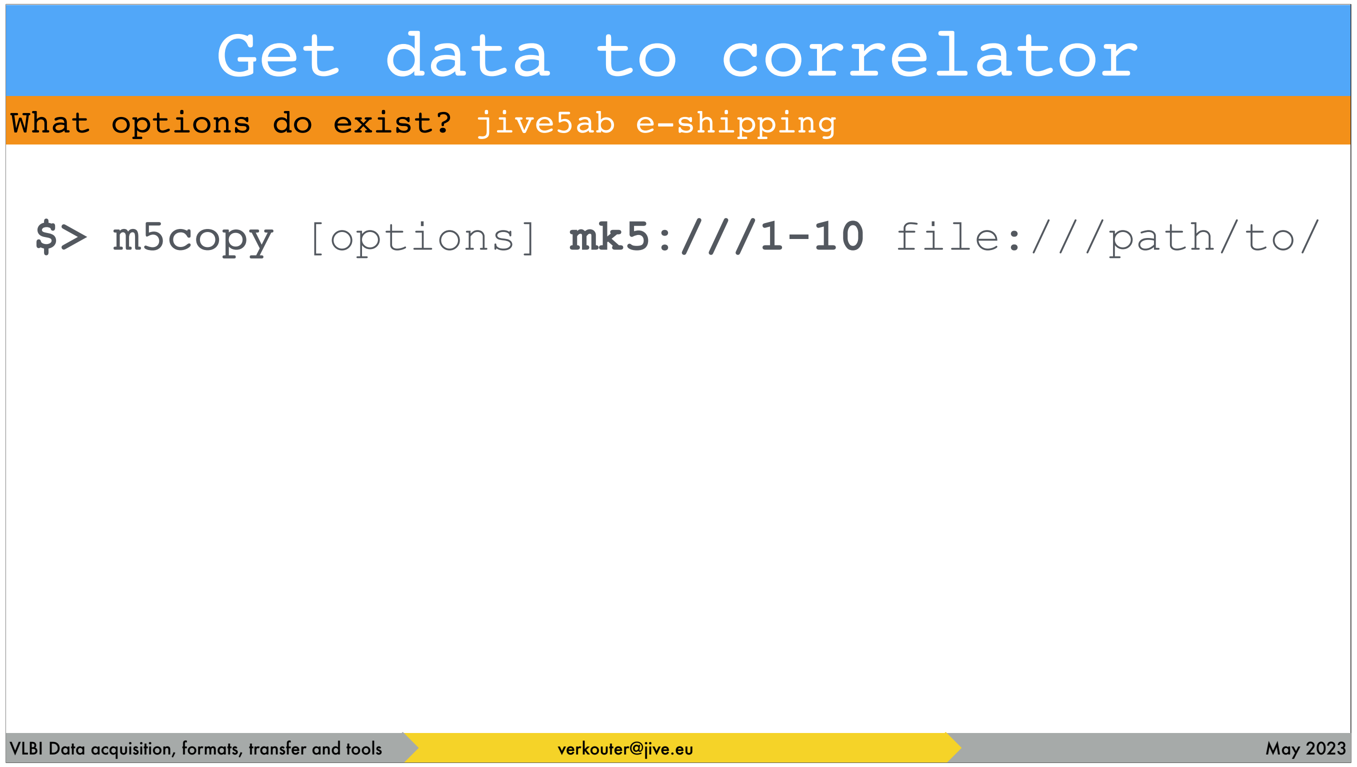


Get data to correlator

What options do exist? jive5ab e-shipping

```
$> m5copy [options] SRC DST
```

then you can use the m5copy command line script basically like this - copy data from source to destination

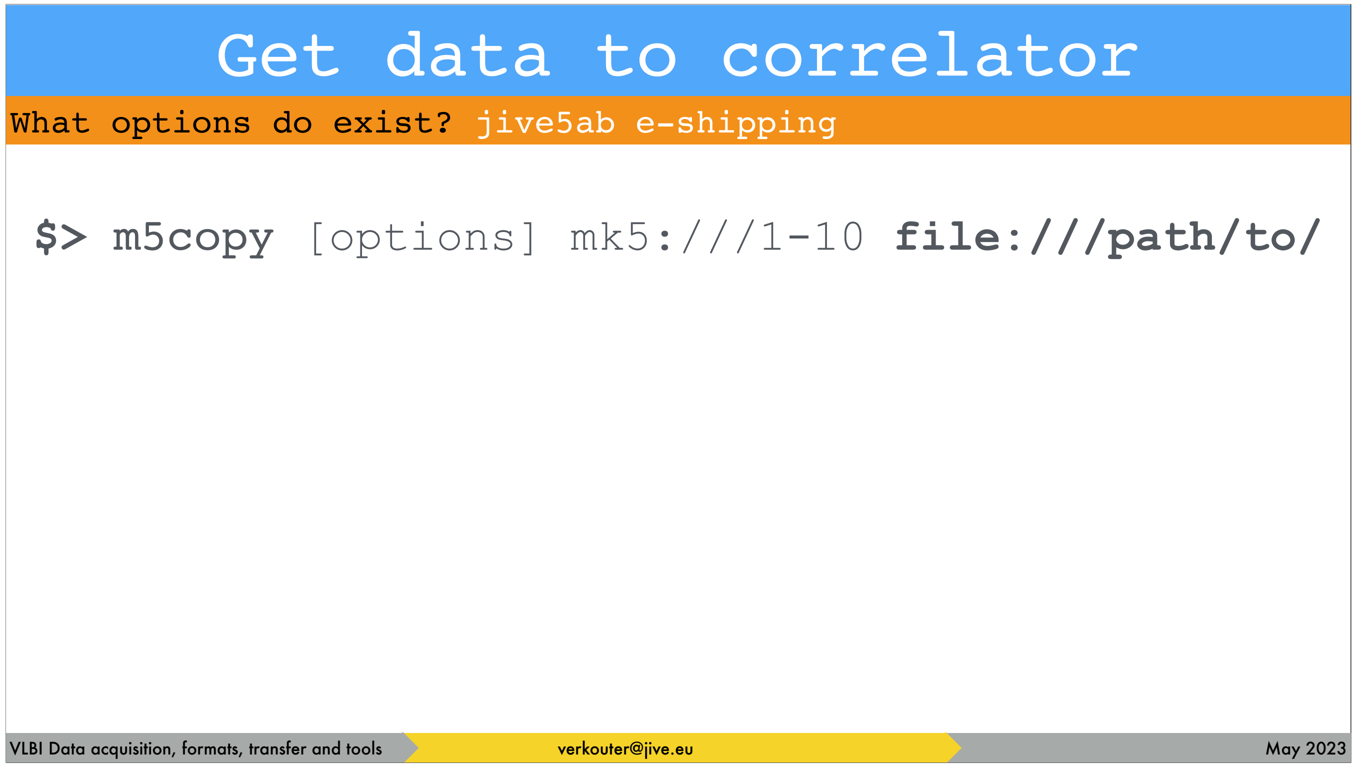


Get data to correlator

What options do exist? jive5ab e-shipping

```
$> m5copy [options] mk5:///1-10 file:///path/to/
```

the source and

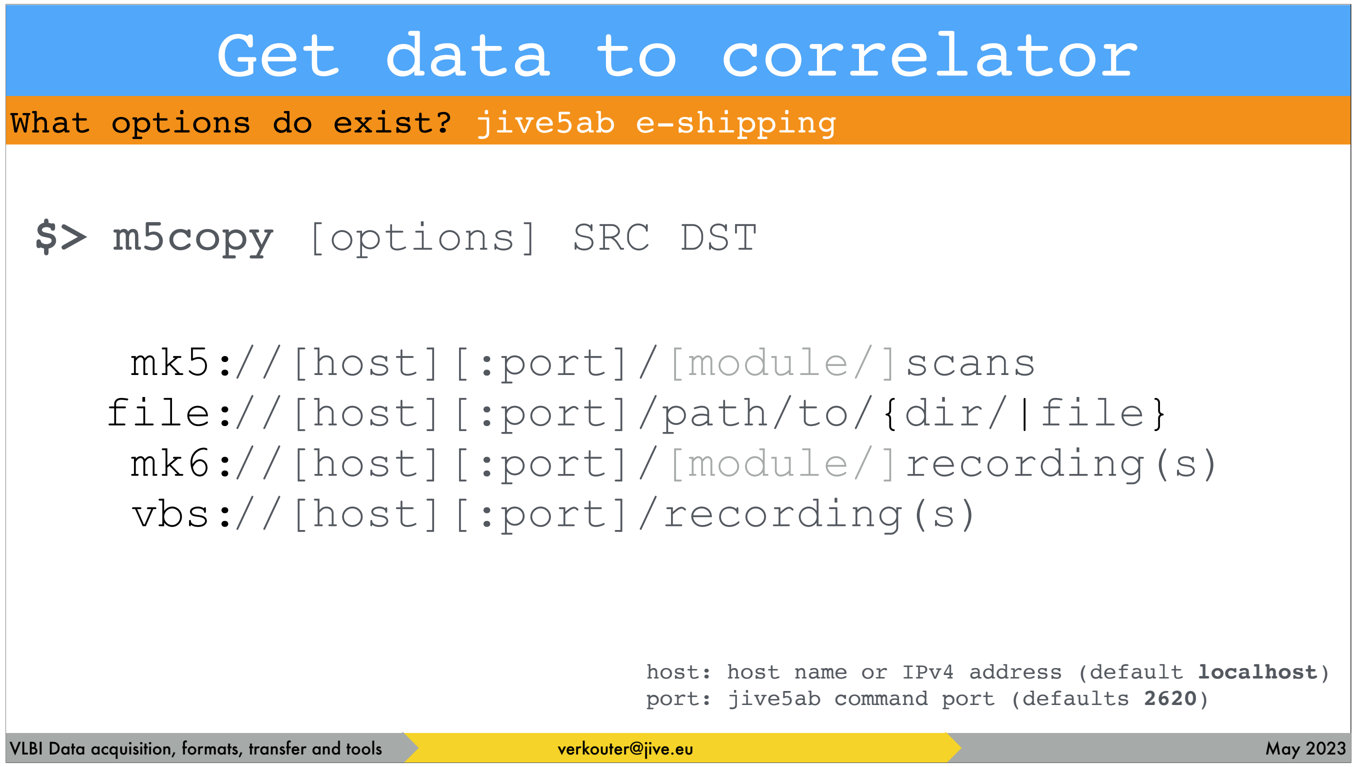


Get data to correlator

What options do exist? jive5ab e-shipping

```
$> m5copy [options] mk5:///1-10 file:///path/to/
```

destinations are URL like specifications



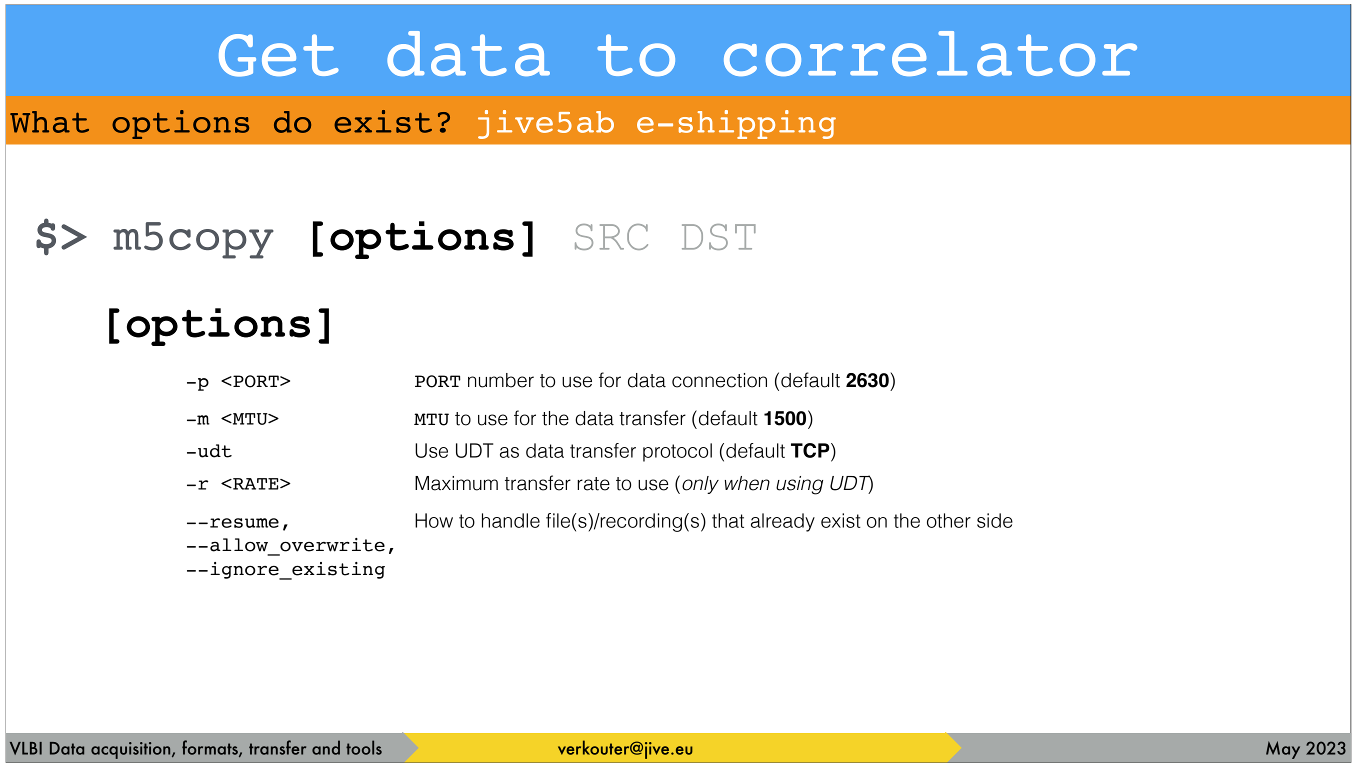
Get data to correlator

What options do exist? jive5ab e-shipping

```
$> m5copy [options] SRC DST

    mk5://[host][:port]/[module/]scans
   file://[host][:port]/path/to/{dir/|file}
    mk6://[host][:port]/[module/]recording(s)
    vbs://[host][:port]/recording(s)
```

```
host: host name or IPv4 address (default localhost)
port: jive5ab command port (defaults 2620)
```

As you can see jive5ab can address most current VLBI data formats and media.



Get data to correlator

What options do exist? jive5ab e-shipping

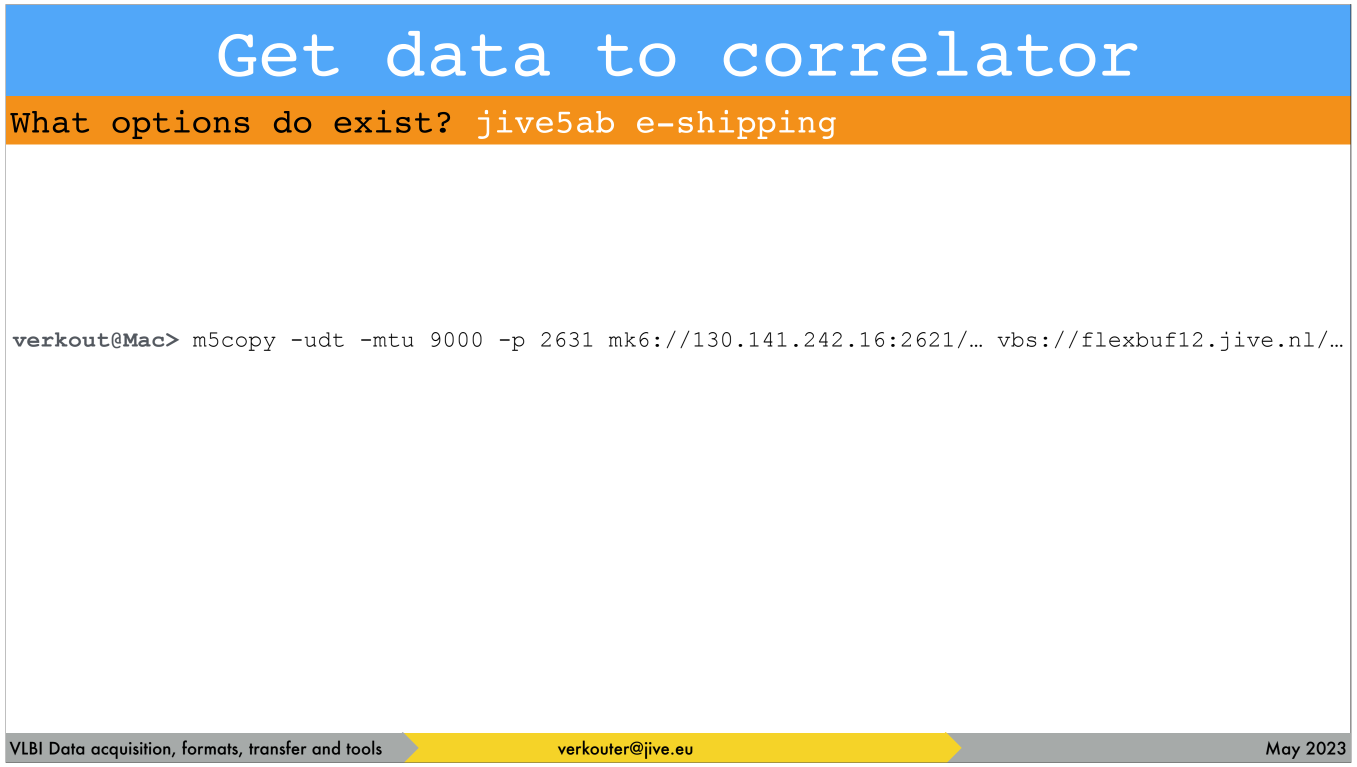
```
$> m5copy [options] SRC DST
```

[options]

`-p <PORT>`	`PORT` number to use for data connection (default **2630**)
`-m <MTU>`	`MTU` to use for the data transfer (default **1500**)
`-udt`	Use UDT as data transfer protocol (default **TCP**)
`-r <RATE>`	Maximum transfer rate to use (*only when using UDT*)

`--resume,`
`--allow_overwrite,`
`--ignore_existing`
	How to handle file(s)/recording(s) that already exist on the other side

And this is a summary of the most important options

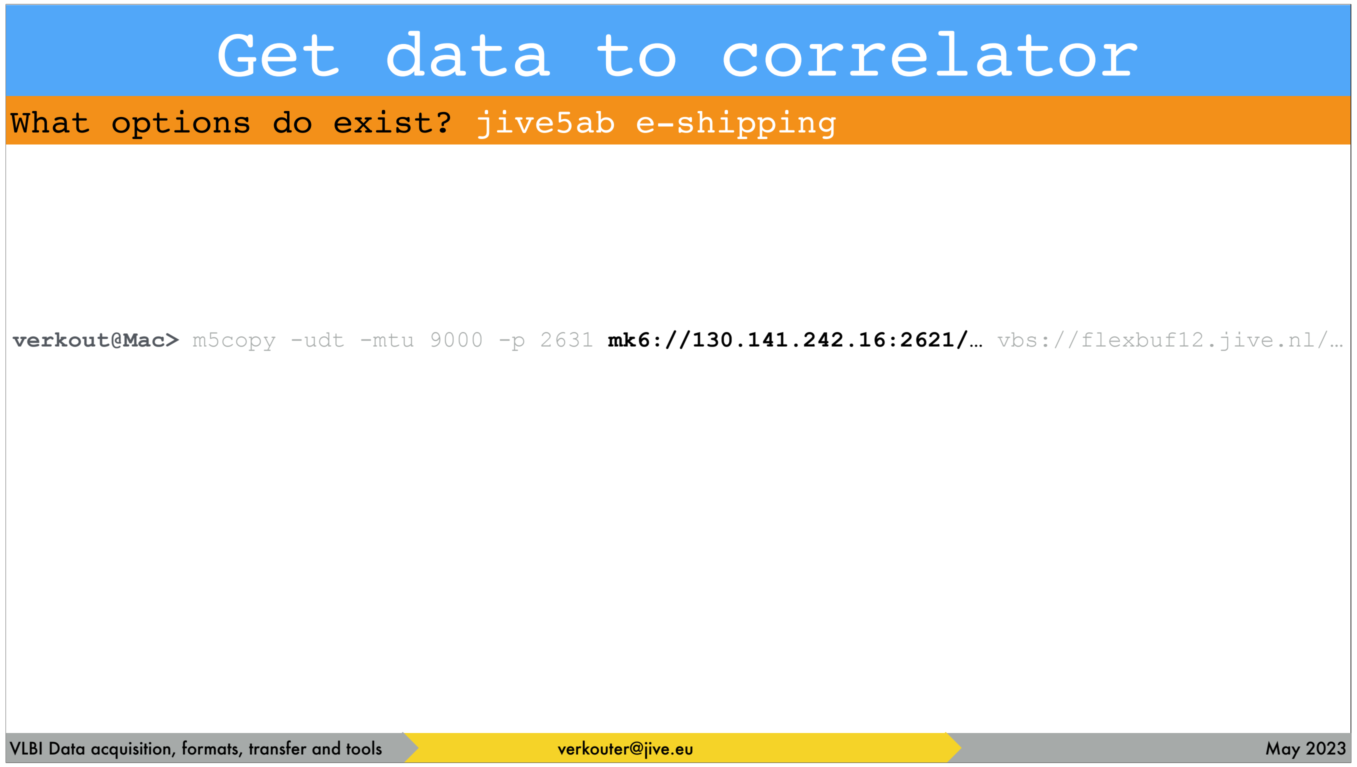


Get data to correlator

What options do exist? jive5ab e-shipping

```
verkout@Mac> m5copy -udt -mtu 9000 -p 2631 mk6://130.141.242.16:2621/… vbs://flexbuf12.jive.nl/…
```

If you issue a command like this

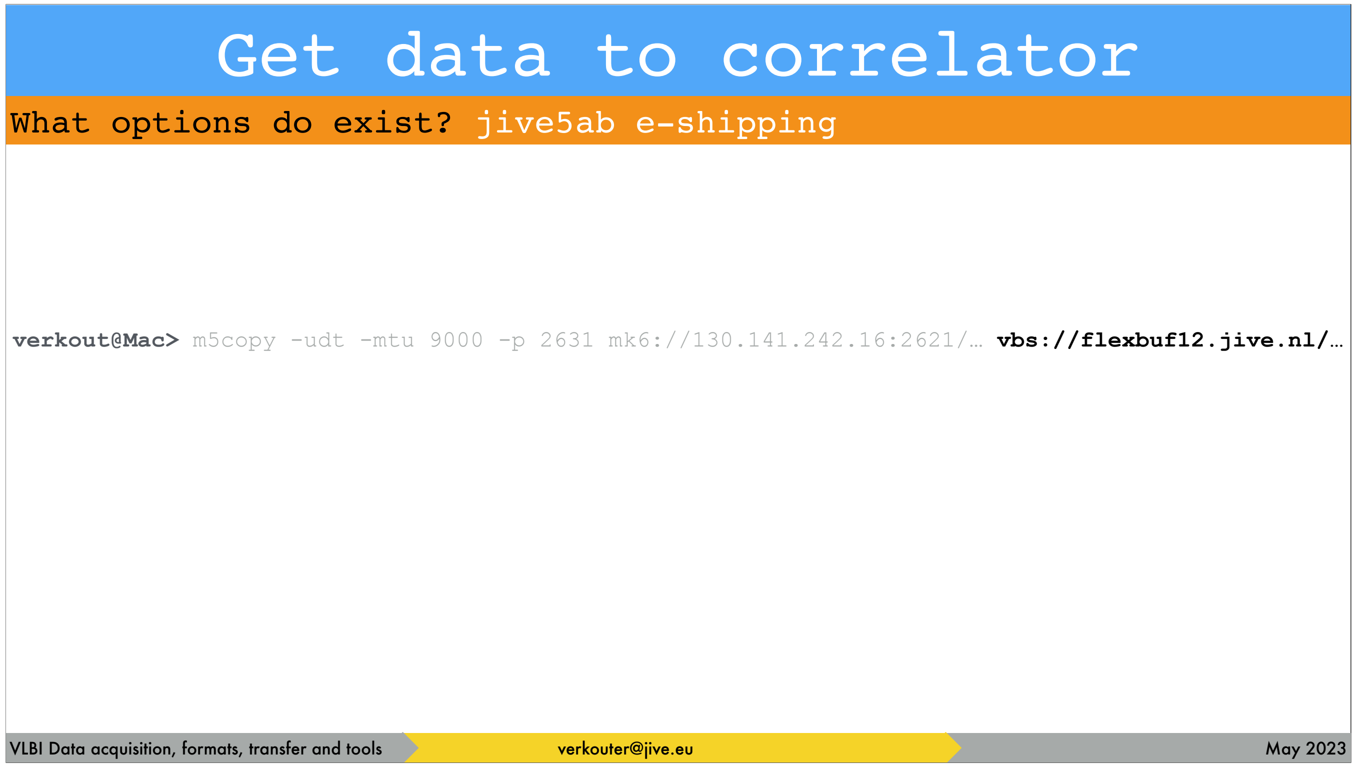


Get data to correlator

What options do exist? jive5ab e-shipping

```
verkout@Mac> m5copy -udt -mtu 9000 -p 2631 mk6://130.141.242.16:2621/… vbs://flexbuf12.jive.nl/…
```

transferring mark6 data from somewhere

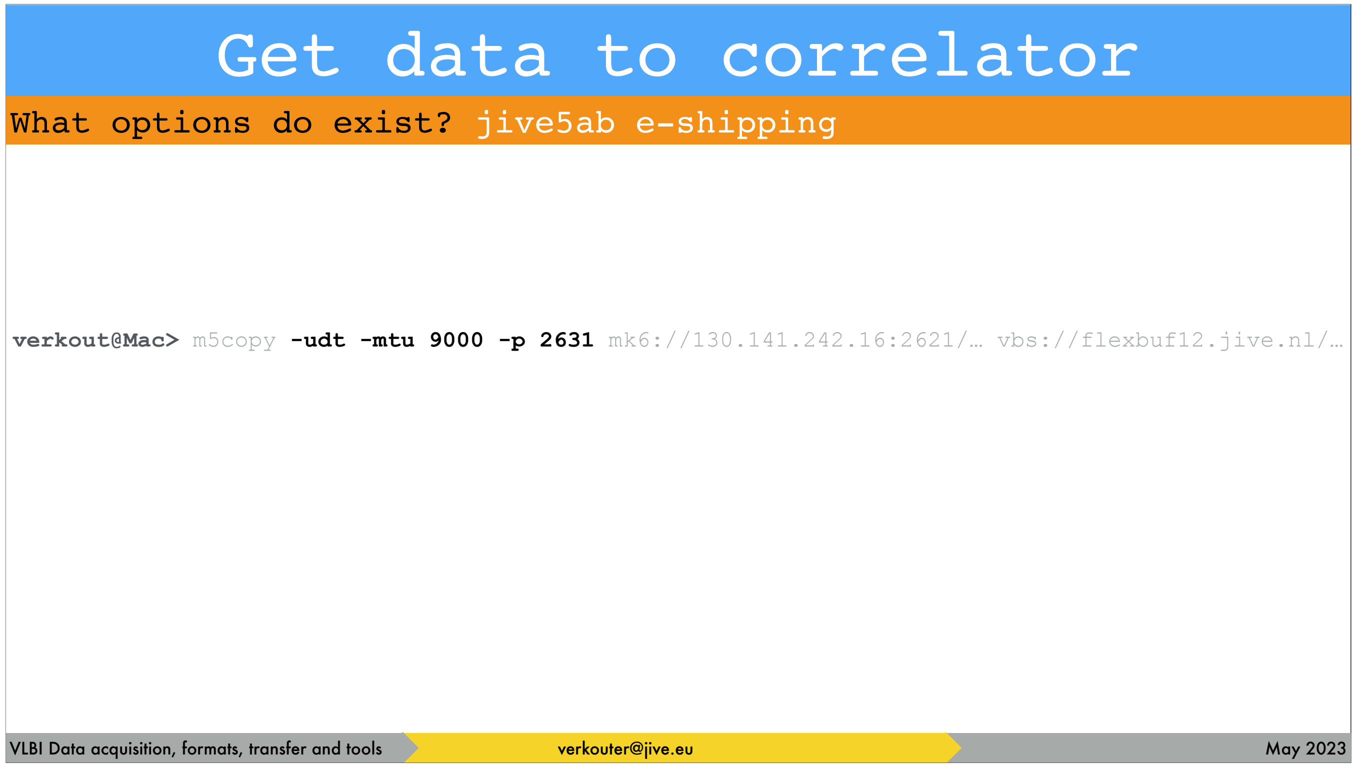


Get data to correlator

What options do exist? jive5ab e-shipping

```
verkout@Mac> m5copy -udt -mtu 9000 -p 2631 mk6://130.141.242.16:2621/… vbs://flexbuf12.jive.nl/…
```

to a flexbuff at JIVE

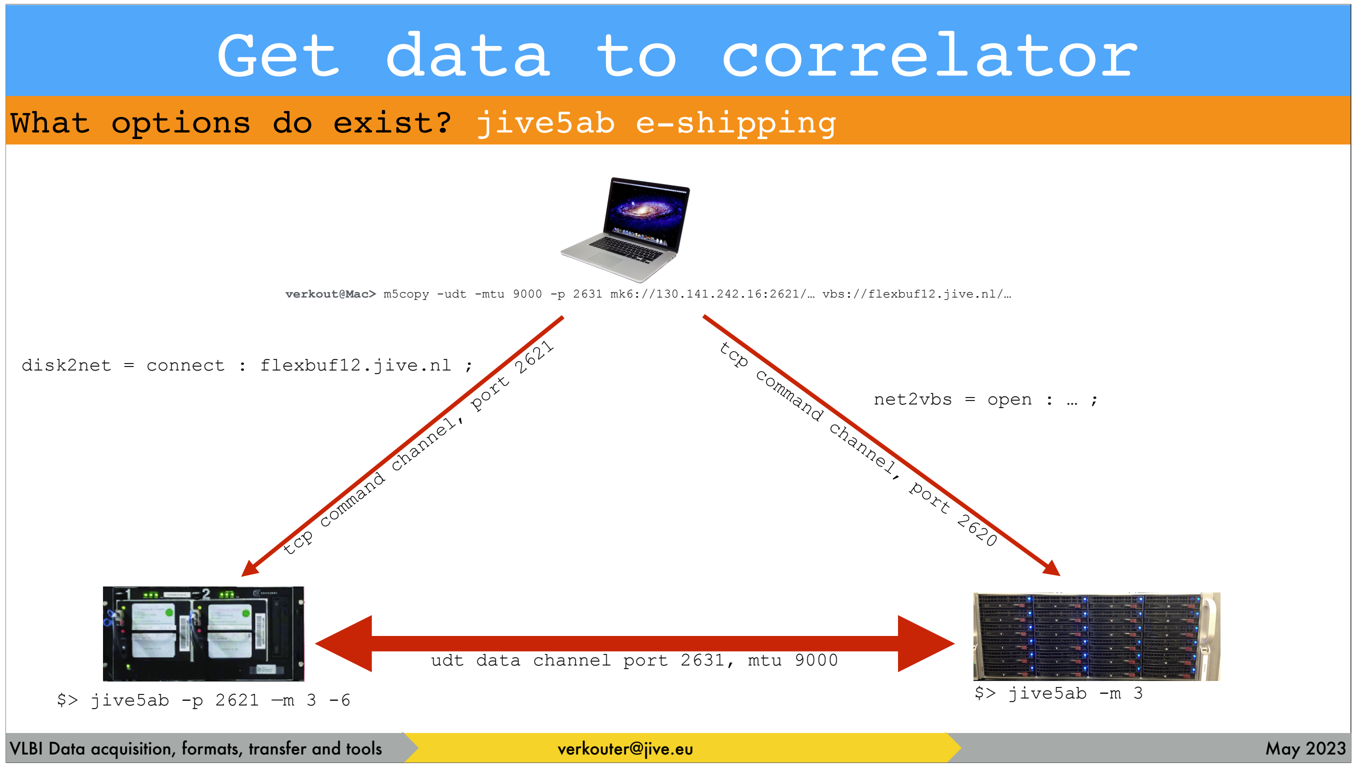


Get data to correlator

What options do exist? jive5ab e-shipping

```
verkout@Mac> m5copy -udt -mtu 9000 -p 2631 mk6://130.141.242.16:2621/… vbs://flexbuf12.jive.nl/…
```

using the UDT protocol over port 2631 using an MTU of 9000



Get data to correlator

What options do exist? jive5ab e-shipping

```
verkout@Mac> m5copy -udt -mtu 9000 -p 2631 mk6://130.141.242.16:2621/… vbs://flexbuf12.jive.nl/…
```

```
disk2net = connect : flexbuf12.jive.nl ;
```

```
net2vbs = open : … ;
```

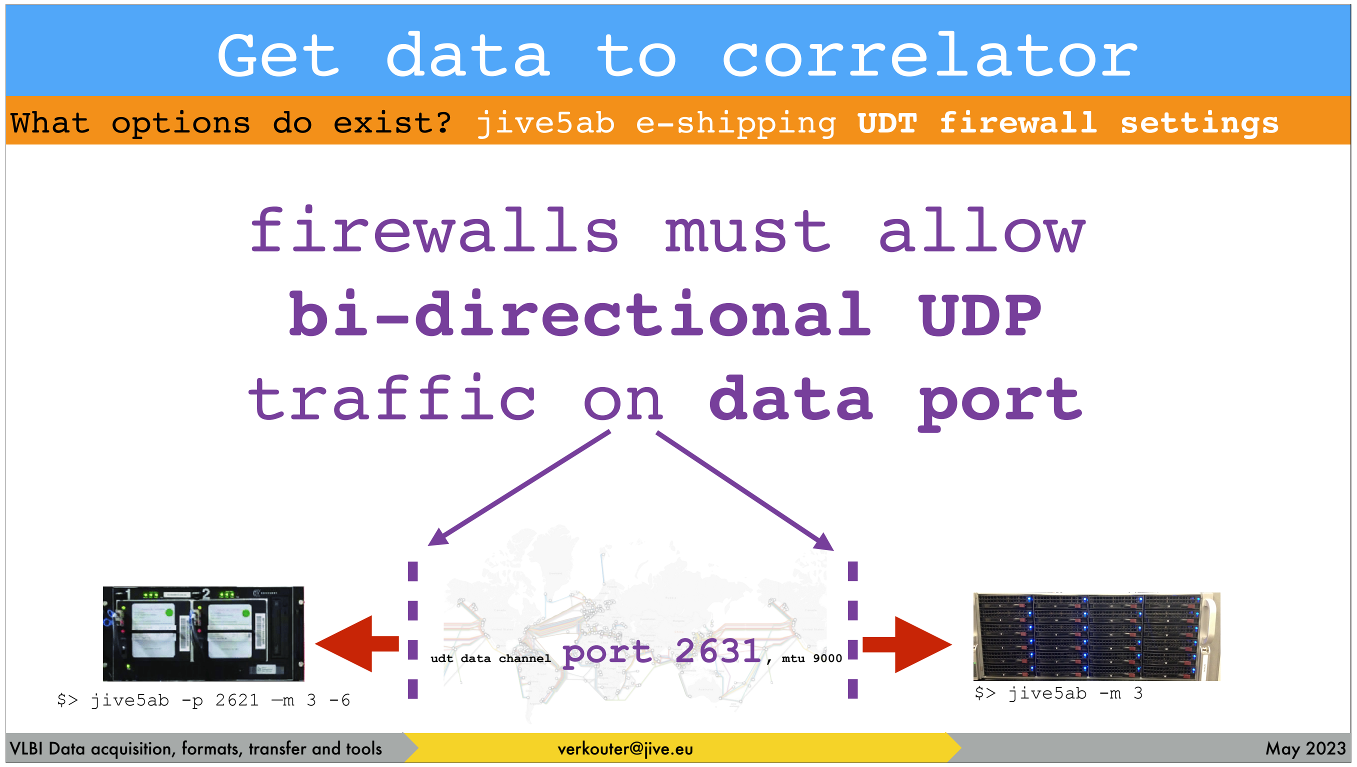
tcp command channel, port 2621
tcp command channel, port 2620
udt data channel port 2631, mtu 9000

```
$> jive5ab -p 2621 –m 3 -6
```

```
$> jive5ab -m 3
```

VLBI Data acquisition, formats, transfer and tools verkouter@jive.eu May 2023

what happens is this. m5copy sends the net2vbs command to the flexbuff and tells the disk2net command on the mark6 to connect directly to the flexbuff using UDT over port 2631 on the FAT LINK. The data will NOT go through your laptop



Get data to correlator

What options do exist? jive5ab e-shipping **UDT firewall settings**

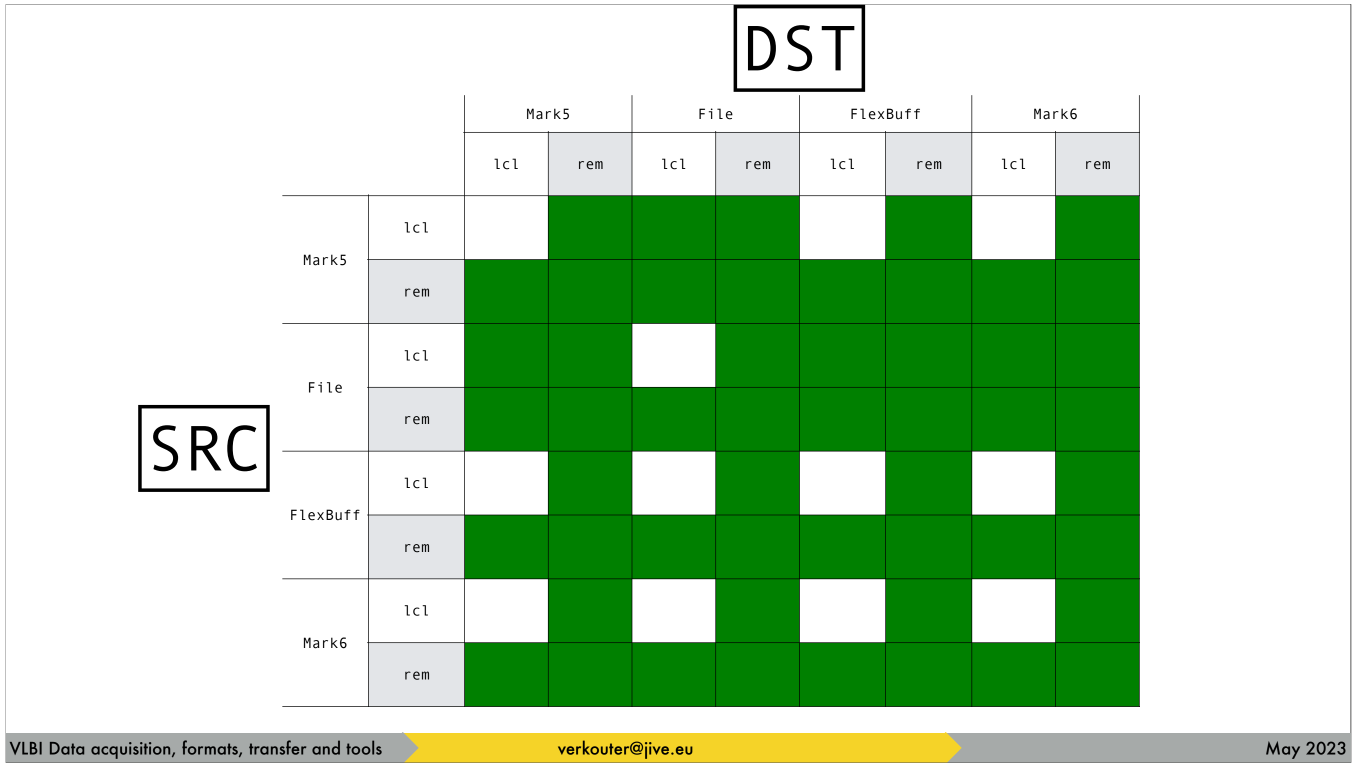
firewalls must allow
bi-directional UDP
traffic on **data port**

udt data channel port 2631, mtu 9000

```
$> jive5ab -p 2621 —m 3 -6
```

```
$> jive5ab -m 3
```

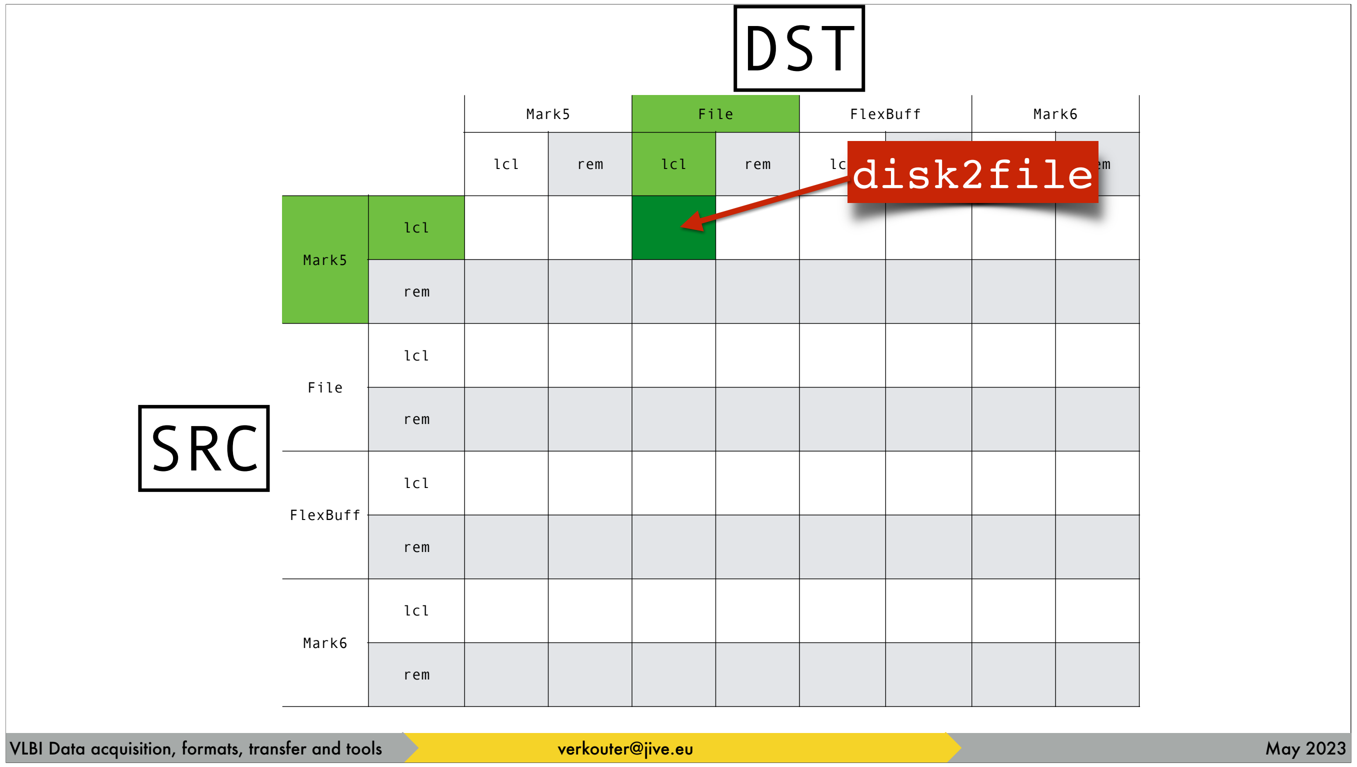
Most systems are behind firewalls. In order to USE the fast UDT protocol both firewalls must allow bi-directional UDP traffic - i.e. in AND outgoing on the data port.



DST

SRC		Mark5		File		FlexBuff		Mark6	
		lcl	rem	lcl	rem	lcl	rem	lcl	rem
Mark5	lcl		🟩	🟩	🟩		🟩		🟩
	rem	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
File	lcl	🟩	🟩		🟩	🟩	🟩	🟩	🟩
	rem	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
FlexBuff	lcl		🟩		🟩		🟩		🟩
	rem	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
Mark6	lcl		🟩		🟩		🟩		🟩
	rem	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩

For reference I include the full connectivity matrix and this might look a bit daunting but … the thing is that both source and destination can be either “the local machine” or remote. The matrix is _mostly_ filled but some combinations just don’t make sense.



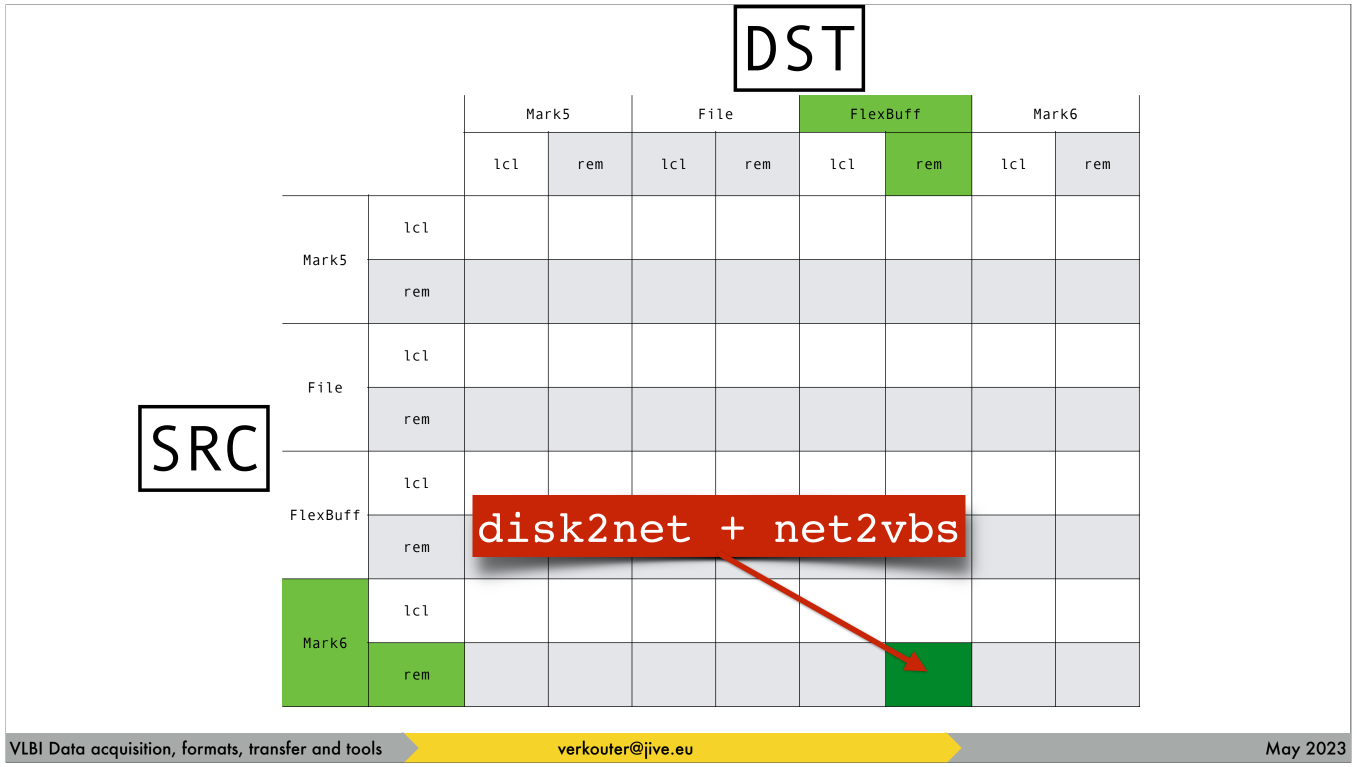
DST

SRC

		Mark5		File		FlexBuff		Mark6	
		lcl	rem	lcl	rem	lcl	rem	lcl	rem
Mark5	lcl			🟩					
	rem								
File	lcl								
	rem								
FlexBuff	lcl								
	rem								
Mark6	lcl								
	rem								

disk2file

the way to read this is for example, for local Mark5 to local file [click] m5copy just executes disk2file



DST

SRC

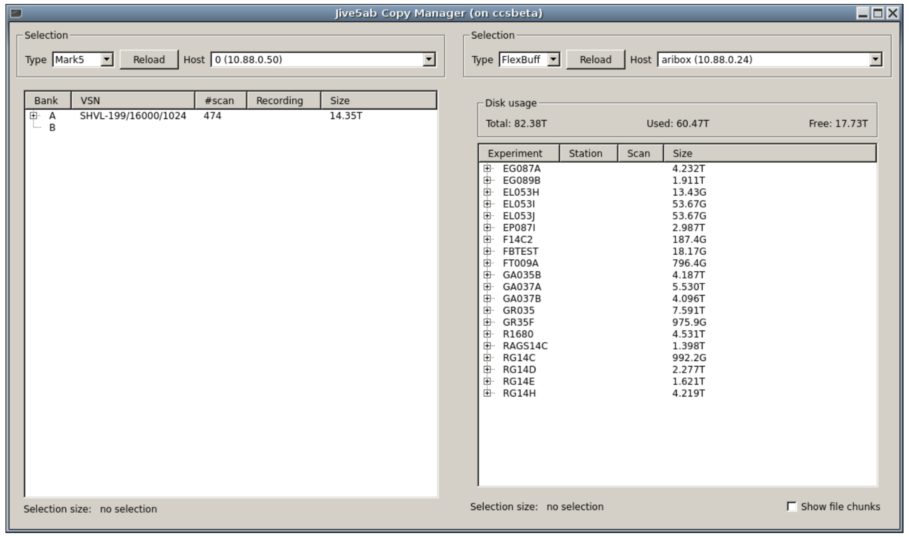
		Mark5		File		FlexBuff		Mark6	
		lcl	rem	lcl	rem	lcl	rem	lcl	rem
Mark5	lcl								
	rem								
File	lcl								
	rem								
FlexBuff	lcl								
	rem								
Mark6	lcl								
	rem						🟩		

disk2net + net2vbs

whilst for remote mark6 to remote flexbuff [click] m5copy couples these two transfers

Get data to correlator

What options do exist? jive5ab + m5copy **GUI frontend**



Jive5ab Copy Manager (on ccsbeta)

Selection

Type Mark5 ▾ Reload Host 0 (10.88.0.50) ▾

Bank	VSN	#scan	Recording	Size
⊞ A	SHVL-199/16000/1024	474		14.35T
B				

Selection size: no selection

Selection

Type FlexBuff ▾ Reload Host aribox (10.88.0.24) ▾

Disk usage

Total: 82.38T Used: 60.47T Free: 17.73T

Experiment	Station	Scan	Size
⊞ EG087A			4.232T
⊞ EG089B			1.911T
⊞ EL053H			13.43G
⊞ EL053I			53.67G
⊞ EL053J			53.67G
⊞ EP087I			2.987T
⊞ F14C2			187.4G
⊞ FBTEST			18.17G
⊞ FT009A			796.4G
⊞ GA035B			4.187T
⊞ GA037A			5.530T
⊞ GA037B			4.096T
⊞ GR035			7.591T
⊞ GR35F			975.9G
⊞ R1680			4.531T
⊞ RAGS14C			1.398T
⊞ RG14C			992.2G
⊞ RG14D			2.277T
⊞ RG14E			1.621T
⊞ RG14H			4.219T

Selection size: no selection

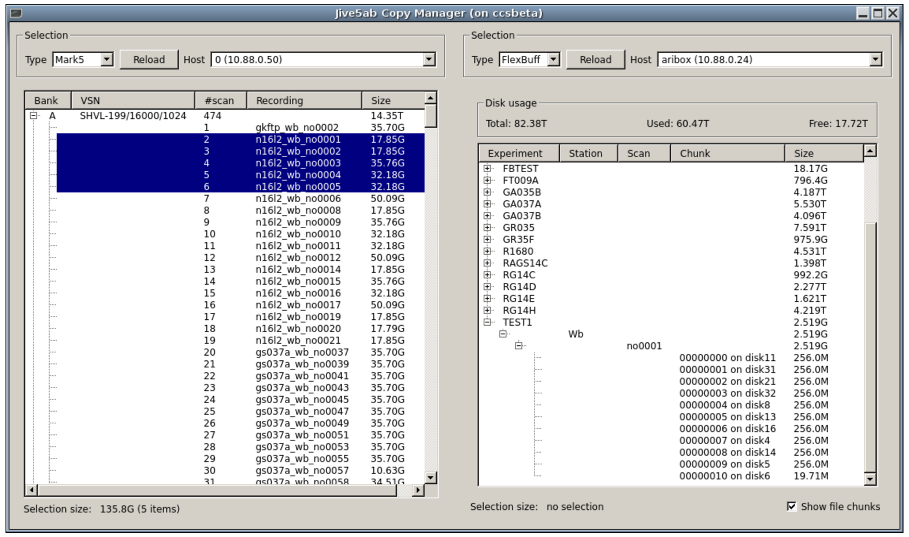
□ Show file chunks

https://github.com/jive-vlbi/jive5ab-copy-manager

There is also a GUI frontend to that drives m5copy for you!

Get data to correlator

What options do exist? jive5ab + m5copy **GUI frontend**



Jive5ab Copy Manager (on ccsbeta)

Selection — Type: Mark5 | Reload | Host: 0 (10.88.0.50)

Bank	VSN	#scan	Recording	Size
A	SHVL-199/16000/1024	474		14.35T
		1	gkftp_wb_no0002	35.70G
		2	n16l2_wb_no0001	17.85G
		3	n16l2_wb_no0002	17.85G
		4	n16l2_wb_no0003	35.76G
		5	n16l2_wb_no0004	32.18G
		6	n16l2_wb_no0005	32.18G
		7	n16l2_wb_no0006	50.09G
		8	n16l2_wb_no0008	17.85G
		9	n16l2_wb_no0009	35.76G
		10	n16l2_wb_no0010	32.18G
		11	n16l2_wb_no0011	32.18G
		12	n16l2_wb_no0012	50.09G
		13	n16l2_wb_no0014	17.85G
		14	n16l2_wb_no0015	35.76G
		15	n16l2_wb_no0016	32.18G
		16	n16l2_wb_no0017	50.09G
		17	n16l2_wb_no0019	17.85G
		18	n16l2_wb_no0020	17.79G
		19	n16l2_wb_no0021	17.85G
		20	gs037a_wb_no0037	35.70G
		21	gs037a_wb_no0039	35.70G
		22	gs037a_wb_no0041	35.70G
		23	gs037a_wb_no0043	35.70G
		24	gs037a_wb_no0045	35.70G
		25	gs037a_wb_no0047	35.70G
		26	gs037a_wb_no0049	35.70G
		27	gs037a_wb_no0051	35.70G
		28	gs037a_wb_no0053	35.70G
		29	gs037a_wb_no0055	35.70G
		30	gs037a_wb_no0057	10.63G
		31	gs037a_wb_no0058	34.51G

Selection size: 135.8G (5 items)

Selection — Type: FlexBuff | Reload | Host: aribox (10.88.0.24)

Disk usage — Total: 82.38T Used: 60.47T Free: 17.72T

Experiment	Station	Scan	Chunk	Size
FBTEST				18.17G
FT009A				796.4G
GA035B				4.187T
GA037A				5.530T
GA037B				4.096T
GR035				7.591T
GR35F				975.9G
R1680				4.531T
RAGS14C				1.398T
RG14C				992.2G
RG14D				2.277T
RG14E				1.621T
RG14H				4.219T
TEST1				2.519G
	Wb			2.519G
		no0001		2.519G
			00000000 on disk11	256.0M
			00000001 on disk31	256.0M
			00000002 on disk21	256.0M
			00000003 on disk32	256.0M
			00000004 on disk8	256.0M
			00000005 on disk13	256.0M
			00000006 on disk16	256.0M
			00000007 on disk4	256.0M
			00000008 on disk14	256.0M
			00000009 on disk5	256.0M
			00000010 on disk6	19.71M

Selection size: no selection

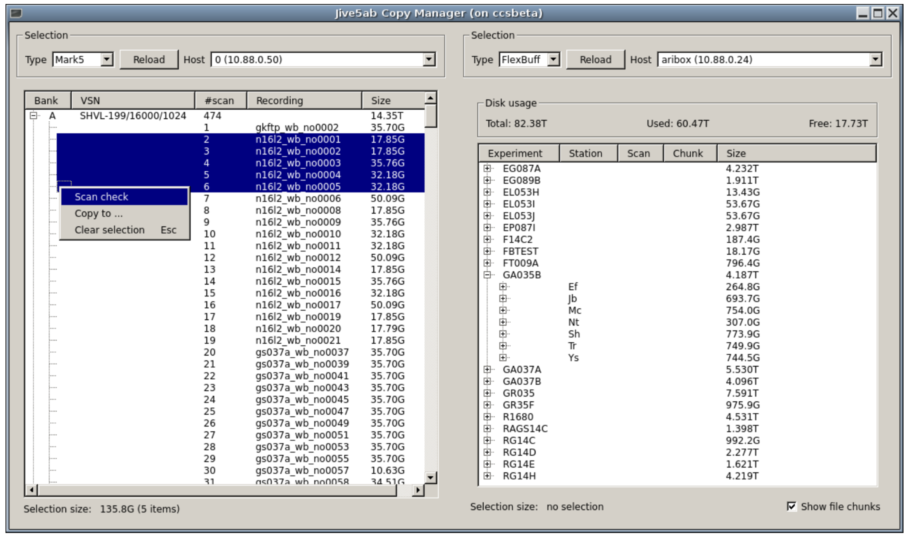
☑ Show file chunks

https://github.com/jive-vlbi/jive5ab-copy-manager

you can easily select multiple scans

Get data to correlator

What options do exist? jive5ab + m5copy **GUI frontend**



Jive5ab Copy Manager (on ccsbeta)

Selection — Type: Mark5 — Reload — Host: 0 (10.88.0.50)

Bank	VSN	#scan	Recording	Size
A	SHVL-199/16000/1024	474		14.35T
		1	gkftp_wb_no0002	35.70G
		2	n16l2_wb_no0001	17.85G
		3	n16l2_wb_no0002	17.85G
		4	n16l2_wb_no0003	35.76G
		5	n16l2_wb_no0004	32.18G
		6	n16l2_wb_no0005	32.18G
		7	n16l2_wb_no0006	50.09G
		8	n16l2_wb_no0008	17.85G
		9	n16l2_wb_no0009	35.76G
		10	n16l2_wb_no0010	32.18G
		11	n16l2_wb_no0011	32.18G
		12	n16l2_wb_no0012	50.09G
		13	n16l2_wb_no0014	17.85G
		14	n16l2_wb_no0015	35.76G
		15	n16l2_wb_no0016	32.18G
		16	n16l2_wb_no0017	50.09G
		17	n16l2_wb_no0019	17.85G
		18	n16l2_wb_no0020	17.79G
		19	n16l2_wb_no0021	17.85G
		20	gs037a_wb_no0037	35.70G
		21	gs037a_wb_no0039	35.70G
		22	gs037a_wb_no0041	35.70G
		23	gs037a_wb_no0043	35.70G
		24	gs037a_wb_no0045	35.70G
		25	gs037a_wb_no0047	35.70G
		26	gs037a_wb_no0049	35.70G
		27	gs037a_wb_no0051	35.70G
		28	gs037a_wb_no0053	35.70G
		29	gs037a_wb_no0055	35.70G
		30	gs037a_wb_no0057	10.63G
		31	gs037a_wb_no0058	34.51G

Selection size: 135.8G (5 items)

Selection — Type: FlexBuff — Reload — Host: aribox (10.88.0.24)

Disk usage — Total: 82.38T Used: 60.47T Free: 17.73T

Experiment	Station	Scan	Chunk	Size
EG087A				4.232T
EG089B				1.911T
EL053H				13.43G
EL053I				53.67G
EL053J				53.67G
EP087I				2.987T
F14C2				187.4G
FBTEST				18.17G
FT009A				796.4G
GA035B				4.187T
	Ef			264.8G
	Jb			693.7G
	Mc			754.0G
	Nt			307.0G
	Sh			773.9G
	Tr			749.9G
	Ys			744.5G
GA037A				5.530T
GA037B				4.096T
GR035				7.591T
GR35F				975.9G
R1680				4.531T
RAGS14C				1.398T
RG14C				992.2G
RG14D				2.277T
RG14E				1.621T
RG14H				4.219T

Selection size: no selection

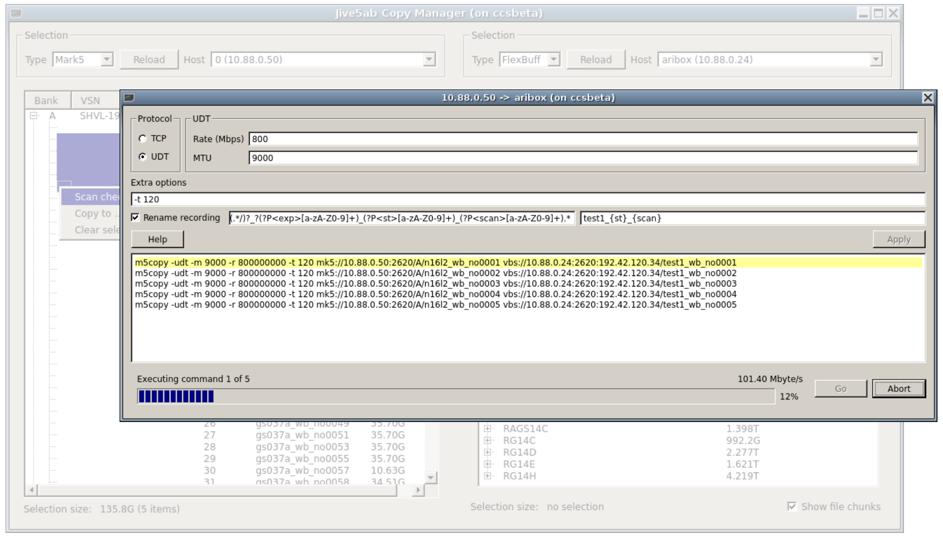
☑ Show file chunks

https://github.com/jive-vlbi/jive5ab-copy-manager

and with a right-click

Get data to correlator

What options do exist? jive5ab + m5copy **GUI frontend**



Type Mark5 Host 0 (10.88.0.50)

Type FlexBuff Host aribox (10.88.0.24)

○ TCP
● UDT

Rate (Mbps) 800
MTU 9000

Extra options: -t 120

☑ Rename recording (.*/)?_?(?P<exp>[a-zA-Z0-9]+)_(?P<st>[a-zA-Z0-9]+)_(?P<scan>[a-zA-Z0-9]+).* test1_{st}_{scan}

```
m5copy -udt -m 9000 -r 800000000 -t 120 mk5://10.88.0.50:2620/A/n16l2_wb_no0001 vbs://10.88.0.24:2620:192.42.120.34/test1_wb_no0001
m5copy -udt -m 9000 -r 800000000 -t 120 mk5://10.88.0.50:2620/A/n16l2_wb_no0002 vbs://10.88.0.24:2620:192.42.120.34/test1_wb_no0002
m5copy -udt -m 9000 -r 800000000 -t 120 mk5://10.88.0.50:2620/A/n16l2_wb_no0003 vbs://10.88.0.24:2620:192.42.120.34/test1_wb_no0003
m5copy -udt -m 9000 -r 800000000 -t 120 mk5://10.88.0.50:2620/A/n16l2_wb_no0004 vbs://10.88.0.24:2620:192.42.120.34/test1_wb_no0004
m5copy -udt -m 9000 -r 800000000 -t 120 mk5://10.88.0.50:2620/A/n16l2_wb_no0005 vbs://10.88.0.24:2620:192.42.120.34/test1_wb_no0005
```

Executing command 1 of 5 101.40 Mbyte/s 12%

```
26  gs037a_wb_no0049  35.70G
27  gs037a_wb_no0051  35.70G
28  gs037a_wb_no0053  35.70G
29  gs037a_wb_no0055  35.70G
30  gs037a_wb_no0057  10.63G
31  gs037a_wb_no0058  34.51G
```

```
RAGS14C  1.398T
RG14C    992.2G
RG14D    2.277T
RG14E    1.621T
RG14H    4.219T
```

Selection size: 135.8G (5 items)

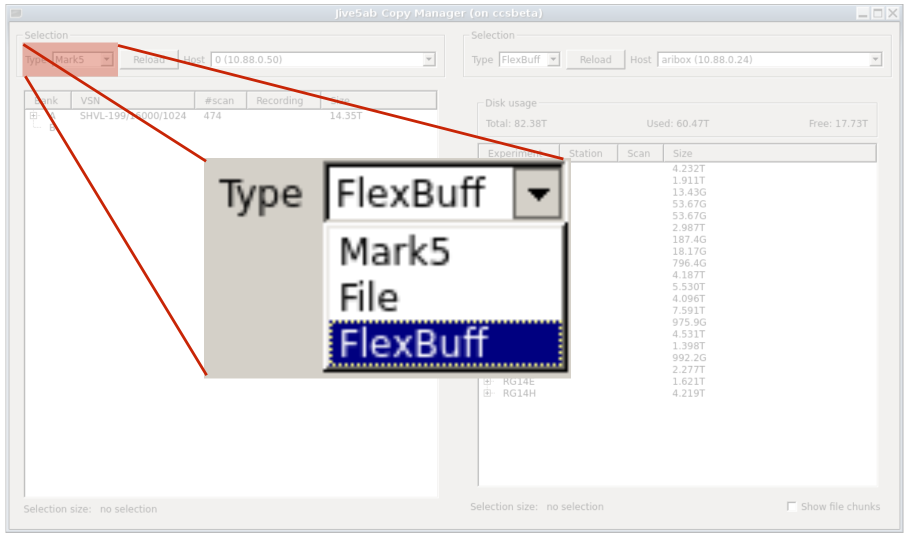
Selection size: no selection ☑ Show file chunks

https://github.com/jive-vlbi/jive5ab-copy-manager

easily copy them across

Get data to correlator

What options do exist? jive5ab + m5copy **GUI frontend**

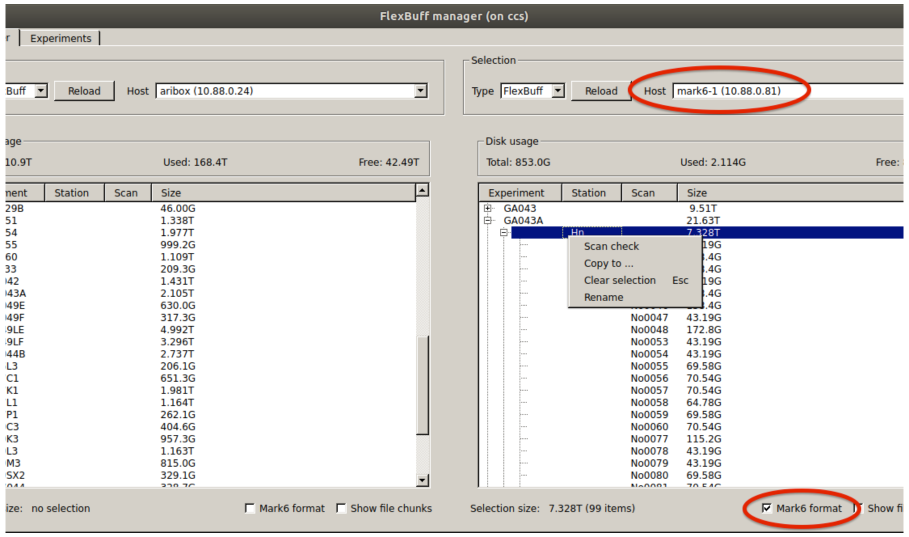


https://github.com/jive-vlbi/jive5ab-copy-manager

You can copy from any of the supported types

Get data to correlator

What options do exist? jive5ab + m5copy **GUI frontend**



FlexBuff manager (on ccs)

Buff | Reload | Host aribox (10.88.0.24)

age
10.9T   Used: 168.4T   Free: 42.49T

ment	Station	Scan	Size
29B			46.00G
51			1.338T
54			1.977T
55			999.2G
60			1.109T
33			209.3G
42			1.431T
43A			2.105T
49E			630.0G
49F			317.3G
9LE			4.992T
9LF			3.296T
44B			2.737T
L3			206.1G
C1			651.3G
K1			1.981T
L1			1.164T
P1			262.1G
C3			404.6G
K3			957.3G
L3			1.163T
M3			815.0G
SX2			329.1G

ize: no selection   □ Mark6 format   □ Show file chunks

Selection
Type FlexBuff | Reload | Host mark6-1 (10.88.0.81)

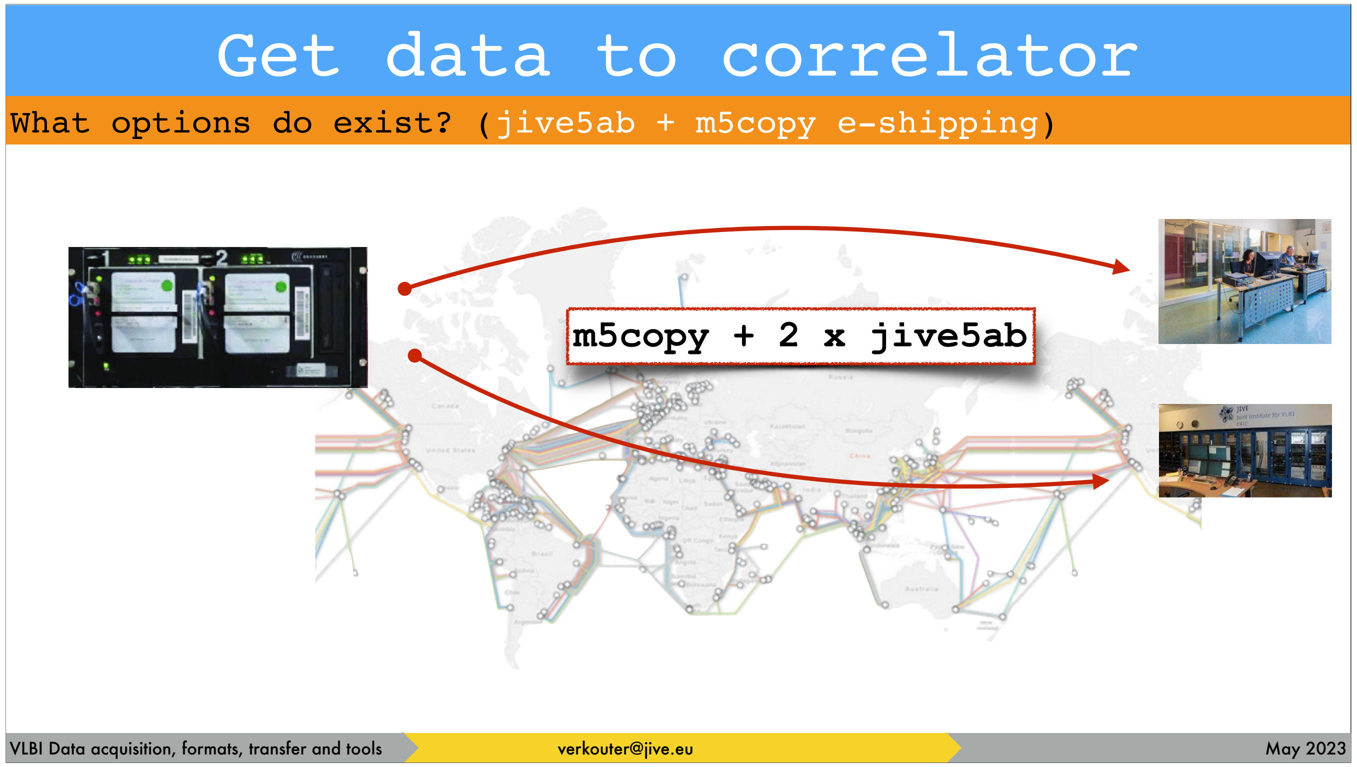
Disk usage
Total: 853.0G   Used: 2.114G   Free:

Experiment	Station	Scan	Size
⊞ GA043			9.51T
⊟ GA043A			21.63T
⊟	Hn		7.328T
			19G
			.4G
			.4G
			19G
			.4G
			.4G
		No0047	43.19G
		No0048	172.8G
		No0053	43.19G
		No0054	43.19G
		No0055	69.58G
		No0056	70.54G
		No0057	70.54G
		No0058	64.78G
		No0059	69.58G
		No0060	70.54G
		No0077	115.2G
		No0078	43.19G
		No0079	43.19G
		No0080	69.58G

Selection size: 7.328T (99 items)   ☑ Mark6 format   □ Show fi

https://github.com/jive-vlbi/jive5ab-copy-manager

and the latest version also supports Mark6 natively

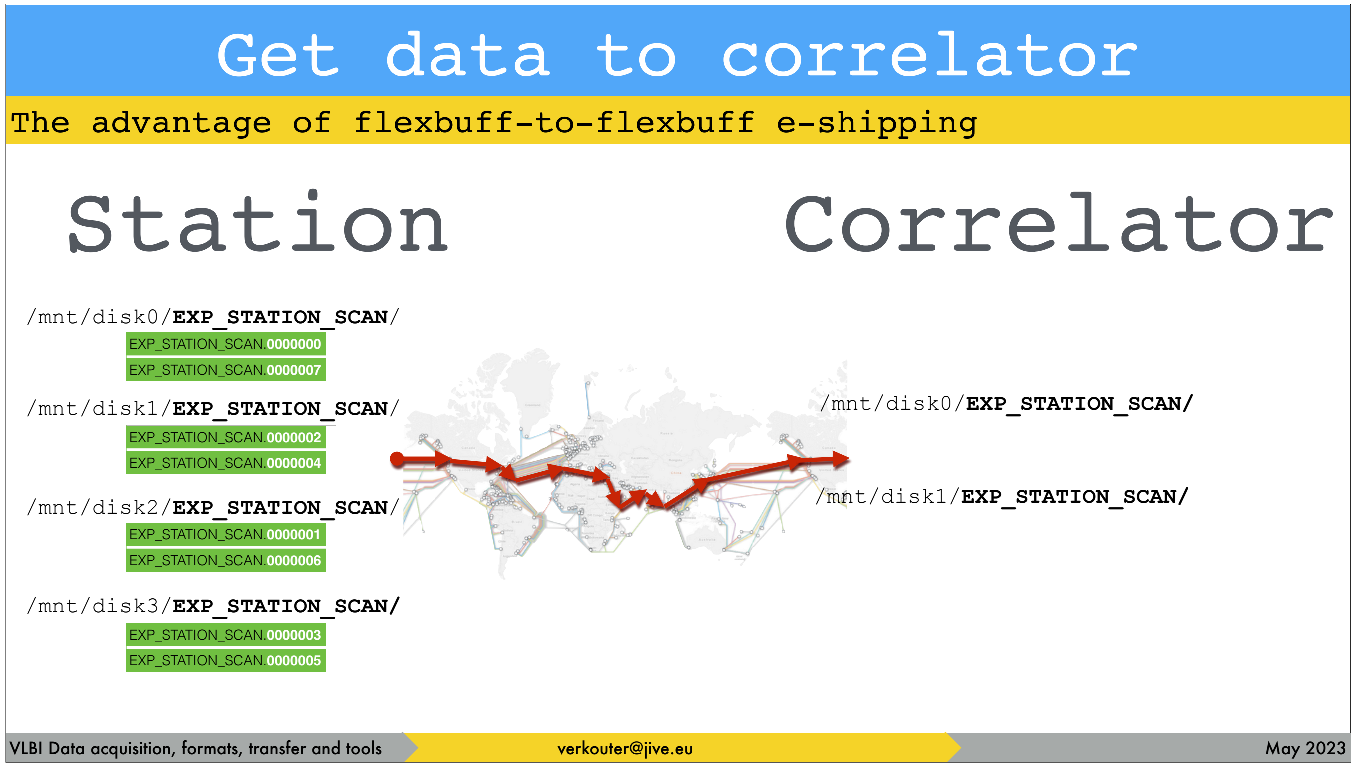


Get data to correlator

What options do exist? (jive5ab + m5copy e-shipping)

m5copy + 2 x jive5ab

So using e-transfer you can ship data to several correlators without mucking with the modules!



Get data to correlator

The advantage of flexbuff-to-flexbuff e-shipping

Station

```
/mnt/disk0/EXP_STATION_SCAN/
        EXP_STATION_SCAN.0000000
        EXP_STATION_SCAN.0000007
/mnt/disk1/EXP_STATION_SCAN/
        EXP_STATION_SCAN.0000002
        EXP_STATION_SCAN.0000004
/mnt/disk2/EXP_STATION_SCAN/
        EXP_STATION_SCAN.0000001
        EXP_STATION_SCAN.0000006
/mnt/disk3/EXP_STATION_SCAN/
        EXP_STATION_SCAN.0000003
        EXP_STATION_SCAN.0000005
```

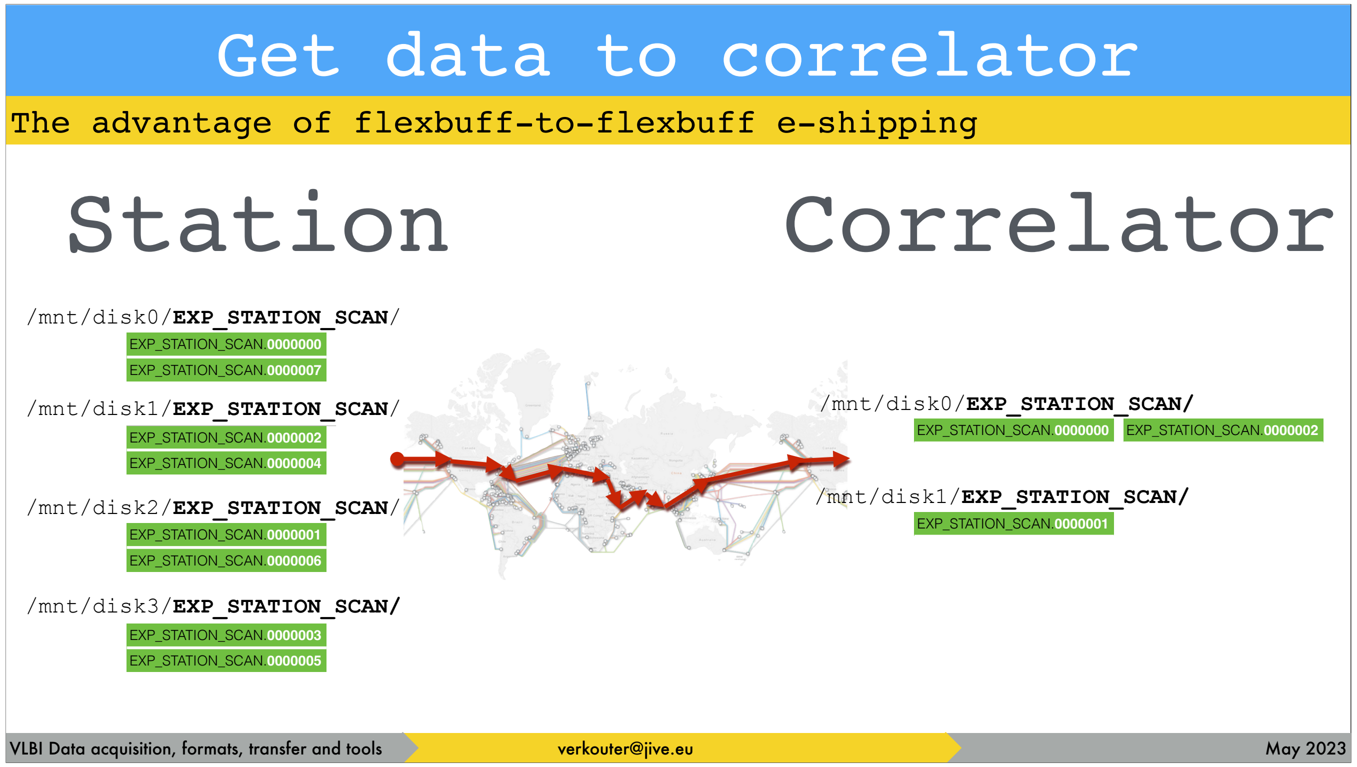
Correlator

```
/mnt/disk0/EXP_STATION_SCAN/
/mnt/disk1/EXP_STATION_SCAN/
```

VLBI Data acquisition, formats, transfer and tools verkouter@jive.eu May 2023

The flexbuff recording format is very suitable for e-shipping. It is independent of the number of disks present at source or destination. Let’s assume the station [click] has a flexbuff with four mountpoints with data and the [click] correlator one with just two.

Initially there is no data from the experiment at the correlator. [click] After starting the transfer the two jive5ab’s exchange information which chunks are missing …



Get data to correlator

The advantage of flexbuff-to-flexbuff e-shipping

Station

/mnt/disk0/**EXP_STATION_SCAN**/

EXP_STATION_SCAN.0000000
EXP_STATION_SCAN.0000007

/mnt/disk1/**EXP_STATION_SCAN**/

EXP_STATION_SCAN.0000002
EXP_STATION_SCAN.0000004

/mnt/disk2/**EXP_STATION_SCAN**/

EXP_STATION_SCAN.0000001
EXP_STATION_SCAN.0000006

/mnt/disk3/**EXP_STATION_SCAN/**

EXP_STATION_SCAN.0000003
EXP_STATION_SCAN.0000005

Correlator

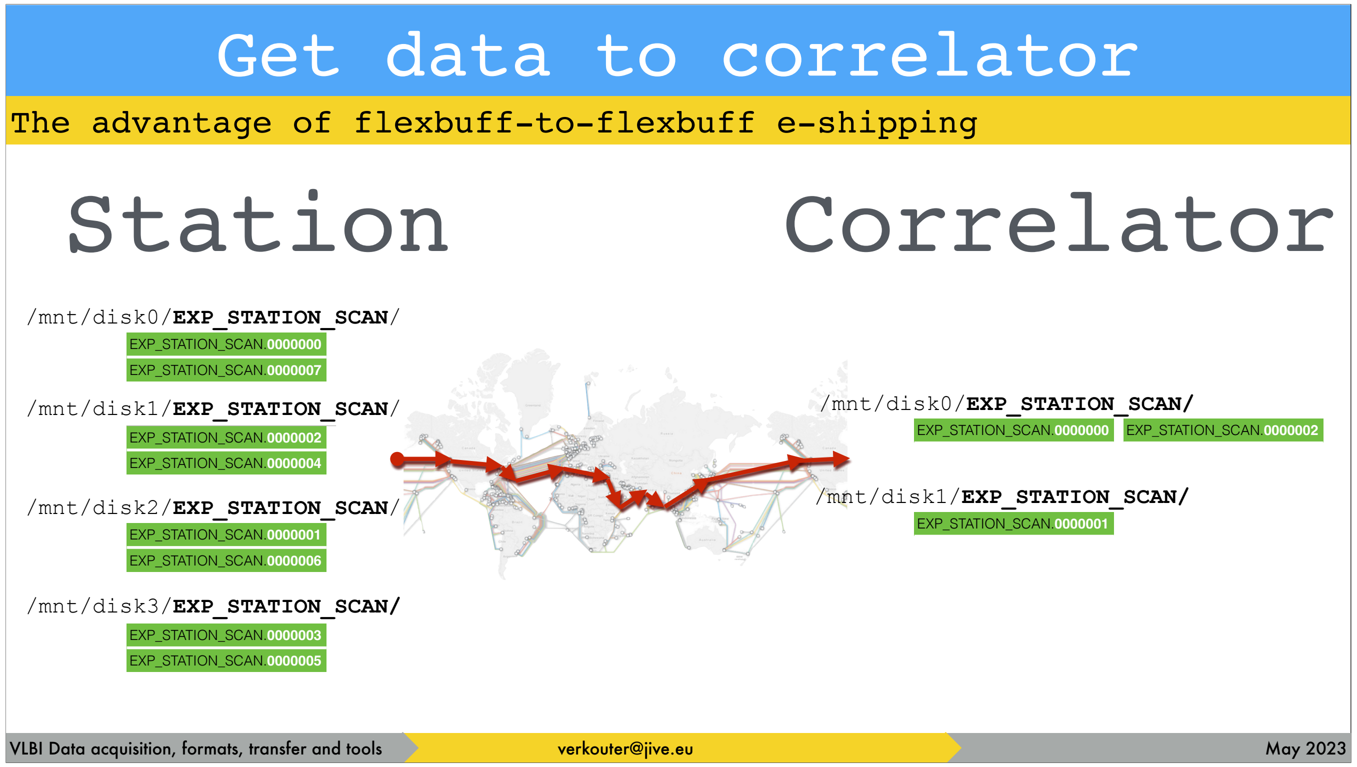
/mnt/disk0/**EXP_STATION_SCAN/**

EXP_STATION_SCAN.0000000 EXP_STATION_SCAN.0000002

/mnt/disk1/**EXP_STATION_SCAN/**

EXP_STATION_SCAN.0000001

and start transferring them. [click] If the connection breaks or the program is stopped …



Get data to correlator

The advantage of flexbuff-to-flexbuff e-shipping

Station

/mnt/disk0/**EXP_STATION_SCAN**/

- EXP_STATION_SCAN.0000000
- EXP_STATION_SCAN.0000007

/mnt/disk1/**EXP_STATION_SCAN**/

- EXP_STATION_SCAN.0000002
- EXP_STATION_SCAN.0000004

/mnt/disk2/**EXP_STATION_SCAN**/

- EXP_STATION_SCAN.0000001
- EXP_STATION_SCAN.0000006

/mnt/disk3/**EXP_STATION_SCAN**/

- EXP_STATION_SCAN.0000003
- EXP_STATION_SCAN.0000005

Correlator

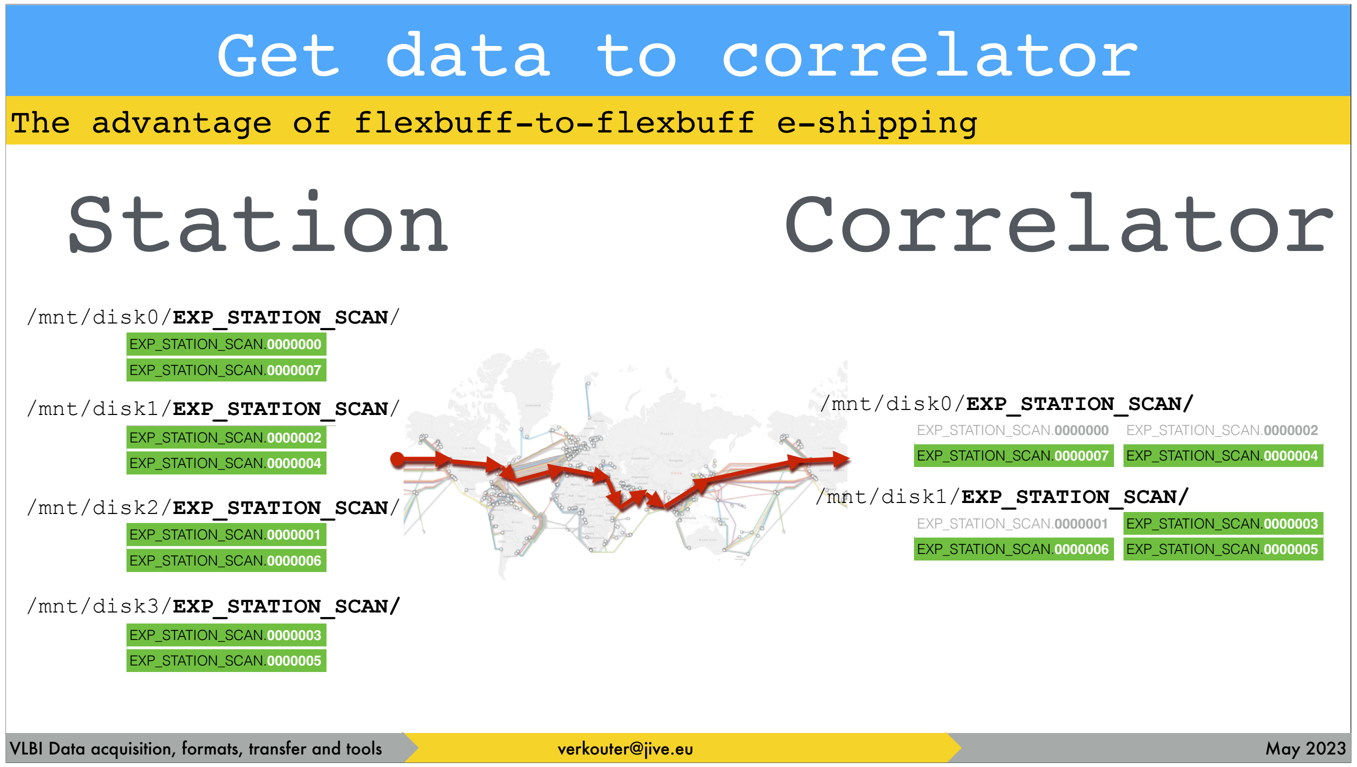
/mnt/disk0/**EXP_STATION_SCAN/**

- EXP_STATION_SCAN.0000000
- EXP_STATION_SCAN.0000002

/mnt/disk1/**EXP_STATION_SCAN/**

- EXP_STATION_SCAN.0000001

… and later on [click] restarted, the receiving jive5ab sees that some chunks are already present …



Get data to correlator

The advantage of flexbuff-to-flexbuff e-shipping

Station

/mnt/disk0/**EXP_STATION_SCAN**/

EXP_STATION_SCAN.**0000000**

EXP_STATION_SCAN.**0000007**

/mnt/disk1/**EXP_STATION_SCAN**/

EXP_STATION_SCAN.**0000002**

EXP_STATION_SCAN.**0000004**

/mnt/disk2/**EXP_STATION_SCAN**/

EXP_STATION_SCAN.**0000001**

EXP_STATION_SCAN.**0000006**

/mnt/disk3/**EXP_STATION_SCAN/**

EXP_STATION_SCAN.**0000003**

EXP_STATION_SCAN.**0000005**

Correlator

/mnt/disk0/**EXP_STATION_SCAN/**

EXP_STATION_SCAN.**0000000** EXP_STATION_SCAN.**0000002**

EXP_STATION_SCAN.**0000007** EXP_STATION_SCAN.**0000004**

/mnt/disk1/**EXP_STATION_SCAN/**

EXP_STATION_SCAN.**0000001** EXP_STATION_SCAN.**0000003**

EXP_STATION_SCAN.**0000006** EXP_STATION_SCAN.**0000005**

… and only the remaining chunks are transferred.

This can only be done because of the uniqueness of the file names of the individual chunks!

Dealing with scattered data is a

CENSORED

Now you have a lot of data scattered over many hard disks and that is not very nice to deal with!

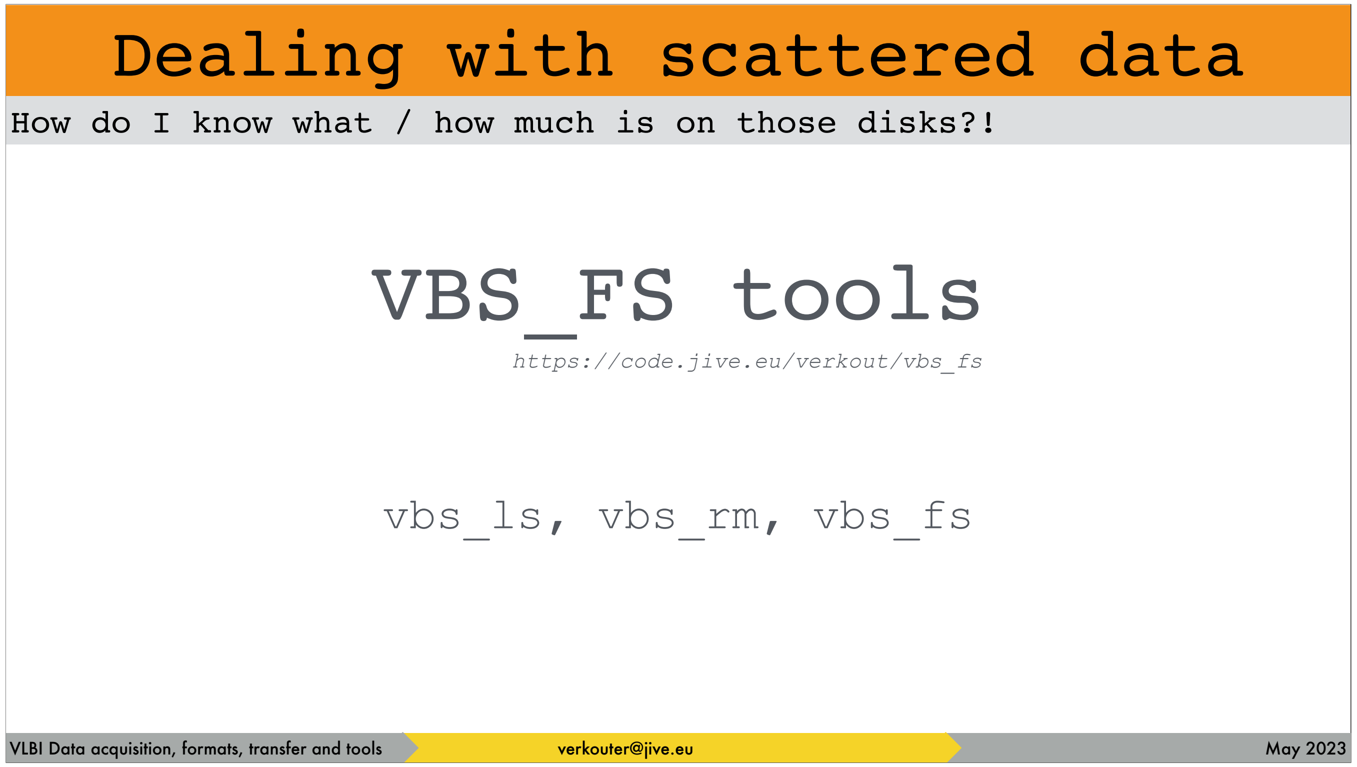
Dealing with scattered data

One of the first questions is:

Dealing with scattered data

How do I know what / how much is on those disks?!

One of the first questions is: what is actually on those disks and how large is it?



Dealing with scattered data

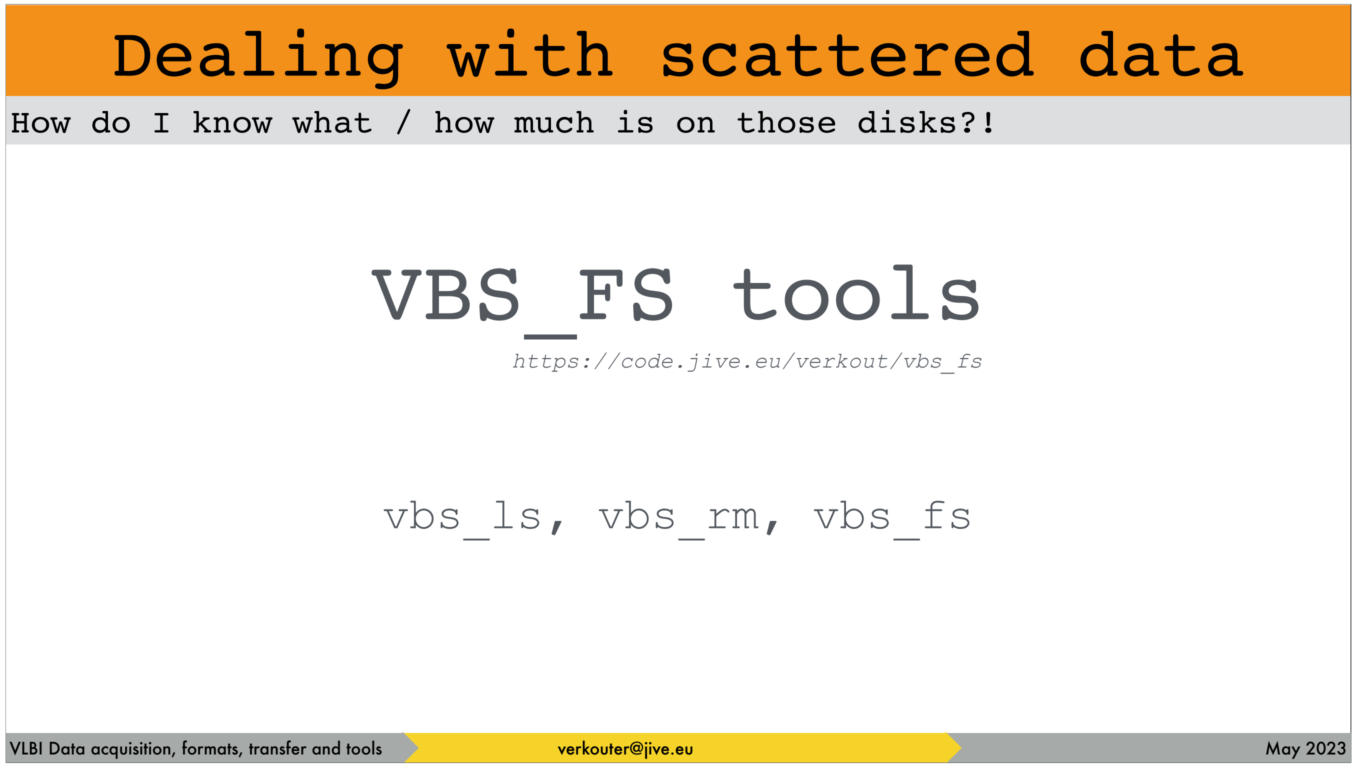
How do I know what / how much is on those disks?!

VBS_FS tools

https://code.jive.eu/verkout/vbs_fs

vbs_ls, vbs_rm, vbs_fs

The vbs_fs toolset is a collection of programs consisting of vbs_ls, vbs_rm and vbs_fs.



Dealing with scattered data

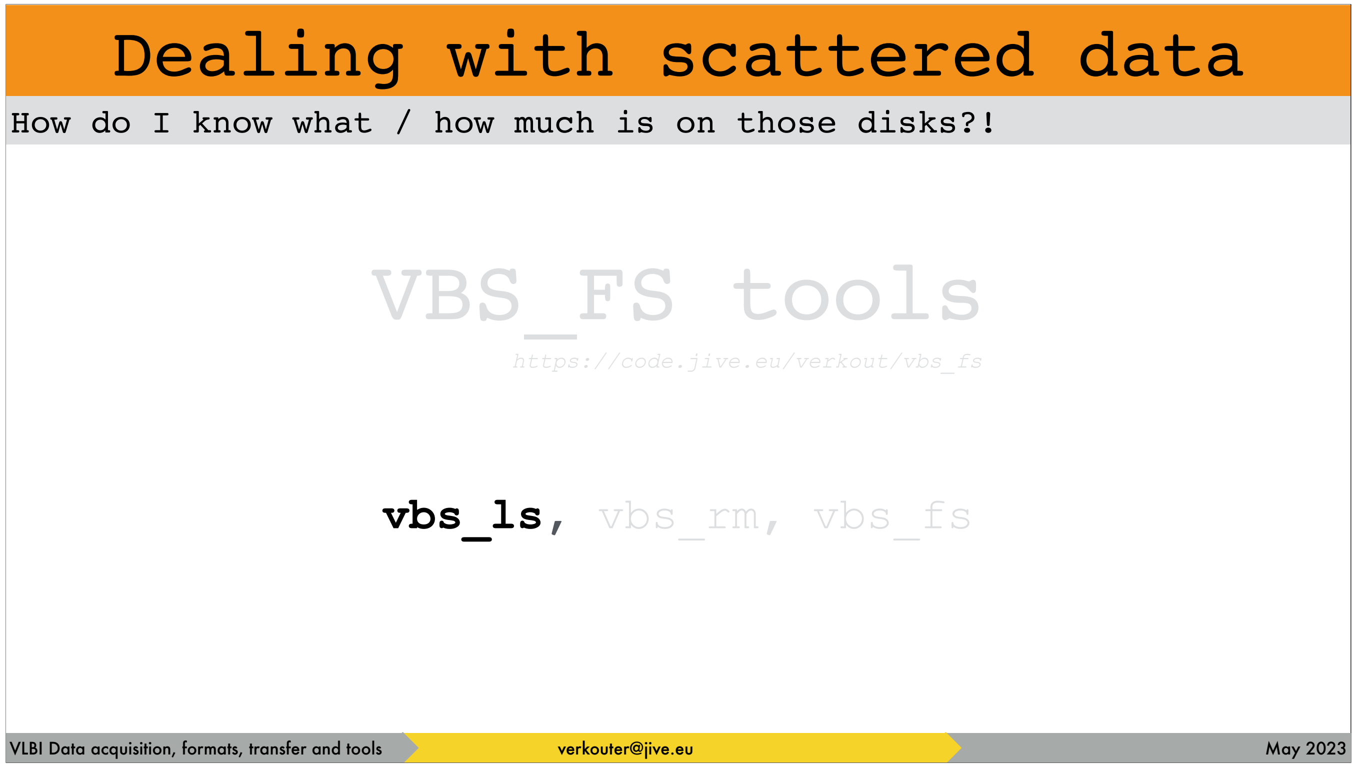
How do I know what / how much is on those disks?!

VBS_FS tools

https://code.jive.eu/verkout/vbs_fs

vbs_ls, vbs_rm, vbs_fs

The vbs_fs toolset is a collection of programs consisting of



Dealing with scattered data

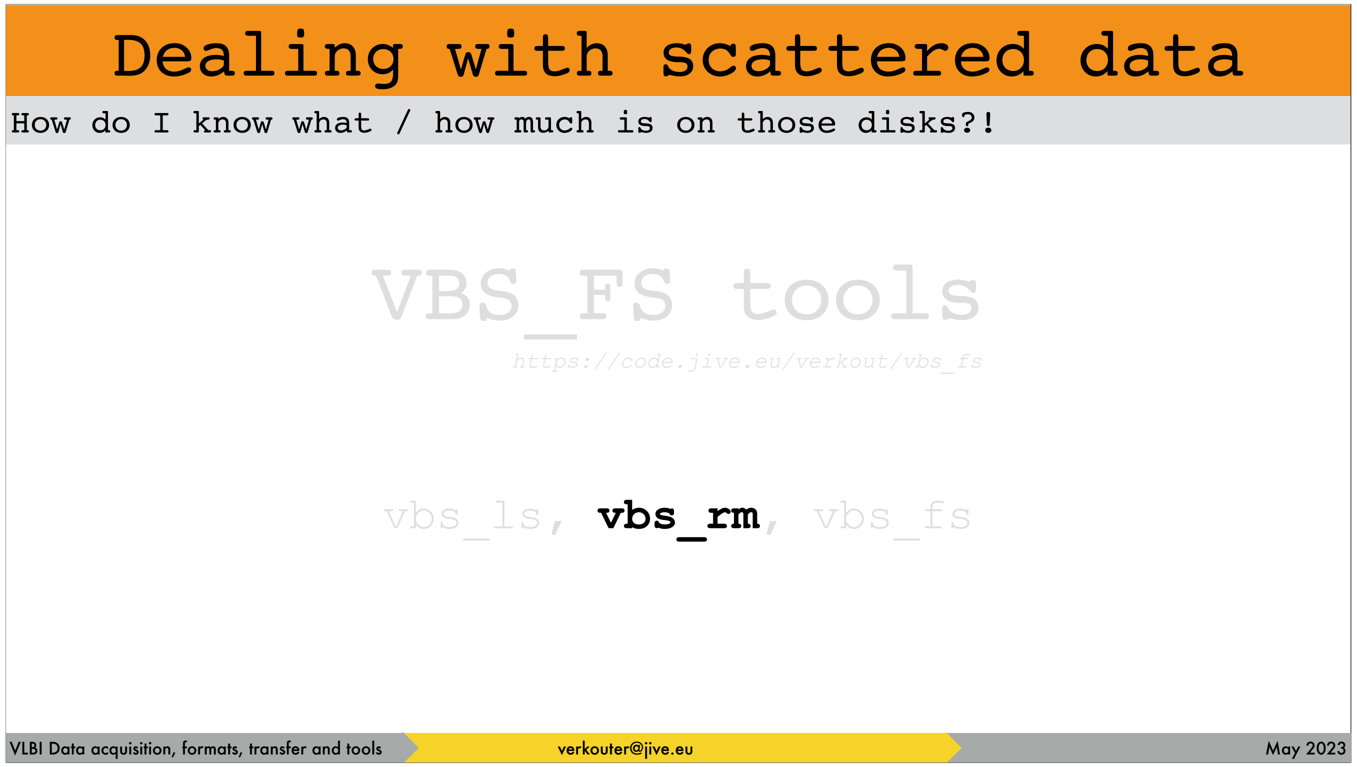
How do I know what / how much is on those disks?!

VBS_FS tools

https://code.jive.eu/verkout/vbs_fs

vbs_ls, vbs_rm, vbs_fs

vbs_ls



Dealing with scattered data

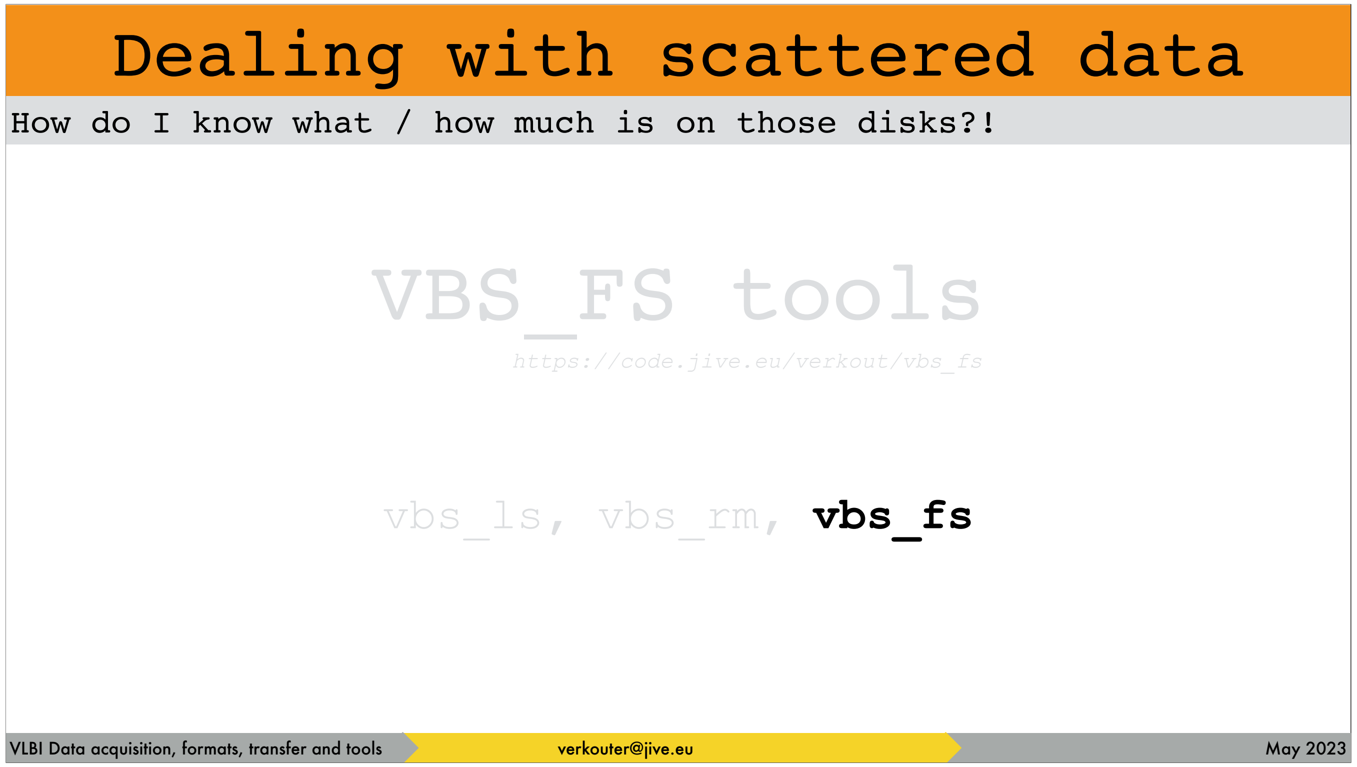
How do I know what / how much is on those disks?!

VBS_FS tools

https://code.jive.eu/verkout/vbs_fs

vbs_ls, **vbs_rm**, vbs_fs

vbs_rm



Dealing with scattered data

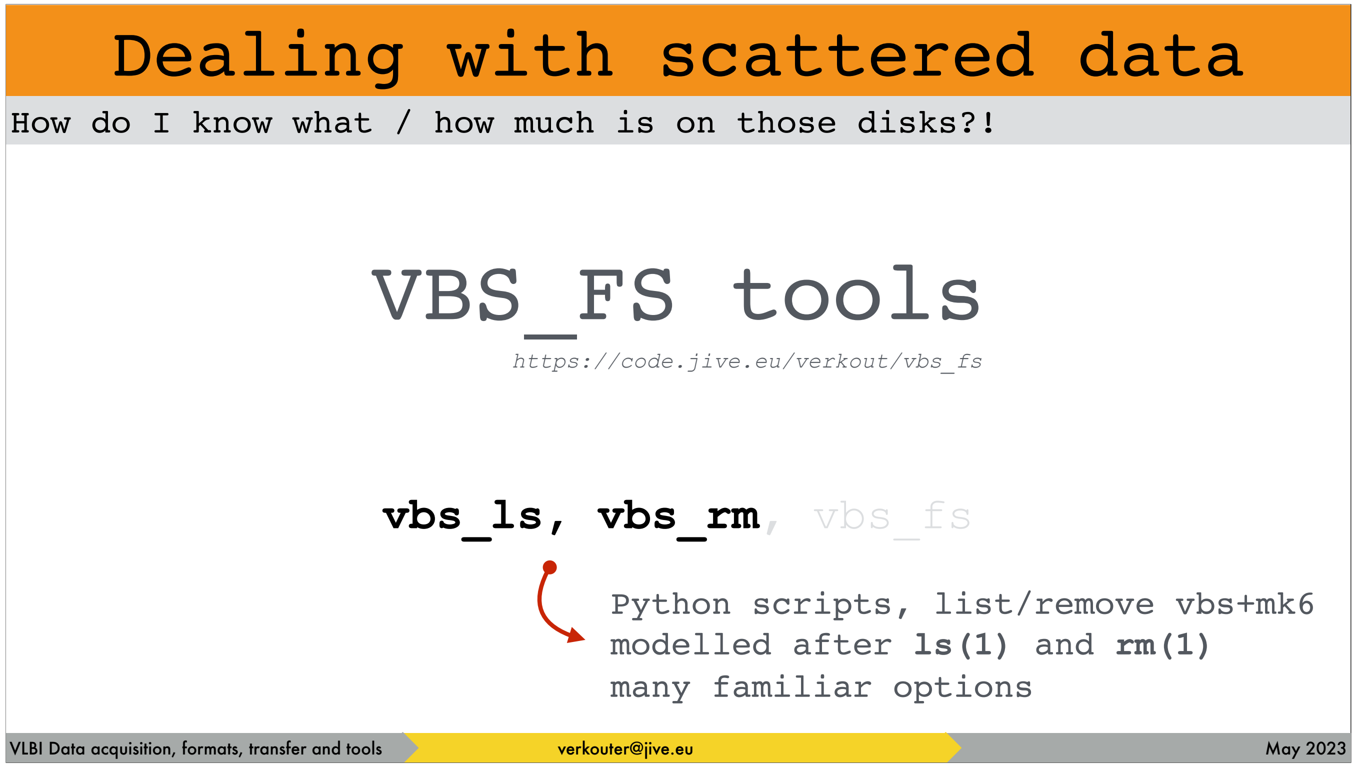
How do I know what / how much is on those disks?!

VBS_FS tools

https://code.jive.eu/verkout/vbs_fs

vbs_ls, vbs_rm, **vbs_fs**

and vbs_fs.



Dealing with scattered data

How do I know what / how much is on those disks?!

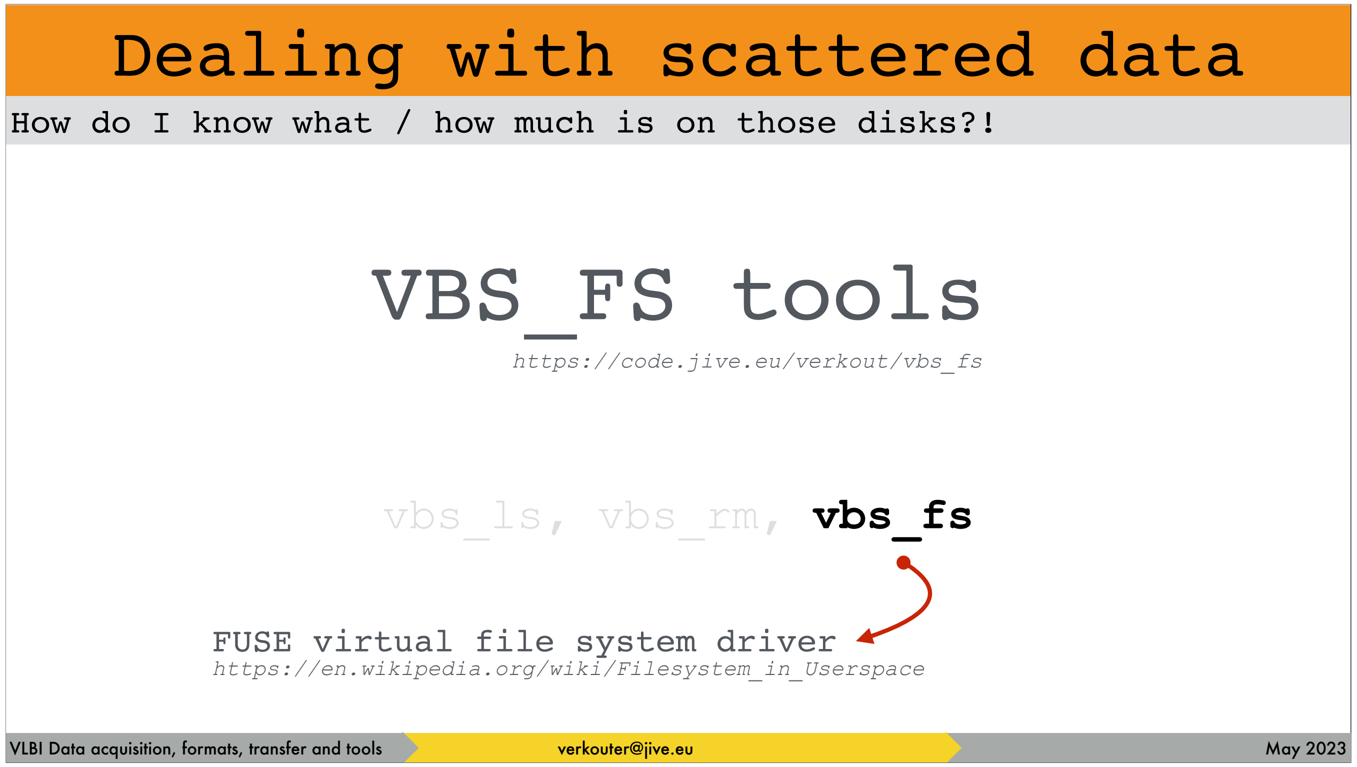
VBS_FS tools

https://code.jive.eu/verkout/vbs_fs

vbs_ls, vbs_rm, vbs_fs

Python scripts, list/remove vbs+mk6
modelled after **ls(1)** and **rm(1)**
many familiar options

As their name implies, vbs_ls and vbs_rm are modelled after their illustrious UNIX counterparts.



Dealing with scattered data

How do I know what / how much is on those disks?!

VBS_FS tools

https://code.jive.eu/verkout/vbs_fs

vbs_ls, vbs_rm, **vbs_fs**

FUSE virtual file system driver
https://en.wikipedia.org/wiki/Filesystem_in_Userspace

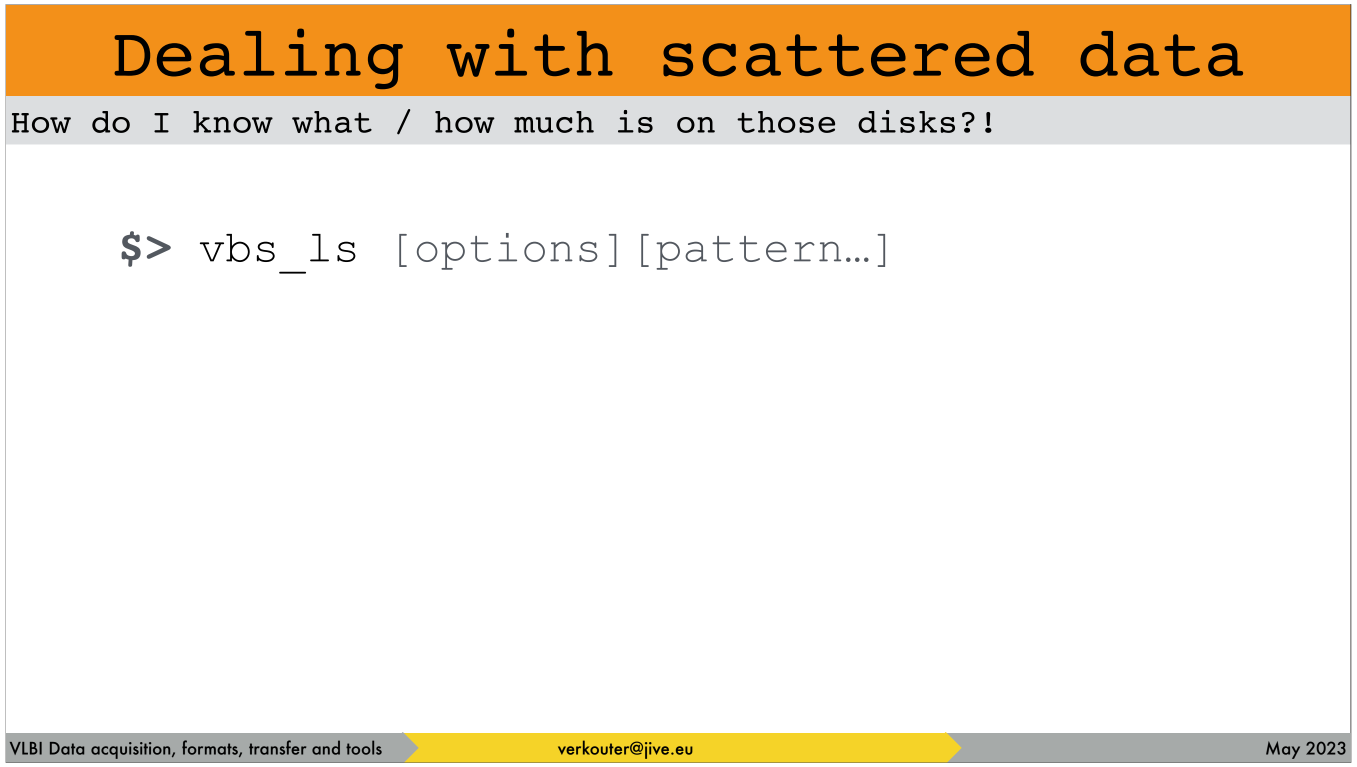
whilst vbs_fs is a FUSE file system in user space, which may be convenient

Dealing with scattered data

How do I know what / how much is on those disks?!

```
$>
```

to come back to the original question: what is ON those drives?

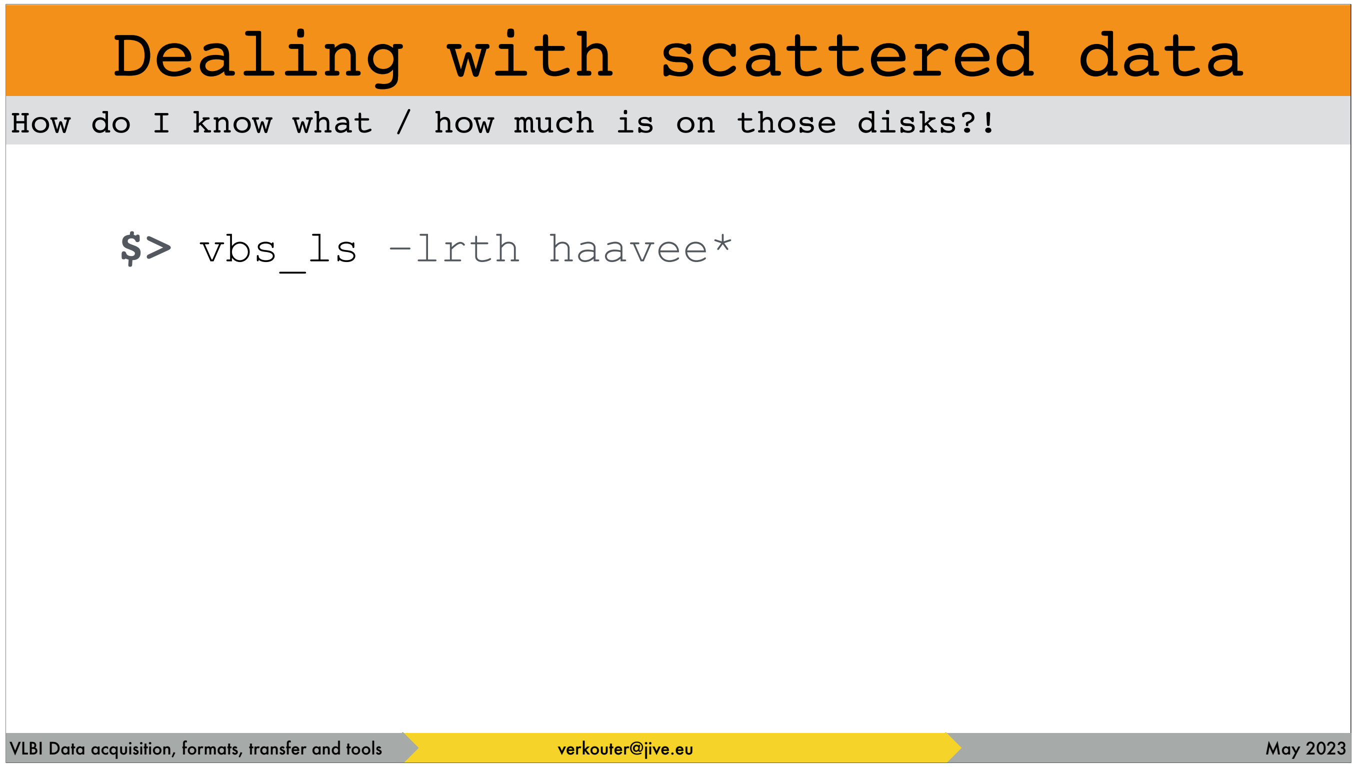


Dealing with scattered data

How do I know what / how much is on those disks?!

```
$> vbs_ls [options][pattern…]
```

vbs_ls can be used

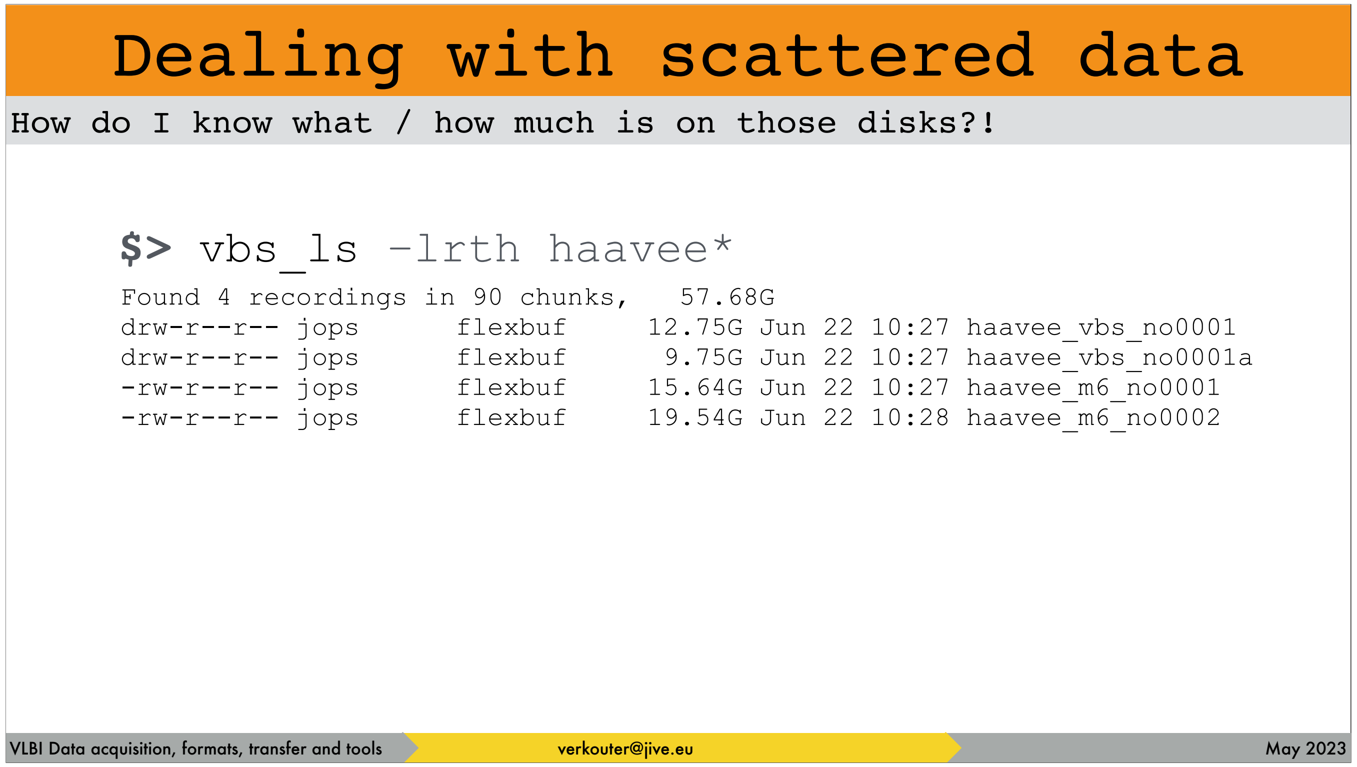


Dealing with scattered data

How do I know what / how much is on those disks?!

```
$> vbs_ls -lrth haavee*
```

for example like this

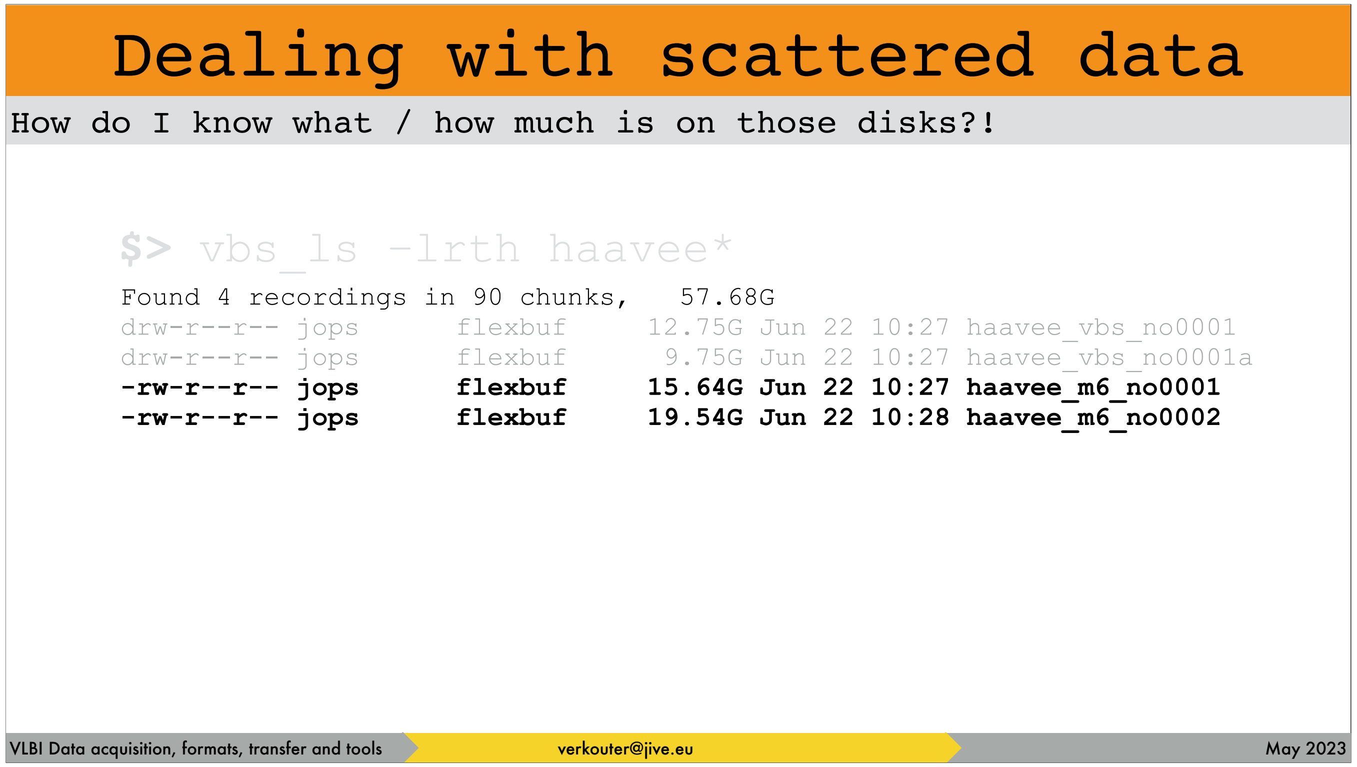


Dealing with scattered data

How do I know what / how much is on those disks?!

```
$> vbs_ls -lrth haavee*
Found 4 recordings in 90 chunks,   57.68G
drw-r--r-- jops      flexbuf     12.75G Jun 22 10:27 haavee_vbs_no0001
drw-r--r-- jops      flexbuf      9.75G Jun 22 10:27 haavee_vbs_no0001a
-rw-r--r-- jops      flexbuf     15.64G Jun 22 10:27 haavee_m6_no0001
-rw-r--r-- jops      flexbuf     19.54G Jun 22 10:28 haavee_m6_no0002
```

to give reasonably familiar output.

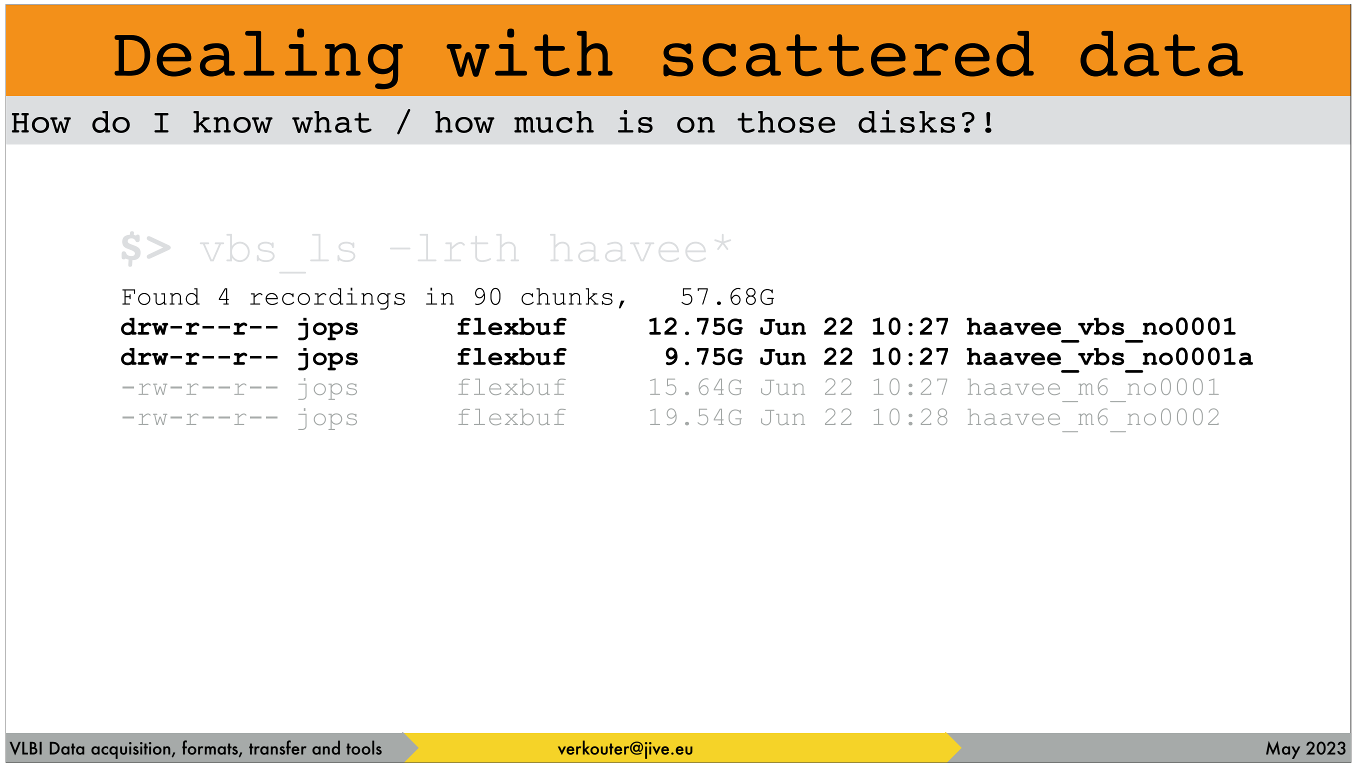


Dealing with scattered data

How do I know what / how much is on those disks?!

```
$> vbs_ls -lrth haavee*
Found 4 recordings in 90 chunks,   57.68G
drw-r--r-- jops      flexbuf     12.75G Jun 22 10:27 haavee_vbs_no0001
drw-r--r-- jops      flexbuf      9.75G Jun 22 10:27 haavee_vbs_no0001a
-rw-r--r-- jops      flexbuf     15.64G Jun 22 10:27 haavee_m6_no0001
-rw-r--r-- jops      flexbuf     19.54G Jun 22 10:28 haavee_m6_no0002
```

as you can see, vbs_ls deals with both mark6

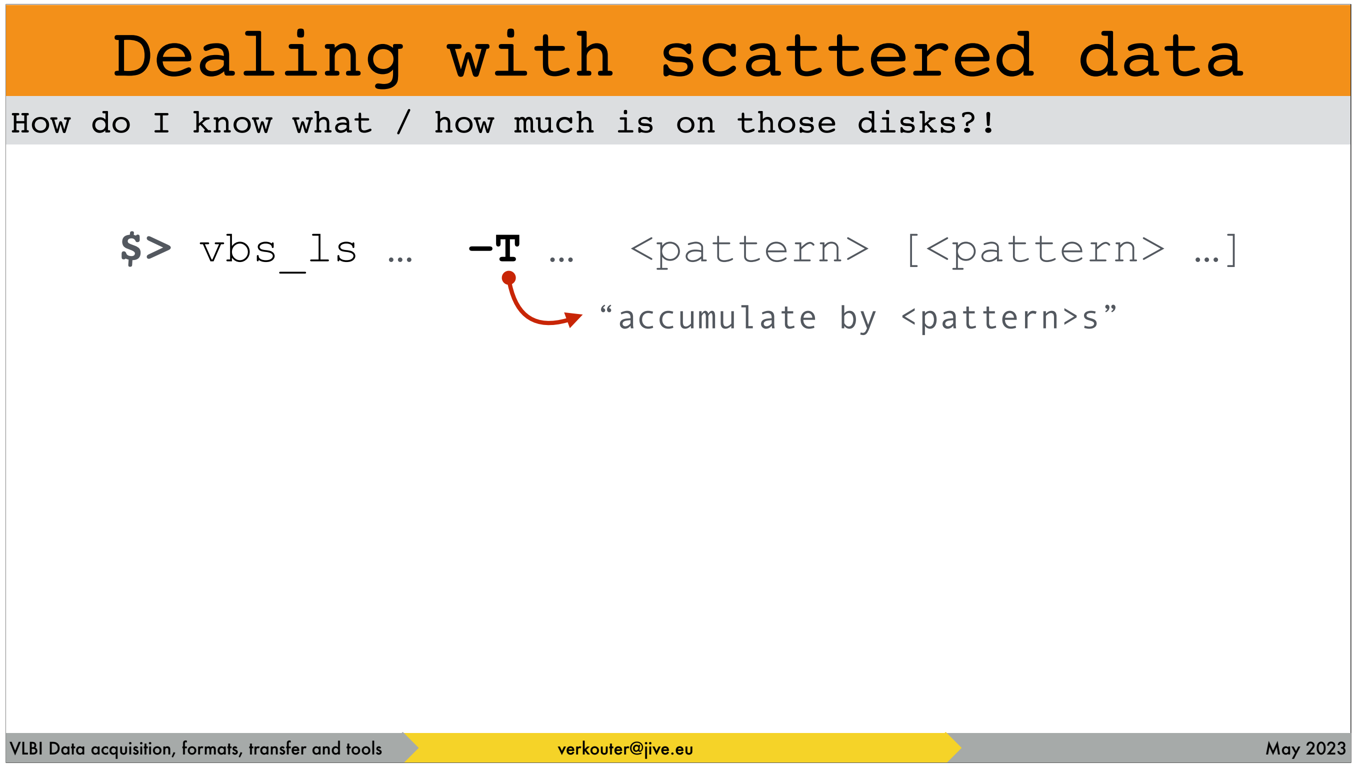


Dealing with scattered data

How do I know what / how much is on those disks?!

```
$> vbs_ls -lrth haavee*
Found 4 recordings in 90 chunks,   57.68G
drw-r--r-- jops      flexbuf     12.75G Jun 22 10:27 haavee_vbs_no0001
drw-r--r-- jops      flexbuf      9.75G Jun 22 10:27 haavee_vbs_no0001a
-rw-r--r-- jops      flexbuf     15.64G Jun 22 10:27 haavee_m6_no0001
-rw-r--r-- jops      flexbuf     19.54G Jun 22 10:28 haavee_m6_no0002
```

as well as flexbuff style recordings, even if present on the same media



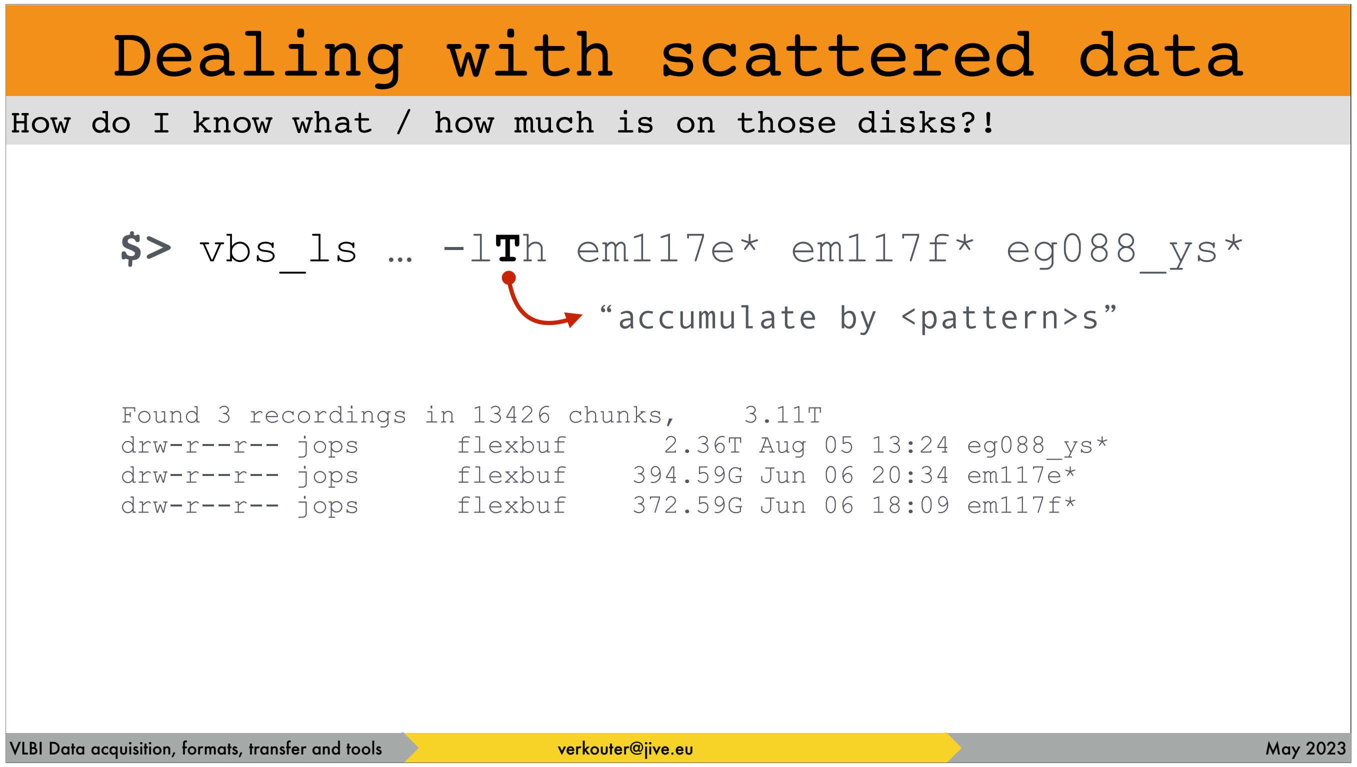
Dealing with scattered data

How do I know what / how much is on those disks?!

```
$> vbs_ls …  -T …  <pattern> [<pattern> …]
```

"accumulate by <pattern>s"

There is an option that is NOT in ls, that allows accumulation by pattern



Dealing with scattered data

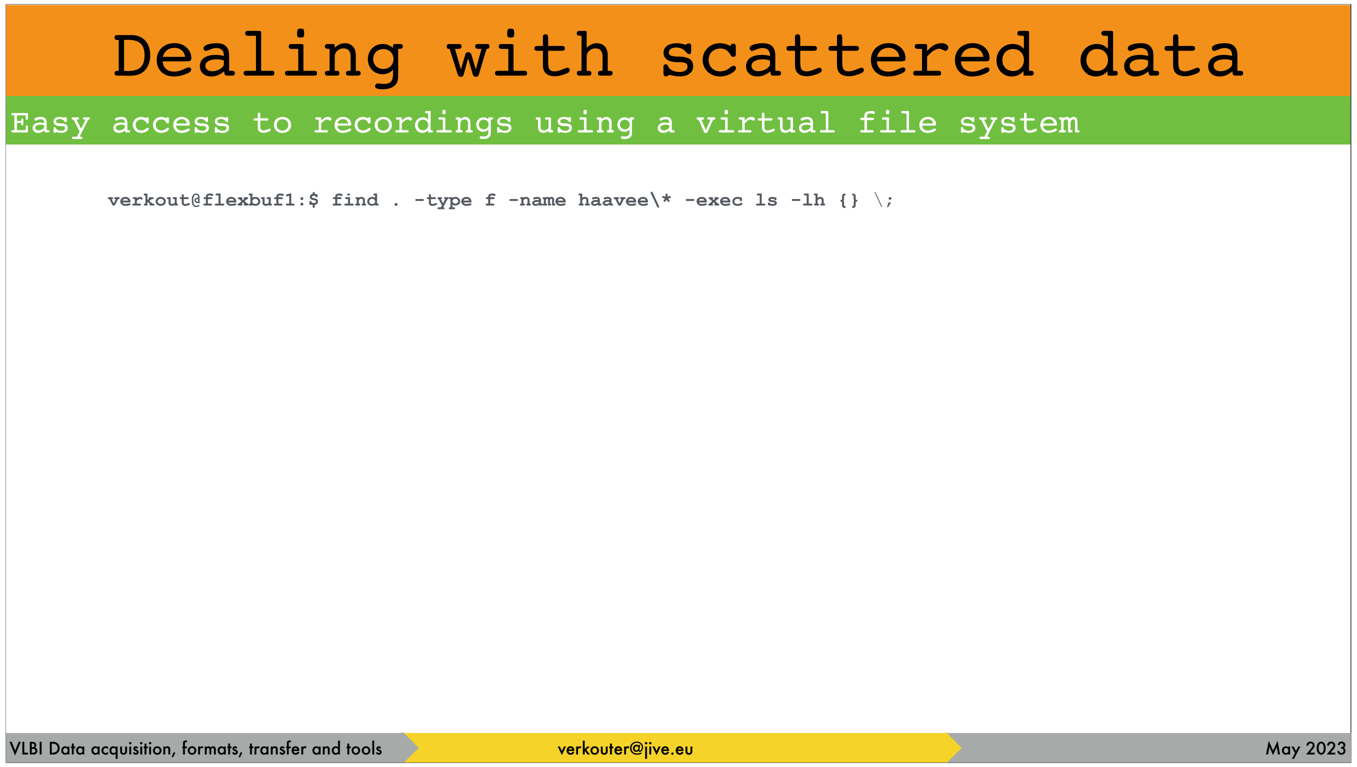
How do I know what / how much is on those disks?!

```
$> vbs_ls … -lTh em117e* em117f* eg088_ys*
```

“accumulate by <pattern>s”

```
Found 3 recordings in 13426 chunks,    3.11T
drw-r--r-- jops      flexbuf      2.36T Aug 05 13:24 eg088_ys*
drw-r--r-- jops      flexbuf    394.59G Jun 06 20:34 em117e*
drw-r--r-- jops      flexbuf    372.59G Jun 06 18:09 em117f*
```

and for example can be used to assess how much data per experiment is present.

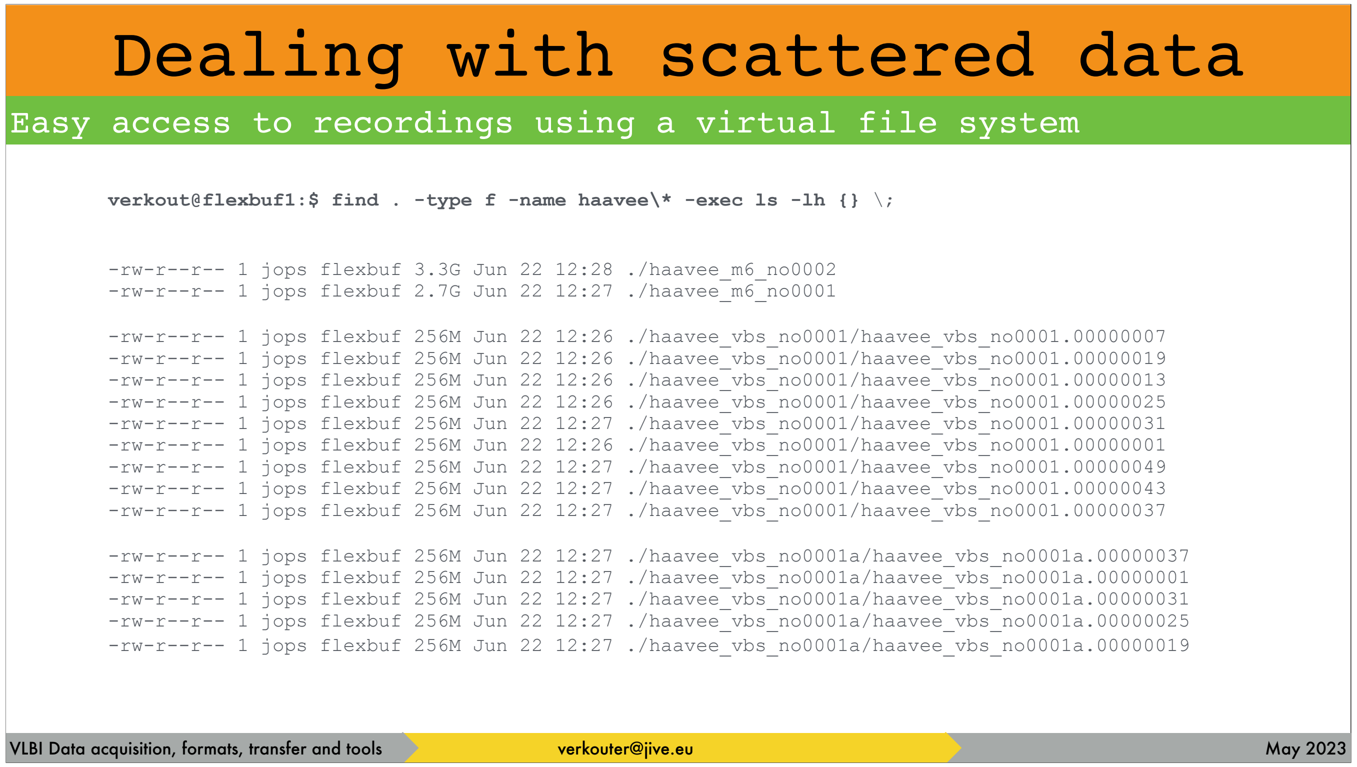


Dealing with scattered data

Easy access to recordings using a virtual file system

```
verkout@flexbuf1:$ find . -type f -name haavee\* -exec ls -lh {} \;
```

Another big issue with scattered data is …

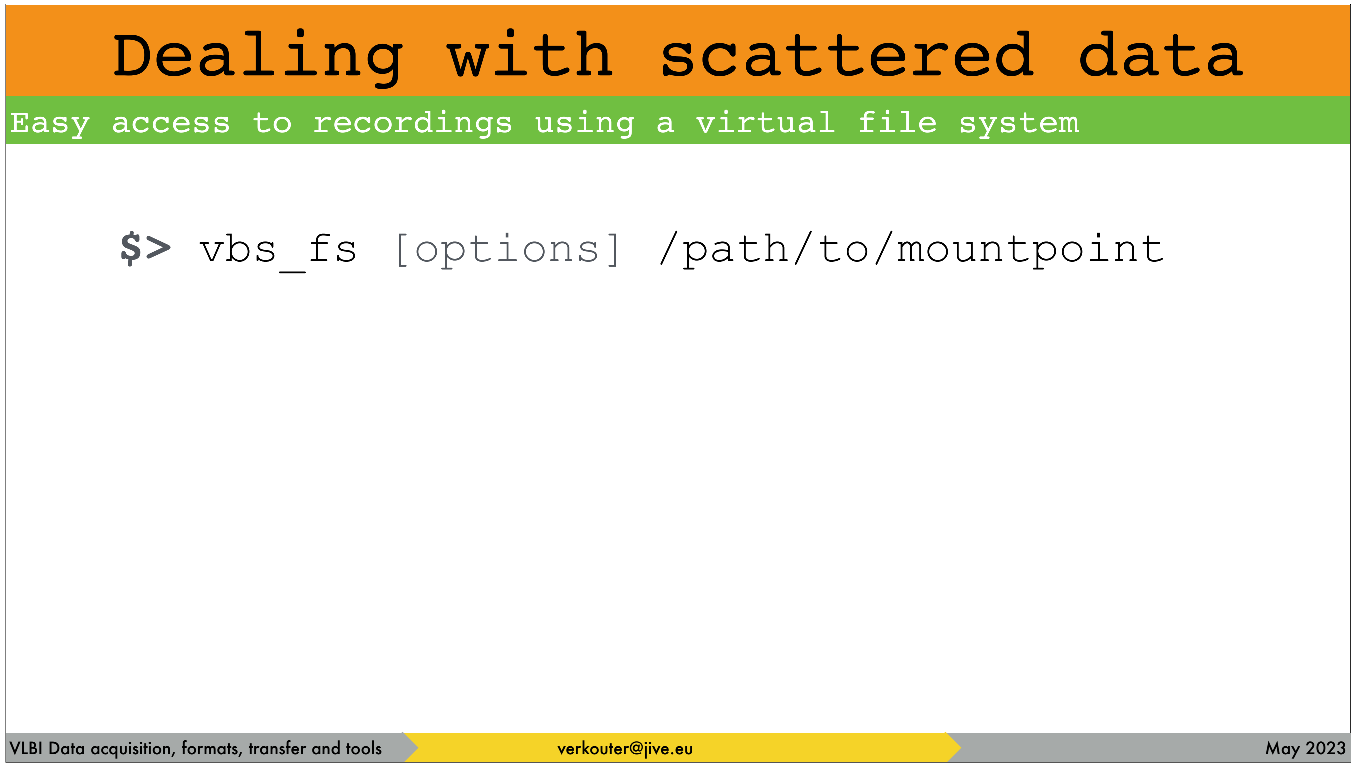


Dealing with scattered data

Easy access to recordings using a virtual file system

```
verkout@flexbuf1:$ find . -type f -name haavee\* -exec ls -lh {} \;


-rw-r--r-- 1 jops flexbuf 3.3G Jun 22 12:28 ./haavee_m6_no0002
-rw-r--r-- 1 jops flexbuf 2.7G Jun 22 12:27 ./haavee_m6_no0001

-rw-r--r-- 1 jops flexbuf 256M Jun 22 12:26 ./haavee_vbs_no0001/haavee_vbs_no0001.00000007
-rw-r--r-- 1 jops flexbuf 256M Jun 22 12:26 ./haavee_vbs_no0001/haavee_vbs_no0001.00000019
-rw-r--r-- 1 jops flexbuf 256M Jun 22 12:26 ./haavee_vbs_no0001/haavee_vbs_no0001.00000013
-rw-r--r-- 1 jops flexbuf 256M Jun 22 12:26 ./haavee_vbs_no0001/haavee_vbs_no0001.00000025
-rw-r--r-- 1 jops flexbuf 256M Jun 22 12:27 ./haavee_vbs_no0001/haavee_vbs_no0001.00000031
-rw-r--r-- 1 jops flexbuf 256M Jun 22 12:26 ./haavee_vbs_no0001/haavee_vbs_no0001.00000001
-rw-r--r-- 1 jops flexbuf 256M Jun 22 12:27 ./haavee_vbs_no0001/haavee_vbs_no0001.00000049
-rw-r--r-- 1 jops flexbuf 256M Jun 22 12:27 ./haavee_vbs_no0001/haavee_vbs_no0001.00000043
-rw-r--r-- 1 jops flexbuf 256M Jun 22 12:27 ./haavee_vbs_no0001/haavee_vbs_no0001.00000037

-rw-r--r-- 1 jops flexbuf 256M Jun 22 12:27 ./haavee_vbs_no0001a/haavee_vbs_no0001a.00000037
-rw-r--r-- 1 jops flexbuf 256M Jun 22 12:27 ./haavee_vbs_no0001a/haavee_vbs_no0001a.00000001
-rw-r--r-- 1 jops flexbuf 256M Jun 22 12:27 ./haavee_vbs_no0001a/haavee_vbs_no0001a.00000031
-rw-r--r-- 1 jops flexbuf 256M Jun 22 12:27 ./haavee_vbs_no0001a/haavee_vbs_no0001a.00000025
-rw-r--r-- 1 jops flexbuf 256M Jun 22 12:27 ./haavee_vbs_no0001a/haavee_vbs_no0001a.00000019
```

… it is spread like shrapnel over many disks!

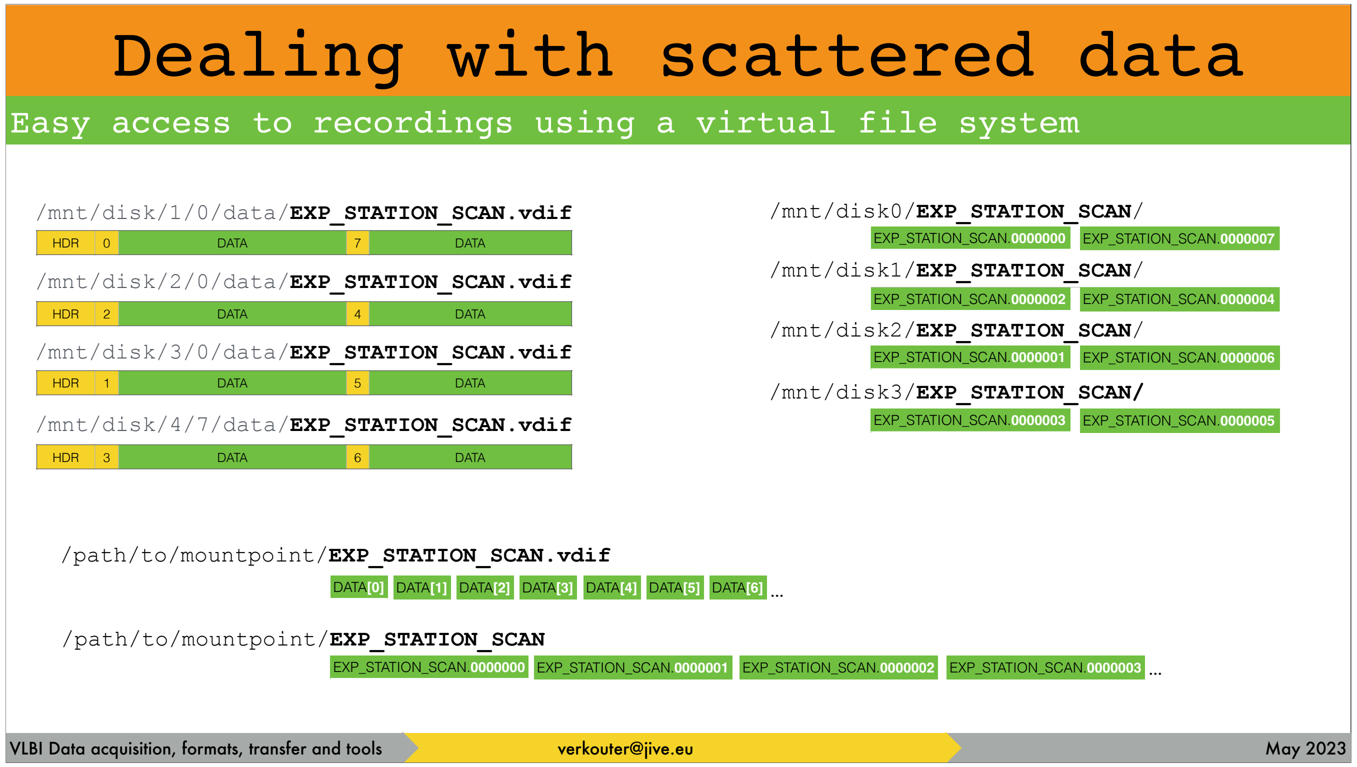


Dealing with scattered data

Easy access to recordings using a virtual file system

```
$> vbs_fs [options] /path/to/mountpoint
```

Enter the vbs_fs virtual file system. After startup, this

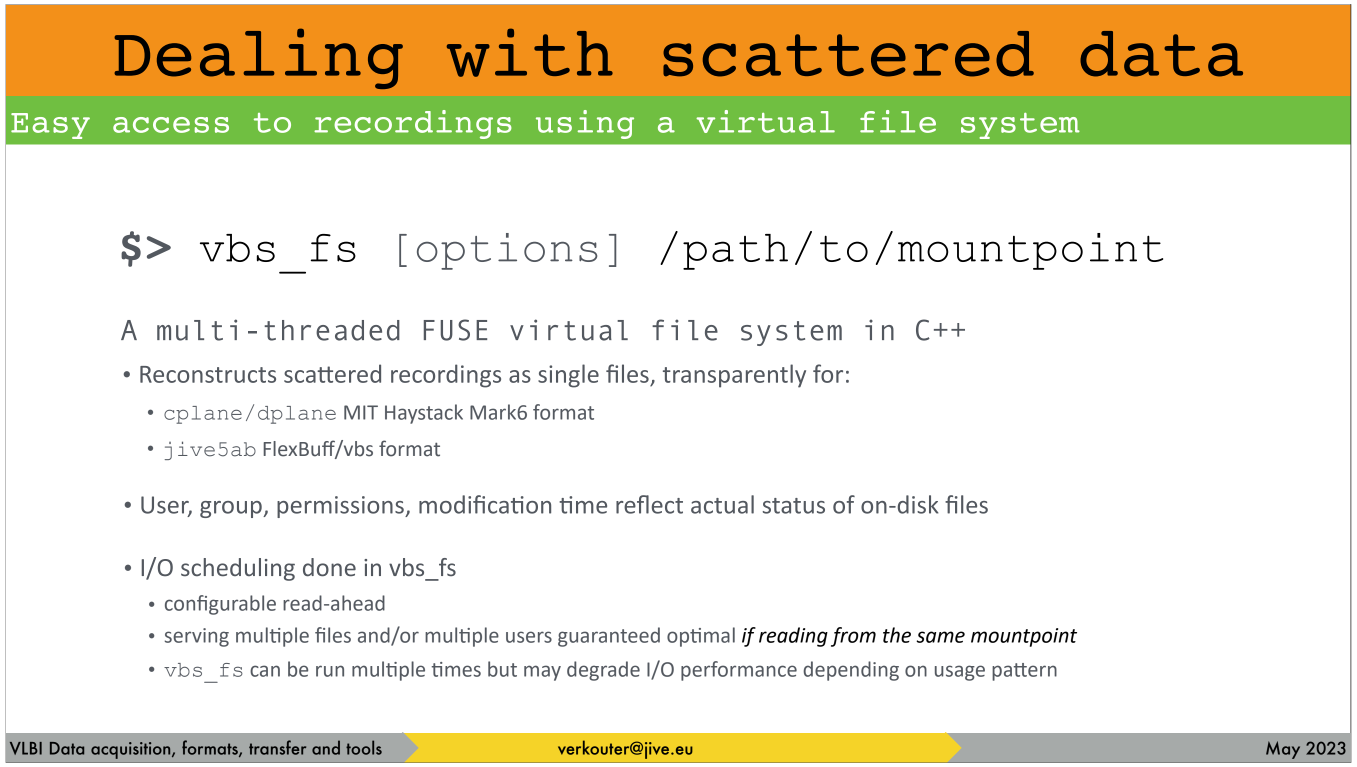


Dealing with scattered data

Easy access to recordings using a virtual file system

/mnt/disk/1/0/data/**EXP_STATION_SCAN.vdif**
HDR
0
DATA
7
DATA
/mnt/disk/2/0/data/**EXP_STATION_SCAN.vdif**
HDR
2
DATA
4
DATA
/mnt/disk/3/0/data/**EXP_STATION_SCAN.vdif**
HDR
1
DATA
5
DATA
/mnt/disk/4/7/data/**EXP_STATION_SCAN.vdif**
HDR
3
DATA
6
DATA
/mnt/disk0/**EXP_STATION_SCAN**/
EXP_STATION_SCAN.0000000
EXP_STATION_SCAN.0000007
/mnt/disk1/**EXP_STATION_SCAN**/
EXP_STATION_SCAN.0000002
EXP_STATION_SCAN.0000004
/mnt/disk2/**EXP_STATION_SCAN**/
EXP_STATION_SCAN.0000001
EXP_STATION_SCAN.0000006
/mnt/disk3/**EXP_STATION_SCAN/**
EXP_STATION_SCAN.0000003
EXP_STATION_SCAN.0000005
/path/to/mountpoint/**EXP_STATION_SCAN.vdif**
DATA[0] DATA[1] DATA[2] DATA[3] DATA[4] DATA[5] DATA[6] ...
/path/to/mountpoint/**EXP_STATION_SCAN**
EXP_STATION_SCAN.0000000 EXP_STATION_SCAN.0000001 EXP_STATION_SCAN.0000002 EXP_STATION_SCAN.0000003 ...

… virtual file system takes the scattered data from recordings [click] and reconstructs them as single files under the mountpoint.



Dealing with scattered data

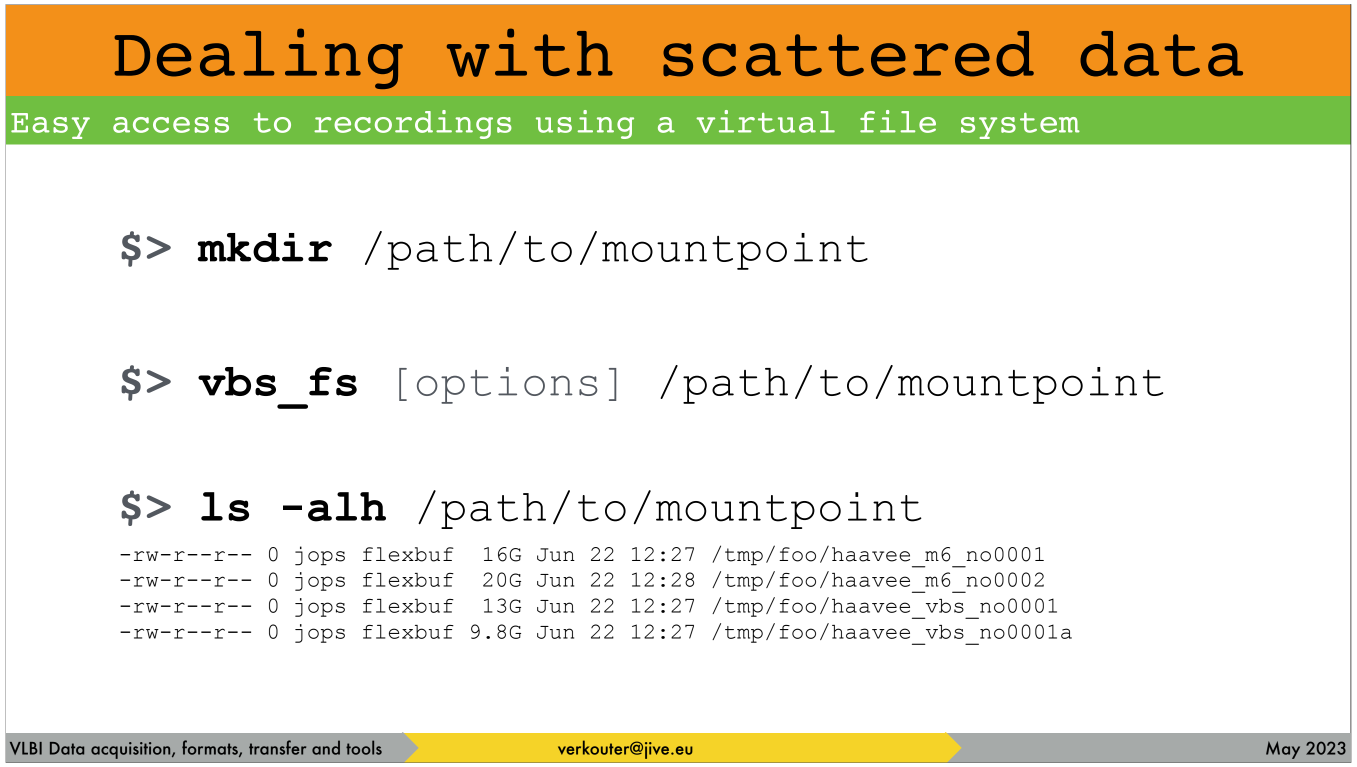
Easy access to recordings using a virtual file system

```
$> vbs_fs [options] /path/to/mountpoint
```

A multi-threaded FUSE virtual file system in C++

- Reconstructs scattered recordings as single files, transparently for:
  - `cplane/dplane` MIT Haystack Mark6 format
  - `jive5ab` FlexBuff/vbs format
- User, group, permissions, modification time reflect actual status of on-disk files
- I/O scheduling done in vbs_fs
  - configurable read-ahead
  - serving multiple files and/or multiple users guaranteed optimal *if reading from the same mountpoint*
  - `vbs_fs` can be run multiple times but may degrade I/O performance depending on usage pattern

The vbs_fs program is written in multithreaded c++ for raw speed and does the i/o scheduling for you, even if multiple users are accessing the recordings through the same mountpoint.

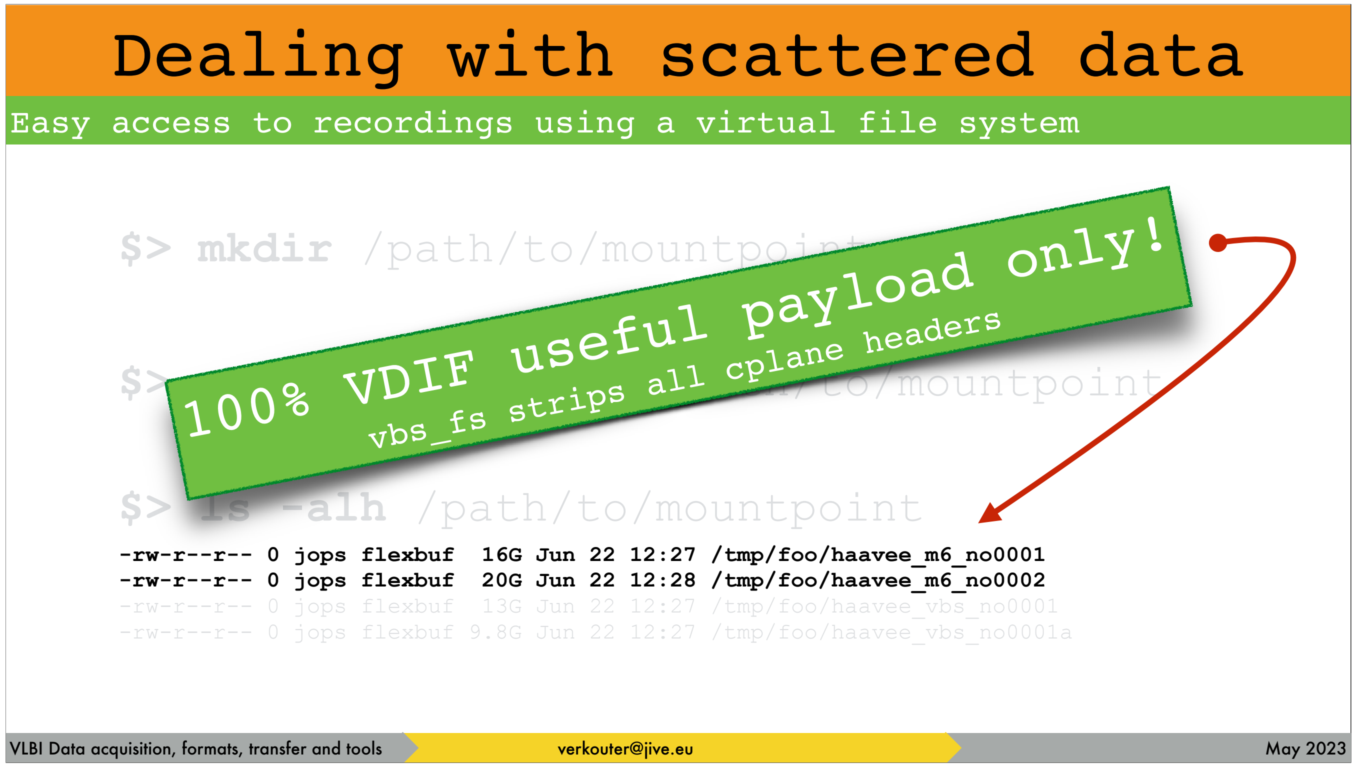


Dealing with scattered data

Easy access to recordings using a virtual file system

```
$> mkdir /path/to/mountpoint

$> vbs_fs [options] /path/to/mountpoint

$> ls -alh /path/to/mountpoint
-rw-r--r-- 0 jops flexbuf  16G Jun 22 12:27 /tmp/foo/haavee_m6_no0001
-rw-r--r-- 0 jops flexbuf  20G Jun 22 12:28 /tmp/foo/haavee_m6_no0002
-rw-r--r-- 0 jops flexbuf  13G Jun 22 12:27 /tmp/foo/haavee_vbs_no0001
-rw-r--r-- 0 jops flexbuf 9.8G Jun 22 12:27 /tmp/foo/haavee_vbs_no0001a
```

In order to use it, you create a directory for the mountpoint first and then [click] it is a matter of mounting the virtual file system on that mountpoint, [click] after which recordings show up in that mountpoint as ordinary files



Dealing with scattered data

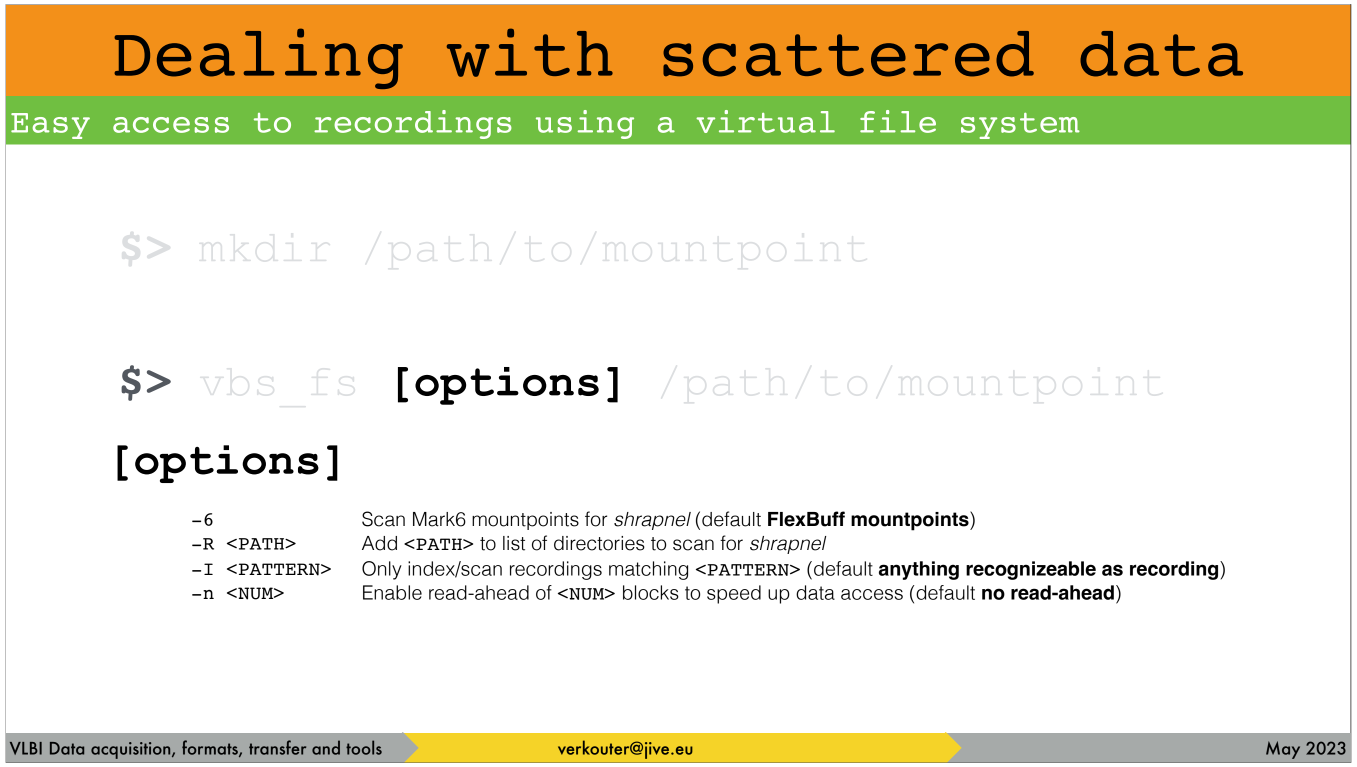
Easy access to recordings using a virtual file system

```
$> mkdir /path/to/mountpoint

$>                      /to/mountpoint

$> ls -alh /path/to/mountpoint
-rw-r--r-- 0 jops flexbuf  16G Jun 22 12:27 /tmp/foo/haavee_m6_no0001
-rw-r--r-- 0 jops flexbuf  20G Jun 22 12:28 /tmp/foo/haavee_m6_no0002
-rw-r--r-- 0 jops flexbuf  13G Jun 22 12:27 /tmp/foo/haavee_vbs_no0001
-rw-r--r-- 0 jops flexbuf 9.8G Jun 22 12:27 /tmp/foo/haavee_vbs_no0001a
```

100% VDIF useful payload only!

vbs_fs strips all cplane headers

Interesting to know is that for Mark6 files vbs_fs strips all the application headers so this is 100% VDIF payload!!!



Dealing with scattered data

Easy access to recordings using a virtual file system

```
$> mkdir /path/to/mountpoint

$> vbs_fs [options] /path/to/mountpoint
```

[options]

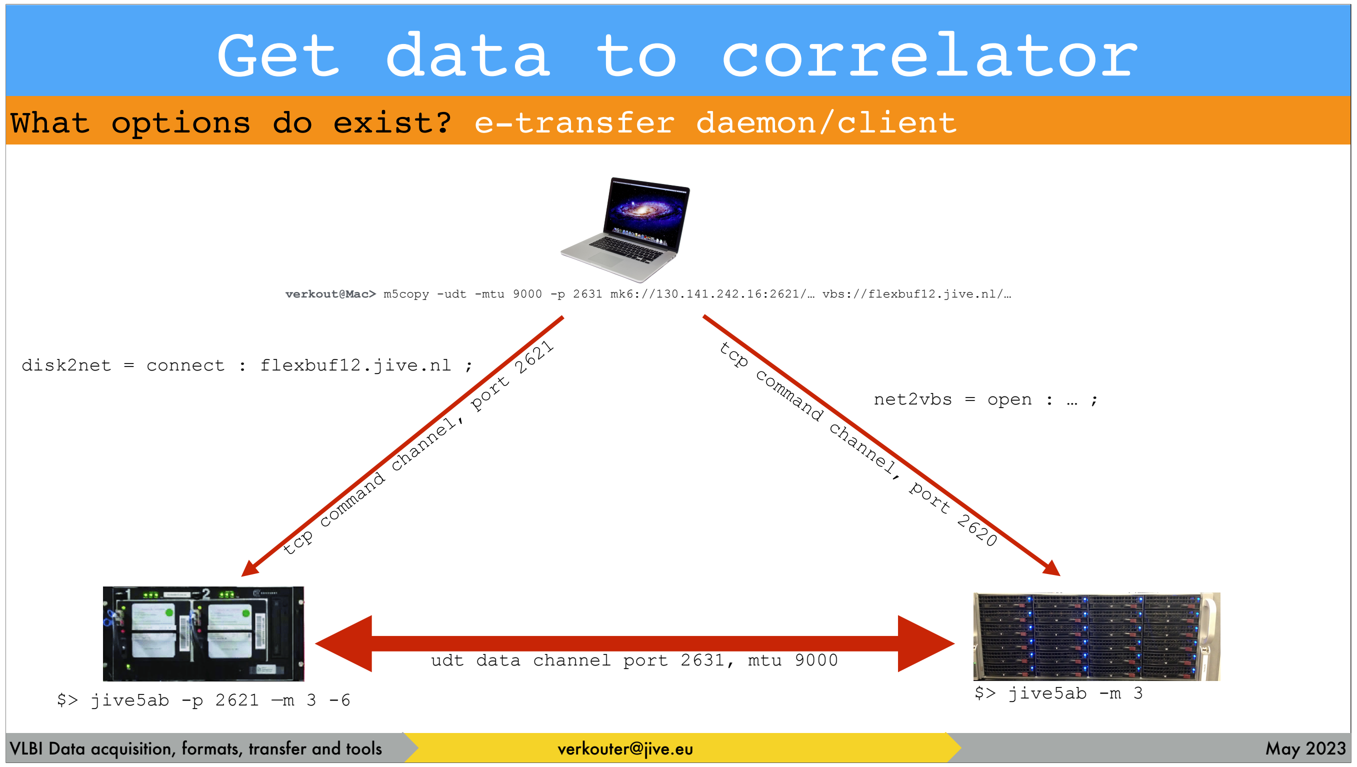
`-6`	Scan Mark6 mountpoints for *shrapnel* (default **FlexBuff mountpoints**)
`-R <PATH>`	Add `<PATH>` to list of directories to scan for *shrapnel*
`-I <PATTERN>`	Only index/scan recordings matching `<PATTERN>` (default **anything recognizeable as recording**)
`-n <NUM>`	Enable read-ahead of `<NUM>` blocks to speed up data access (default **no read-ahead**)

Useful options for vbs_fs to know about are those influencing the mountpoints to scan for recordings, a filter to index only recordings matching a certain pattern, or to enable read-ahead to make data access a lot faster.

Get data to correlator

What options do exist? e-transfer daemon/client

Now that you know about the fuse file system to present recordings as single files, it might be good to introduce another e-shipping option that is being developed.



Get data to correlator

What options do exist? e-transfer daemon/client

```
verkout@Mac> m5copy -udt -mtu 9000 -p 2631 mk6://130.141.242.16:2621/… vbs://flexbuf12.jive.nl/…
```

```
disk2net = connect : flexbuf12.jive.nl ;
```

tcp command channel, port 2621

```
net2vbs = open : … ;
```

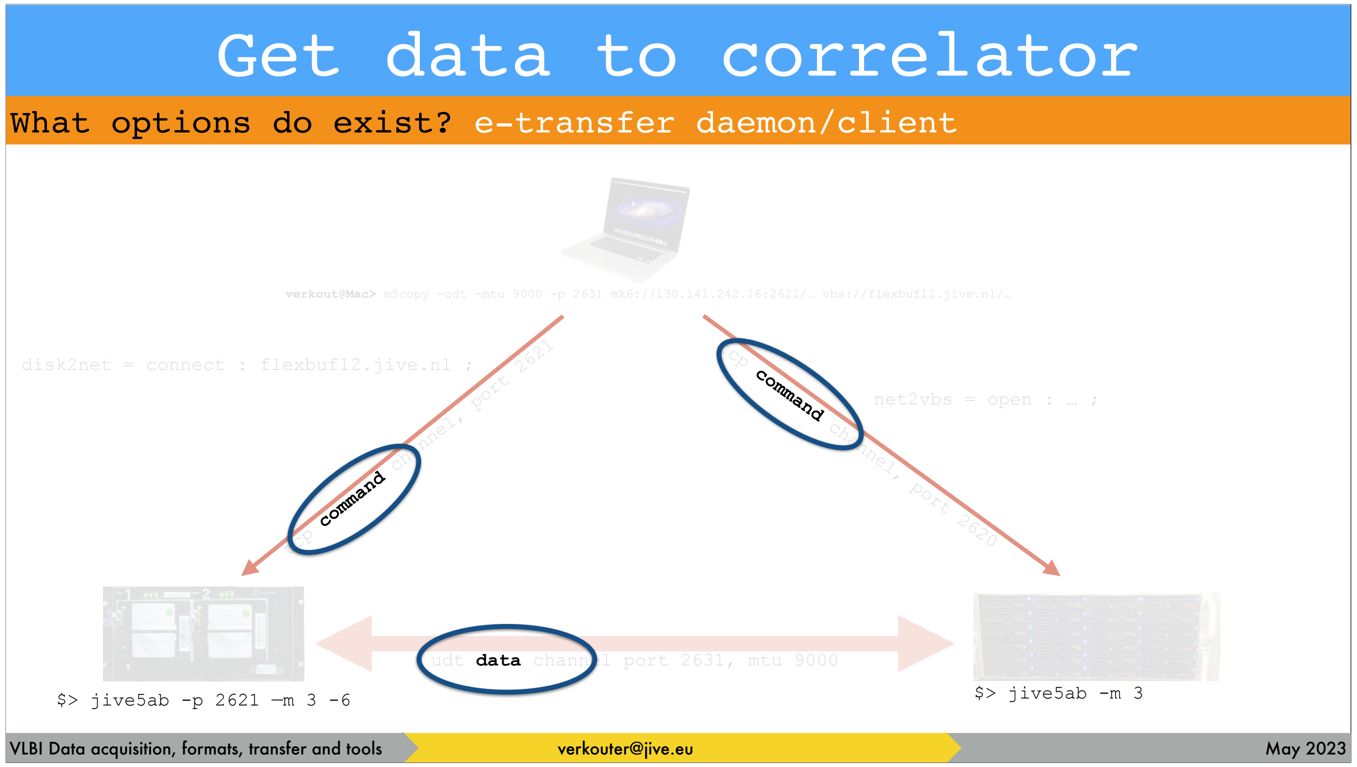
tcp command channel, port 2620

udt data channel port 2631, mtu 9000

```
$> jive5ab -p 2621 —m 3 -6
```

```
$> jive5ab -m 3
```

Remember the jive5ab + m5copy situation?



Get data to correlator

What options do exist? e-transfer daemon/client

```
verkout@Mac> m5copy -udt -mtu 9000 -p 2631 mk6://130.141.242.16:2621/… vbs://flexbuf12.jive.nl/…
```

```
disk2net = connect : flexbuf12.jive.nl ;
```

```
net2vbs = open : … ;
```

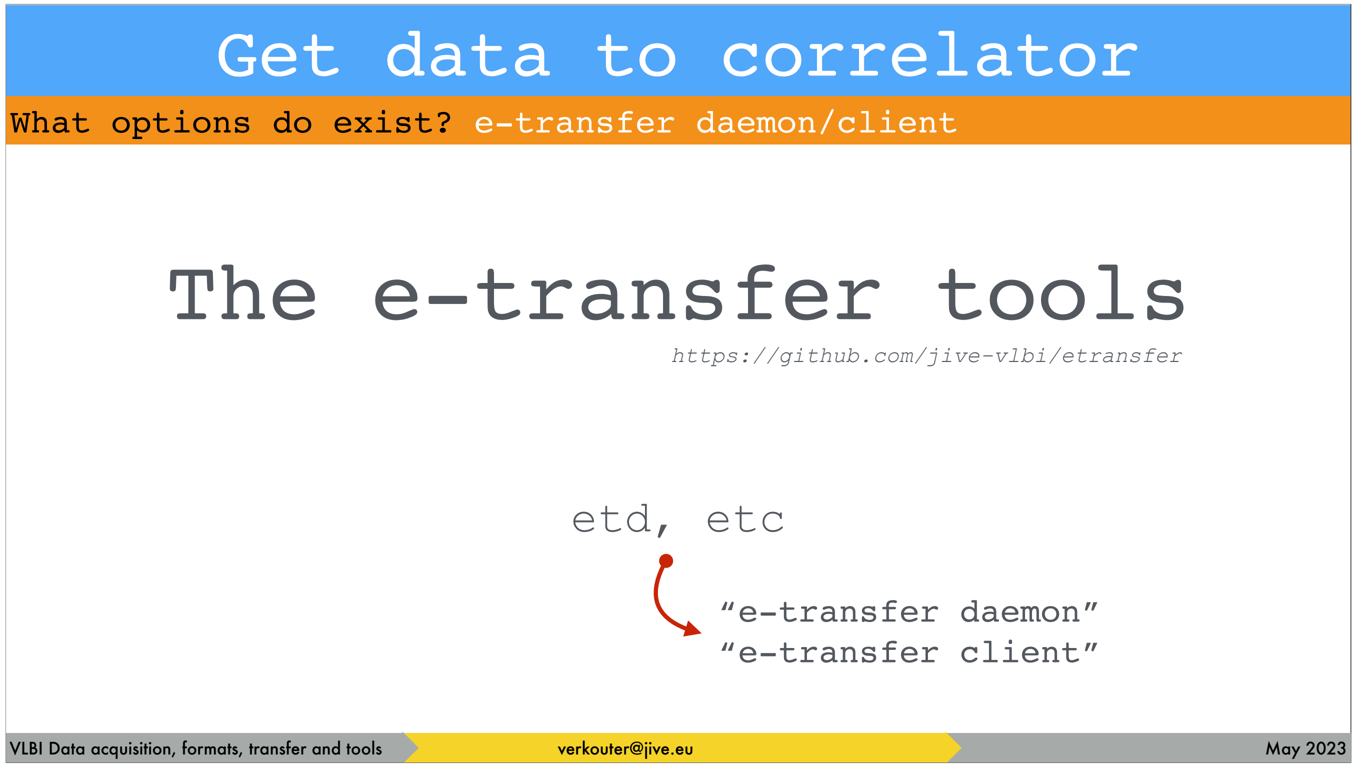
```
$> jive5ab -p 2621 —m 3 -6
```

```
$> jive5ab -m 3
```

tcp command channel, port 2621
tcp command channel, port 2620
udt data channel port 2631, mtu 9000

One of the biggest issues is that there is [click] no control channel between the sender and receiver of the data!

That doesn't sound like a big thing but it is.



Get data to correlator

What options do exist? e-transfer daemon/client

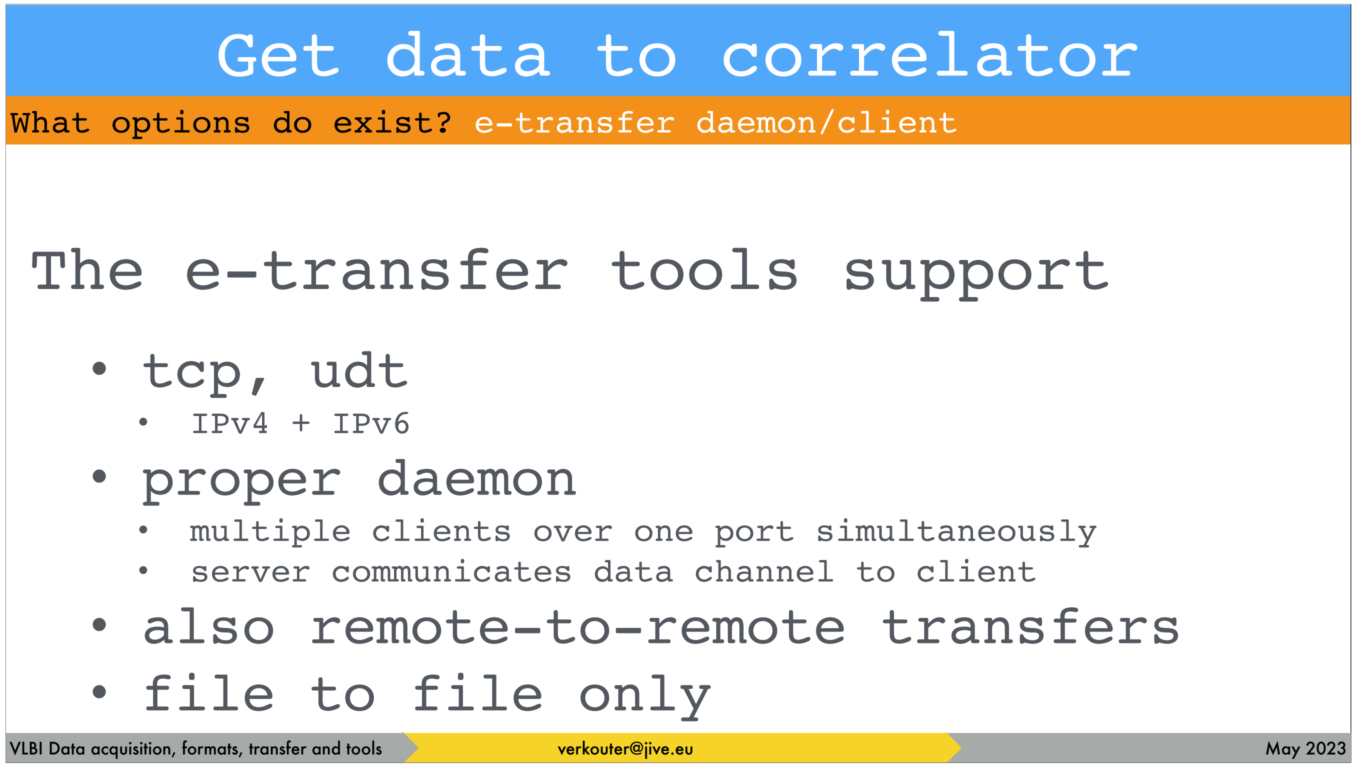
The e-transfer tools

https://github.com/jive-vlbi/etransfer

etd, etc

"e-transfer daemon"
"e-transfer client"

Based on the experiences with UDT, jive5ab and m5copy a proper daemon/client pair of programs were developed, [click] the etransfer daemon and client



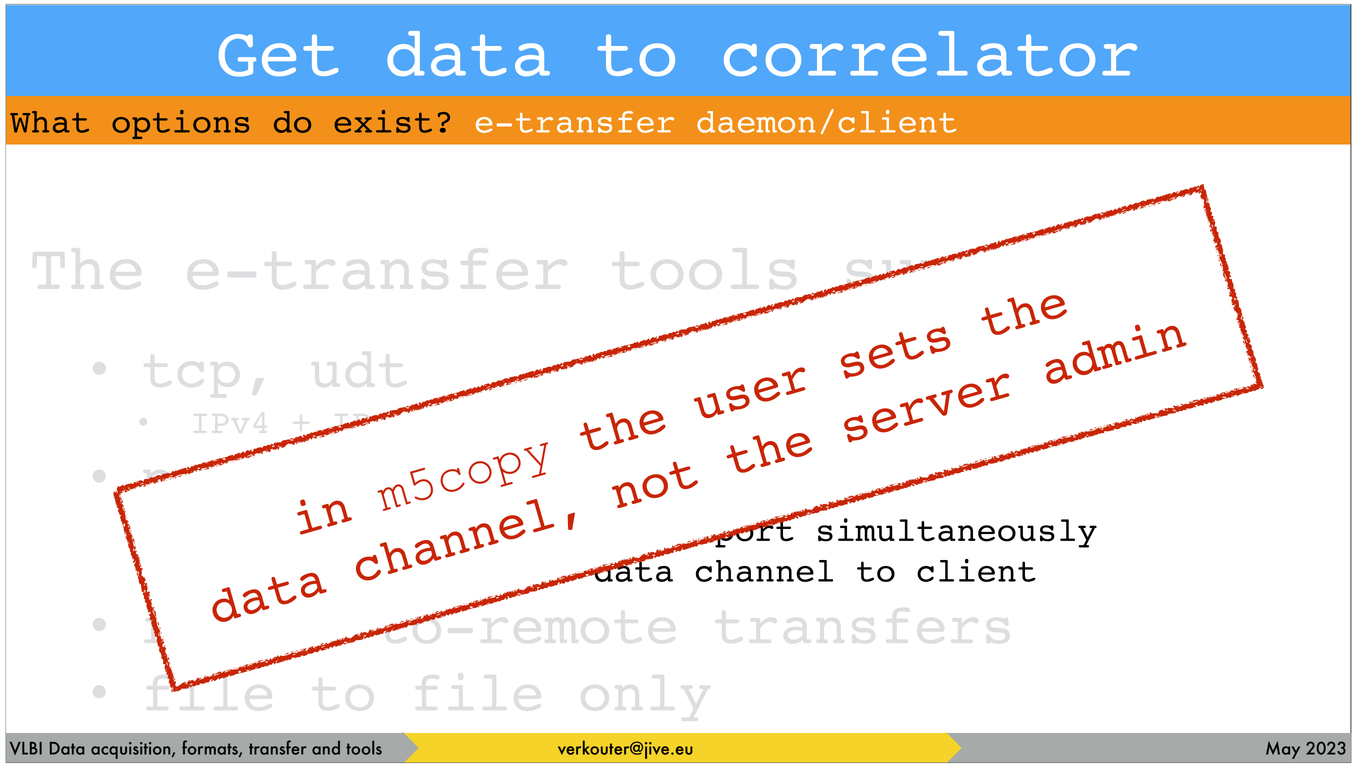
Get data to correlator

What options do exist? e-transfer daemon/client

The e-transfer tools support

- tcp, udt
  - IPv4 + IPv6
- proper daemon
  - multiple clients over one port simultaneously
  - server communicates data channel to client
- also remote-to-remote transfers
- file to file only

The important properties are these



Get data to correlator

What options do exist? e-transfer daemon/client

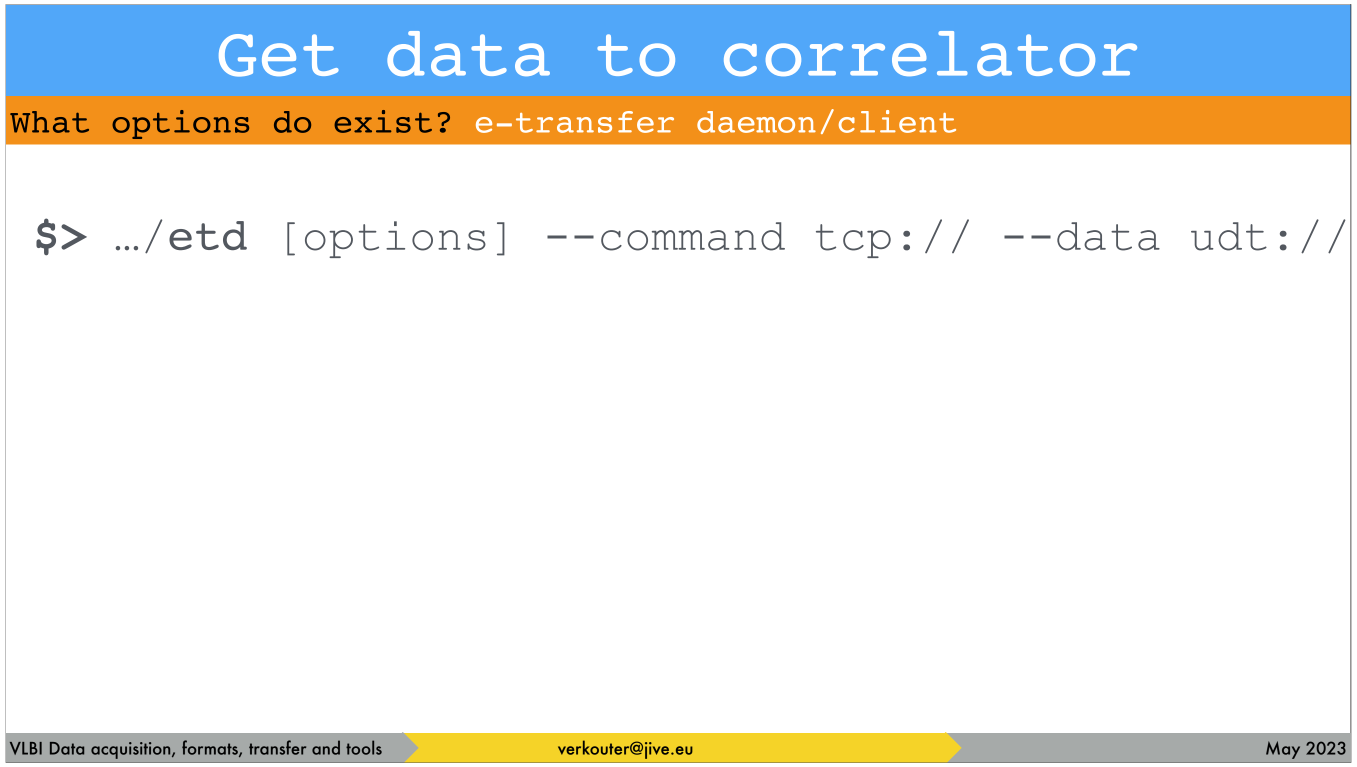
The e-transfer tools su

- tcp, udt
  - IPv4 + IP
- r
  - port simultaneously
  - ata channel to client
- to-remote transfers
- file to file only

in m5copy the user sets the data channel, not the server admin

and for server administrators these properties are the most important!

Because [click] in m5copy the CLIENT sets the data channel but that is basically just wrong!

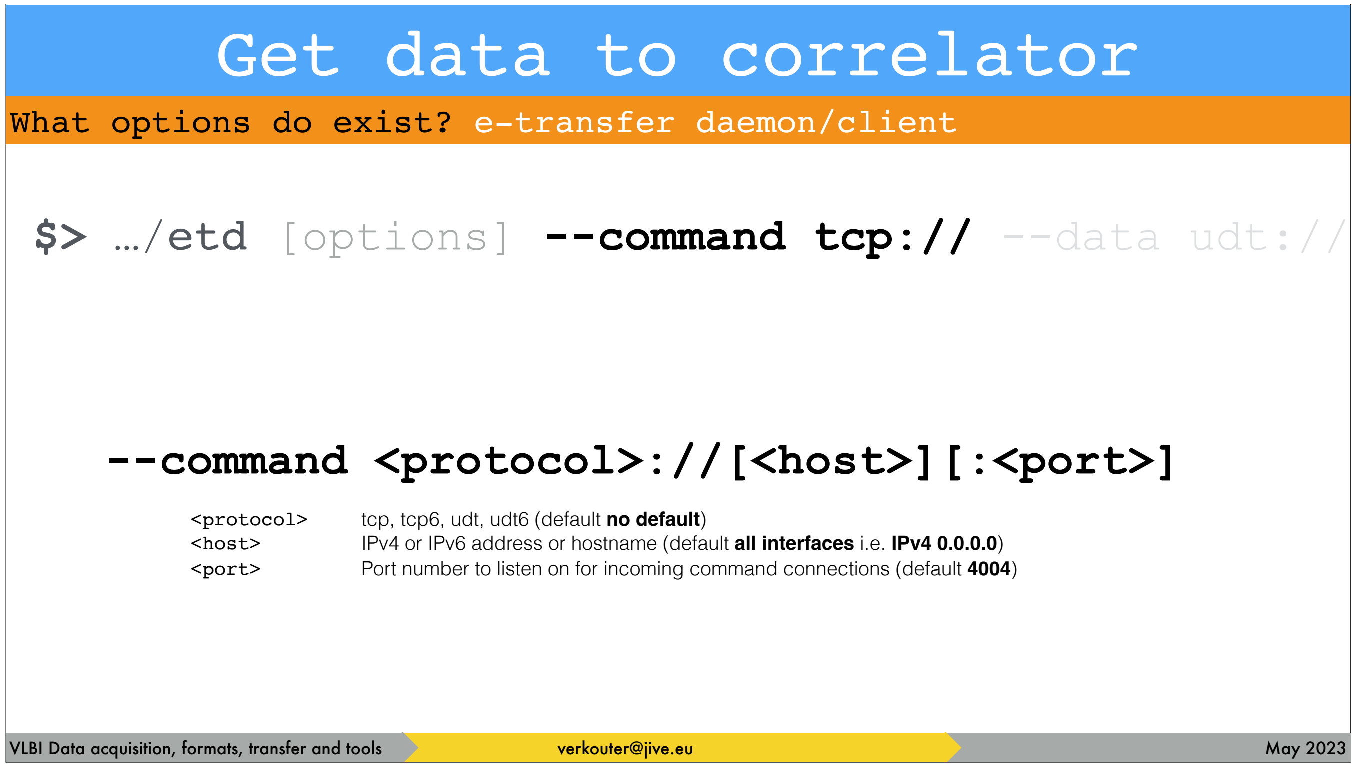


Get data to correlator

What options do exist? e-transfer daemon/client

```
$> …/etd [options] --command tcp:// --data udt://
```

In order to use the system, run the daemon in the data center. The daemon requires you to specify



Get data to correlator

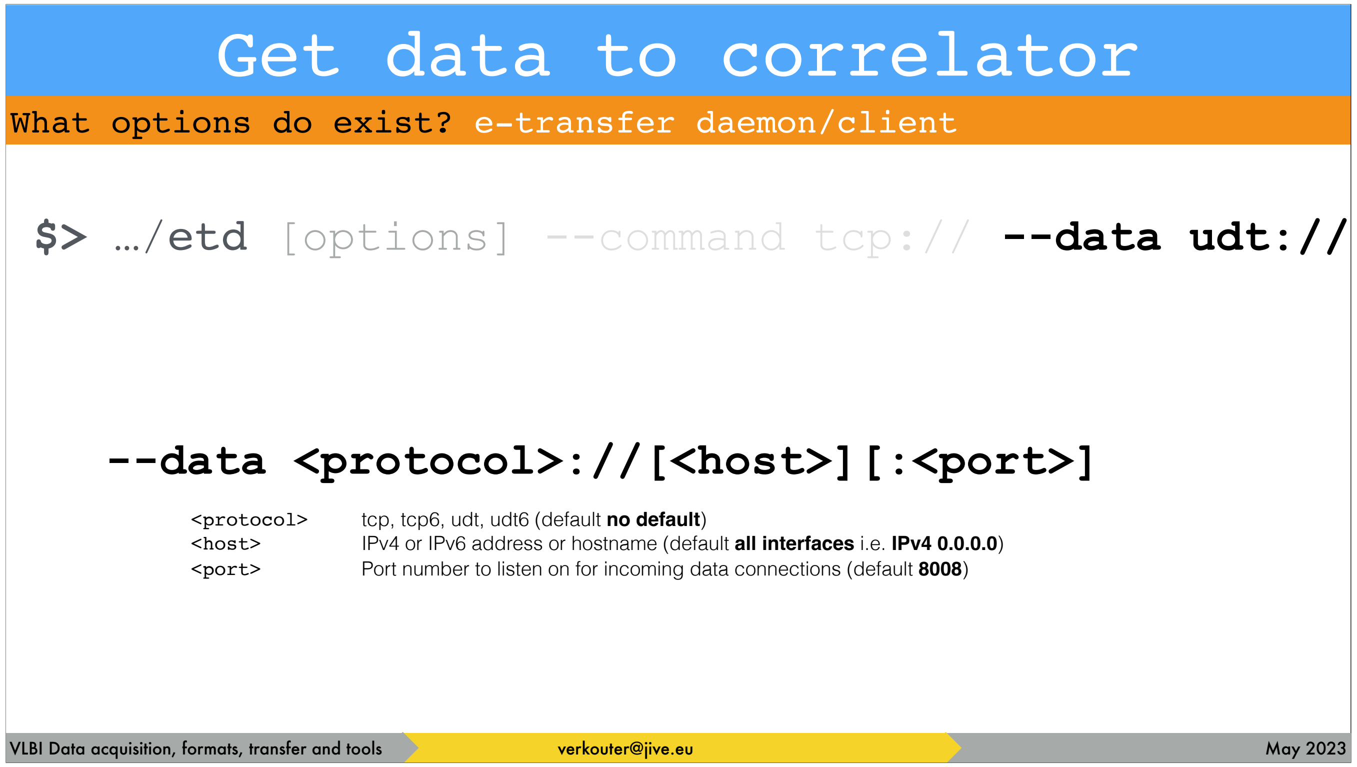
What options do exist? e-transfer daemon/client

```
$> …/etd [options] --command tcp:// --data udt://
```

```
--command <protocol>://[<host>][:<port>]
```

`<protocol>`	tcp, tcp6, udt, udt6 (default **no default**)
`<host>`	IPv4 or IPv6 address or hostname (default **all interfaces** i.e. **IPv4 0.0.0.0**)
`<port>`	Port number to listen on for incoming command connections (default **4004**)

a command channel where to listen on for incoming client connections



Get data to correlator

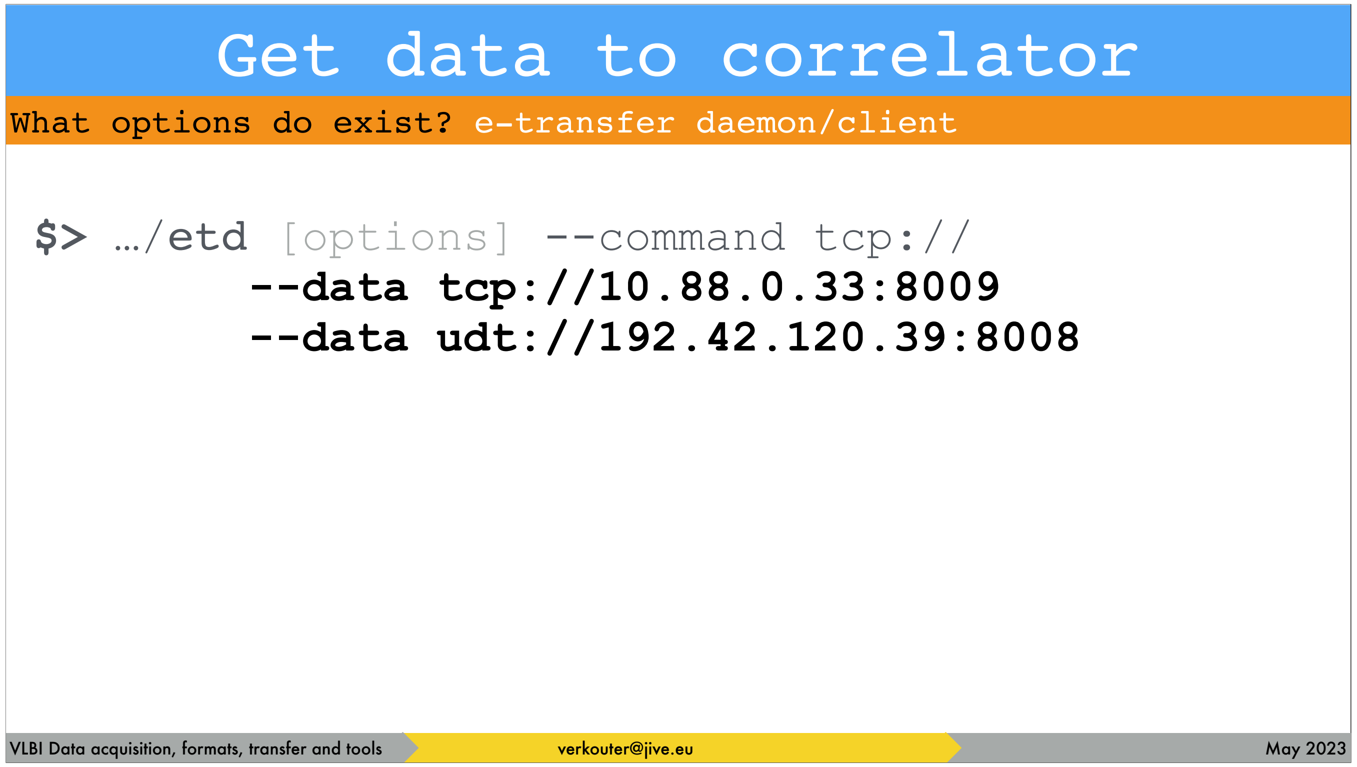
What options do exist? e-transfer daemon/client

```
$> …/etd [options] --command tcp:// --data udt://
```

```
--data <protocol>://[<host>][:<port>]
```

`<protocol>`	tcp, tcp6, udt, udt6 (default **no default**)
`<host>`	IPv4 or IPv6 address or hostname (default **all interfaces** i.e. **IPv4 0.0.0.0**)
`<port>`	Port number to listen on for incoming data connections (default **8008**)

and at least one data channel over which you allow incoming data

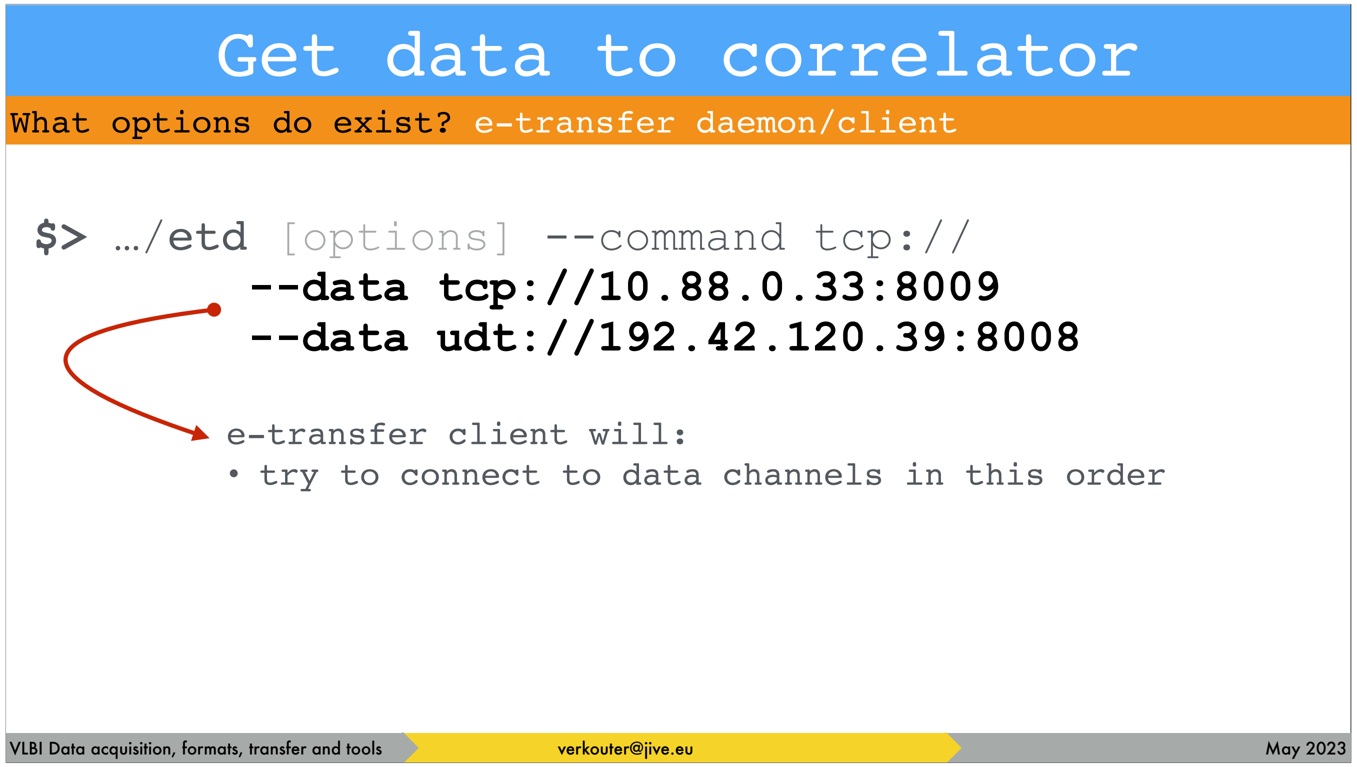


Get data to correlator

What options do exist? e-transfer daemon/client

```
$> …/etd [options] --command tcp://
        --data tcp://10.88.0.33:8009
        --data udt://192.42.120.39:8008
```

if you specify multiple data channels interesting things can happen!



Get data to correlator

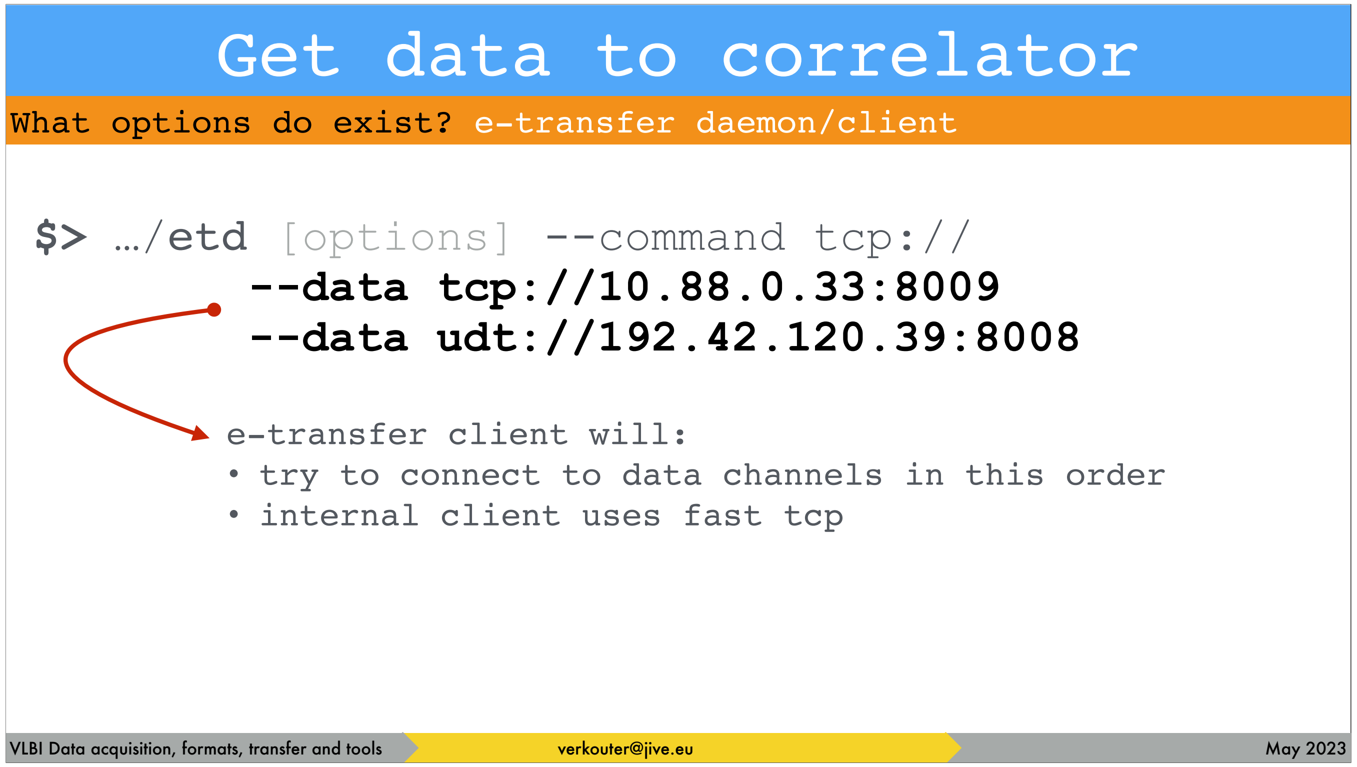
What options do exist? e-transfer daemon/client

```
$> …/etd [options] --command tcp://
            --data tcp://10.88.0.33:8009
            --data udt://192.42.120.39:8008
```

e-transfer client will:

- try to connect to data channels in this order

the e-transfer client will attempt to connect the data channels in the order they were specified on the daemon command line!



Get data to correlator

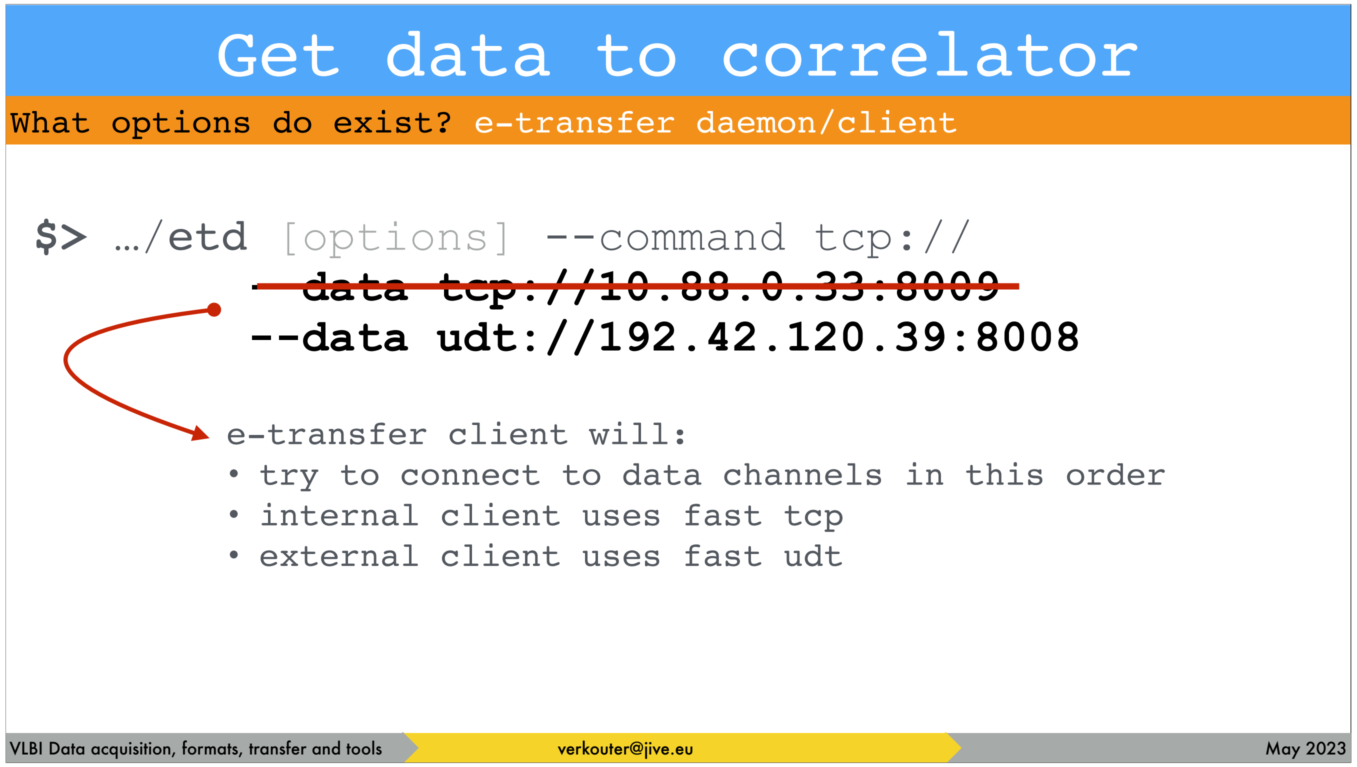
What options do exist? e-transfer daemon/client

```
$> …/etd [options] --command tcp://
          --data tcp://10.88.0.33:8009
          --data udt://192.42.120.39:8008
```

e-transfer client will:

- try to connect to data channels in this order
- internal client uses fast tcp

and in this case it means that if you do an internal data transfer, it uses tcp which is faster inside your institute.



Get data to correlator

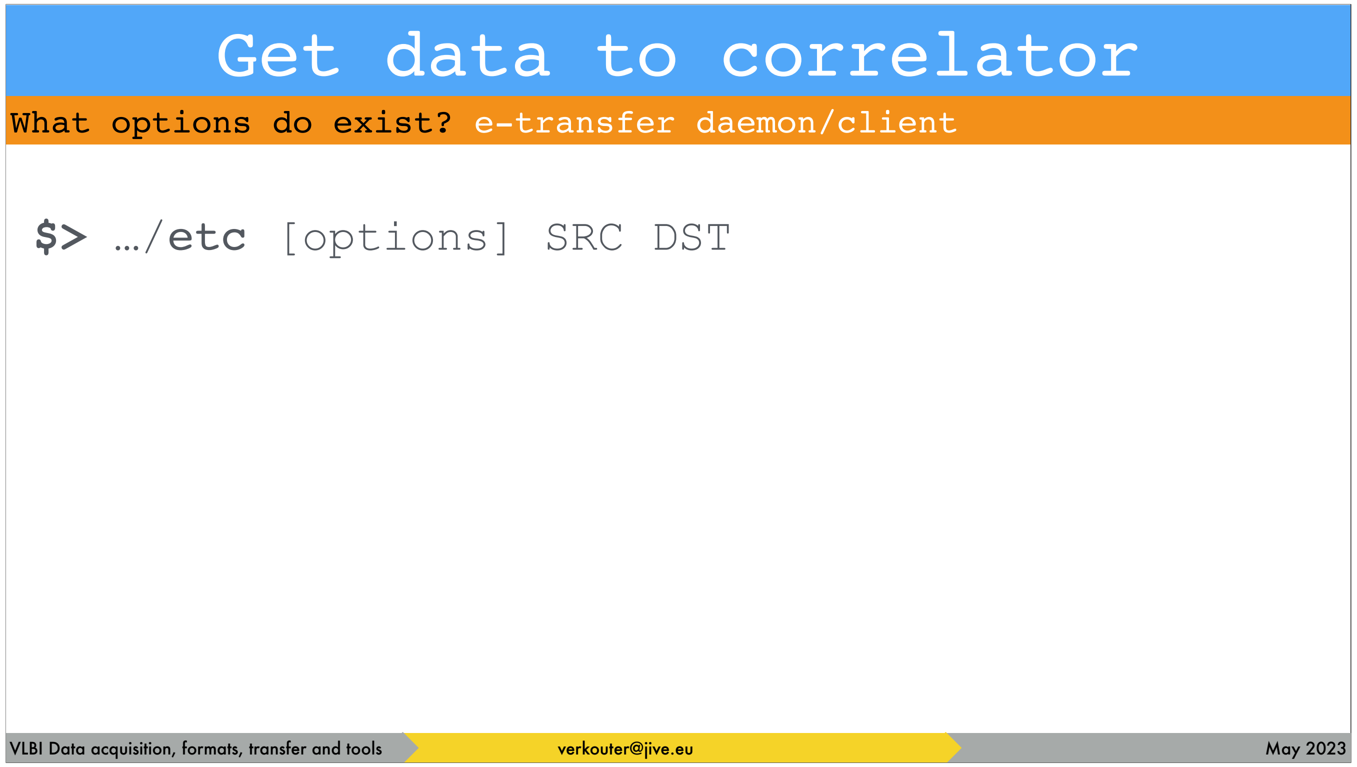
What options do exist? e-transfer daemon/client

$> …/etd [options] --command tcp://
~~--data tcp://10.88.0.33:8009~~
--data udt://192.42.120.39:8008

e-transfer client will:

- try to connect to data channels in this order
- internal client uses fast tcp
- external client uses fast udt

but an external client cannot connect to the internal address and so will use the fast UDT channel

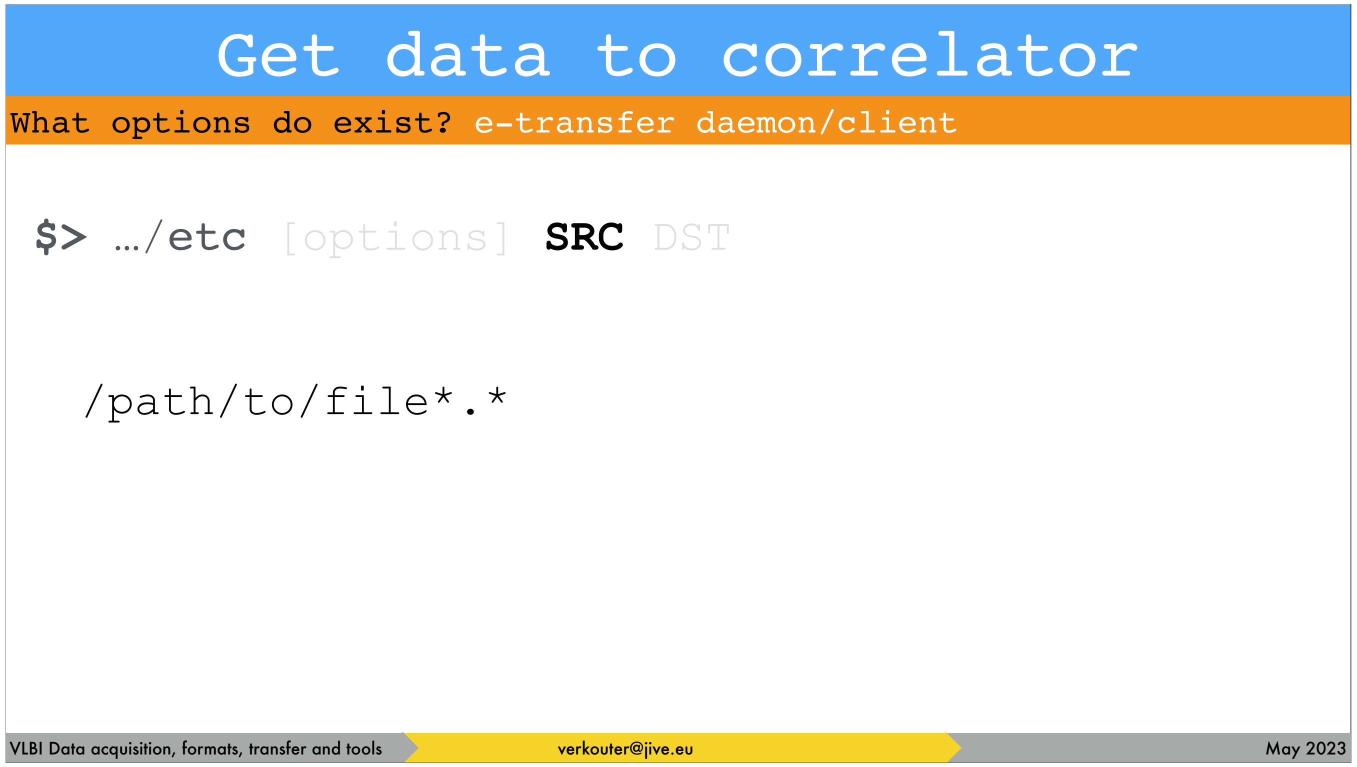


Get data to correlator

What options do exist? e-transfer daemon/client

```
$> …/etc [options] SRC DST
```

The client command looks a bit like secure copy



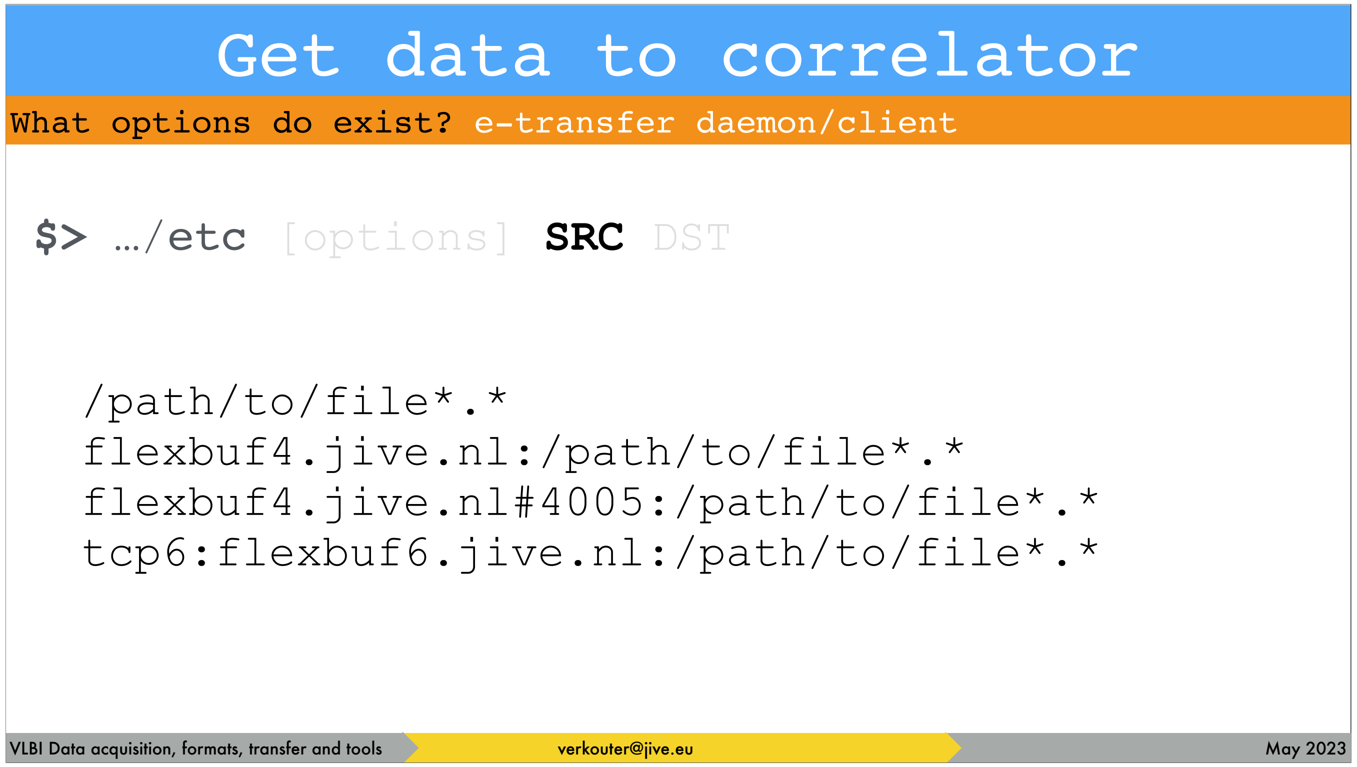
Get data to correlator

What options do exist? e-transfer daemon/client

```
$> …/etc [options] SRC DST
```

```
/path/to/file*.*
```

you can specify local files … or

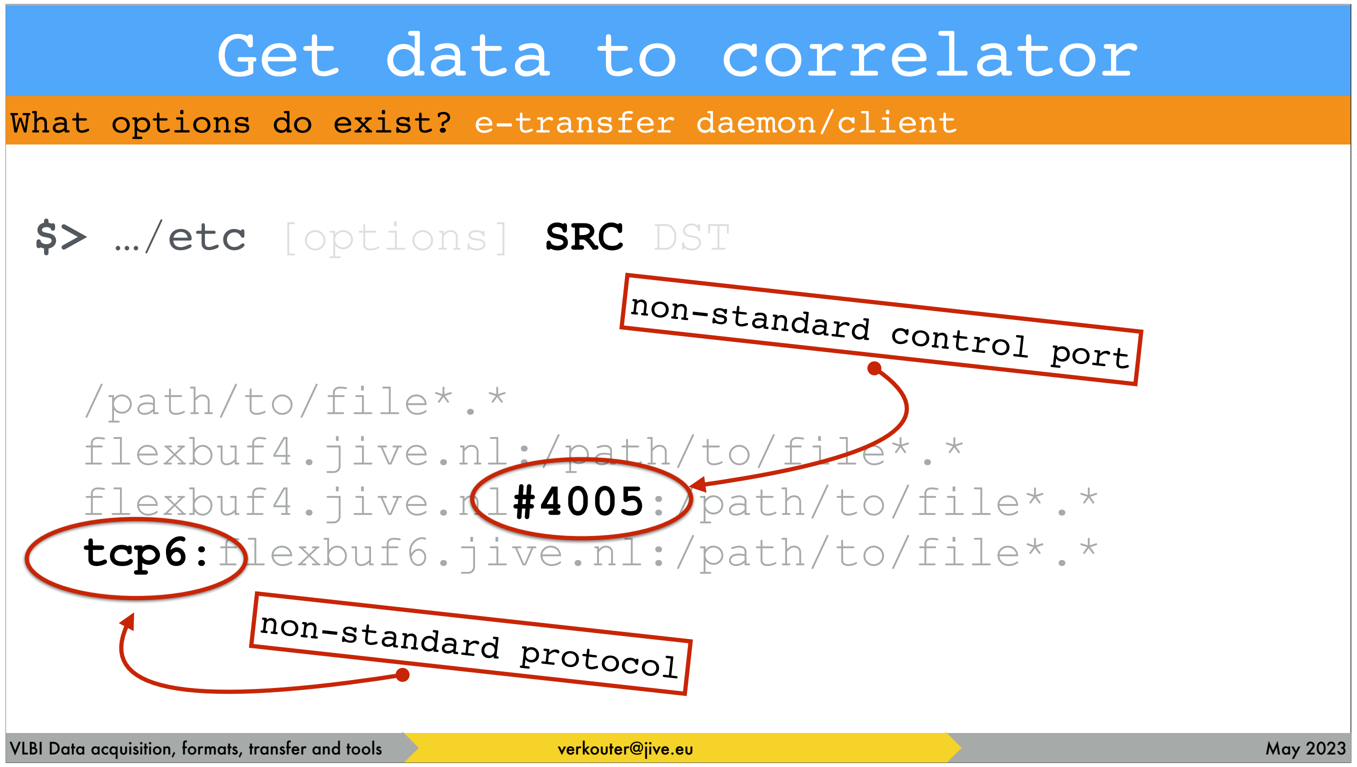


Get data to correlator

What options do exist? e-transfer daemon/client

```
$> …/etc [options] SRC DST

  /path/to/file*.*
  flexbuf4.jive.nl:/path/to/file*.*
  flexbuf4.jive.nl#4005:/path/to/file*.*
  tcp6:flexbuf6.jive.nl:/path/to/file*.*
```

.. remote files - and because the system supports remote to remote transfers



Get data to correlator

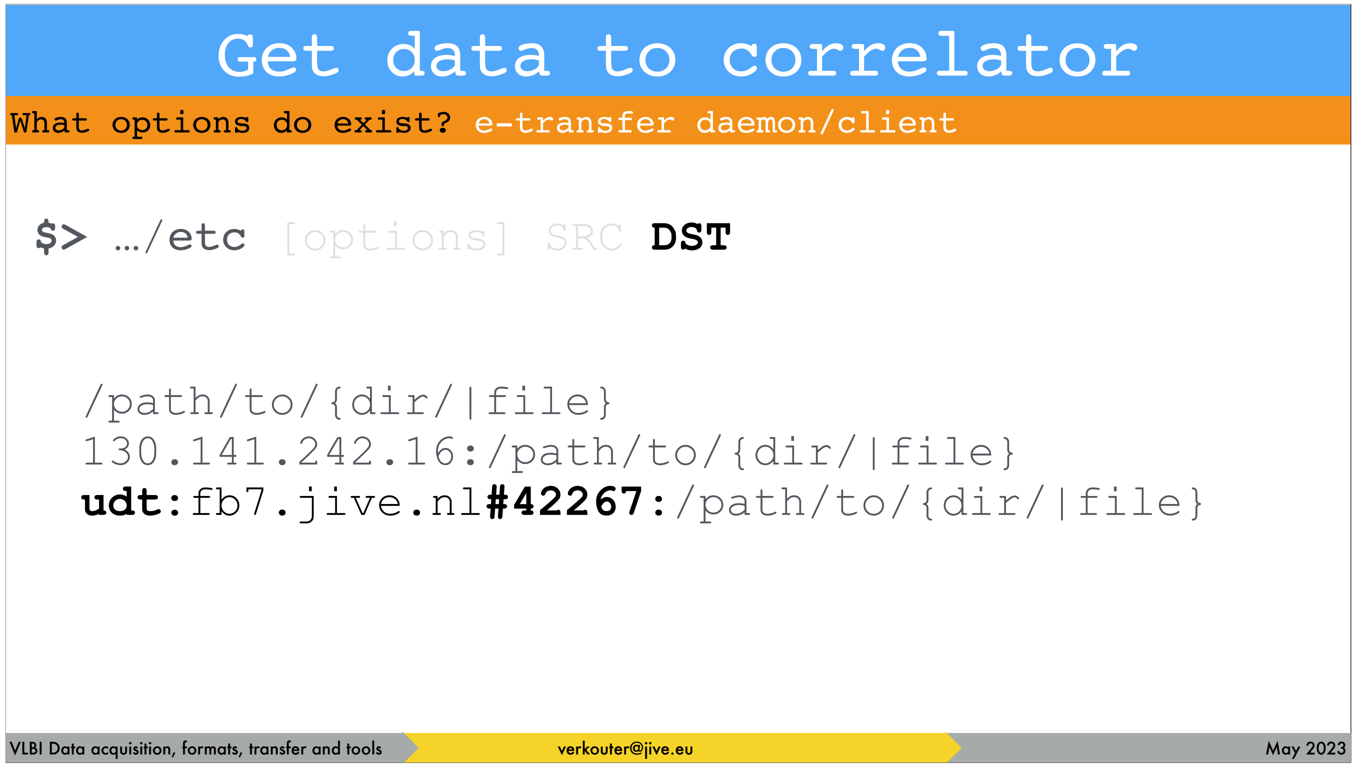
What options do exist? e-transfer daemon/client

```
$> …/etc [options] SRC DST

  /path/to/file*.*
  flexbuf4.jive.nl:/path/to/file*.*
  flexbuf4.jive.nl#4005:/path/to/file*.*
  tcp6:flexbuf6.jive.nl:/path/to/file*.*
```

non-standard control port

non-standard protocol

you may have to encode a lot of information in just one location …



Get data to correlator

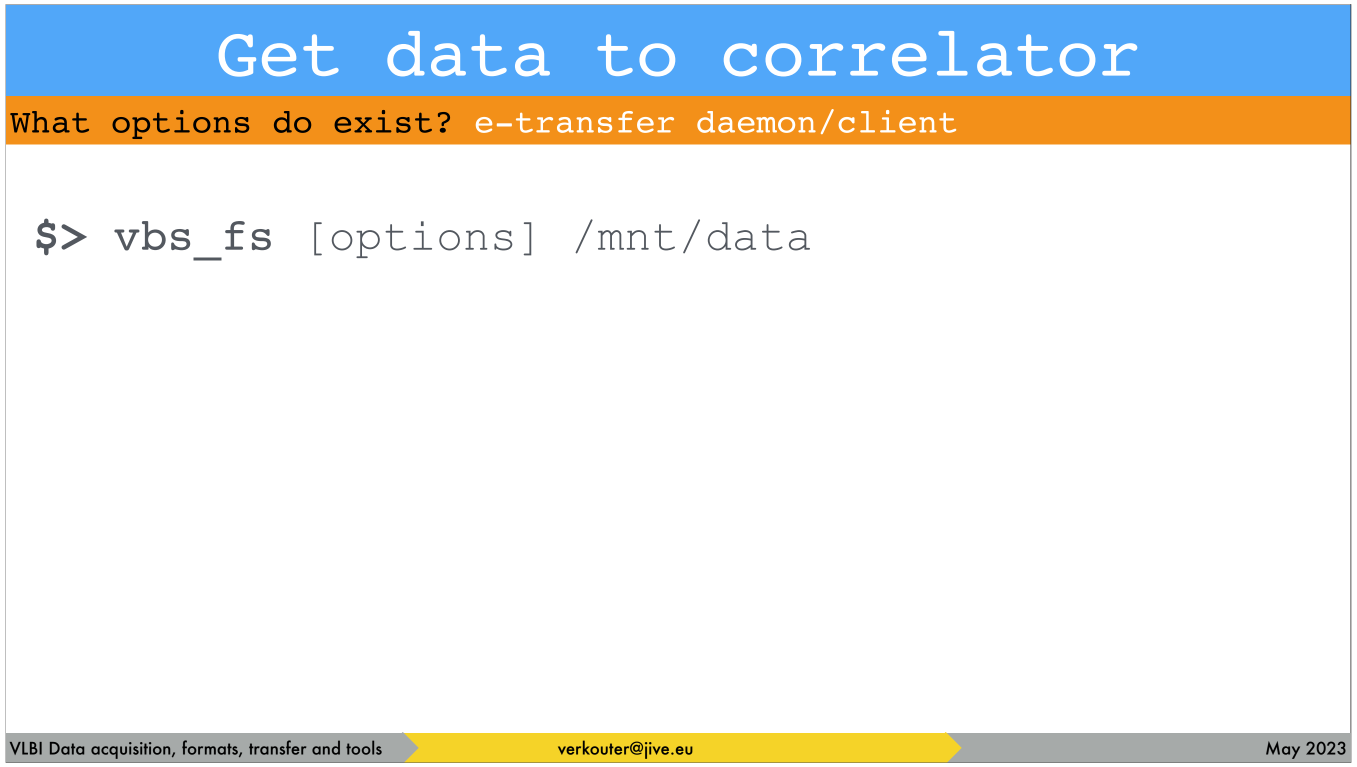
What options do exist? e-transfer daemon/client

```
$> …/etc [options] SRC DST
```

```
/path/to/{dir/|file}
130.141.242.16:/path/to/{dir/|file}
udt:fb7.jive.nl#42267:/path/to/{dir/|file}
```

VLBI Data acquisition, formats, transfer and tools verkouter@jive.eu May 2023

because the destination could be completely different again.

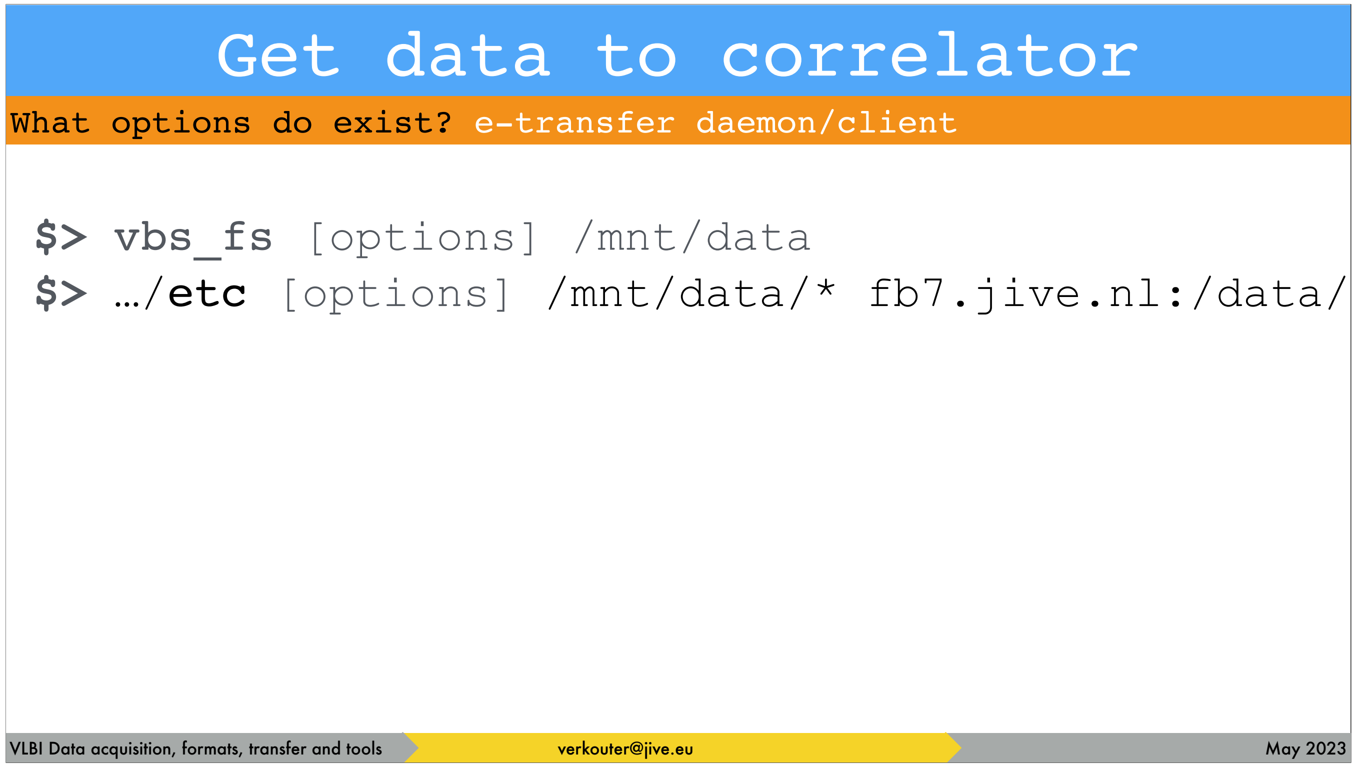


Get data to correlator

What options do exist? e-transfer daemon/client

```
$> vbs_fs [options] /mnt/data
```

The combination of vbs_fs with the e-transfer system …

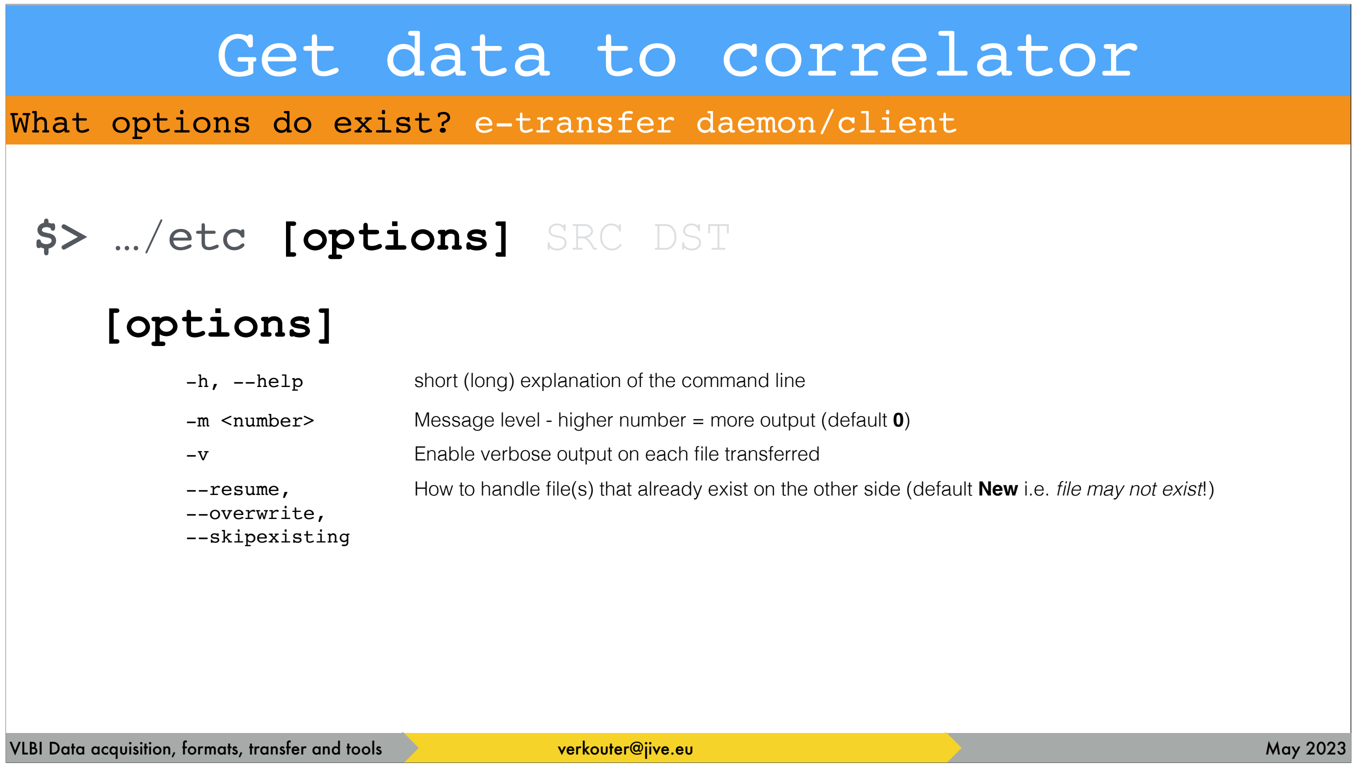


Get data to correlator

What options do exist? e-transfer daemon/client

```
$> vbs_fs [options] /mnt/data
$> …/etc [options] /mnt/data/* fb7.jive.nl:/data/
```

… makes it is easy to transfer the files like this.



Get data to correlator

What options do exist? e-transfer daemon/client

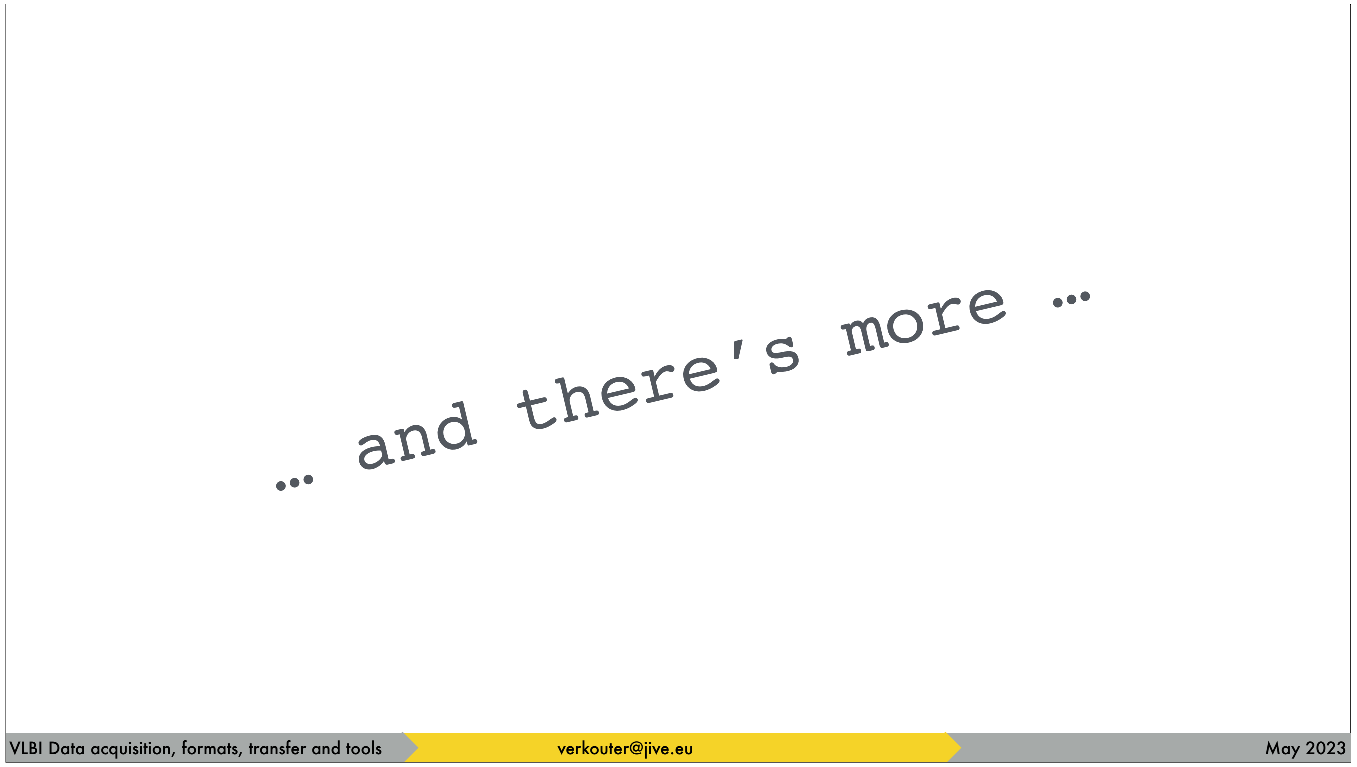
```
$> …/etc [options] SRC DST
```

[options]

`-h, --help`	short (long) explanation of the command line
`-m <number>`	Message level - higher number = more output (default **0**)
`-v`	Enable verbose output on each file transferred

`--resume,`
`--overwrite,`
`--skipexisting`
	How to handle file(s) that already exist on the other side (default **New** i.e. *file may not exist!*)

Some of the important options for the client are these. New ones may be added in the future, depending on your requests.



… and there's more …

And there is so much more that I could be explaining but there is no time!

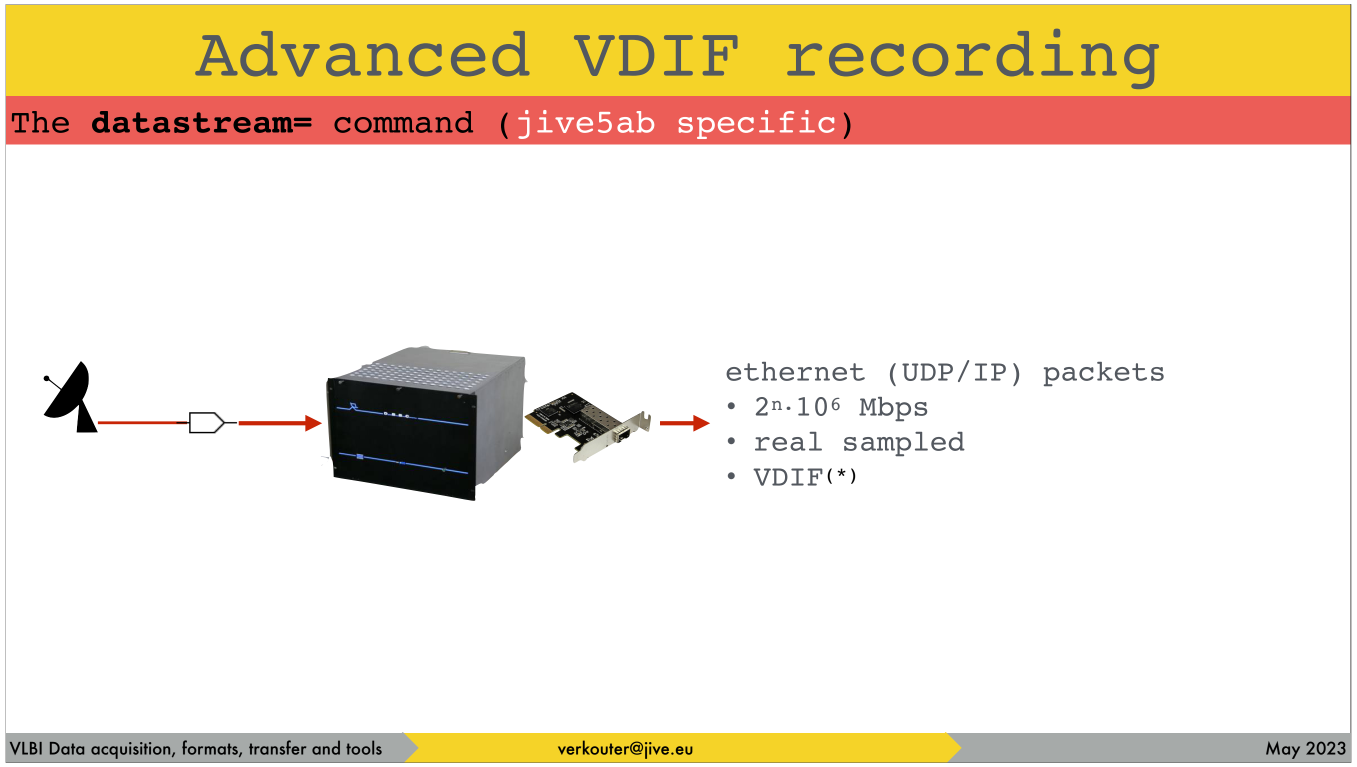
I still hope you learnt a lot in this lecture.

Thanks for attention!

in any case, many thanks for your attention!

(Automatically) collect frames in multiple recordings?

Extra Good Stuff™



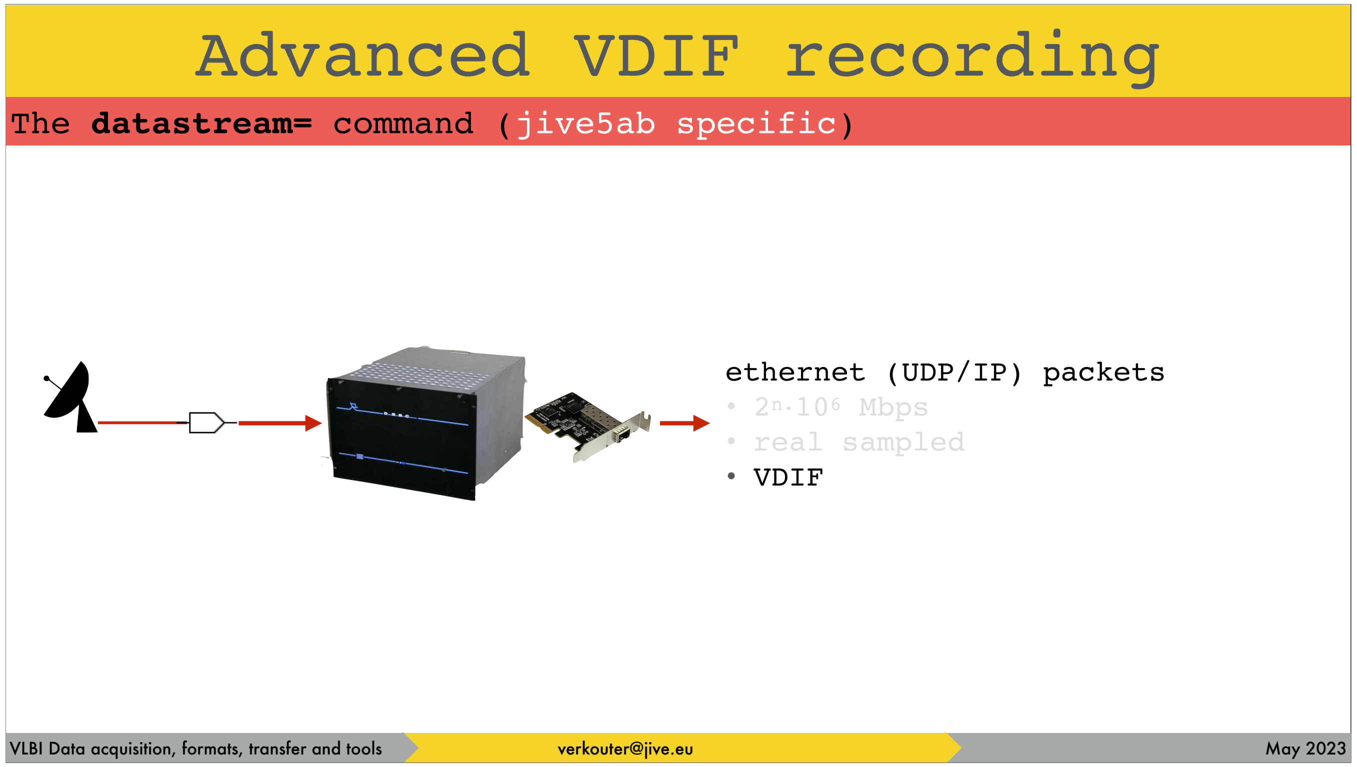
Advanced VDIF recording

The **datastream=** command (jive5ab specific)

ethernet (UDP/IP) packets

- $2^n \cdot 10^6$ Mbps
- real sampled
- VDIF$^{(*)}$

remember this picture?



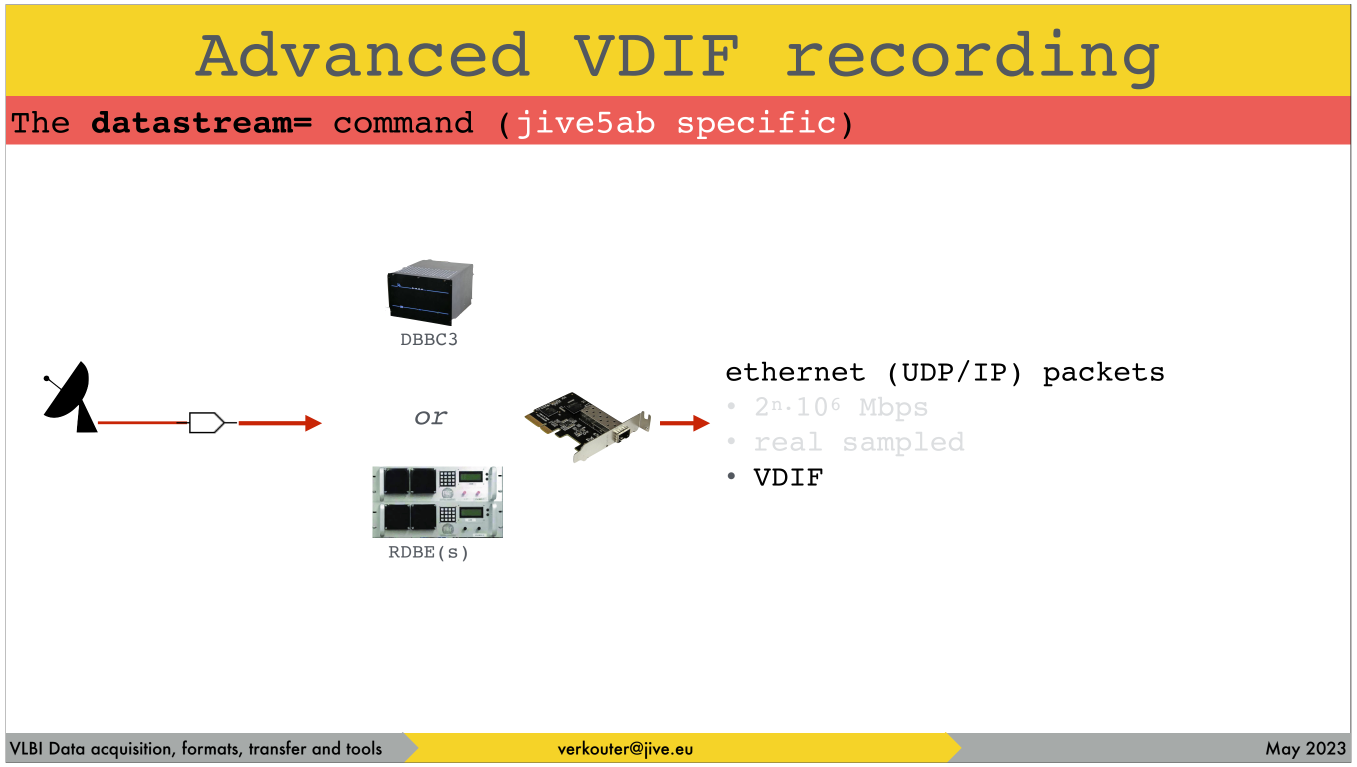
Advanced VDIF recording

The **datastream=** command (jive5ab specific)

ethernet (UDP/IP) packets

- $2^n \cdot 10^6$ Mbps
- real sampled
- VDIF

and let’s focus on this bit



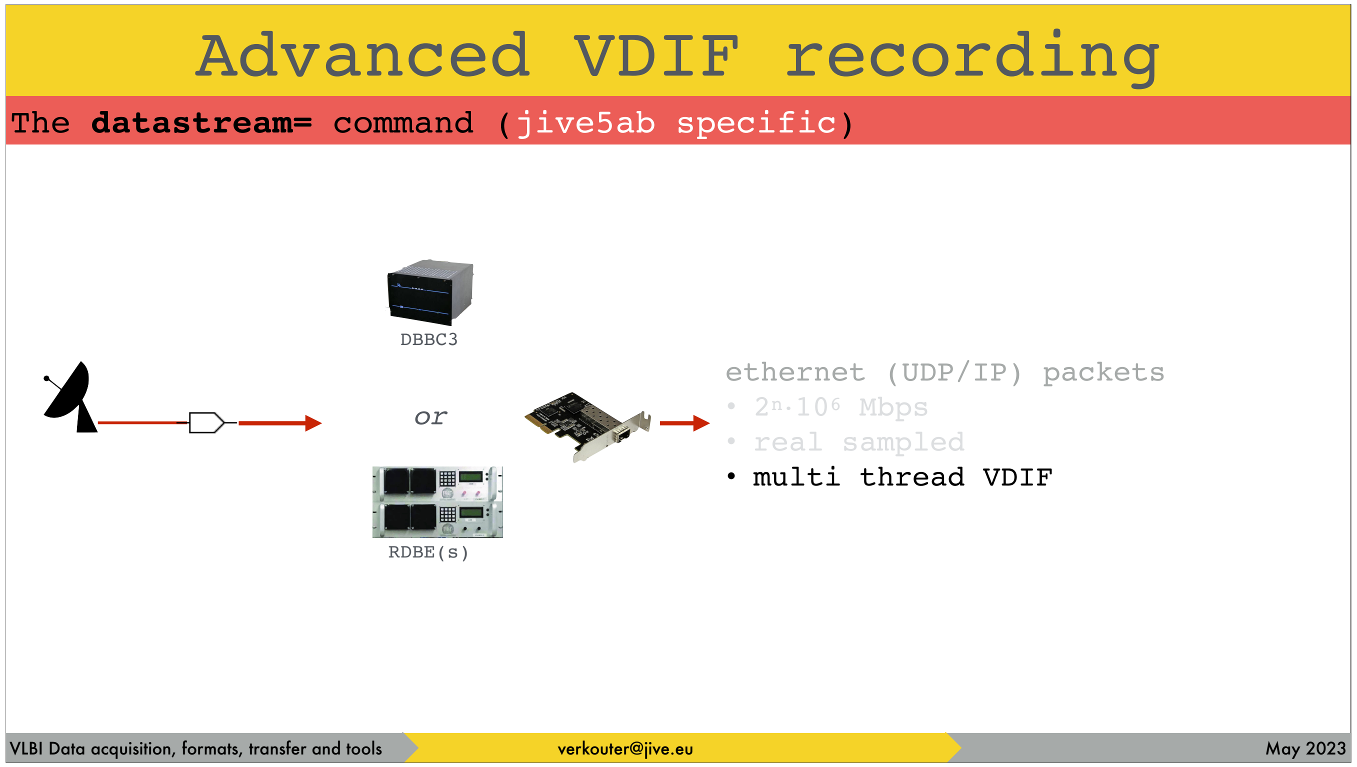
Advanced VDIF recording

The **datastream=** command (jive5ab specific)

ethernet (UDP/IP) packets

- $2^n \cdot 10^6$ Mbps
- real sampled
- VDIF

especially in these situations



Advanced VDIF recording

The **datastream=** command (jive5ab specific)

DBBC3

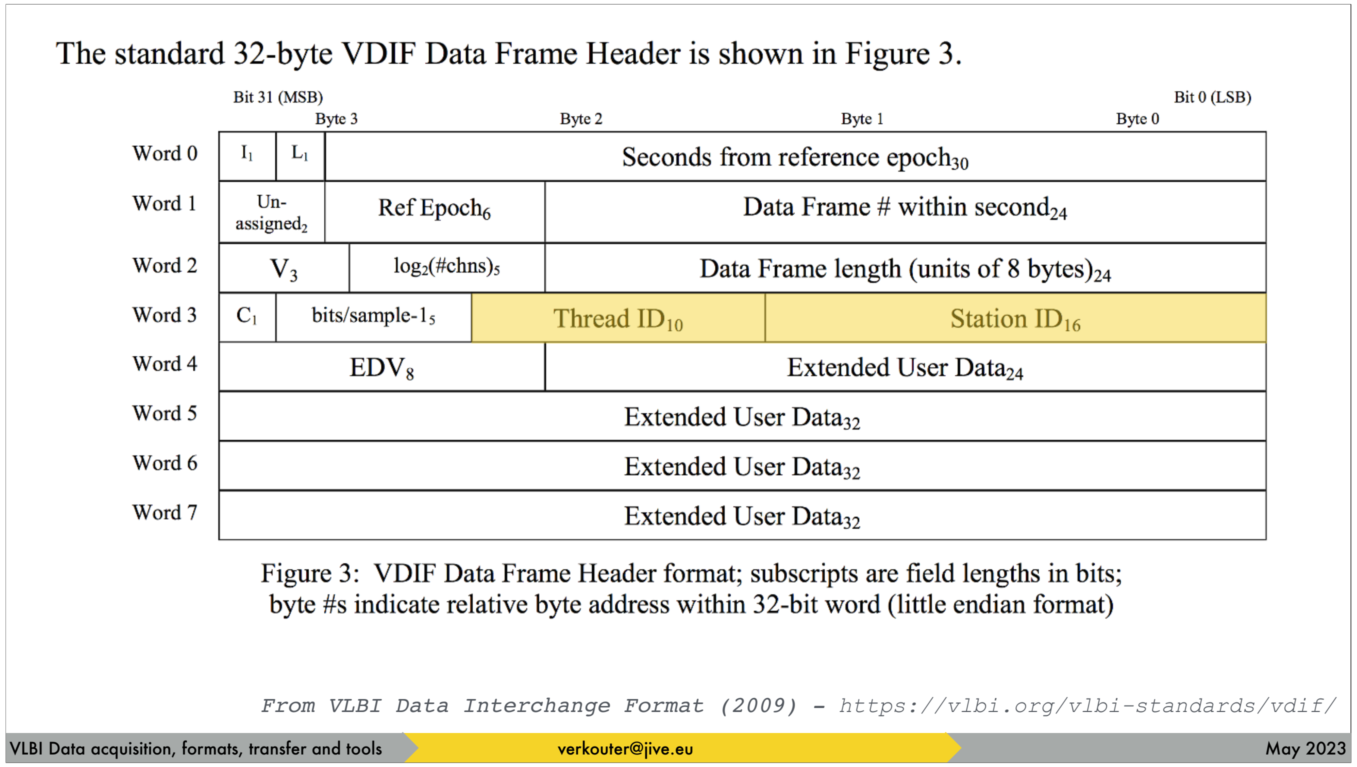
or

RDBE(s)

ethernet (UDP/IP) packets

- $2^n \cdot 10^6$ Mbps
- real sampled
- multi thread VDIF

the hardware sends out multiple VDIF threads



The standard 32-byte VDIF Data Frame Header is shown in Figure 3.

Bit 31 (MSB) Bit 0 (LSB)

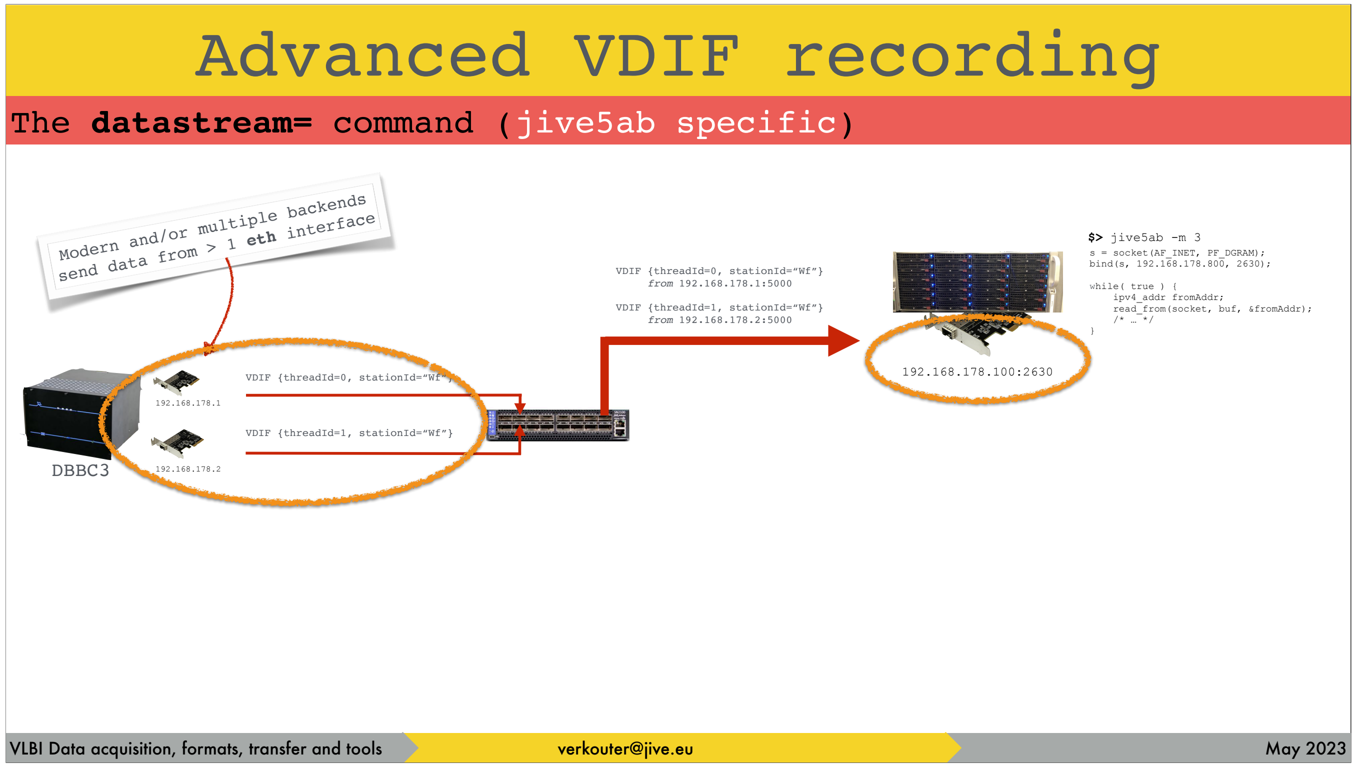
Byte 3 Byte 2 Byte 1 Byte 0

Word 0	I$_1$	L$_1$	Seconds from reference epoch$_{30}$	
Word 1	Un-assigned$_2$	Ref Epoch$_6$	Data Frame # within second$_{24}$	
Word 2	V$_3$	log$_2$(#chns)$_5$	Data Frame length (units of 8 bytes)$_{24}$	
Word 3	C$_1$	bits/sample-1$_5$	Thread ID$_{10}$	Station ID$_{16}$
Word 4	EDV$_8$	Extended User Data$_{24}$		
Word 5	Extended User Data$_{32}$			
Word 6	Extended User Data$_{32}$			
Word 7	Extended User Data$_{32}$			

Figure 3: VDIF Data Frame Header format; subscripts are field lengths in bits; byte #s indicate relative byte address within 32-bit word (little endian format)

From VLBI Data Interchange Format (2009) – https://vlbi.org/vlbi-standards/vdif/

And I’ll reshow the VDIF header definition: there are two fields available for identification of the data a station id and a thread id



Advanced VDIF recording

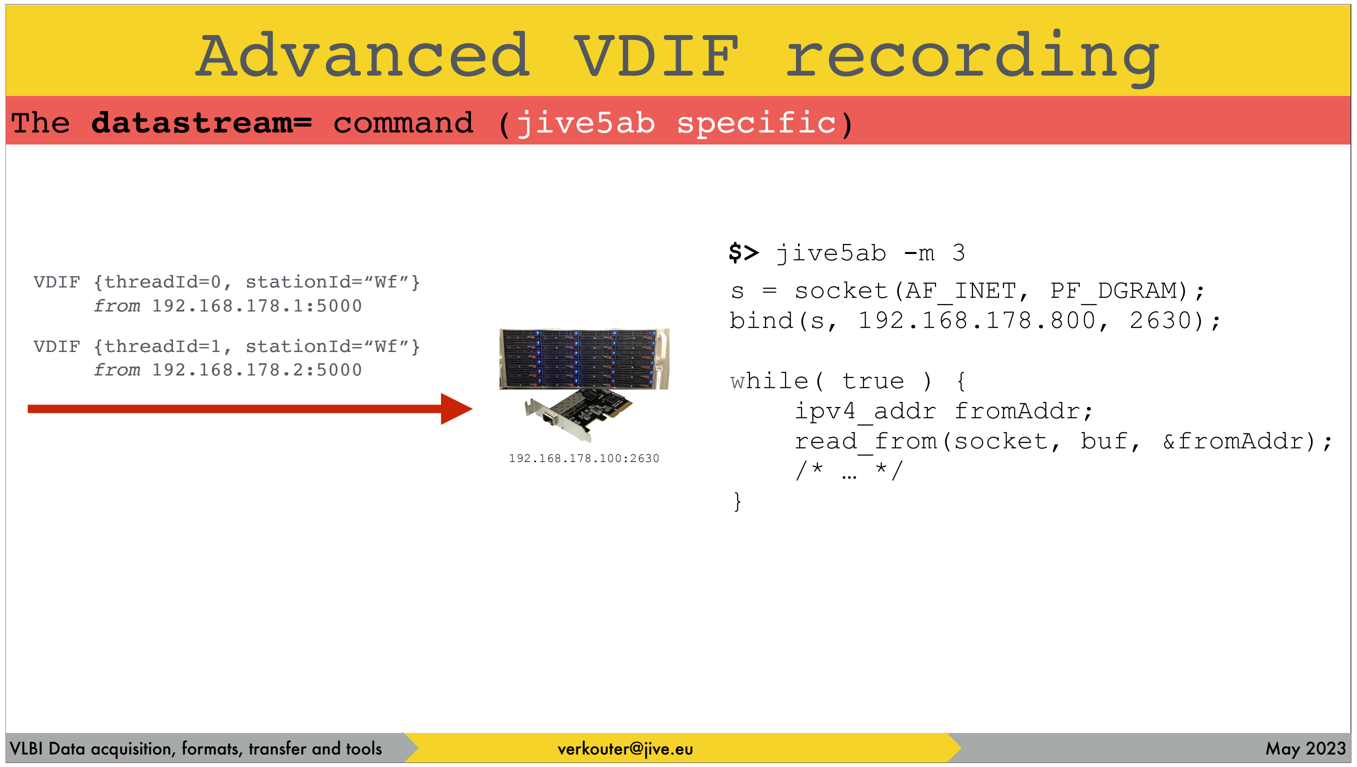
The **datastream=** command (jive5ab specific)

Modern and/or multiple backends send data from > 1 **eth** interface

DBBC3
192.168.178.1
192.168.178.2
VDIF {threadId=0, stationId="Wf"}
VDIF {threadId=1, stationId="Wf"}
VDIF {threadId=0, stationId="Wf"}
from 192.168.178.1:5000
VDIF {threadId=1, stationId="Wf"}
from 192.168.178.2:5000
192.168.178.100:2630

```
$> jive5ab -m 3
s = socket(AF_INET, PF_DGRAM);
bind(s, 192.168.178.800, 2630);

while( true ) {
    ipv4_addr fromAddr;
    read_from(socket, buf, &fromAddr);
    /* … */
}
```

from jive5ab's point of view it sees this. The VDIF frames [click] from the different equipment are sent to its network interface



Advanced VDIF recording

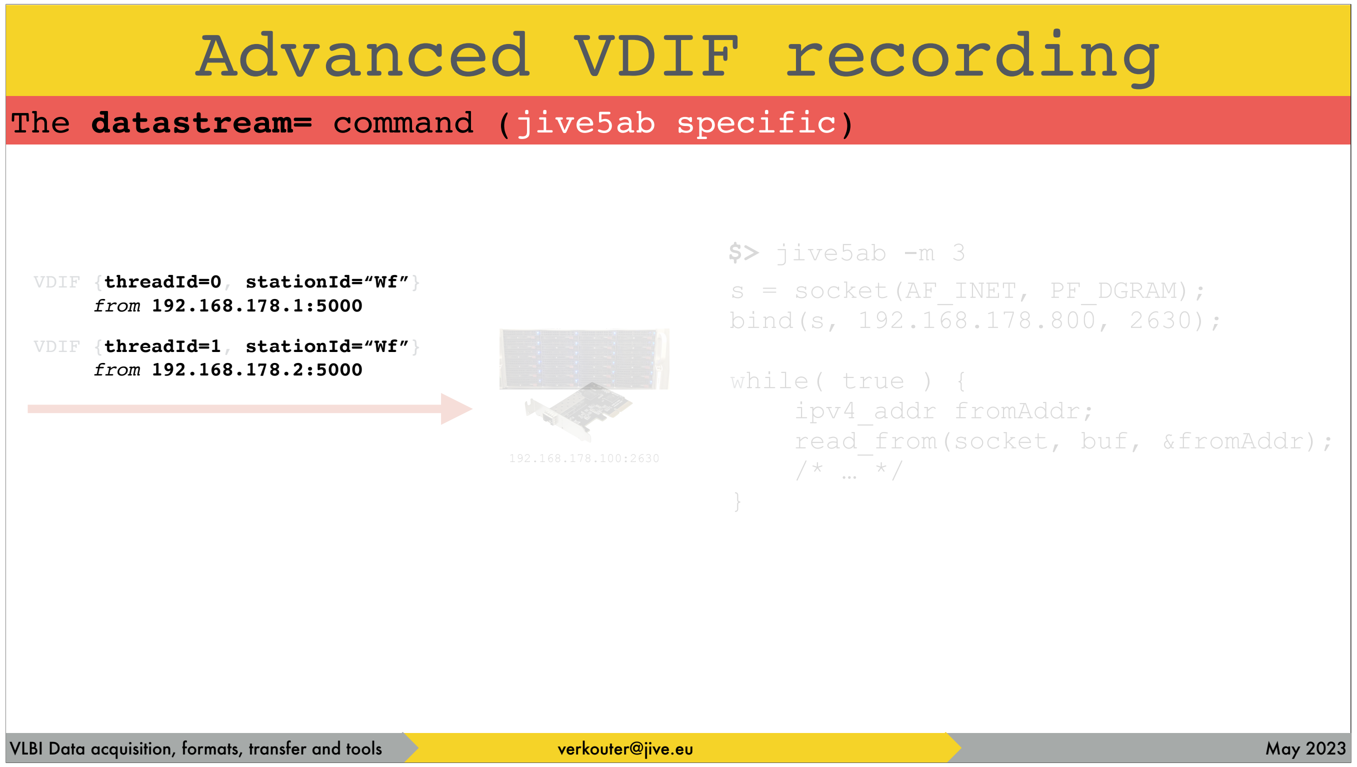
The **datastream=** command (jive5ab specific)

```
VDIF {threadId=0, stationId="Wf"}
     from 192.168.178.1:5000

VDIF {threadId=1, stationId="Wf"}
     from 192.168.178.2:5000
```

192.168.178.100:2630

```
$> jive5ab -m 3
s = socket(AF_INET, PF_DGRAM);
bind(s, 192.168.178.800, 2630);

while( true ) {
    ipv4_addr fromAddr;
    read_from(socket, buf, &fromAddr);
    /* … */
}
```

if we focus on that last part, what does it see incoming



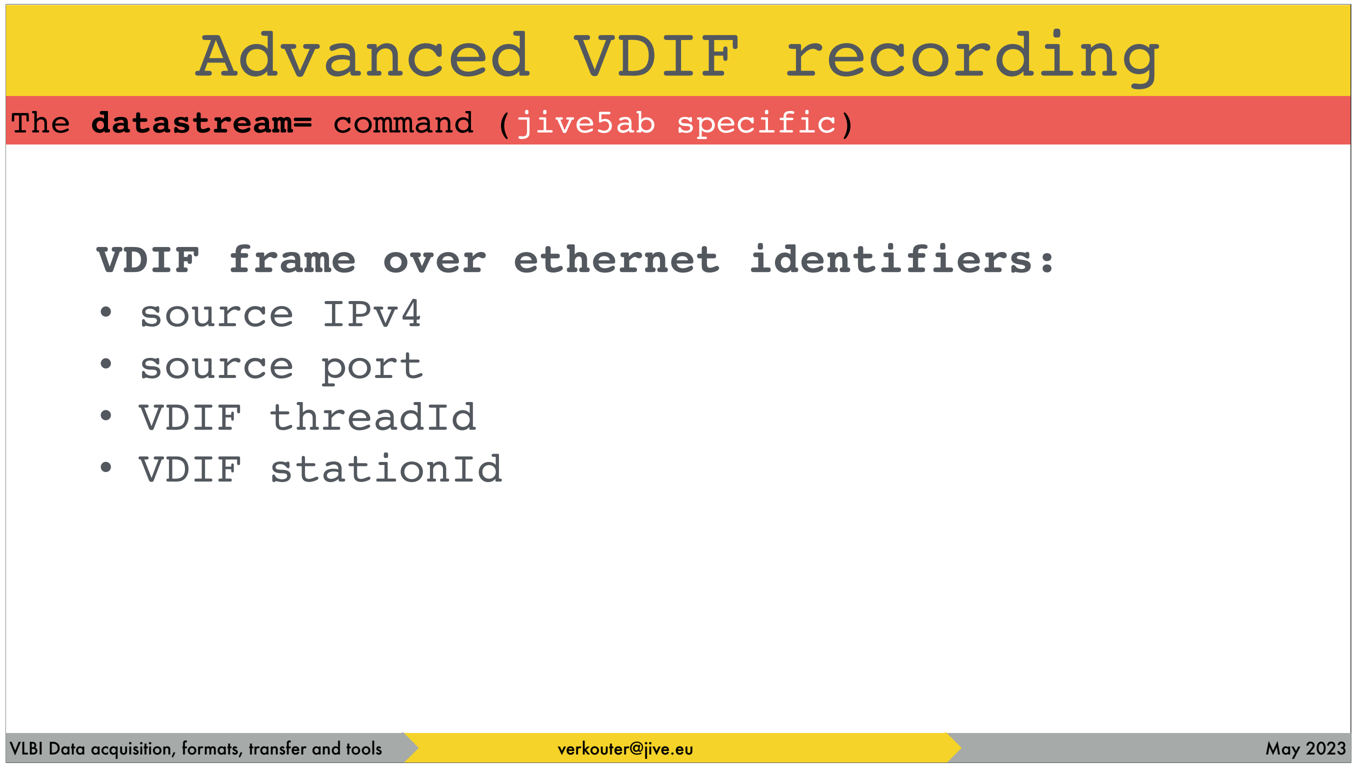
Advanced VDIF recording

The **datastream=** command (jive5ab specific)

```
VDIF {threadId=0, stationId="Wf"}
     from 192.168.178.1:5000

VDIF {threadId=1, stationId="Wf"}
     from 192.168.178.2:5000
```

```
$> jive5ab -m 3
s = socket(AF_INET, PF_DGRAM);
bind(s, 192.168.178.800, 2630);

while( true ) {
    ipv4_addr fromAddr;
    read_from(socket, buf, &fromAddr);
    /* … */
}
```

then we see that each vdif frame has four identifiers



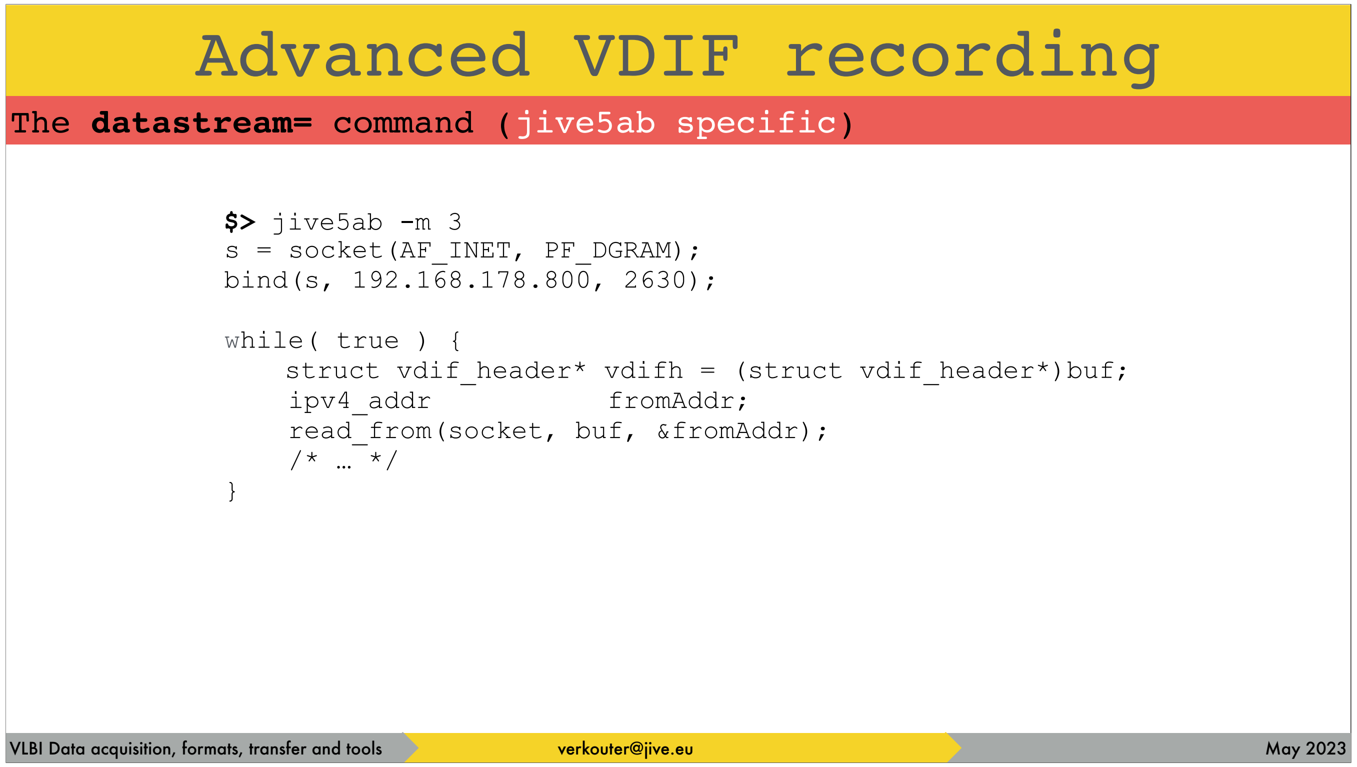
Advanced VDIF recording

The **datastream=** command (jive5ab specific)

VDIF frame over ethernet identifiers:

- source IPv4
- source port
- VDIF threadId
- VDIF stationId

that can be summarized as this

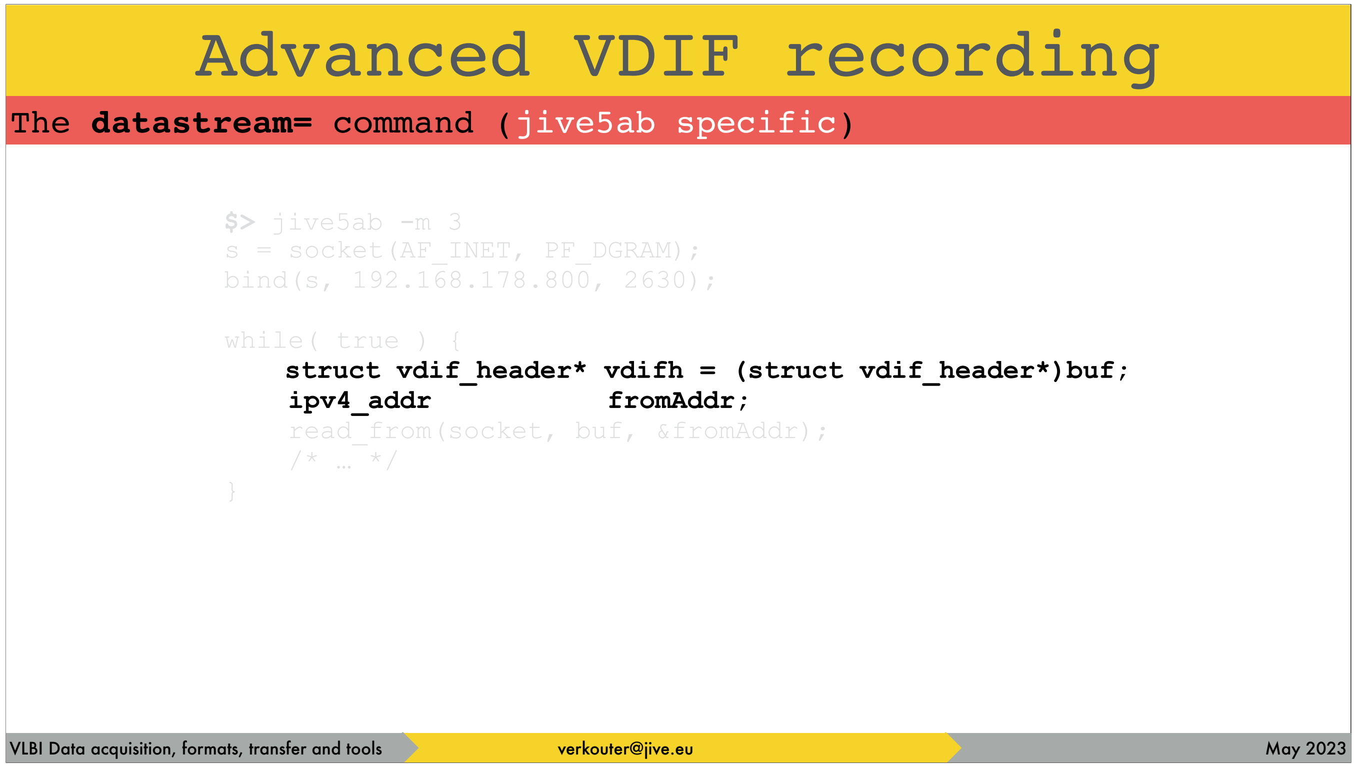


Advanced VDIF recording

The **datastream=** command (jive5ab specific)

```
$> jive5ab -m 3
s = socket(AF_INET, PF_DGRAM);
bind(s, 192.168.178.800, 2630);

while( true ) {
    struct vdif_header* vdifh = (struct vdif_header*)buf;
    ipv4_addr           fromAddr;
    read_from(socket, buf, &fromAddr);
    /* … */
}
```

the jive5ab code that reads packets



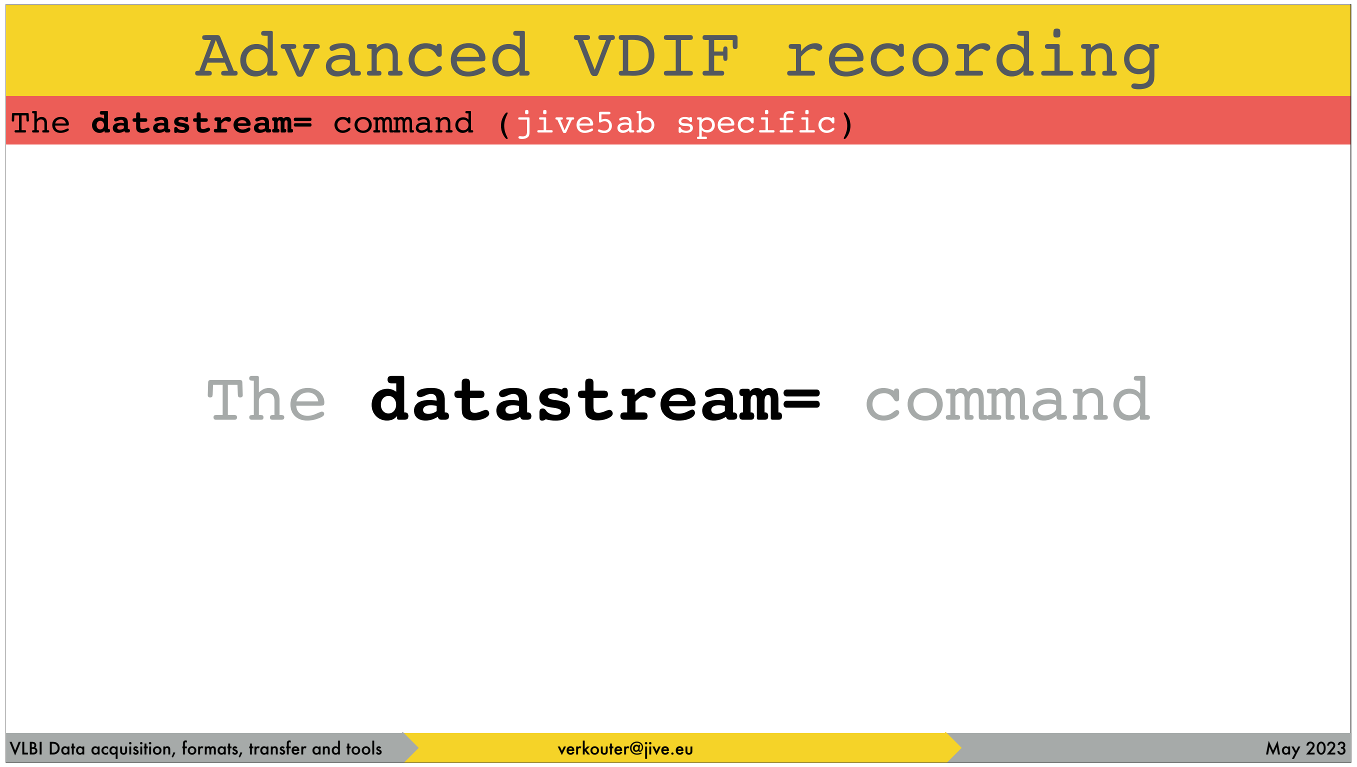
Advanced VDIF recording

The **datastream=** command (jive5ab specific)

```
$> jive5ab -m 3
s = socket(AF_INET, PF_DGRAM);
bind(s, 192.168.178.800, 2630);

while( true ) {
    struct vdif_header* vdifh = (struct vdif_header*)buf;
    ipv4_addr           fromAddr;
    read_from(socket, buf, &fromAddr);
    /* … */
}
```

VLBI Data acquisition, formats, transfer and tools verkouter@jive.eu May 2023

has access to all those fields and lets you do interesting things with it

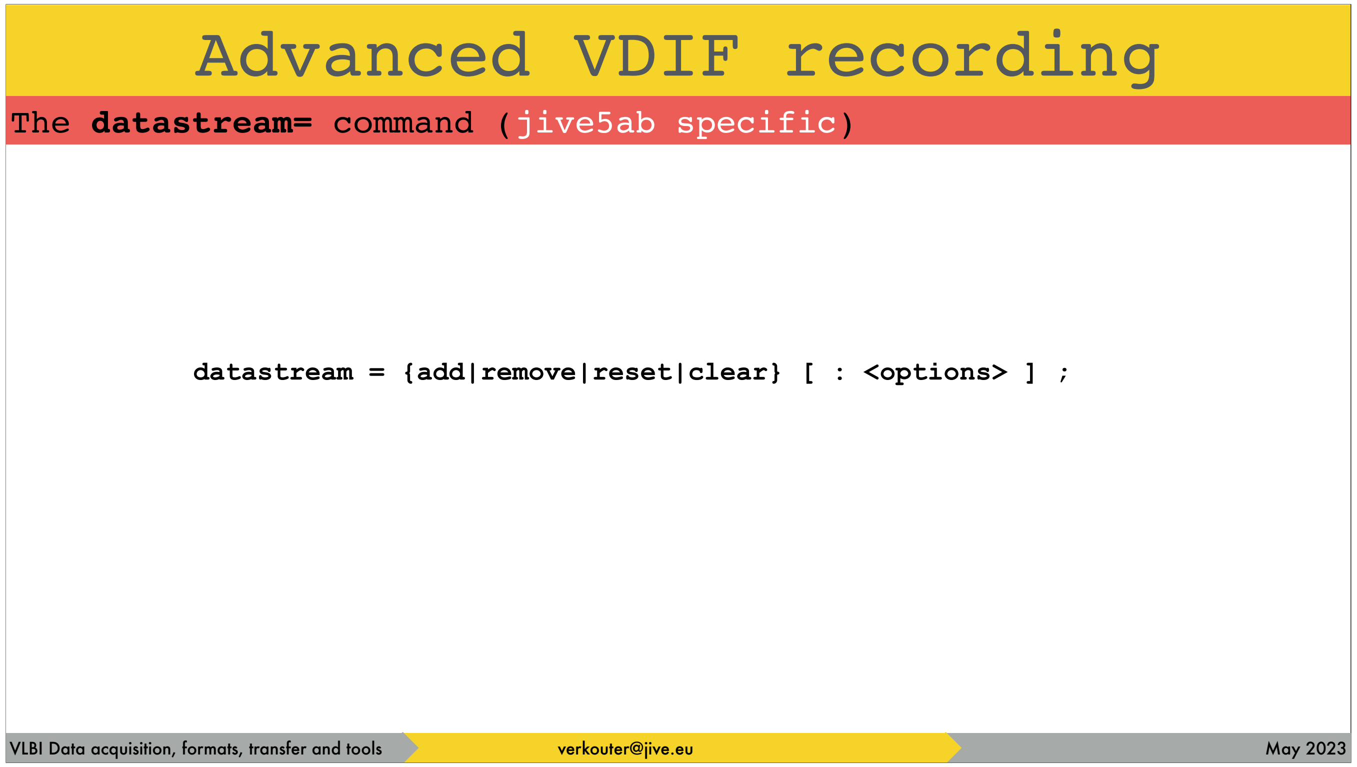


Advanced VDIF recording

The **datastream=** command (jive5ab specific)

The **datastream=** command

using this command.

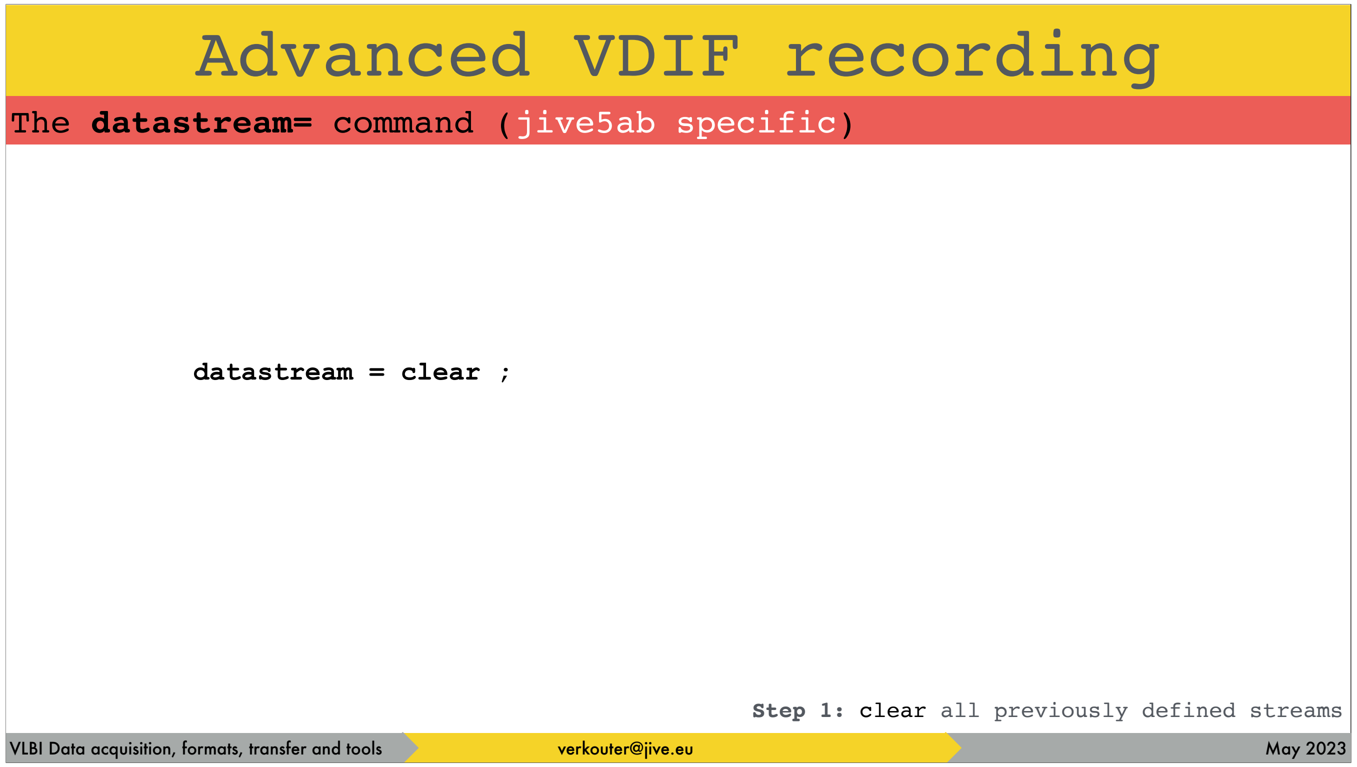


Advanced VDIF recording

The **datastream=** command (jive5ab specific)

```
datastream = {add|remove|reset|clear} [ : <options> ] ;
```

the datastream command has a few subcommands



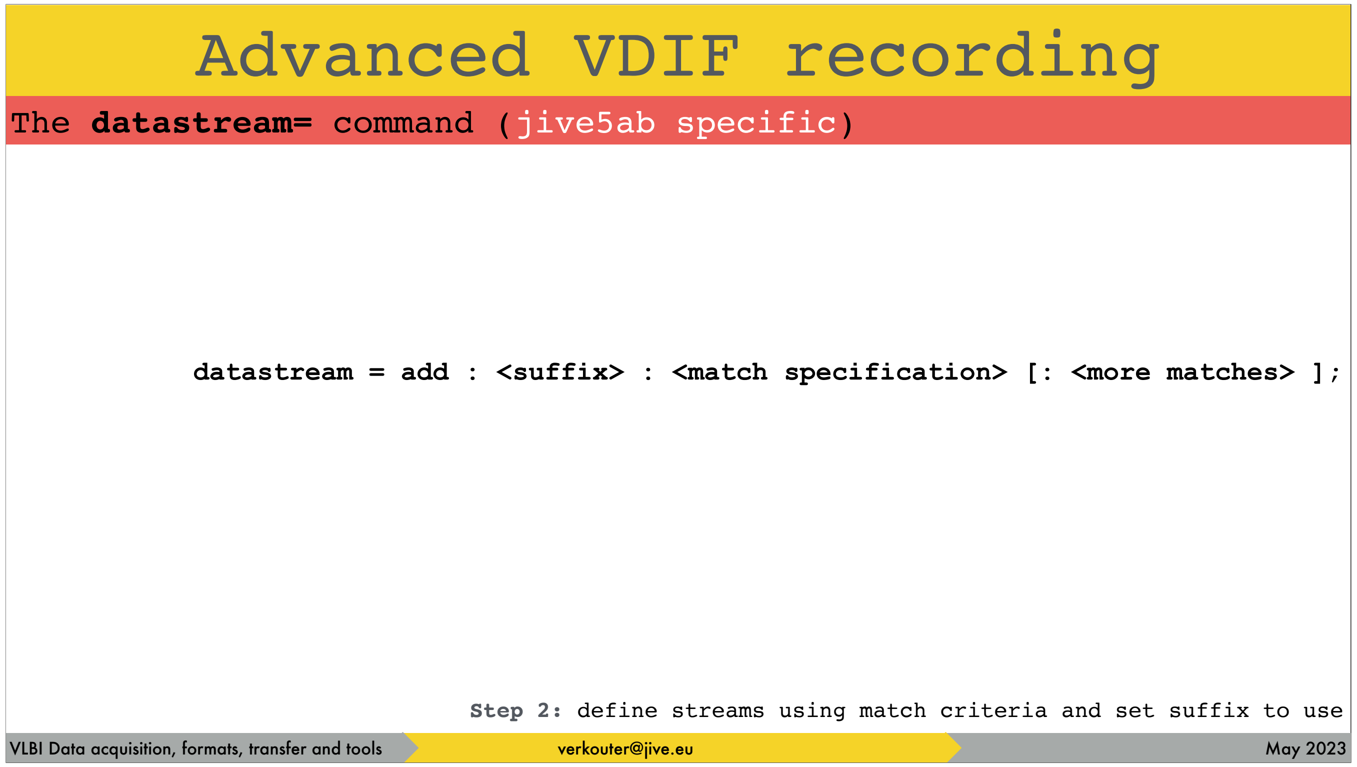
Advanced VDIF recording

The **datastream=** command (jive5ab specific)

```
datastream = clear ;
```

Step 1: clear all previously defined streams

the most reliable operation is at startup to clear whatever was defined



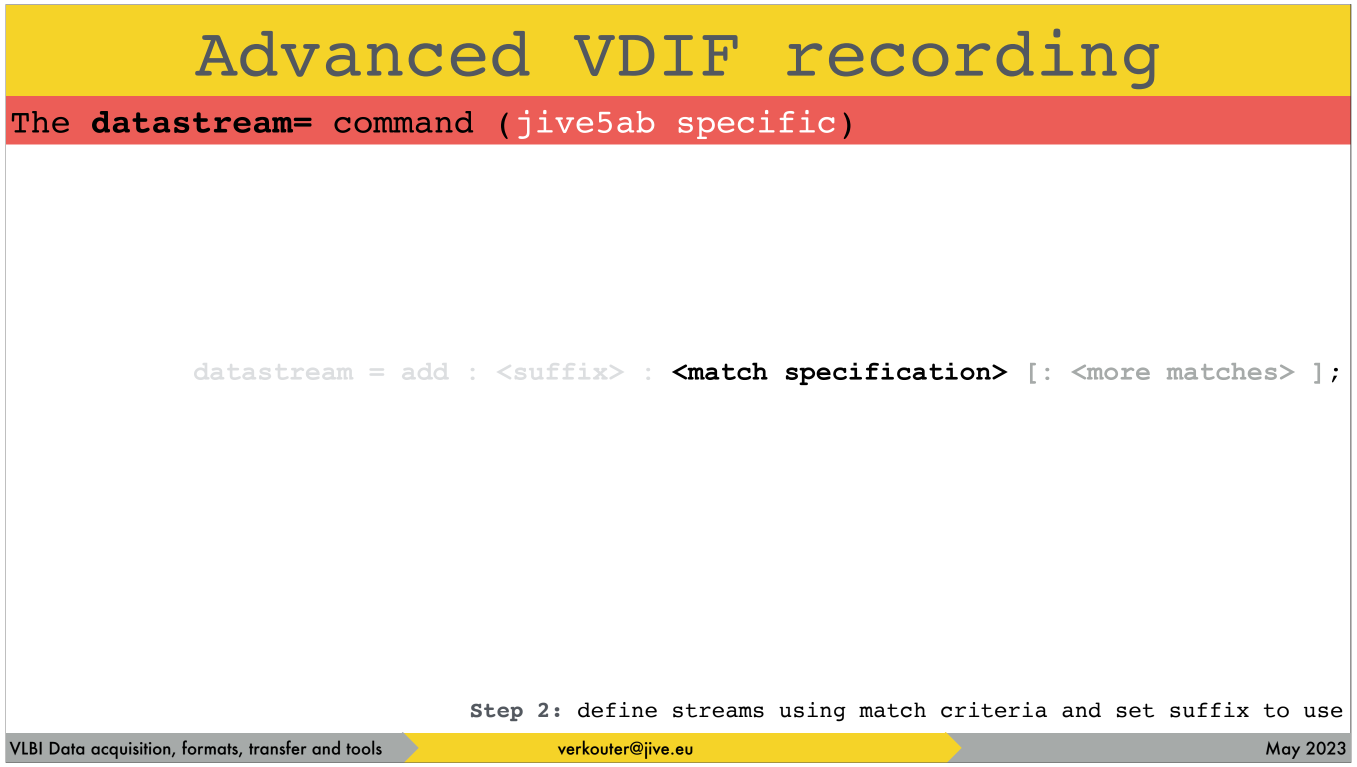
Advanced VDIF recording

The **datastream=** command (jive5ab specific)

```
datastream = add : <suffix> : <match specification> [: <more matches> ];
```

Step 2: define streams using match criteria and set suffix to use

then you need to set up



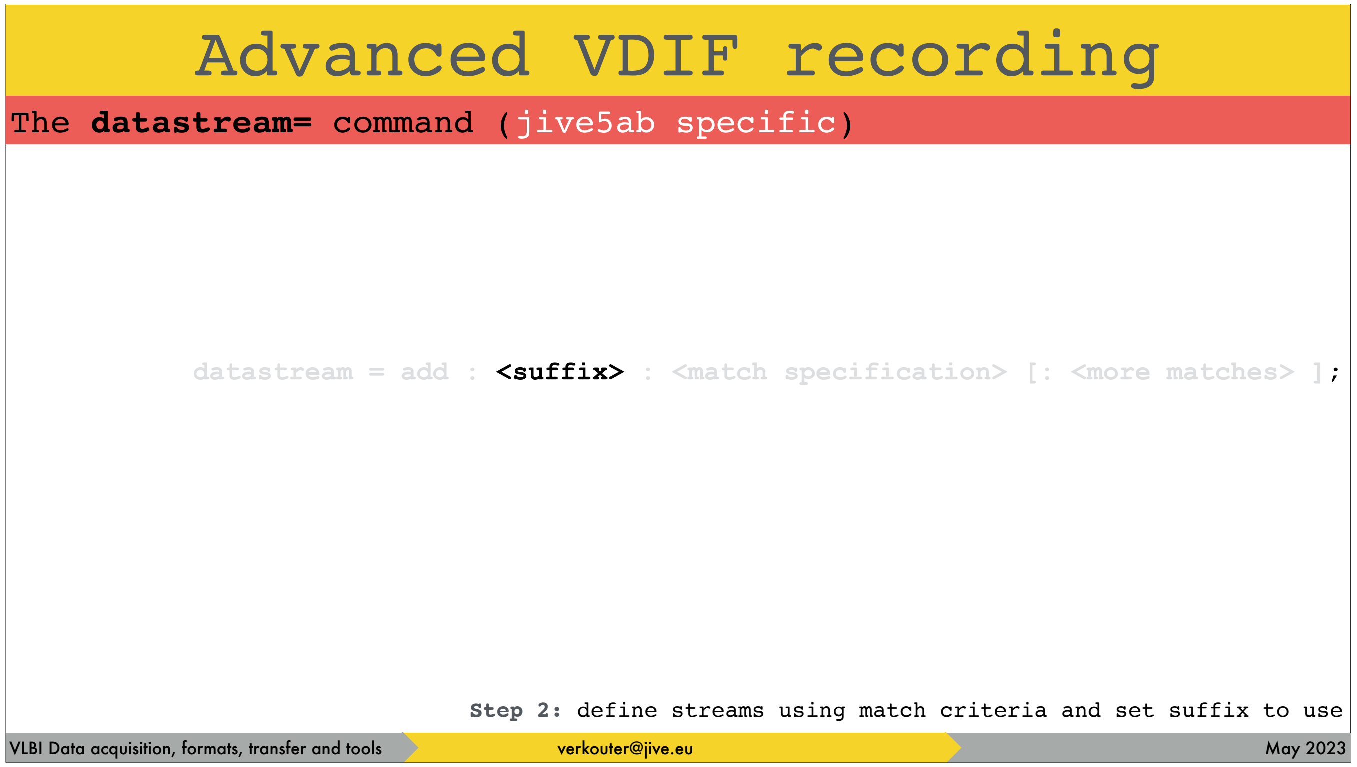
Advanced VDIF recording

The **datastream=** command (jive5ab specific)

```
datastream = add : <suffix> : <match specification> [: <more matches> ];
```

Step 2: define streams using match criteria and set suffix to use

which frames you want to collect



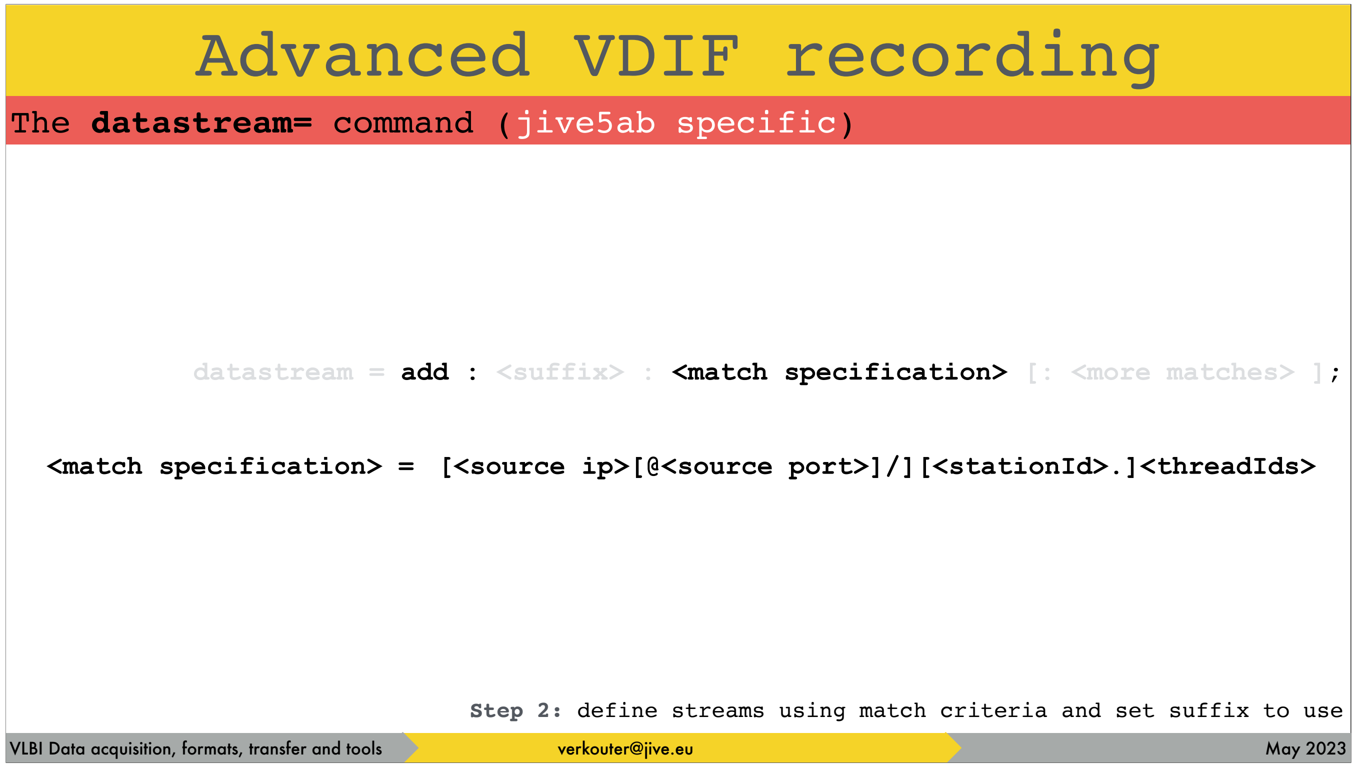
Advanced VDIF recording

The **datastream=** command (jive5ab specific)

```
datastream = add : <suffix> : <match specification> [: <more matches> ];
```

Step 2: define streams using match criteria and set suffix to use

and put into the same recording



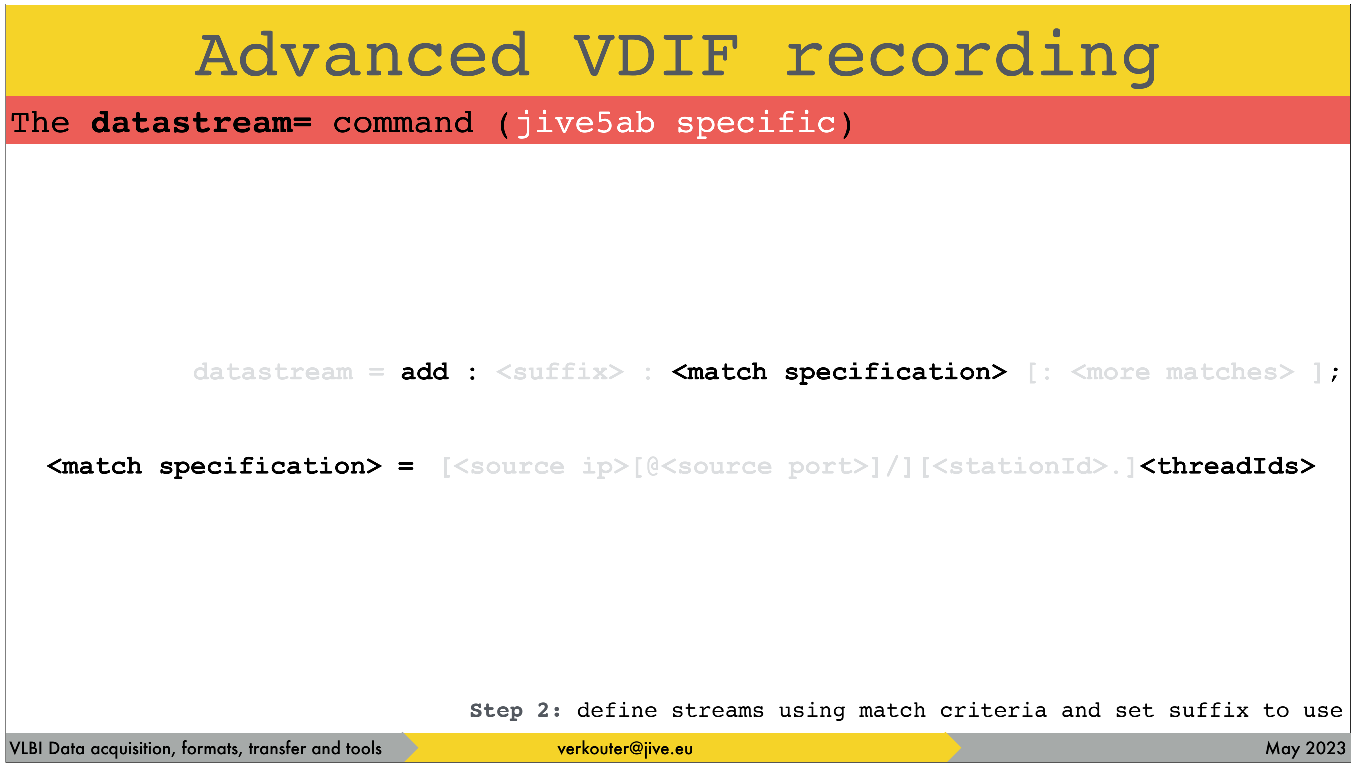
Advanced VDIF recording

The **datastream=** command (jive5ab specific)

```
datastream = add : <suffix> : <match specification> [: <more matches> ];

<match specification> =  [<source ip>[@<source port>]/][<stationId>.]<threadIds>
```

Step 2: define streams using match criteria and set suffix to use

A match specification is a filter expression which looks daunting



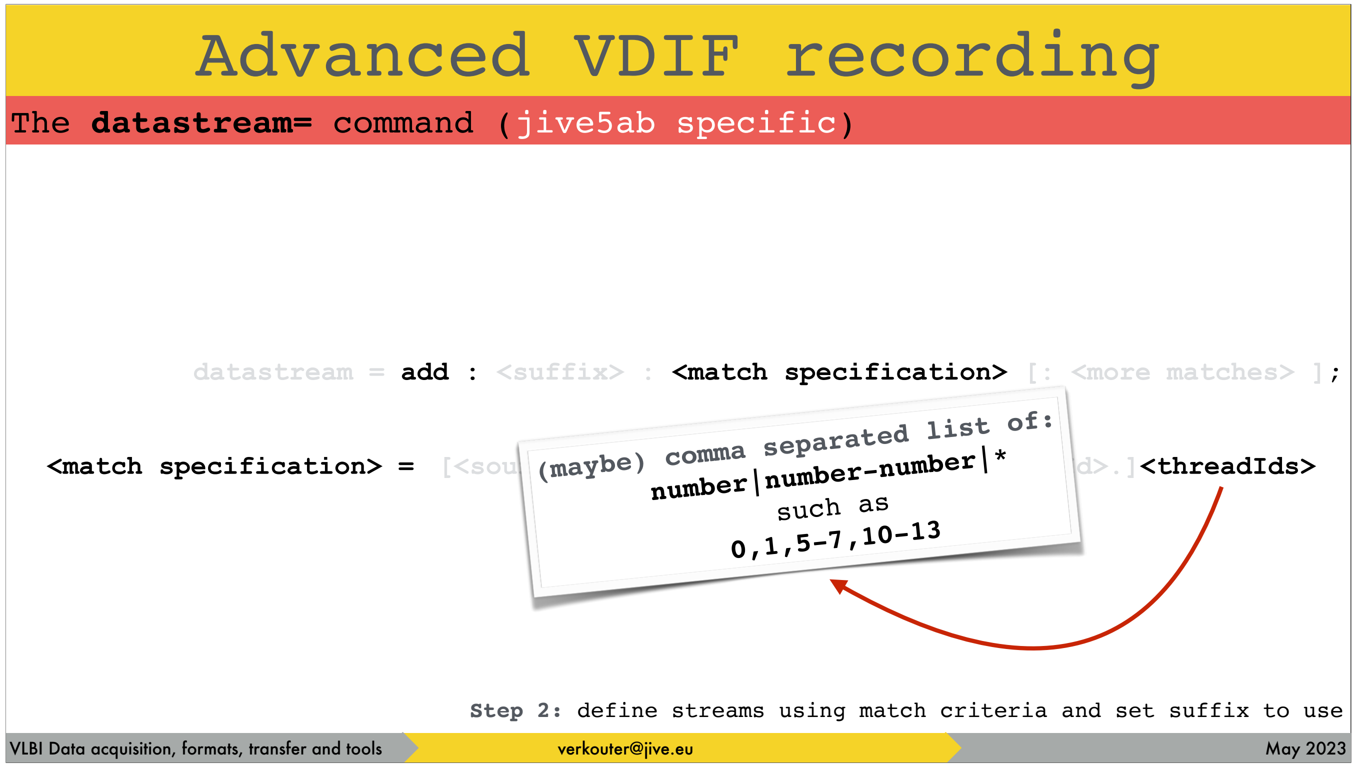
Advanced VDIF recording

The **datastream=** command (jive5ab specific)

```
datastream = add : <suffix> : <match specification> [: <more matches> ];

<match specification> =  [<source ip>[@<source port>]/][<stationId>.]<threadIds>
```

Step 2: define streams using match criteria and set suffix to use

but there is only one _required_ field, which thread id's to match



Advanced VDIF recording

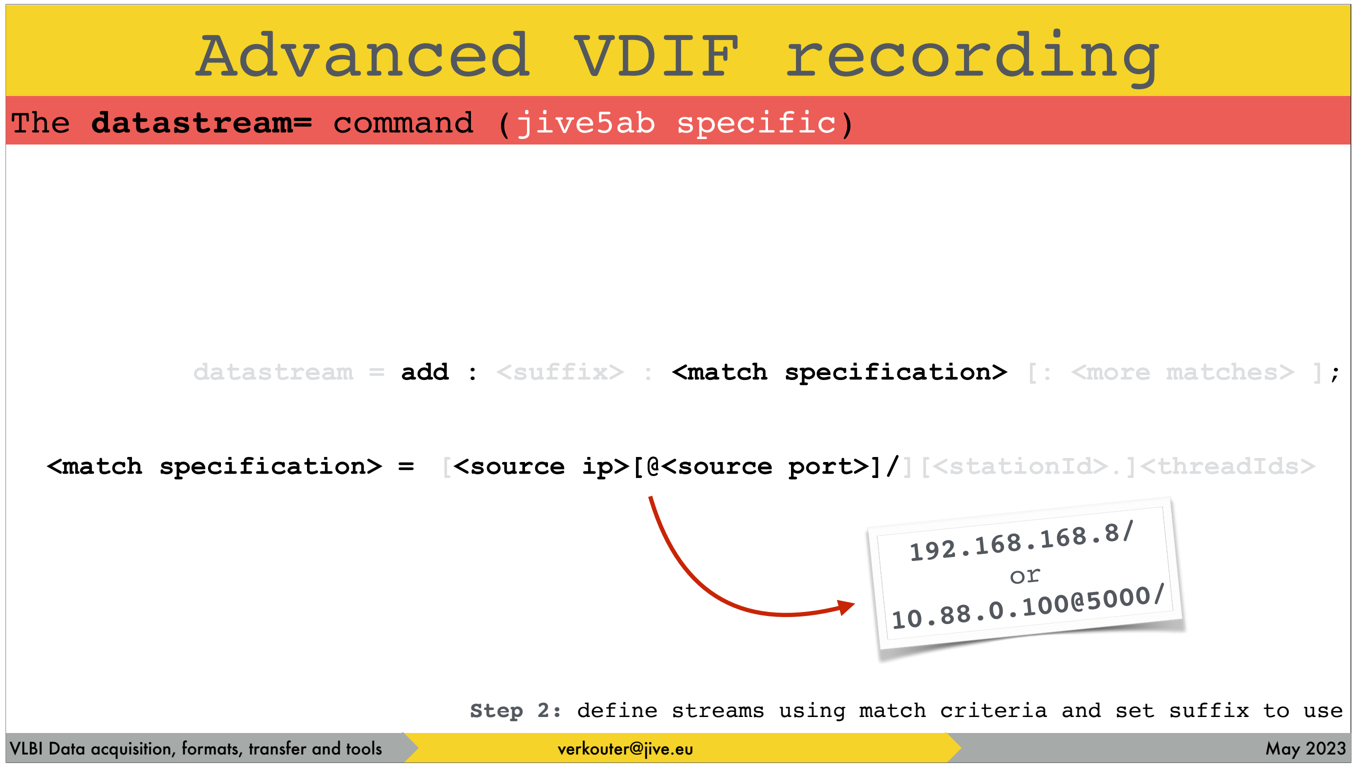
The **datastream=** command (jive5ab specific)

```
datastream = add : <suffix> : <match specification> [: <more matches> ] ;

<match specification> =  [<sou                d>.]<threadIds>
```

(maybe) comma separated list of:
number|number-number|*
such as
0,1,5-7,10-13

Step 2: define streams using match criteria and set suffix to use

and its format is quite simple - just numbers, ranges of numbers, or the asterisk which means “any thread id”



Advanced VDIF recording

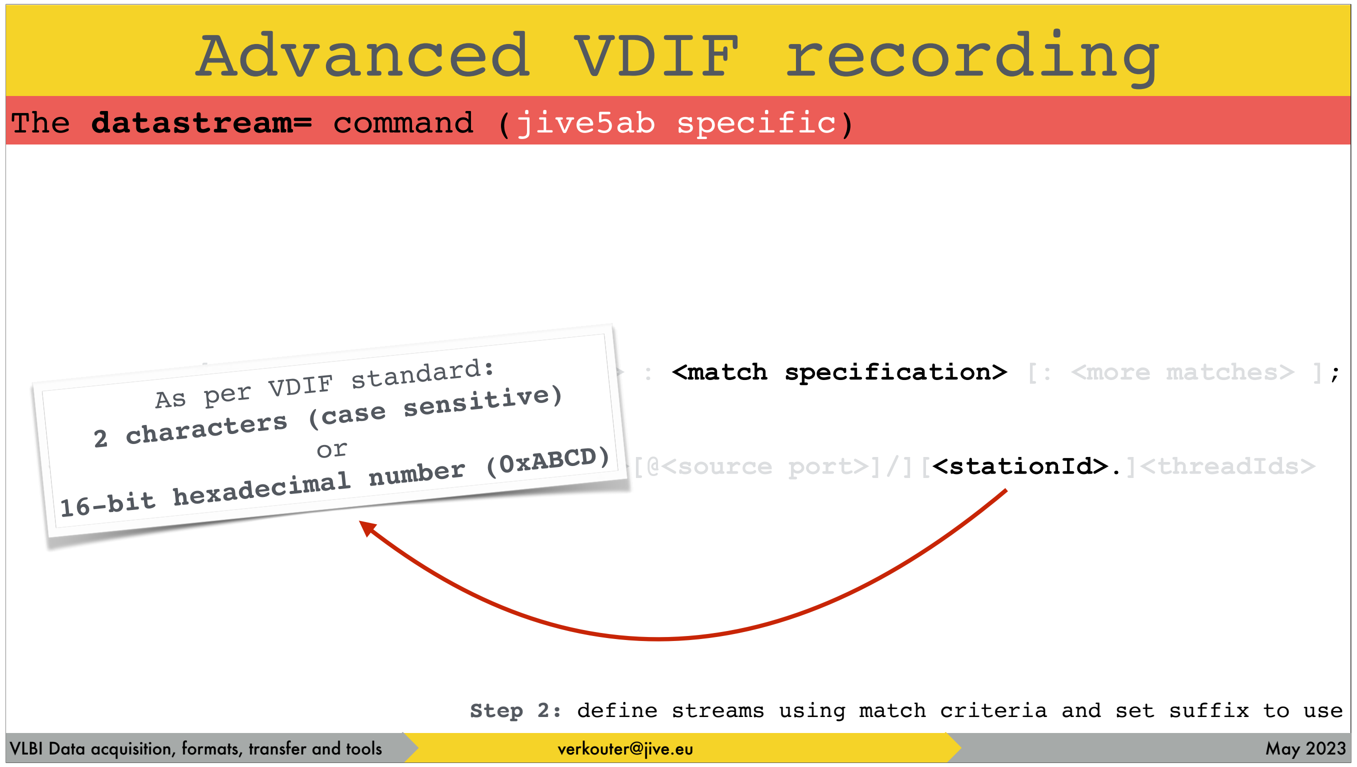
The **datastream=** command (jive5ab specific)

```
datastream = add : <suffix> : <match specification> [: <more matches> ];

<match specification> =  [<source ip>[@<source port>]/] [<stationId>.]<threadIds>
```

192.168.168.8/
or
10.88.0.100@5000/

Step 2: define streams using match criteria and set suffix to use

this part of the match specification should be self-explanatory: filter on which IP address sent the frame



Advanced VDIF recording

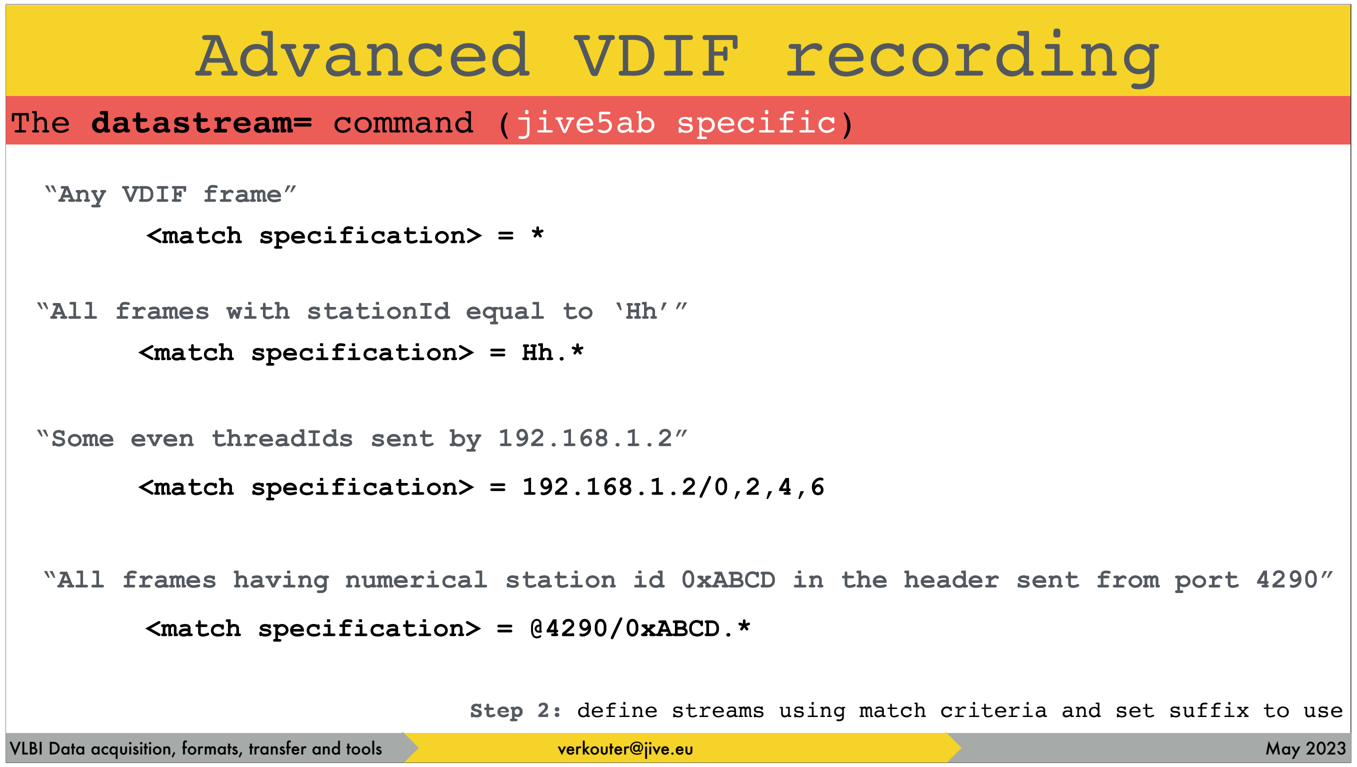
The **datastream=** command (jive5ab specific)

```
> : <match specification> [: <more matches> ];
[@<source port>]/][<stationId>.]<threadIds>
```

As per VDIF standard:
2 characters (case sensitive)
or
16-bit hexadecimal number (0xABCD)

Step 2: define streams using match criteria and set suffix to use

and then there is the possibility to filter on the station id from the VDIF header



Advanced VDIF recording

The **datastream=** command (jive5ab specific)

"Any VDIF frame"

```
<match specification> = *
```

"All frames with stationId equal to 'Hh'"

```
<match specification> = Hh.*
```

"Some even threadIds sent by 192.168.1.2"

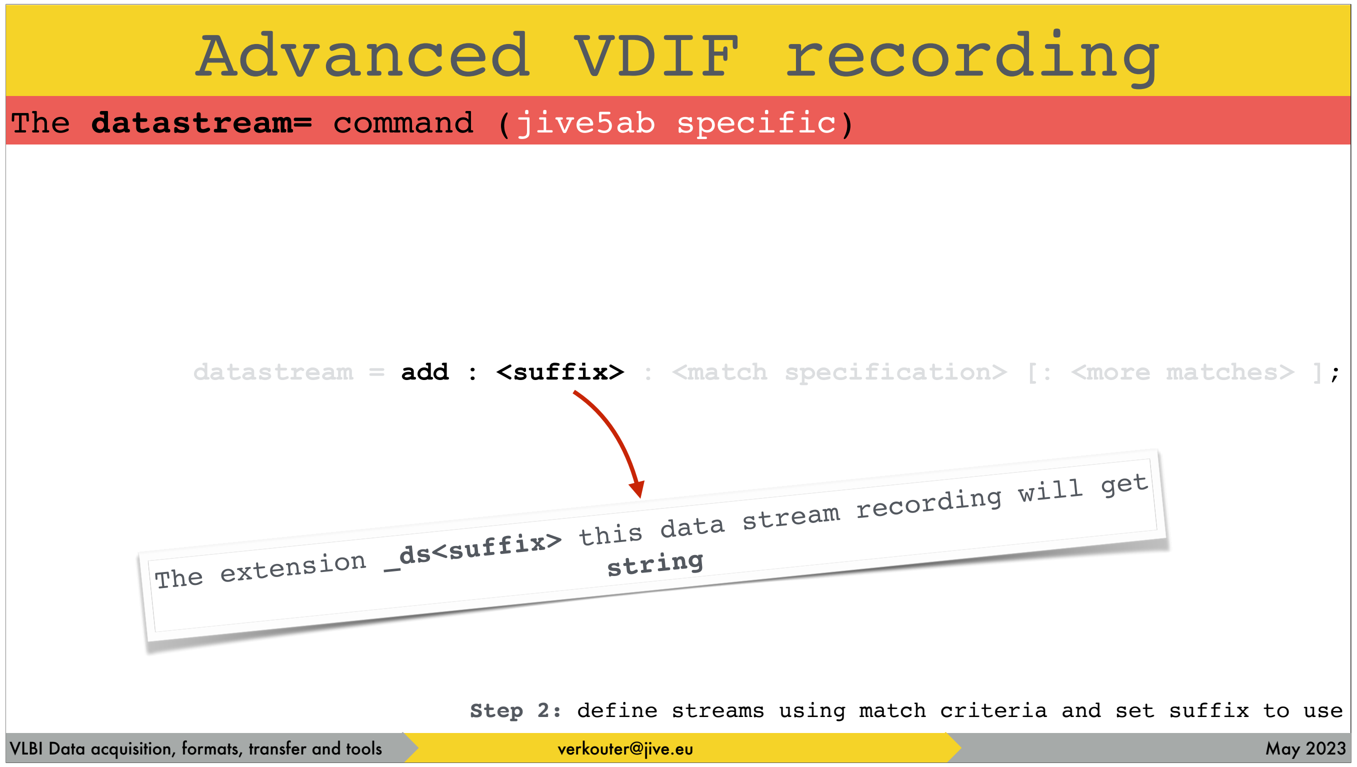
```
<match specification> = 192.168.1.2/0,2,4,6
```

"All frames having numerical station id 0xABCD in the header sent from port 4290"

```
<match specification> = @4290/0xABCD.*
```

Step 2: define streams using match criteria and set suffix to use

some examples of how to translate question into a match specification



Advanced VDIF recording

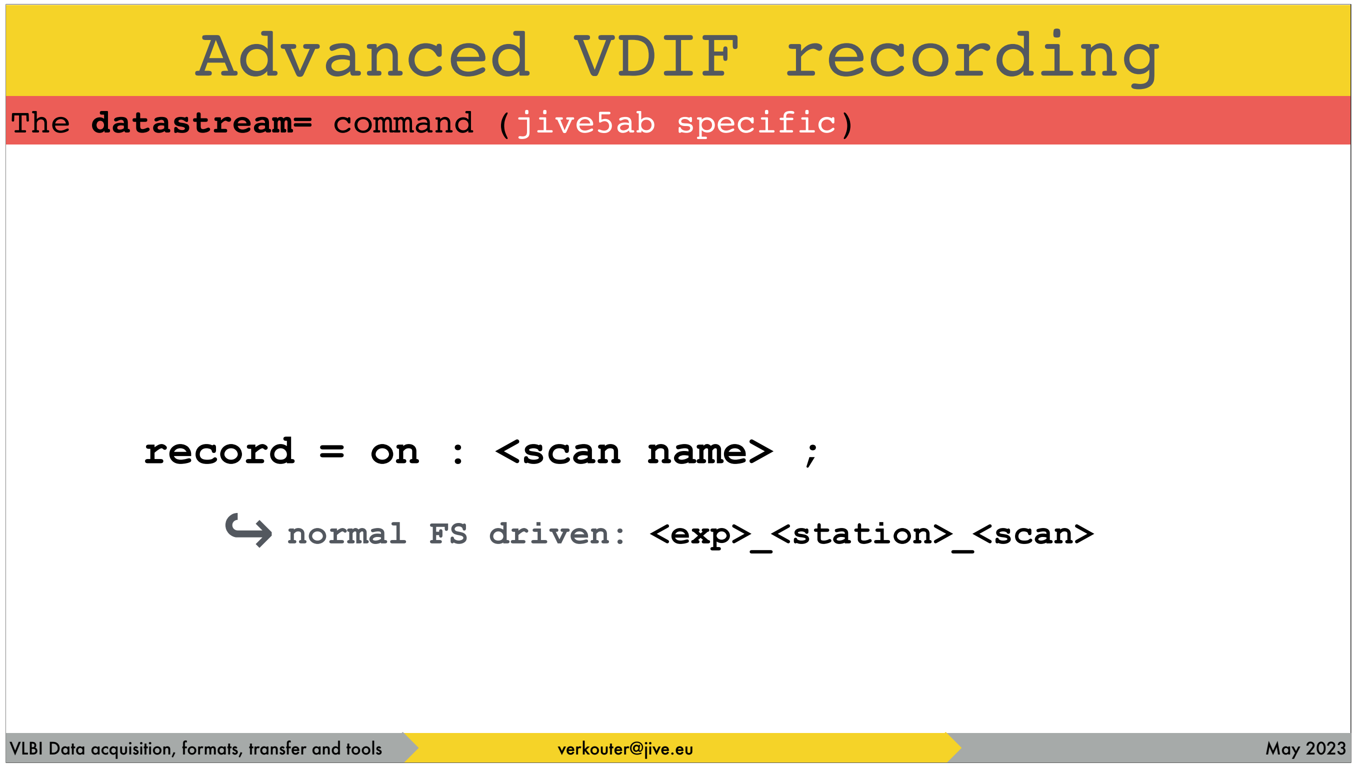
The **datastream=** command (jive5ab specific)

```
datastream = add : <suffix> : <match specification> [: <more matches> ] ;
```

The extension **_ds<suffix>** this data stream recording will get
string

Step 2: define streams using match criteria and set suffix to use

The suffix part of the add command determines the suffix the recording will get that contains the matching VDIF frames



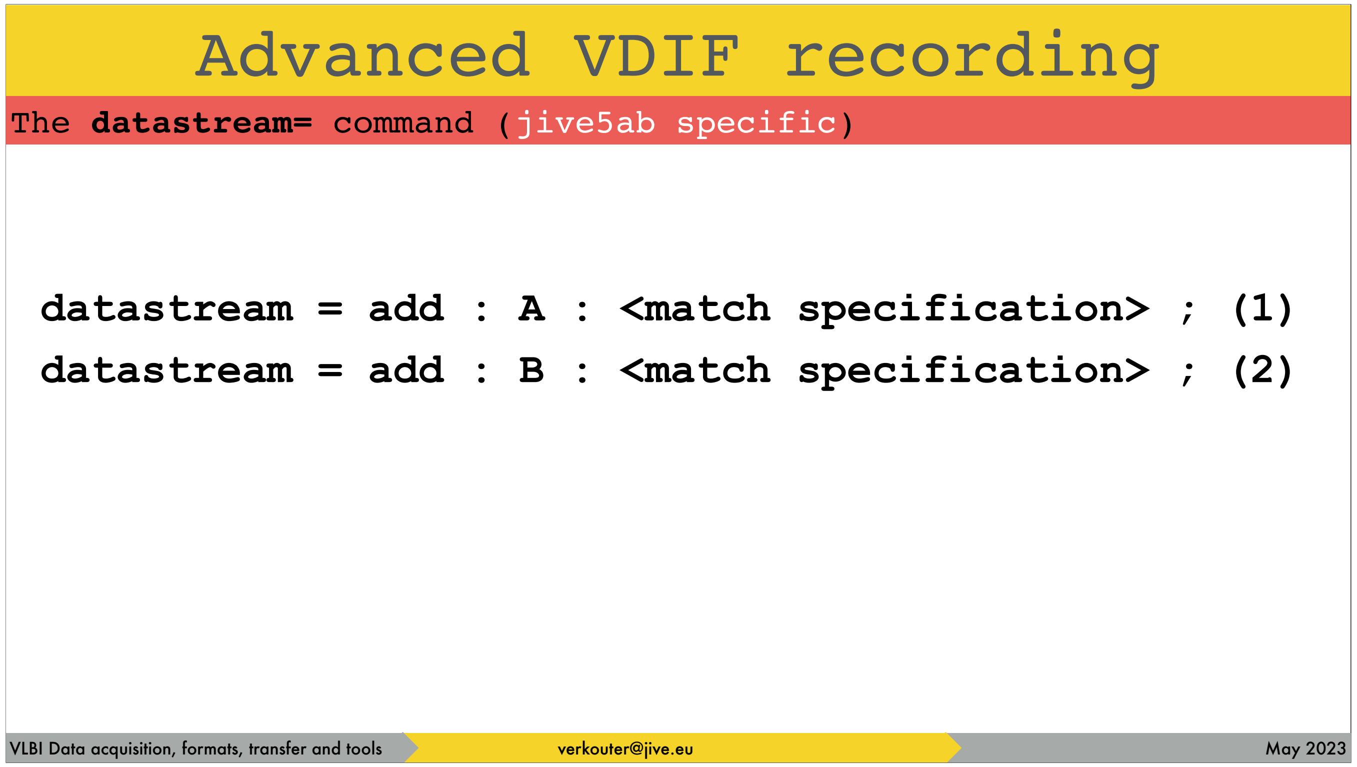
Advanced VDIF recording

The **datastream=** command (jive5ab specific)

```
record = on : <scan name> ;
```

↪ normal FS driven: **<exp>_<station>_<scan>**

during normal operations the standard recording name is exp_station_scan

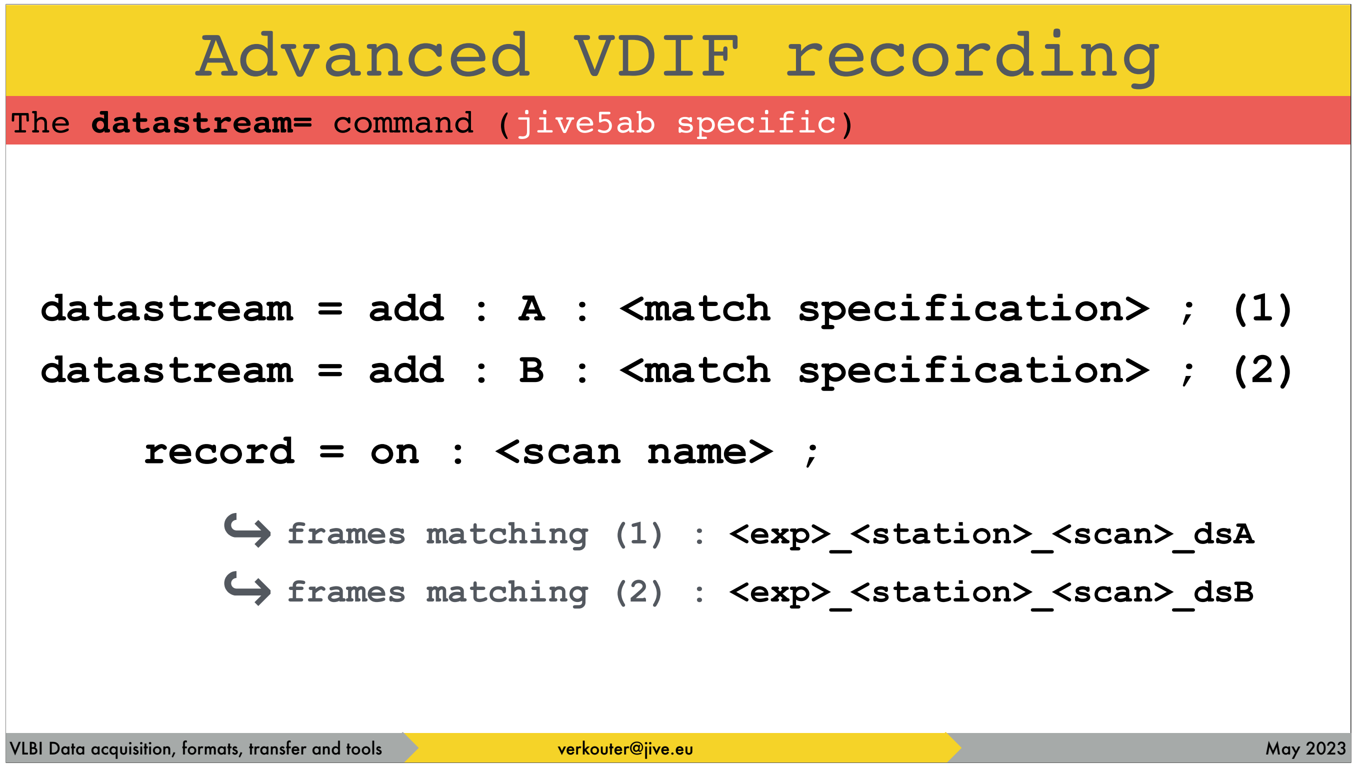


Advanced VDIF recording

The **datastream=** command (jive5ab specific)

```
datastream = add : A : <match specification> ; (1)
datastream = add : B : <match specification> ; (2)
```

but if datastreams were enabled

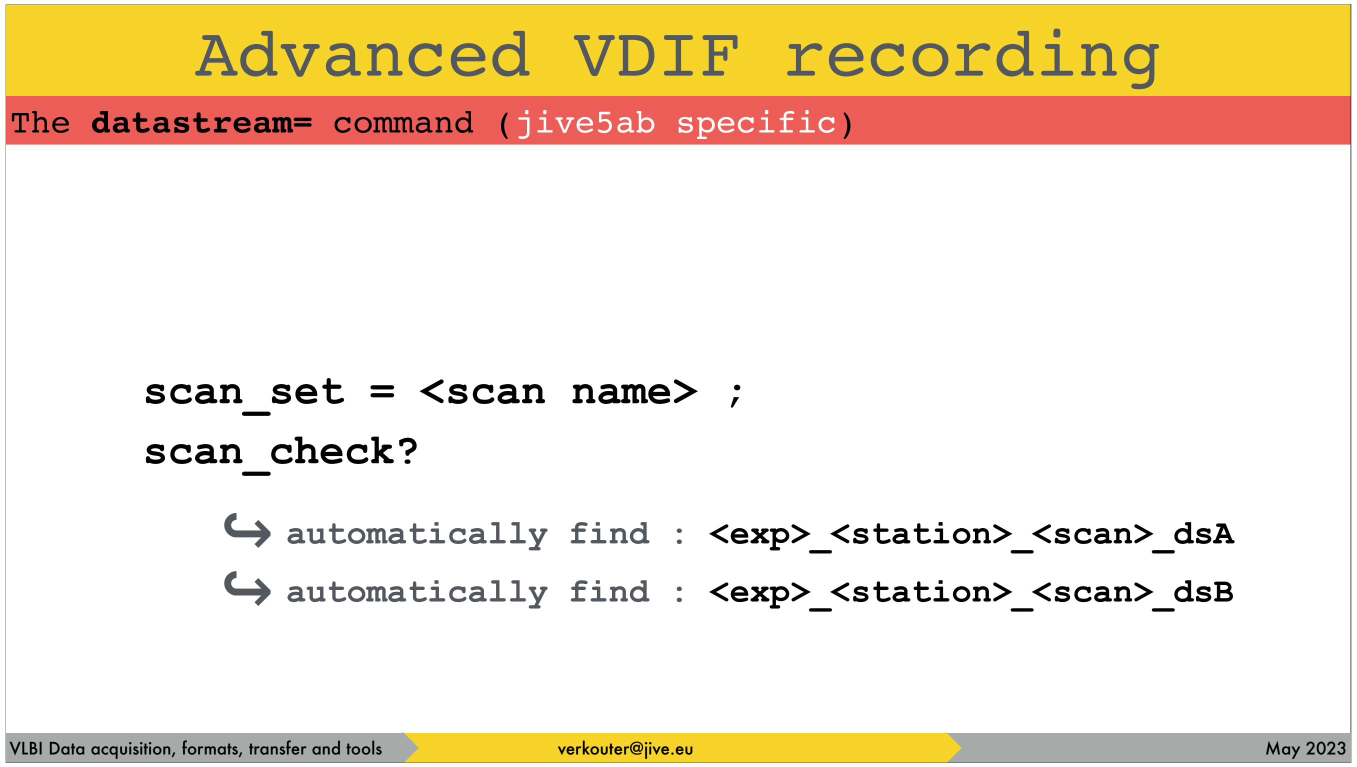


Advanced VDIF recording

The **datastream=** command (jive5ab specific)

```
datastream = add : A : <match specification> ; (1)
datastream = add : B : <match specification> ; (2)

     record = on : <scan name> ;

         ↪ frames matching (1) : <exp>_<station>_<scan>_dsA
         ↪ frames matching (2) : <exp>_<station>_<scan>_dsB
```

and then recording is turned on, the frames are collected into differently named recordings

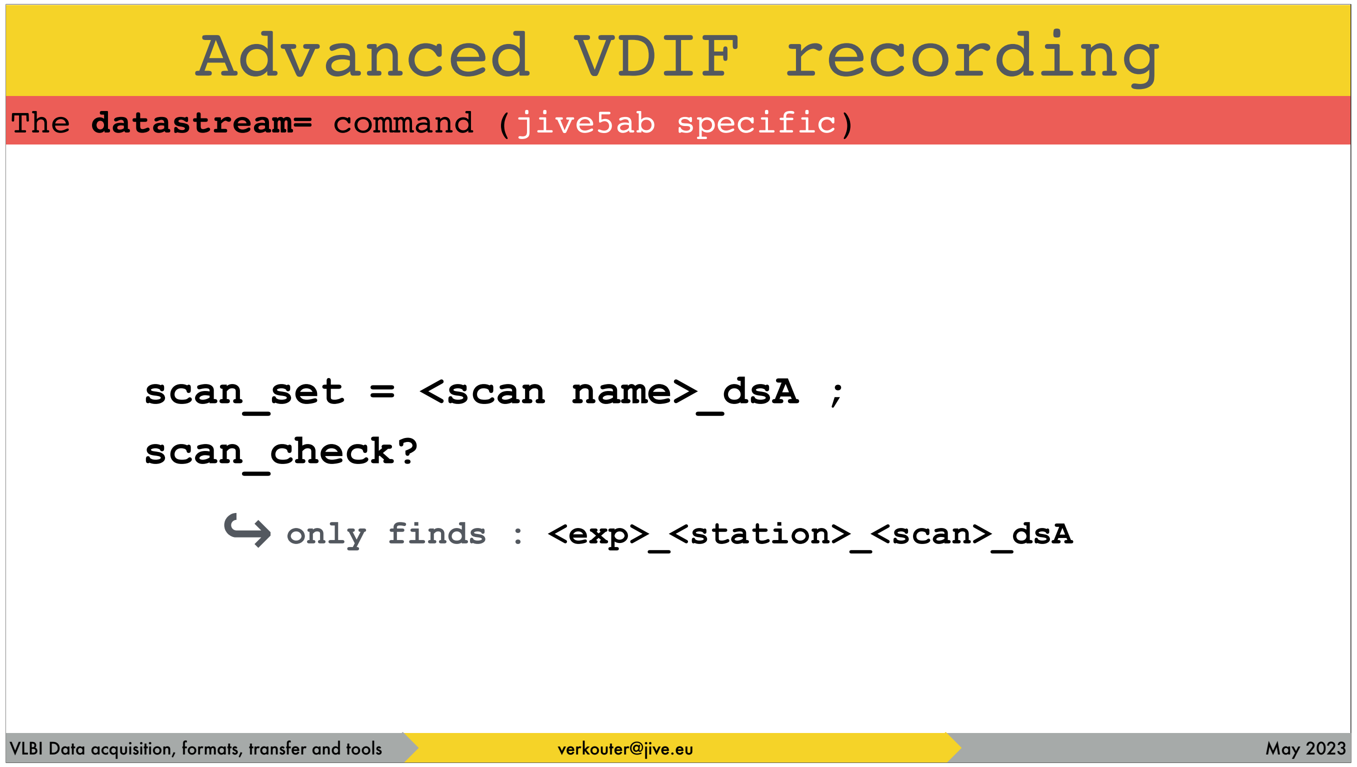


Advanced VDIF recording

The **datastream=** command (jive5ab specific)

```
scan_set = <scan name> ;
scan_check?
    ↪ automatically find : <exp>_<station>_<scan>_dsA
    ↪ automatically find : <exp>_<station>_<scan>_dsB
```

it is important to know that these commands have been updated to automatically find the separate recordings

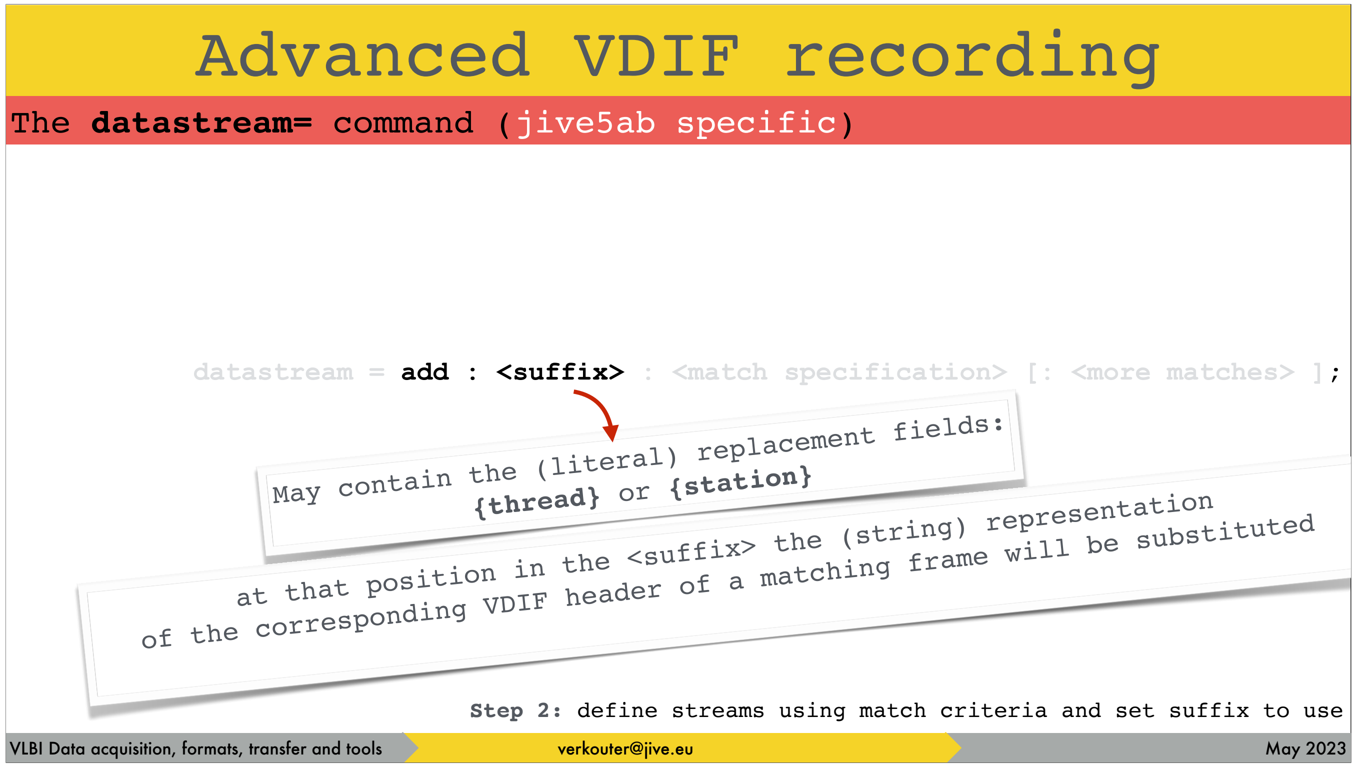


Advanced VDIF recording

The **datastream=** command (jive5ab specific)

```
scan_set = <scan name>_dsA ;
scan_check?
    ↪ only finds : <exp>_<station>_<scan>_dsA
```

but because the recordings are complete recordings by themselves this also works



Advanced VDIF recording

The **datastream=** command (jive5ab specific)

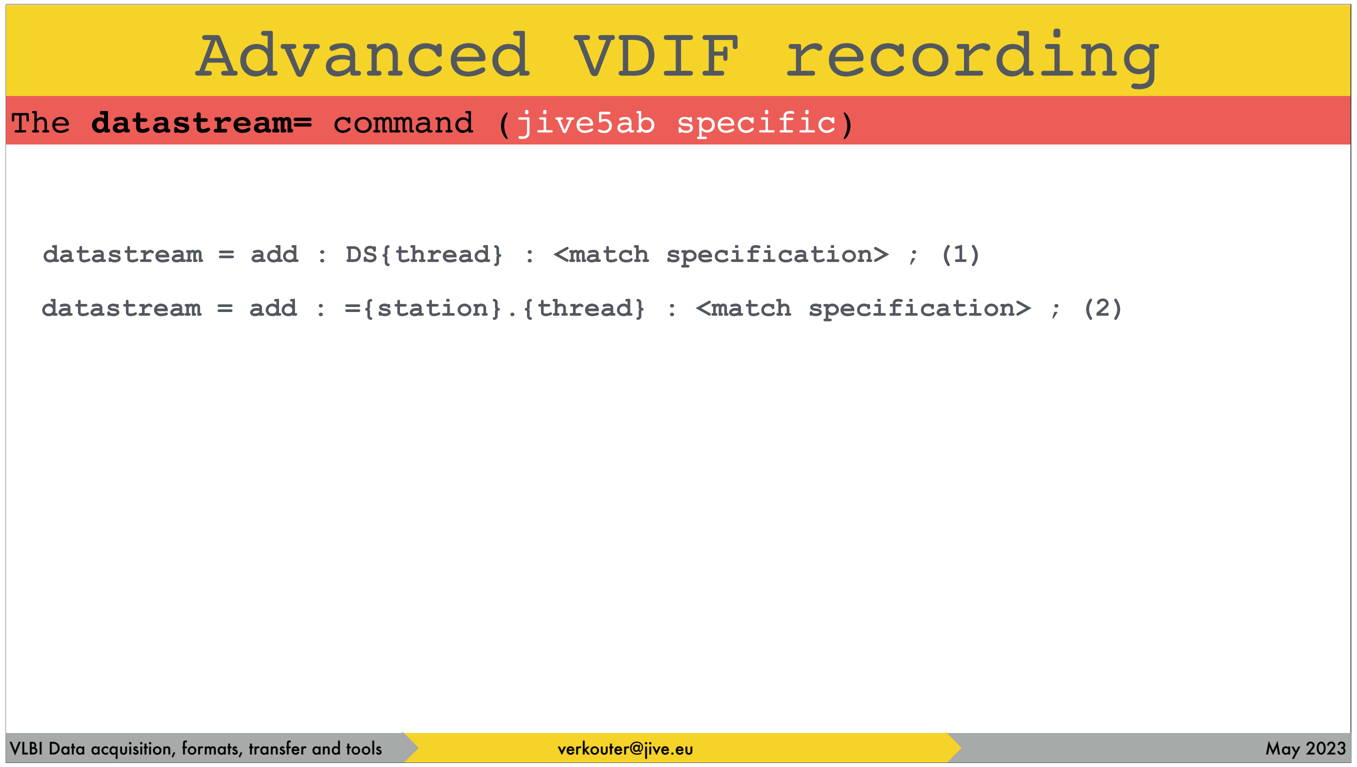
```
datastream = add : <suffix> : <match specification> [: <more matches> ] ;
```

May contain the (literal) replacement fields:
{thread} or **{station}**

at that position in the <suffix> the (string) representation
of the corresponding VDIF header of a matching frame will be substituted

Step 2: define streams using match criteria and set suffix to use

an interesting feature of the suffix string is that it may contain *replacement fields*. What it means is that the code will insert the string representation of the value in a matching VDIF frame

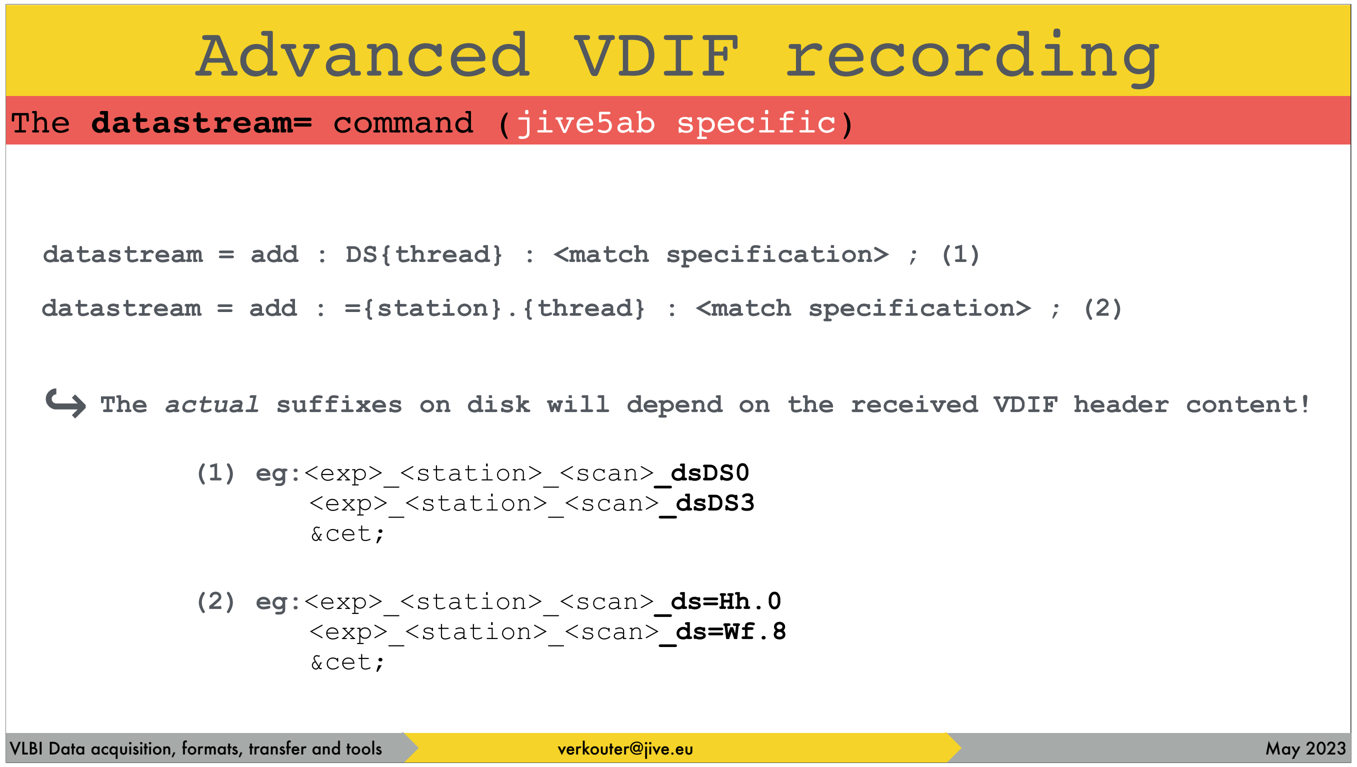


Advanced VDIF recording

The **datastream=** command (jive5ab specific)

```
datastream = add : DS{thread} : <match specification> ; (1)
datastream = add : ={station}.{thread} : <match specification> ; (2)
```

if we look for example at these datastream definitions, and VDIF frames come in



Advanced VDIF recording

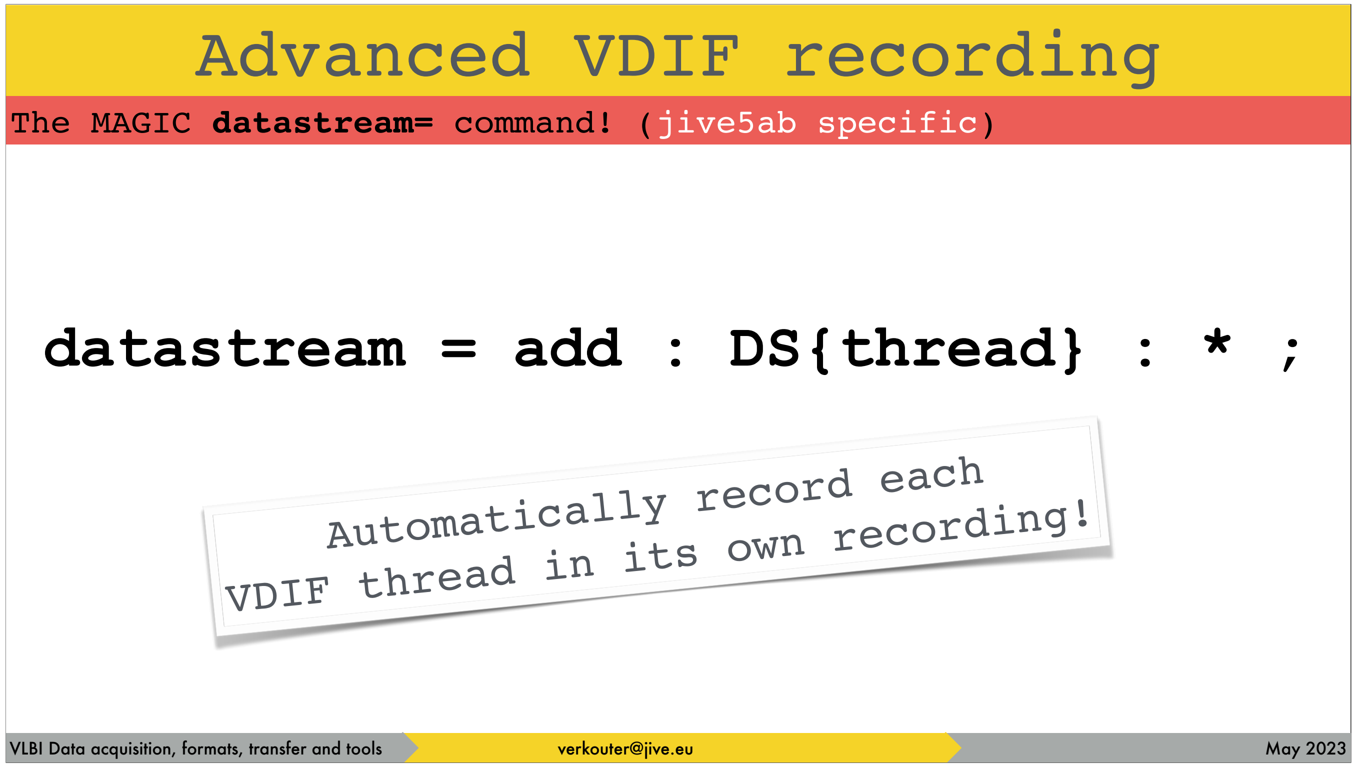
The **datastream=** command (jive5ab specific)

```
datastream = add : DS{thread} : <match specification> ; (1)

datastream = add : ={station}.{thread} : <match specification> ; (2)
```

↪ The *actual* suffixes on disk will depend on the received VDIF header content!

```
(1) eg:<exp>_<station>_<scan>_dsDS0
       <exp>_<station>_<scan>_dsDS3
       &cet;

(2) eg:<exp>_<station>_<scan>_ds=Hh.0
       <exp>_<station>_<scan>_ds=Wf.8
       &cet;
```

then the suffixes on disk will depend on what is in the received VDIF frame headers



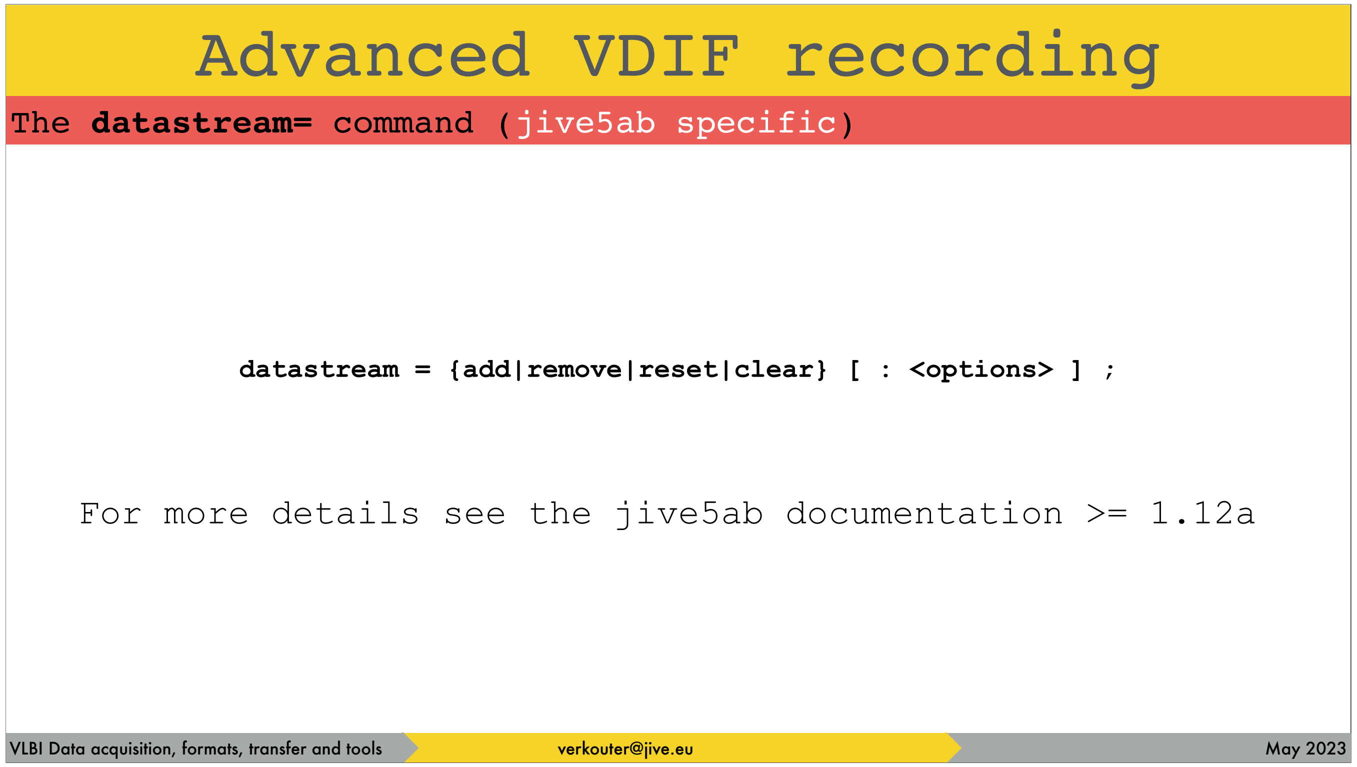
Advanced VDIF recording

The MAGIC **datastream=** command! (jive5ab specific)

```
datastream = add : DS{thread} : * ;
```

Automatically record each VDIF thread in its own recording!

And this single data stream definition is extremely powerful: it records each VDIF thread automatically in its own recording!



Advanced VDIF recording

The **datastream=** command (jive5ab specific)

```
datastream = {add|remove|reset|clear} [ : <options> ] ;
```

For more details see the jive5ab documentation >= 1.12a

And you can read the rest back in the documentation.

Thanks

for Extra!

attention!

and with that I’d like to thank you for you extra attention.