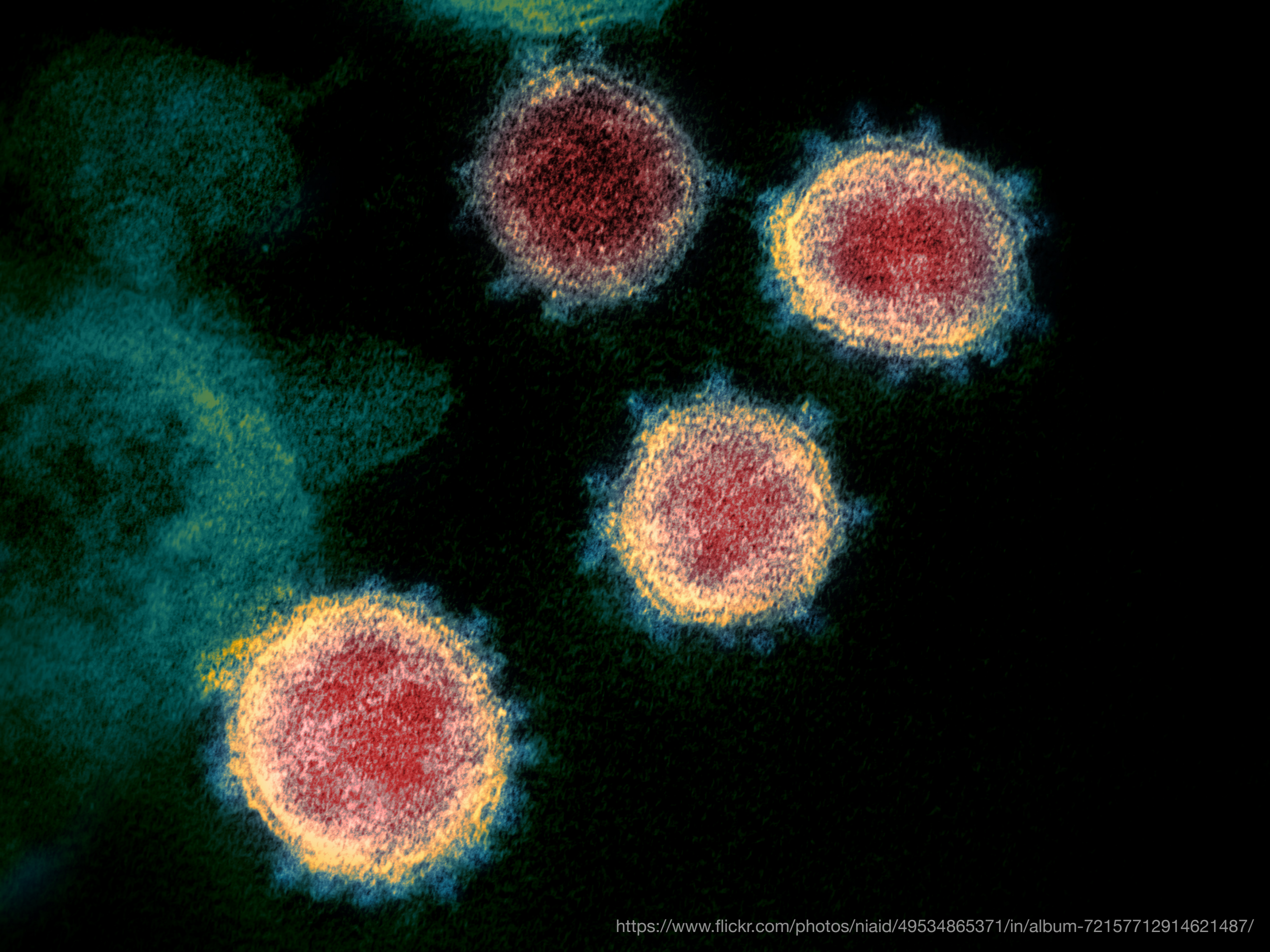


No official release since 2.9.0
April 2018

No official release since 2.9.0
April 2018

3.0 release candidate to testers
March 2020

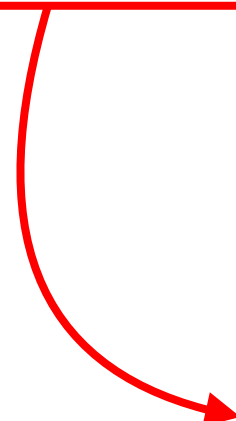




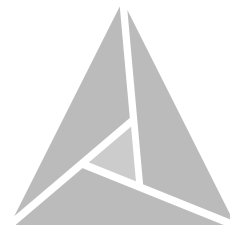
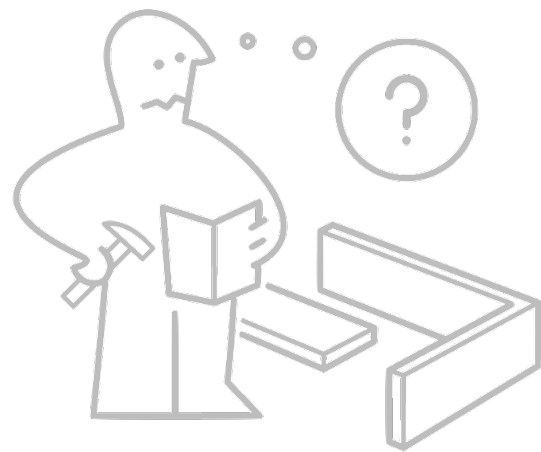
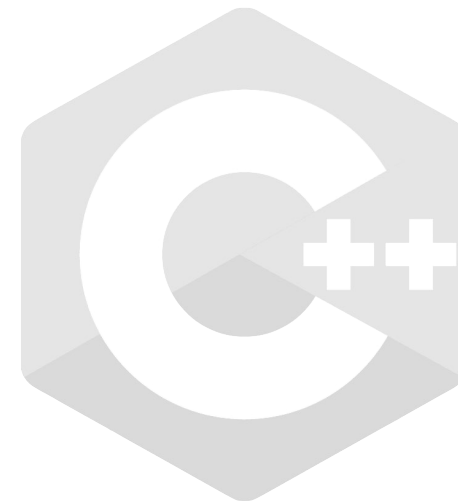
SARS-CoV-2 virus particule

No official release since 2.9.0
April 2018

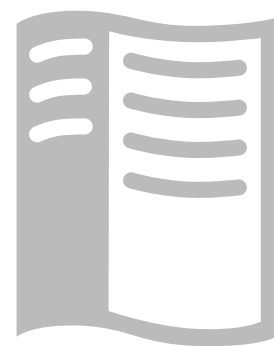
3.0 release candidate to testers
March 2020



*due to COVID-19 never
made it to release*

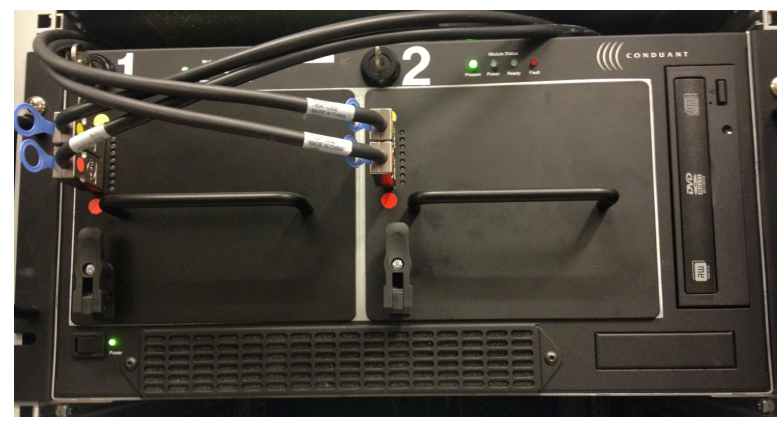


CMake



Read *the* Docs

Improvements in high-speed data transfer and recording with **jive5ab**

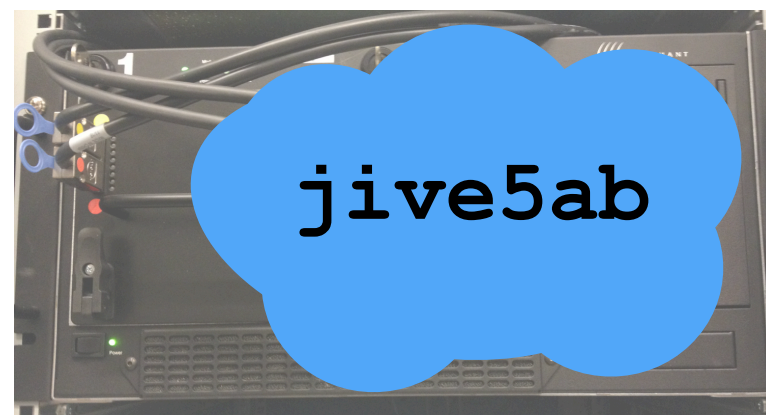
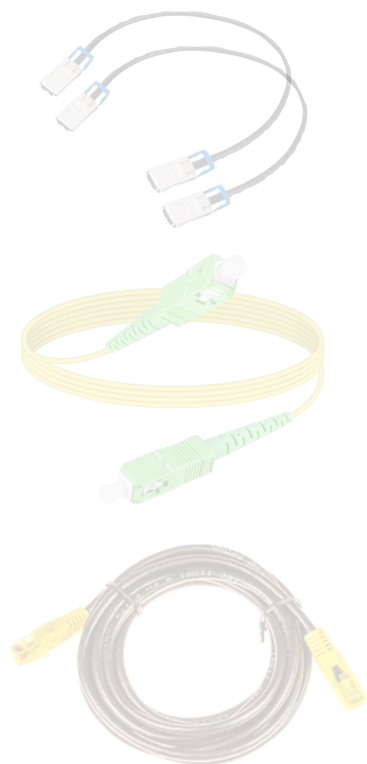


telescope

digital backend(s)

ethernet link(s)

VLBI data recorder



telescope

digital backend(s)

ethernet link(s)

VLBI data recorder

Installation of a pre-compiled version is simple: download the .deb and then, as **root** run:

```
# dpkg -i </path/to/the/.deb>
```

Note: since **2.4.0** the Debian package name has changed from "jive5a" to "jive5ab". It may be necessary to to remove the "jive5a" first, before installing **2.4.0** or **higher**:

```
# dpkg -r jive5a
```

2.9.0 for **Debian Etch** based Mark5A/A+/B/B+/C 's with **SDK9.2.1** or **64bit Debian Wheezy** based Mark5 A/A+/B/B+/C with **SDK9.4**, and **source code** for systems not fitting these parameters.

This release contains code fixes to eliminate compiler warnings (cum errors) on new(er), stricter, standards-enforcing compilers e.g. found in Debian 9.2 and up or GCC7.*.

New features added: remote mount/unmount of Mark5 disk packs as if the disk pack was keyed on/off.

Finer grained hardware detection+checking to support 'crippled' Mark5's - systems with only a StreamStor card. All disk-based functionality remains working but recording (you need an I/O board or daughter card for that).

Recording data rate check implemented to prevent data corruption.

For full details see [the changelog](#)

SDK9.2.1 version: [jive5ab-2.9.0-ETCH-SDK921-i386.deb](#)

64 bit SDK9.4 version: [jive5ab-2.9.0-WHEEZY-SDK94-amd64.deb](#)

Source code: [jive5ab-2.9.0.tar.gz](#)

2.8.1 for **Debian Etch** based Mark5A/A+/B/B+/C 's with **SDK9.2.1** or **64bit Debian Wheezy** based Mark5 A/A+/B/B+/C with **SDK9.4**, **64bit Debian Wheezy** based FlexBuff/Mark6 systems (**no StreamStor dependency**) and **source code** for systems not fitting these parameters.

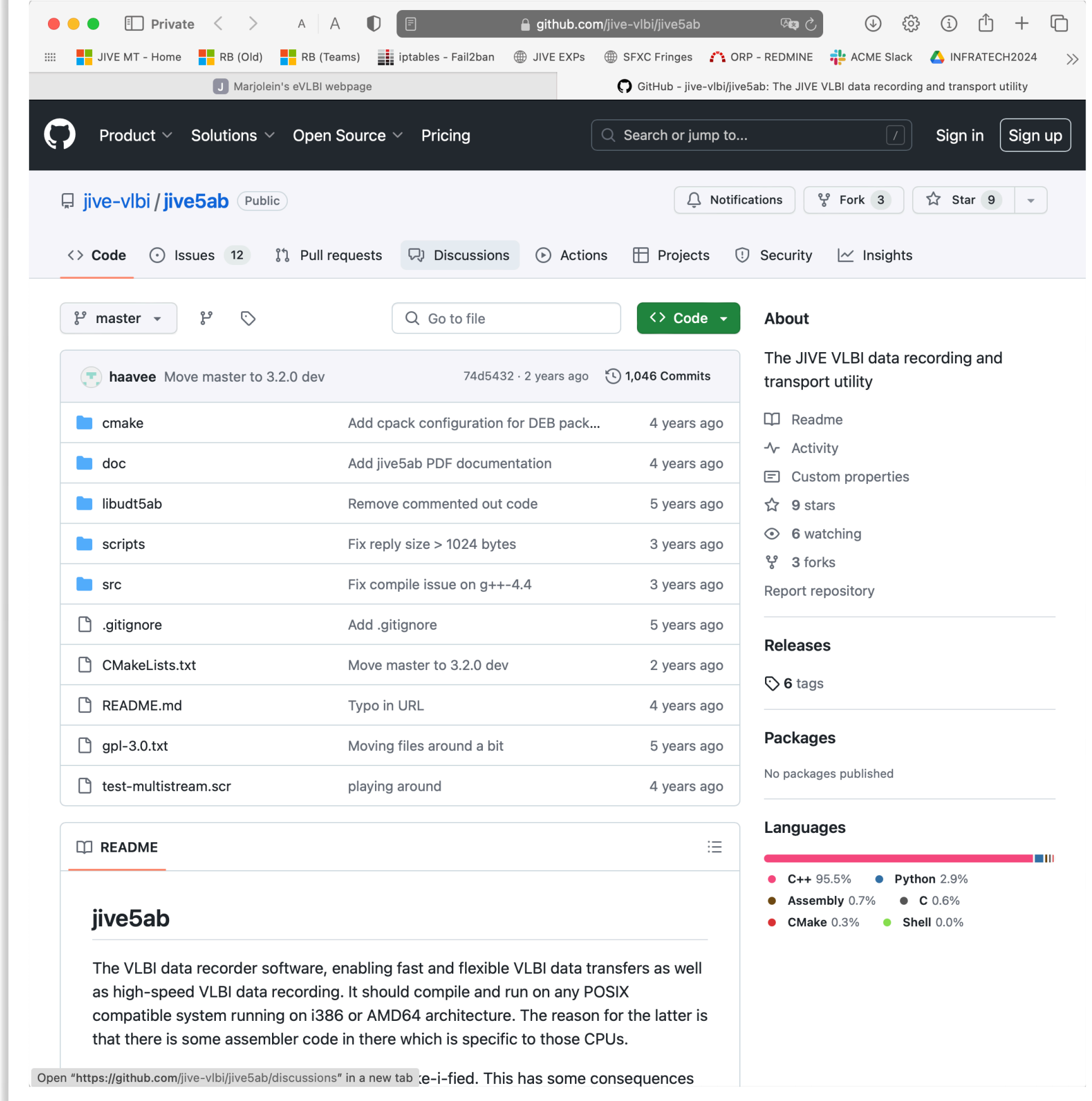
This is a bugfix update for 2.8.0, no new developments. Important fixes include but are not limited to:

There is a multi-thread race condition lurking in FlexBuff "disk2file" and "disk2net" so it is advised to upgrade for non-crashing behaviour.

This version can erase empty disk packs once again (2.8.0 lost this due to overfocus on non-bank mode support)

The "record?" reply on FlexBuff was so broken by 2.8.0 that the Field System correctly complains loudly and bitterly

<https://www.jive.eu/~verkout/> ...



The screenshot shows the GitHub repository page for `jive-vlbi/jive5ab`. The repository is public and has 12 issues, 3 forks, and 9 stars. The main branch is `master`. The repository description is "The JIVE VLBI data recording and transport utility".

Files and Commits:

File	Commit Message	Time Ago
<code>cmake</code>	Add cpack configuration for DEB pack...	4 years ago
<code>doc</code>	Add jive5ab PDF documentation	4 years ago
<code>libudt5ab</code>	Remove commented out code	5 years ago
<code>scripts</code>	Fix reply size > 1024 bytes	3 years ago
<code>src</code>	Fix compile issue on g++-4.4	3 years ago
<code>.gitignore</code>	Add .gitignore	5 years ago
<code>CMakeLists.txt</code>	Move master to 3.2.0 dev	2 years ago
<code>README.md</code>	Typo in URL	4 years ago
<code>gpl-3.0.txt</code>	Moving files around a bit	5 years ago
<code>test-multistream.scr</code>	playing around	4 years ago

README:

jive5ab

The VLBI data recorder software, enabling fast and flexible VLBI data transfers as well as high-speed VLBI data recording. It should compile and run on any POSIX compatible system running on i386 or AMD64 architecture. The reason for the latter is that there is some assembler code in there which is specific to those CPUs.

Open "https://github.com/jive-vlbi/jive5ab/discussions" in a new tab e-i-fied. This has some consequences

About: The JIVE VLBI data recording and transport utility

- Readme
- Activity
- Custom properties
- 9 stars
- 6 watching
- 3 forks

Report repository

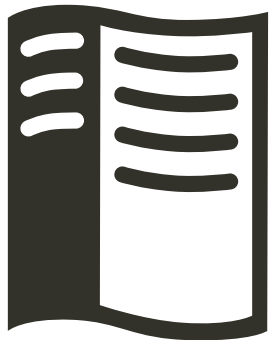
Releases: 6 tags

Packages: No packages published

Languages:

- C++ 95.5%
- Python 2.9%
- Assembly 0.7%
- C 0.6%
- CMake 0.3%
- Shell 0.0%

`https://github.com/jive-vlbi/jive5ab.git`



Private | jive5ab.readthedocs.io/en/latest/ | JIVE MT - Home | RB (Old) | RB (Teams) | iptables - Fail2ban | JIVE EXPs | SFXC Fringes | ORP - REDMINE | ACME Slack | INFRATECH2024

Jive5ab documents
latest

Search docs

CONTENTS:

- jive5ab program
 - Running the jive5ab program
 - Notes on jive5ab command set
- VSI-S Command, Query and Response Syntax
- jive5ab Data Transfer Modes explained
 - Multiple independent Data Transfers
- Data format support across the range of systems
- channel dropping and corner turning
- Comments on Record-, Start-scan- and Stop-scan Pointer
- Scan names, Scan Labels and Linux filenames
- Non-bank mode operation
- FlexBuff/Mark6 operations
- jive5ab Command/Query Summary (by category)
- jive5ab command set

Kong API Gateway: An API gateway built on Nginx to handle 50k microservice reqs/s **Try it for free!**
Ad by EthicalAds

Read the Docs | v: latest

Welcome to Jive5ab documents's documentation! [Edit on GitHub](#)

Welcome to Jive5ab documents's documentation!

Contents:

- jive5ab program
 - JOINT INSTITUTE FOR VLBI IN EUROPE
 - jive5ab program
- Running the jive5ab program
- Notes on jive5ab command set
- VSI-S Command, Query and Response Syntax
 - Command Syntax
 - Command-Response Syntax
 - Query and Query-Response Syntax
- jive5ab Data Transfer Modes explained
- Multiple independent Data Transfers
 - Implementation note — the concept of 'runtimes'
 - Configuring (a) runtime(s)
 - Running a network transfer
- Data format support across the range of systems
 - Configuring data formats
- channel dropping and corner turning
 - Channel dropping
 - Corner turning
- Comments on Record-, Start-scan- and Stop-scan Pointer
 - Record pointer
 - Start-scan and Stop-scan pointers
- Scan names, Scan Labels and Linux filenames
- Non-bank mode operation
 - Introduction
 - Notes on firmware limitation from an application perspective

https://jive5ab.readthedocs.io/en/latest/

谢谢 郭绍光！

many thanks to Shaoguang Guo at SHAO!

on current disk set (query only)
(5B/DIM, 5C)

rtime – Get remaining record time
on current disk set (query only) (G)

runtime – Control multiple
simultaneous transfer environments

scan_check – Check recorded data
between start-scan and stop-scan
pointers (query only)

scan_set – Set start-scan and stop-
scan pointers for data readback

set_disks – Select mount points to
record on (FlexBuff/Mark6)

sp*2* – Configure, start, stop corner
turning: split [in/fill/net/file/disk] to
[net/file]

SS_rev[12] – Get StreamStor
firmware/software revision levels
(query only)

start_stats – Start gathering disk-
performance statistics

status – Get system status (query
only)

task_ID – Set task ID (primarily for
correlator use)

track_check – Check data on
selected track (query only)

trackmask – Configure channel
dropping setup

track_set – Select tracks for
monitoring with DQA or
'track_check'

**tstat – Get current runtime status
and performance**

TVR – Start TVR testing

(un)mount – power up/down bank
(as if keying the bank on or off)
(jive5ab >= 2.8.2)

version – Get detailed version
information of this jive5ab (query
only)

VSN – Write extended-VSN to
permanent area

Monitor arguments:

Parameter**	Type**	Allowed values**	Default**	Comments
<strict>	int	1	1	0 – Enable less
bytestoread>	int		1000000	Amount of dat

Monitor-only parameters:

Parameter**	Type**	Values
<scan#>	int	
<scanlabel>	literalASCII	
<datatype>	char	• tvgr SS ? vdif (legacy)
[STRIKEOUT:<datecode>]<ntrack>	int	
<starttime>	time	
<scanlength>	time	
<totalrecordingrate>	real	Mbps
missingbytes>	int	See Note 5
dataarraysize>]	int	

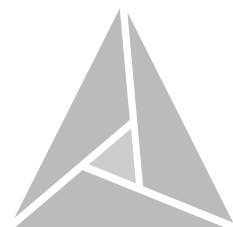
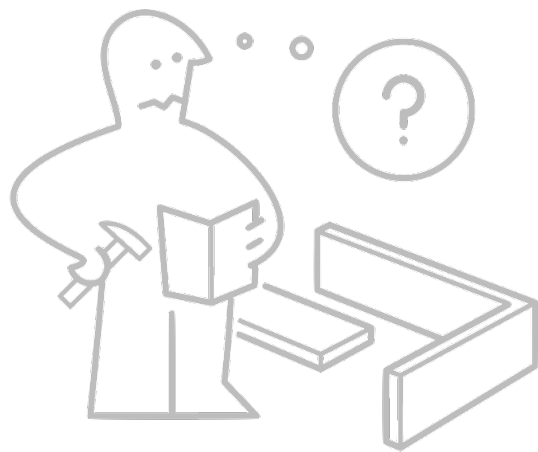
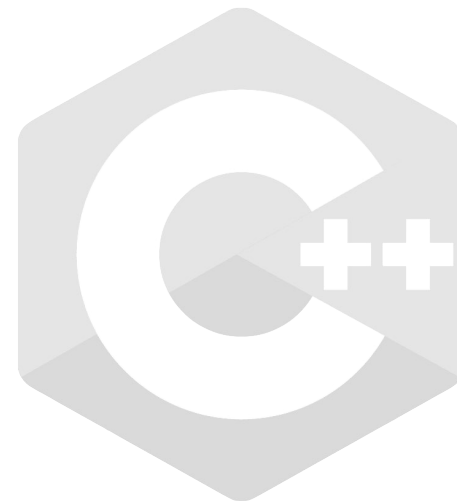
Notes:

1. The 'scan_check' query will be honored only if record and play are both off and does not affect the start-scan or stop-scan pointers.

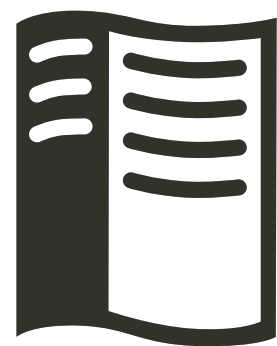
2. The 'scan_check' query essentially executes a 'data_check' starting at the start-scan pointer, followed by a 'data_check' just prior to the stop-scan pointer. This allows information about the selected scan to be conveniently determined.

3. Only tvgr data that were recorded with a bit-stream mask of 0xffffffff and no decimation will

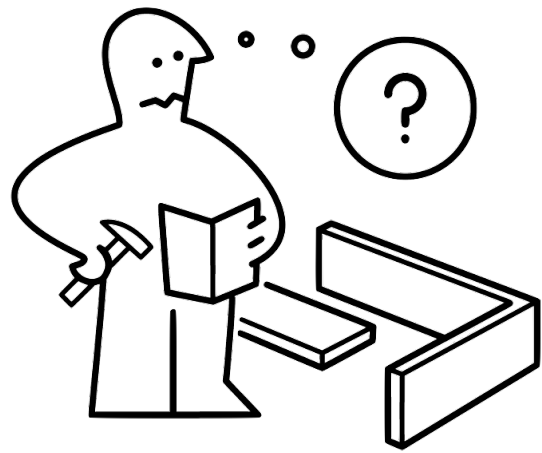
You can help fix up the details!



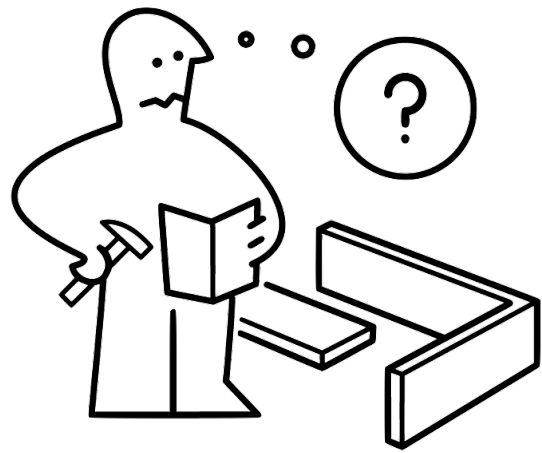
CMake



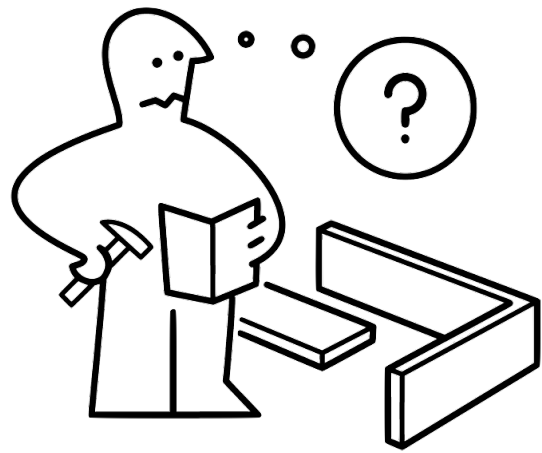
Read *the* Docs



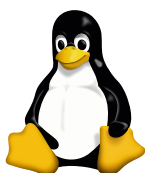
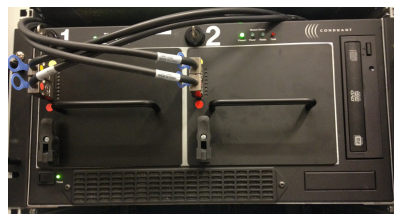
```
$> curl -o - http://www.jive.eu/~verkout/evlbi/jive5ab-X.Y.Z.tar.gz | tar zxvf -
```



```
$> curl -o - http://www.jive.eu/~verkout/evlbi/jive5ab-X.Y.Z.tar.gz | tar zxvf -  
  
$> cd jive5ab-X.Y.Z  
$> make B2B=64 SSAPIROOT=nossapi -j 8  
$> ./jive5ab-X.Y.Z -m 3 [...]
```

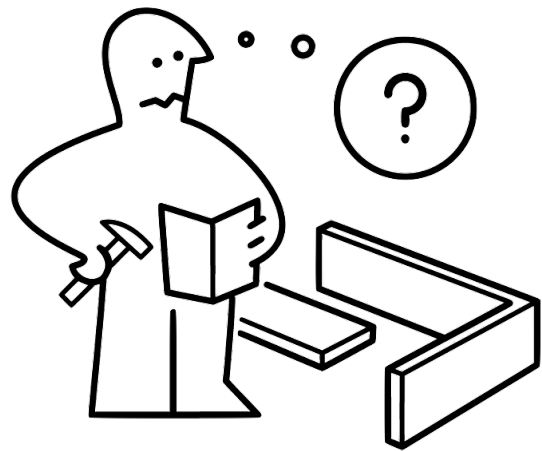



```
$> curl -o - http://www.jive.eu/~verkout/evlbi/jive5ab-X.Y.Z.tar.gz | tar zxvf -  
  
$> cd jive5ab-X.Y.Z  
$> make B2B=64 SSAPIROOT=nossapi -j 8  
$> ./jive5ab-X.Y.Z -m 3 [...]
```

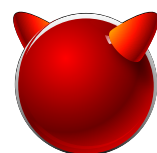
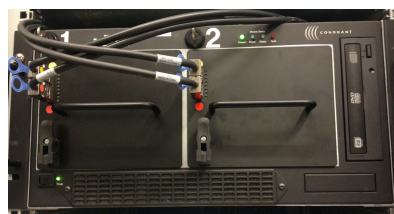


FreeBSD



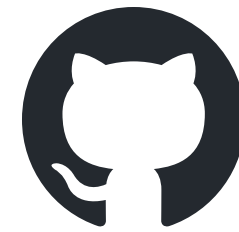
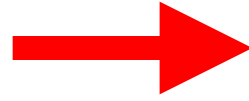
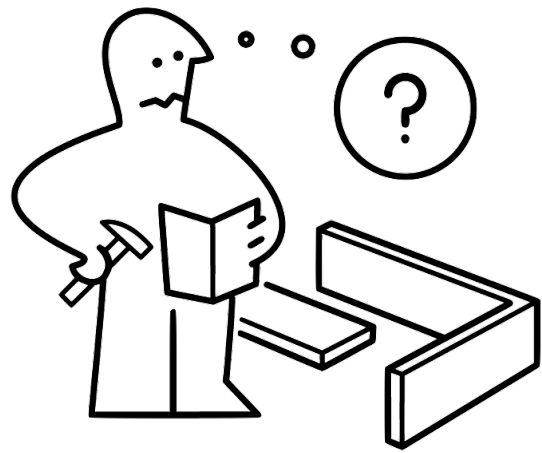


```
$> curl -o - http://www.jive.eu/~verkout/evlbi/jive5ab-X.Y.Z.tgz | xvf -  
$> cd jive5ab-X.Y.Z  
$> make B2B=64 SSAPIROOT=nossapi -j 8  
$> ./jive5ab-X.Y.Z -m 3 [...]
```



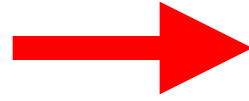
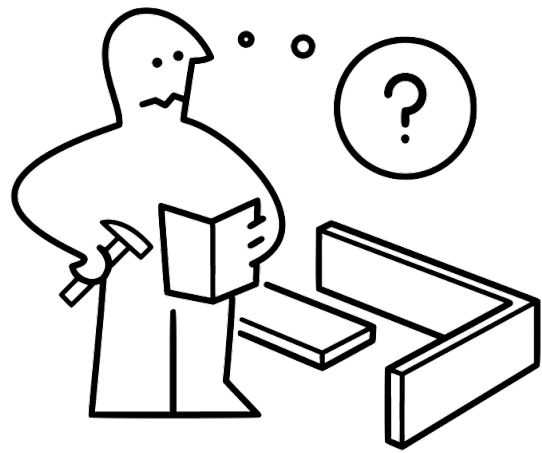
FreeBSD





GitHub

```
$> git clone https://github.com/jive-vlbi/jive5ab.git
```



CMake

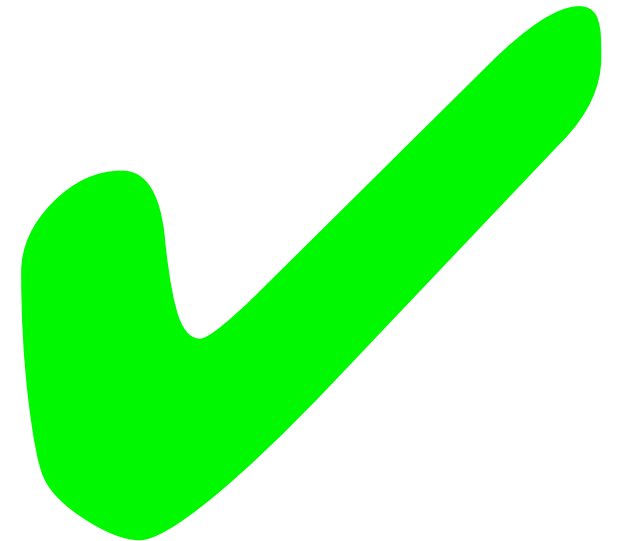
```
$> git clone https://github.com/jive-vlbi/jive5ab.git
```

```
$> mkdir build && cd build
```

```
$> cmake -DSSAPI_ROOT=nossapi ../jive5ab
```




```
$> git clone https://github.com/jive-vlbi/jive5ab.git  
  
$> mkdir build && cd build  
$> cmake -DSSAPI_ROOT=nossapi ../jive5ab  
$> make -j 8 install  
$> /usr/local/bin/jive5ab-X.Y.Z-Nbit-Release -m 3 [...]
```





```
$> git clone https://github.com/jive-vlbi/jive5ab.git
```

```
$> mkdir build-no-sse && cd build-no-sse
```

```
$> cmake -DSSAPI_ROOT=nossapi -DSSE=0 ../jive5ab
```

```
$> make -j 8
```



```
$> git clone https://github.com/jive-vlbi/jive5ab.git
```

```
$> git checkout awesome-feature-branch
```

```
$> mkdir build-awesome-feature-with-c11 &&
```

```
cd build-awesome-feature-with-c11
```

```
$> cmake -DSSAPI_ROOT=nossapi -DC++11=ON ../jive5ab
```

```
$> make -j 8
```

github.com/jive-vlbi/jive5ab

JIVE MT - HomeRB (Old)RB (Teams)iptables - Fail2banJIVE EXPSSFXC FringesORP - REDMINEACME SlackINFRATECH2024

Marjolein's eVLBI webpageGitHub - jive-vlbi/jive5ab: The JIVE VLBI data recording and transport utility

README

Supported cmake options

Category	Option	Description
Debug/Release	CMAKE_BUILD_TYPE= <i>Type</i>	Substitute 'Release' or 'Debu Default: Release
FiLa10G/Mark5B	FILA10G=ON	Generate a jive5ab that can c Mark5B data from FiLa10G/RI the UDPs protocol
SSE Version	SSE=[20 41]	Override automatic 'Streamin Extensions' version detection assembly code).
StreamStor SDK	SSAPI_ROOT= <i>path</i> nossapi	If not given, searches /usr, /usr/local/src/streamstor, /home/streamstor/Sdk for li Otherwise searches <i>path</i> . If n StreamStor hardware present Mark6) or desired (Mark5*) tl must now <i>explicitly</i> pass SSAPI_ROOT=nossapi
	WDAPIVER=XXXX	The StreamStor SDK library v link with. If not given the syst determine the value itself fro is found under SSAPI_ROOT . libwdapiXXXX.so files are fou error. If more than one libwda are found then WDAPIVER=X be given to select which one used
Install location	CMAKE_INSTALL_PREFIX= <i>path</i>	The compiled binary will be ir \${CMAKE_INSTALL_PREFIX}, \${VERSION}~[32 64]bit-\${BL [-FiLa10G], depending on the configuration details
Compiler selection	CMAKE_C_COMPILER= [/path/to/]C-compiler	Select the C-compiler to use

2024.009.15:46:35.17/**mk5=version?**

2024.009.15:46:35.17/mk5/!version? 0 : jive5ab :
3.1.0-dev : 64bit : Debug :
koll : 08-Jan-2024 :
12h26m00s : nossapi :
/usr/local/src/etransfer ;

2024.009.15:48:37.54;mk5=scan_check?

2024.009.15:48:37.59/mk5/scan_check? 0 :?:
xxx_xx_no0001 : VDIF : 128 :
2024y009d15h43m13.3117s :
5.9812s : 256Mbps :
1060224 : 8032 ;

2024.009.15:48:42.56;mk5=evlbi?

2024.009.15:48:42.56/mk5/evlbi? 0 : total: 3062244:
loss : 0 (0.00%) :
out-of-order: 0 (0.00%) :
extent: 0 seqnr/pkt ;

2024.009.15:46:35.17/mk5=version?

2024.009.15:46:35.17/mk5/!version? 0 : jive5ab :

3.1.0-dev : 64bit : Debug :

koll : 08-Jan-2024 :

12h26m00s : nossapi :

/usr/local/src/etransfer ;

2024.009.15:48:37.54;mk5=scan_check?

2024.009.15:48:37.59/mk5/scan_check? 0 :?:

xxx_xx_no0001 : VDIF : 128 :

2024y009d15h43m13.3117s :

5.9812s : 256Mbps :

1060224 : 8032 ;

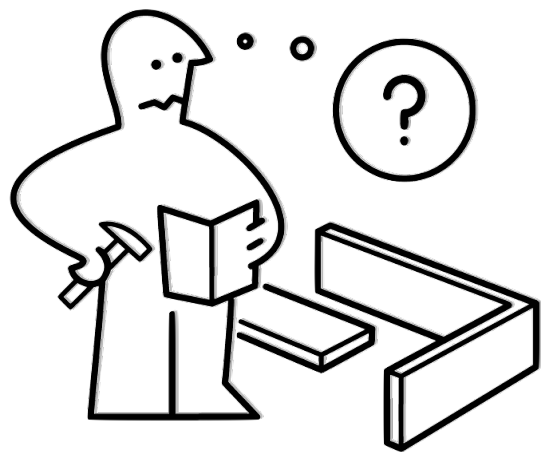
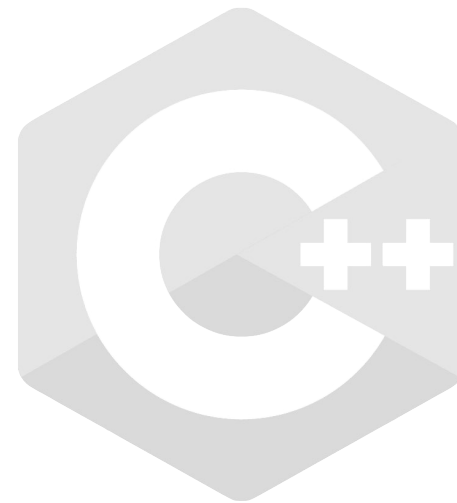
2024.009.15:48:42.56;mk5=evlbi?

2024.009.15:48:42.56/mk5/evlbi? 0 : total: 3062244:

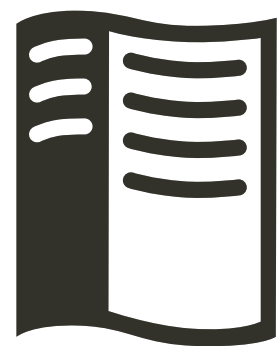
loss : 0 (0.00%) :

out-of-order: 0 (0.00%) :

extent: 0 seqnr/pkt ;



CMake



Read *the* Docs

e-shipping from station to correlator



Mark[56], FlexBuff, any recorder ...

Correlator

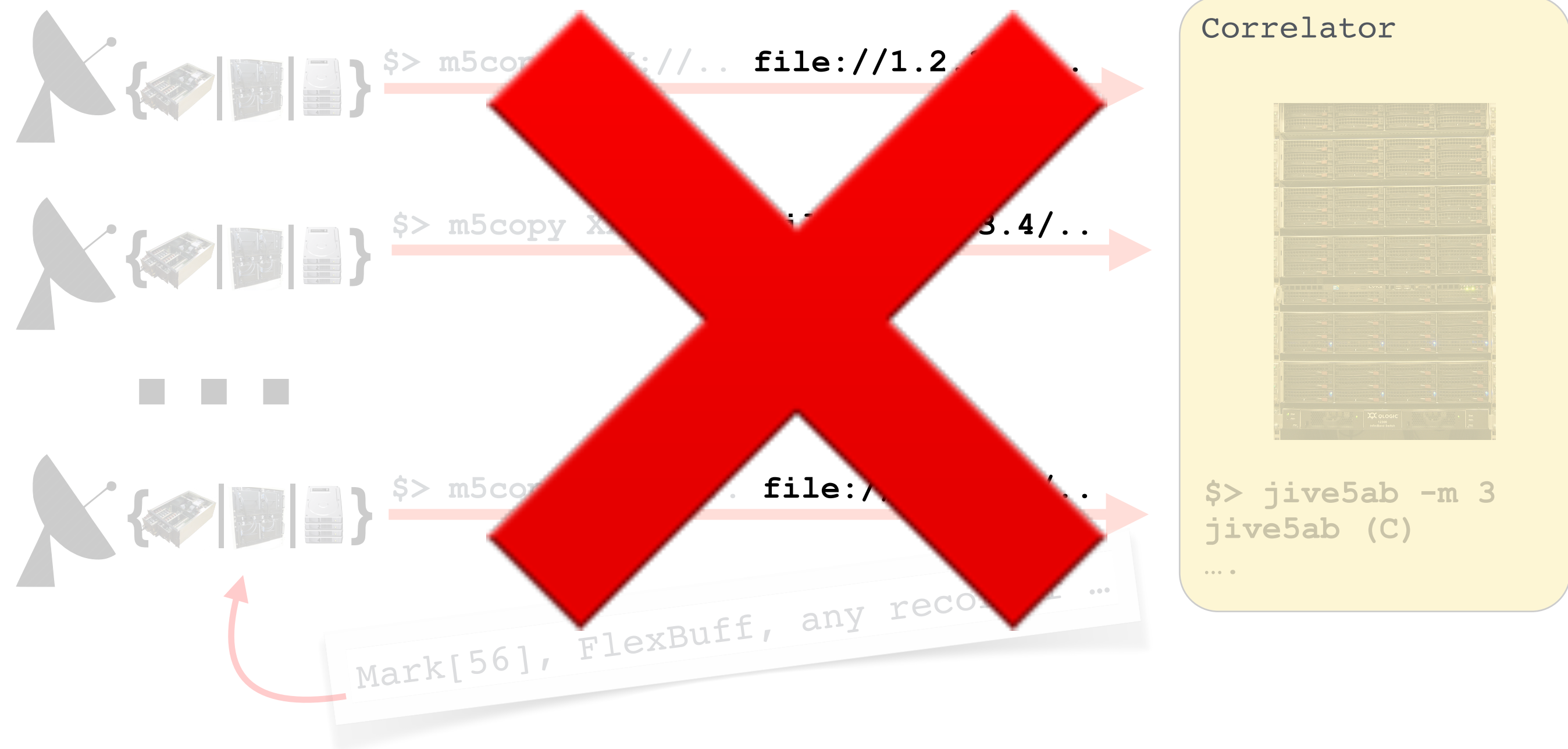


```
$> jive5ab -m 3
jive5ab (C)
...
```

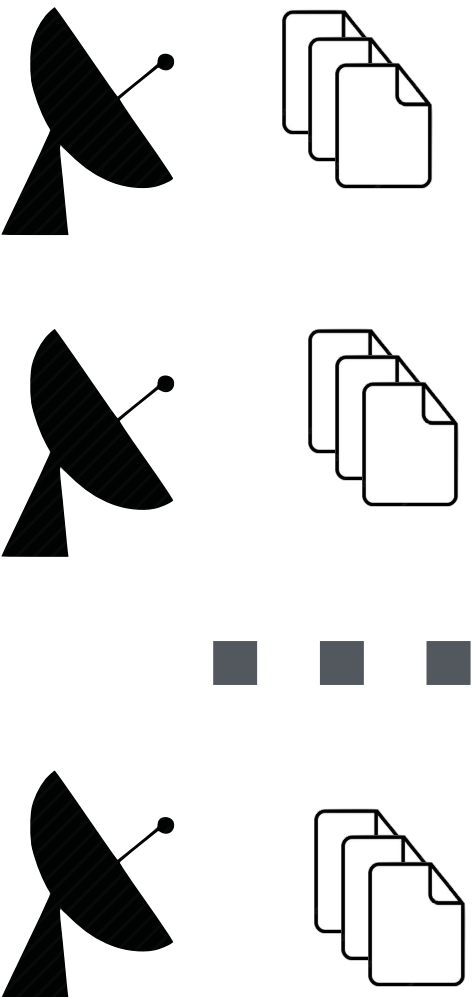
e-shipping from station to correlator



e-shipping from station to correlator



Use e-transfer!

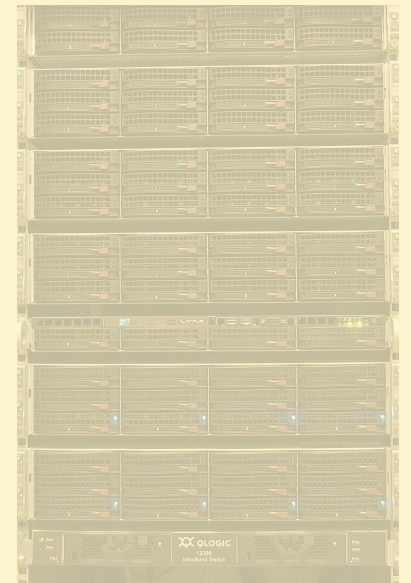


```
$> ./etc /data/scanX tcp://1.2.3.4:4004/
```

```
$> ./etc /data/scanY tcp://1.2.3.4:4004/
```

```
$> ./etc /data/scanZ tcp://1.2.3.4:4004/
```

Correlator

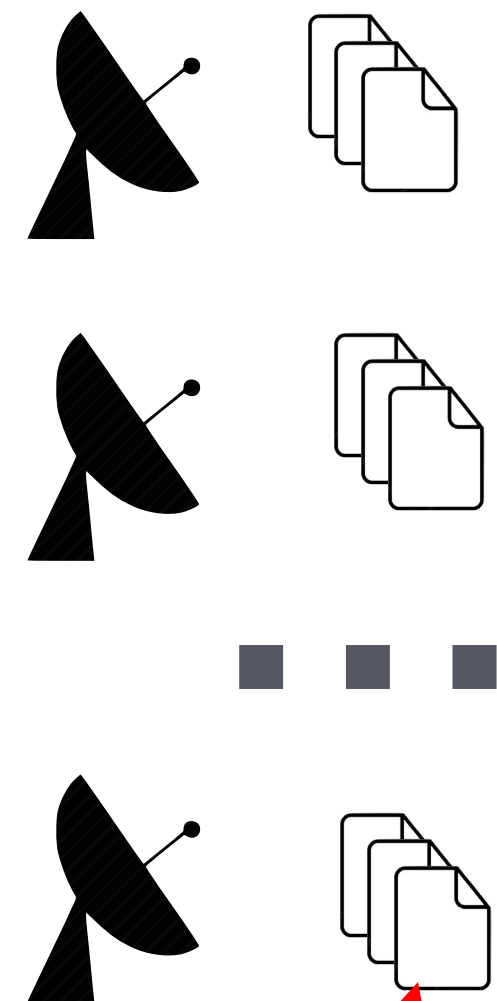


```
$> etd \  
--command tcp://4004  
--data udt://8008
```

...

<https://github.com/jive-vlbi/etransfer.git>

Use e-transfer!



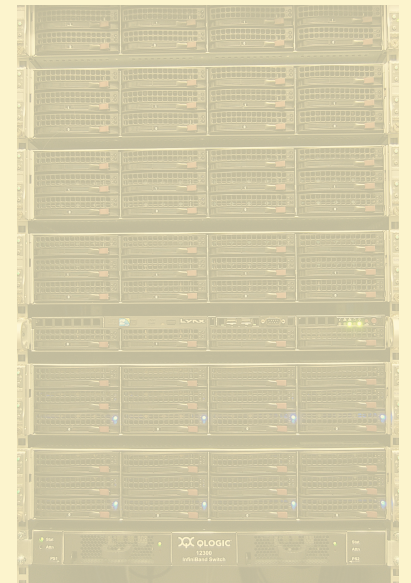
```
$> ./etc /data/scanX tcp://1.2.3.4:4004/
```

```
$> ./etc /data/scanY tcp://1.2.3.4:4004/
```

```
$> ./etc /data/scanZ tcp://1.2.3.4:4004/
```

Only supports files 😞 ...

Correlator



```
$> etd \  
  --command tcp://4004  
  --data udt://8008
```

...

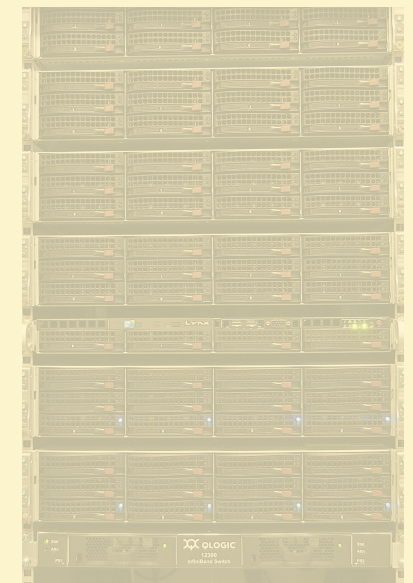
<https://github.com/jive-vlbi/etransfer.git>

Use e-transfer!

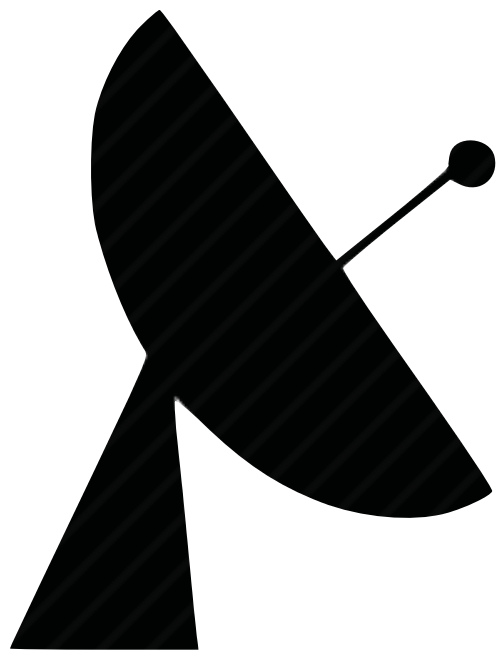


Mark[56], FlexBuff, any recorder!

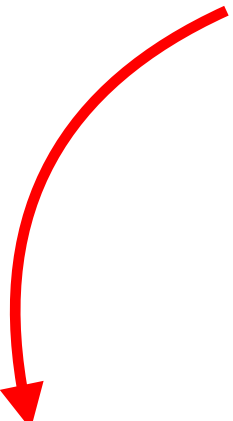
Correlator



```
$> etd \  
  --command tcp://4004  
  --data udt://8008  
...
```



e-transfer only needs checkout!



```
$> git clone https://github.com/jive-vlbi/etransfer.git
$> git clone --branch master https://github.com/jive-vlbi/jive5ab.git

$> mkdir build && cd build
$> cmake -DSSAPI_ROOT=nossapi -DETRANSFER_ROOT=../etransfer ../jive5ab
$> make -j 8

$> /usr/local/bin/jive5ab-X.Y.Z-Nbit-Release -m 3 [...]
```




```
$> git clone https://github.com/jive-vlbi/etransfer.git
$> git clone --branch master https://github.com/jive-vlbi/jive5ab.git

$> mkdir build && cd build
$> cmake -DSSAPI_ROOT=nossapi -DETRANSFER_ROOT=../etransfer ./jive5ab
$> make -j 8

$> /usr/local/bin/jive5ab-X.Y.Z-Nbit-Release -m 3 [...]
```

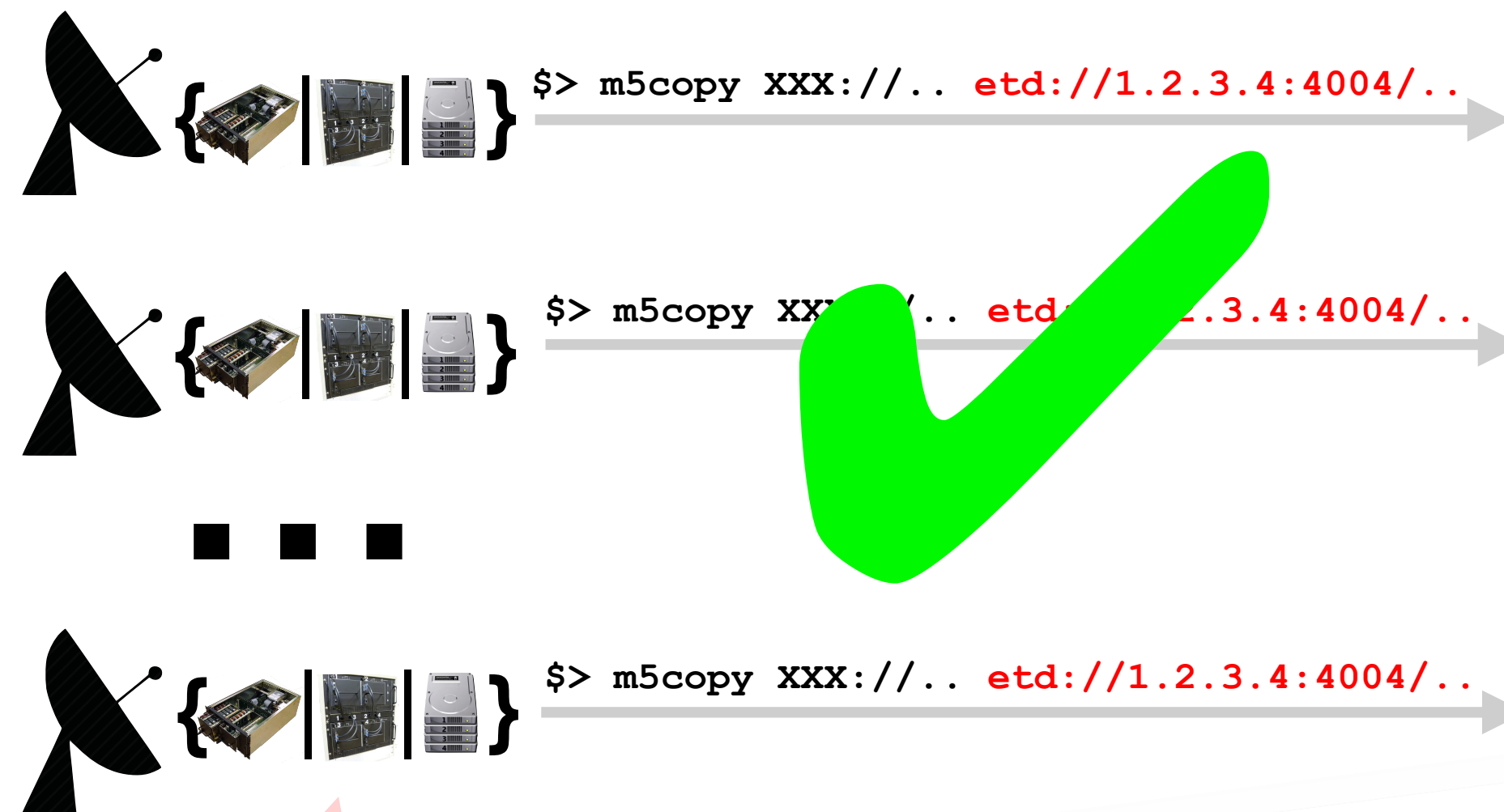


```
$> git clone https://github.com/jive-vlbi/etransfer.git
```

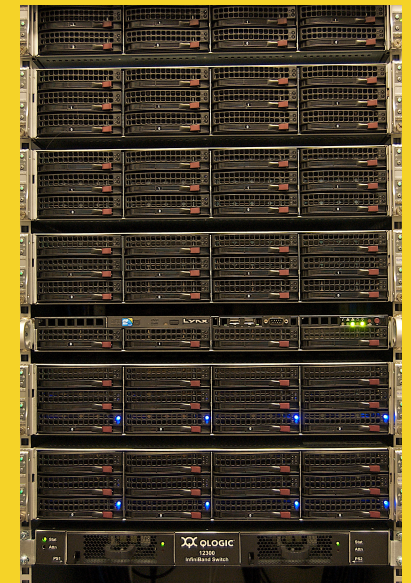
```
$> cd etransfer && make
```

```
$> ./<arch>/etd --command tcp://1.2.3.4:4004 --data udt://1.2.3.4:8008
```

Use e-transfer!



Correlator



```
$> etd \  
  --command tcp://4004  
  --data udt://8008  
...
```

Mark[56], FlexBuff, any recorder ...

2024.009.15:46:35.17/**mk5=version?**

2024.009.15:46:35.17/mk5/!version? 0 : jive5ab :
3.1.0-dev : 64bit : Debug :
koll : 08-Jan-2024 :
12h26m00s : nossapi :
/usr/local/src/etransfer ;

2024.009.15:48:37.54;mk5=scan_check?

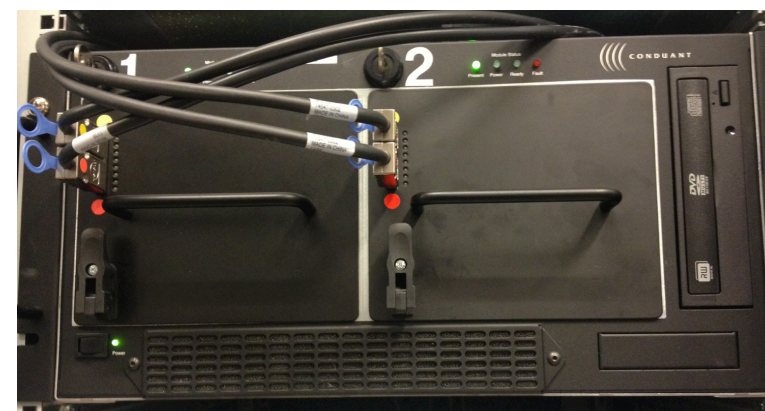
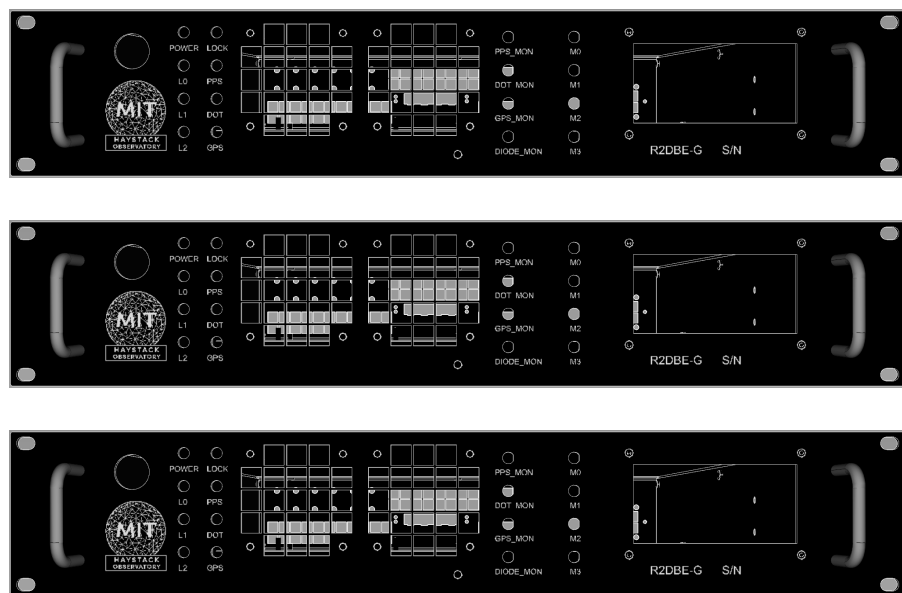
2024.009.15:48:37.59/mk5/scan_check? 0 :?:
xxx_xx_no0001 : VDIF : 128 :
2024y009d15h43m13.3117s :
5.9812s : 256Mbps :
1060224 : 8032 ;

2024.009.15:48:42.56;mk5=evlbi?

2024.009.15:48:42.56/mk5/evlbi? 0 : total: 3062244:
loss : 0 (0.00%) :
out-of-order: 0 (0.00%) :
extent: 0 seqnr/pkt ;

(very)
high recording
data rates*

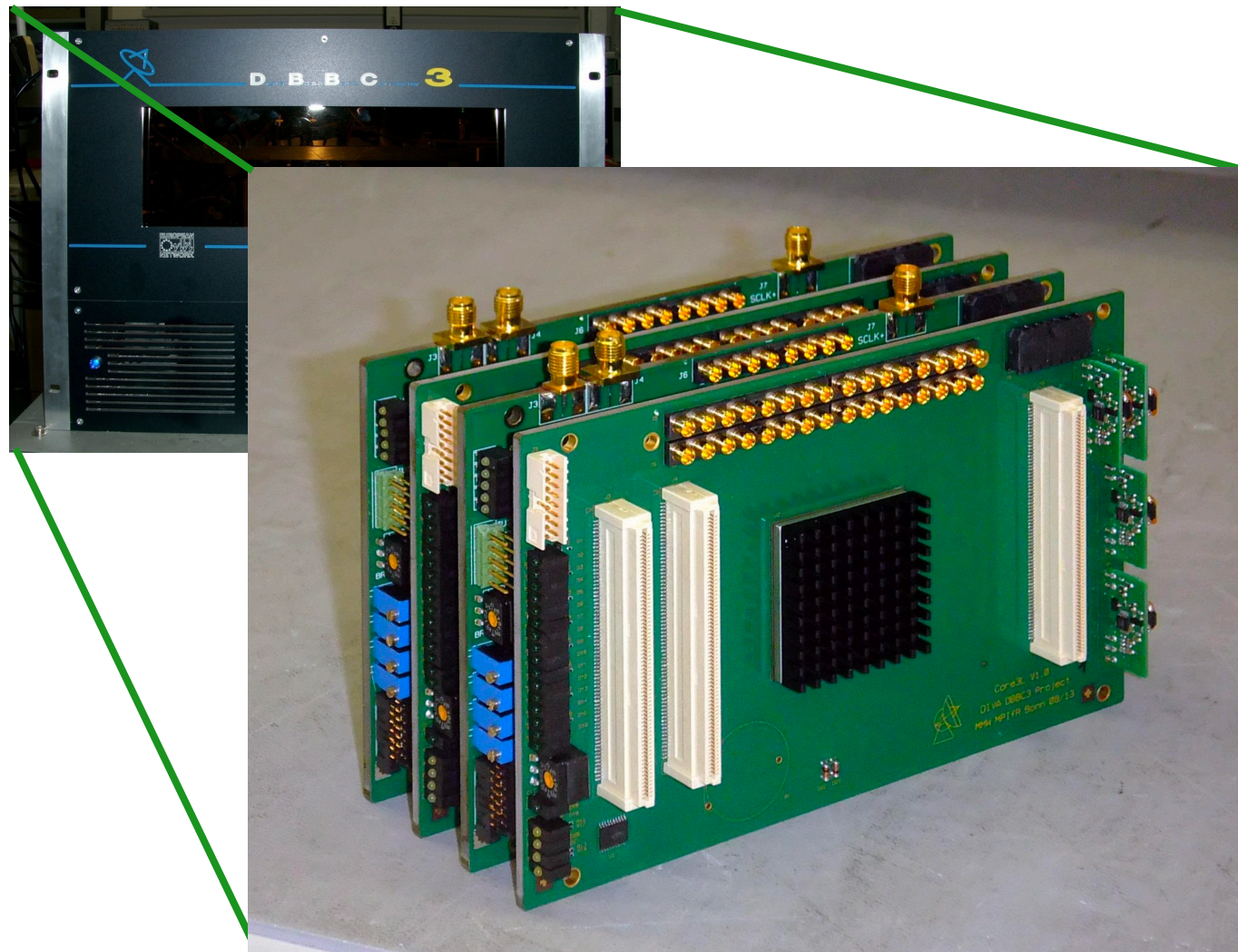
() that's roughly > 8 Gbps in my book*



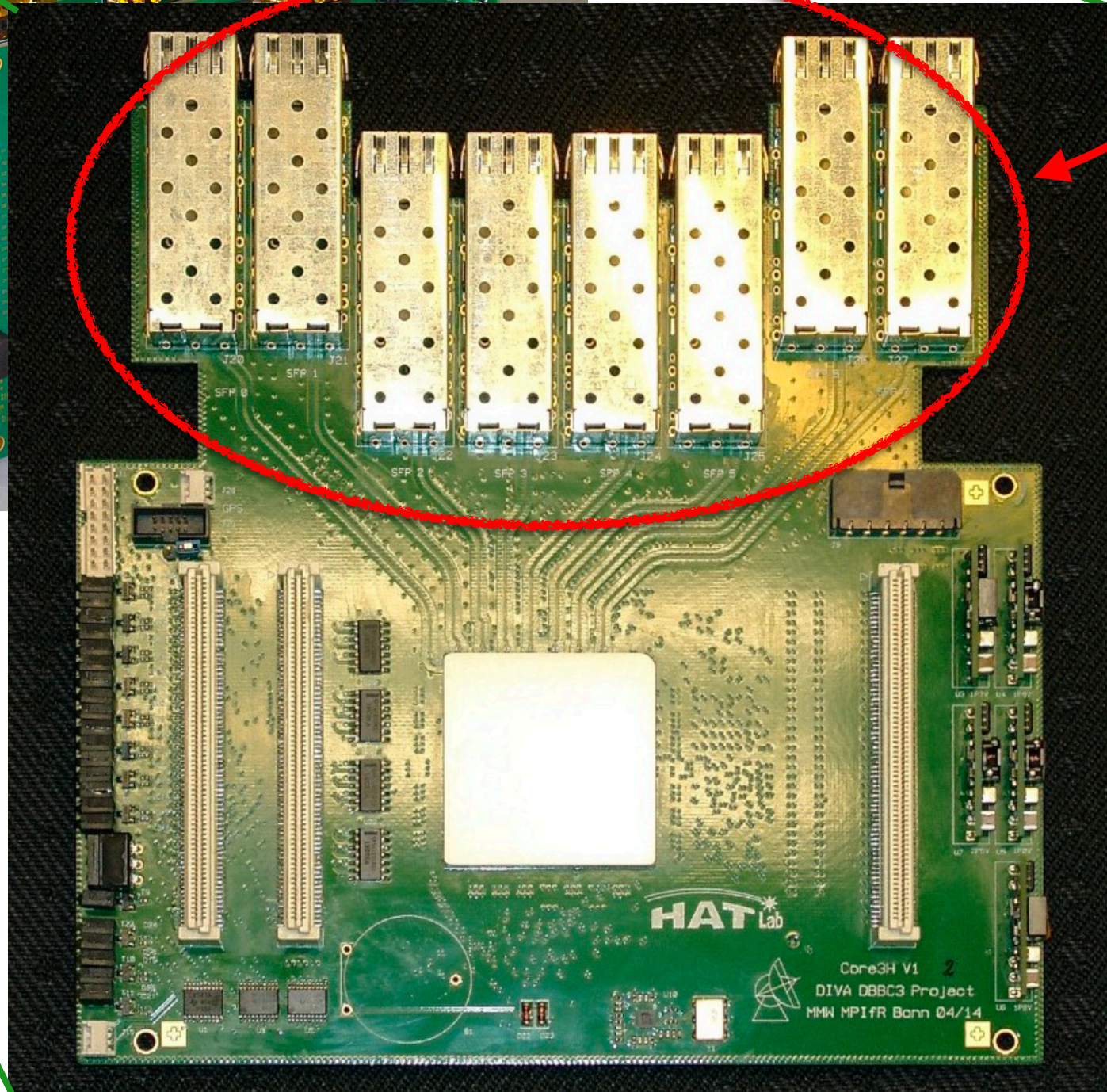
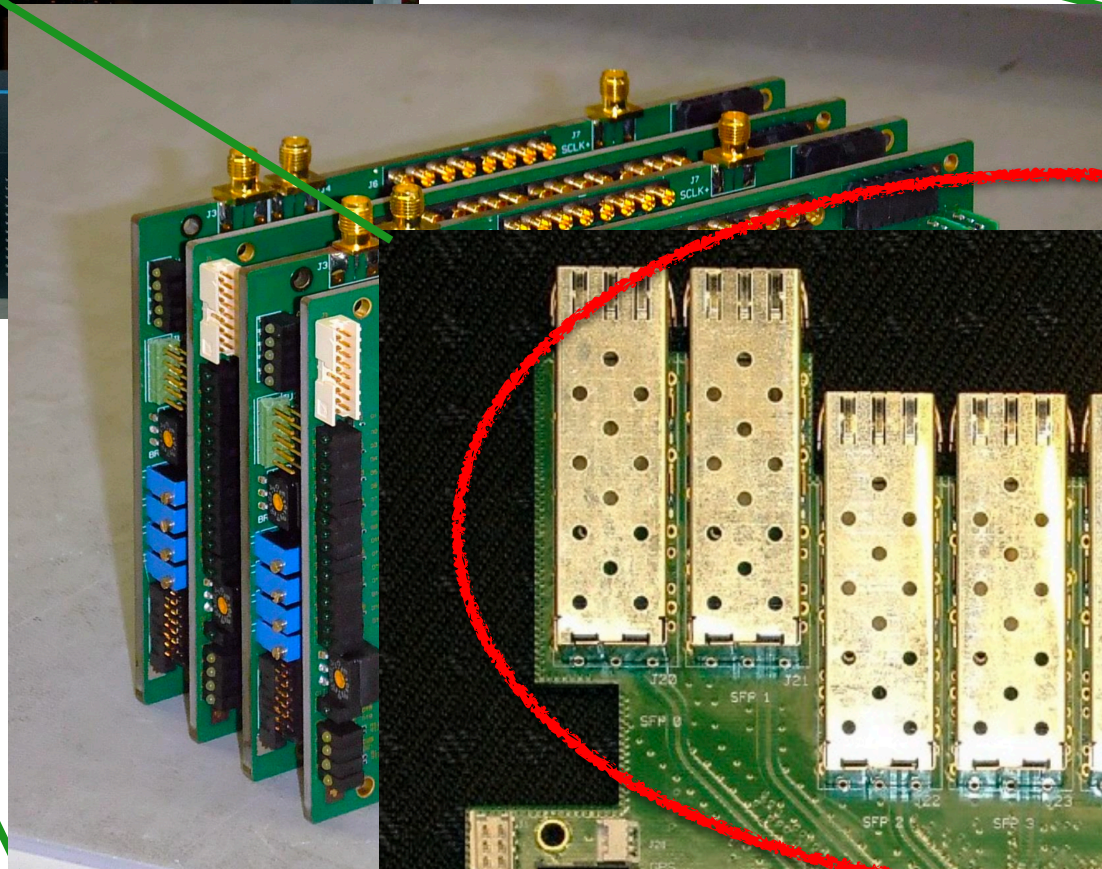
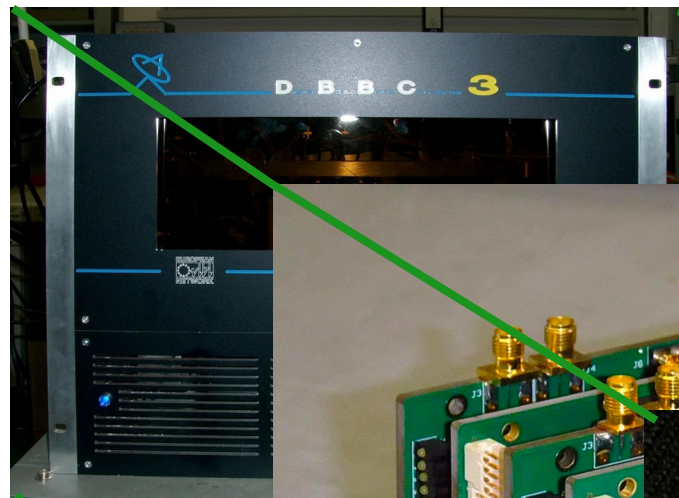
digital backend(s)

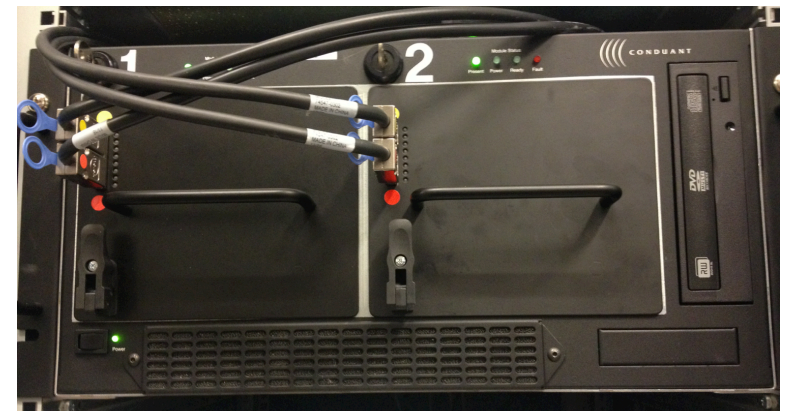
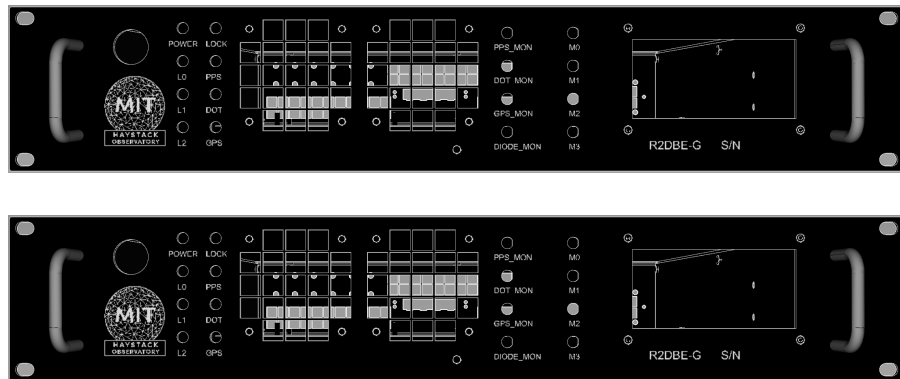
VLBI data recorder(s)





8 x 10 Gbps per CORE-3H board
(max) 8 x CORE-3H per DBBC3





digital backend(s)

VLBI data recorder(s)

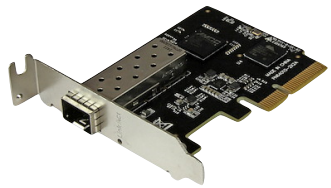
digital backend(s)



VLBI data recorder(s)

Server chassis (Mark6/FlexBuff)

ethernet cards

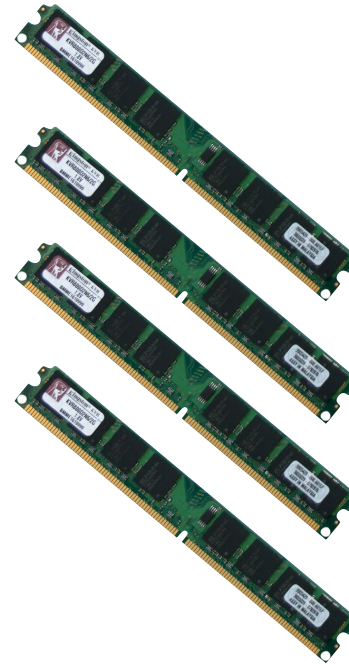


Server chassis (Mark6/FlexBuff)

ethernet cards



memory modules

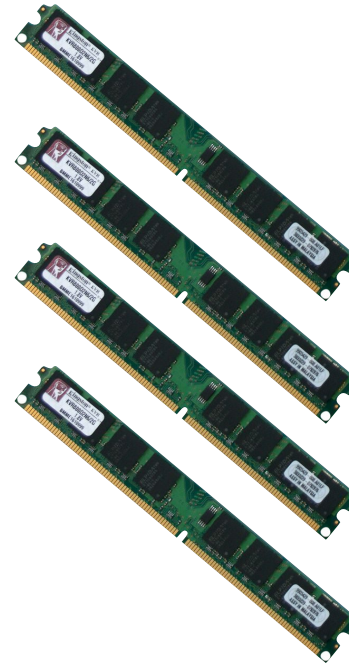


Server chassis (Mark6/FlexBuff)

ethernet cards



memory modules

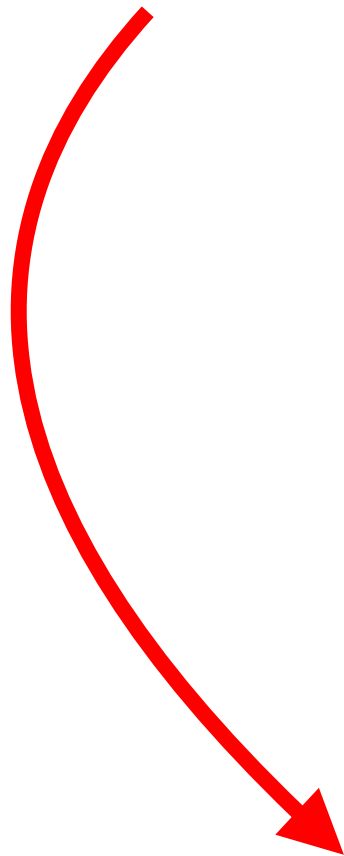


hard disk drives




```
net_port = [ip@]port ;  
    record = on : <scan> ;
```

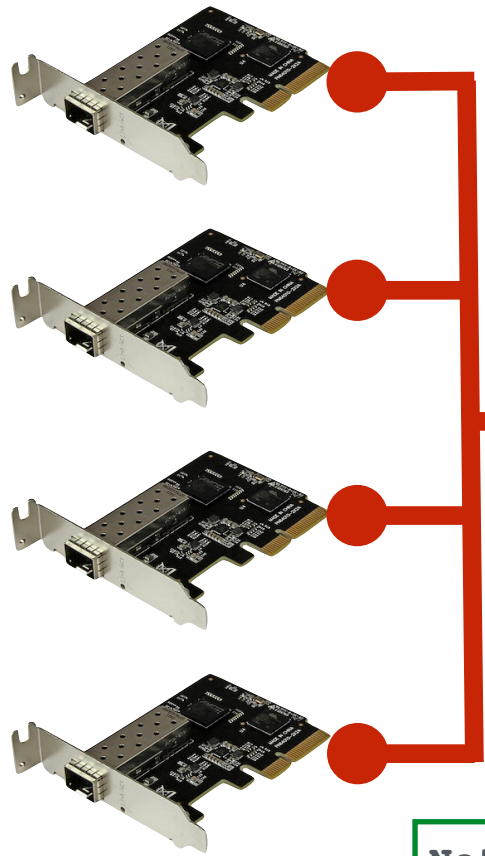
```
net_port = [ip@]port ;  
record = on : <scan> ;
```



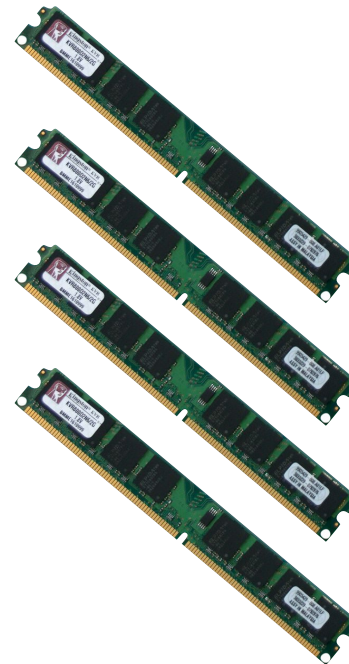
creates one thread, reads from network address
[ip:]port

Server chassis (Mark6/FlexBuff)

ethernet cards



memory modules



hard disk drives



Network reader thread

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

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

Disk writer thread #1

```
fd =
```

```
while
```

```
}
```

Disk writer thread #2

```
fd =
```

```
while
```

```
}
```

Disk writer thread #3

```
fd = open("/mnt/disk0/...");
```

```
while( true ) {
    write(fd, buf, N);
    /* ... */
}
```

Hit single CPU core
bottleneck:

max 16-ish* Gbps

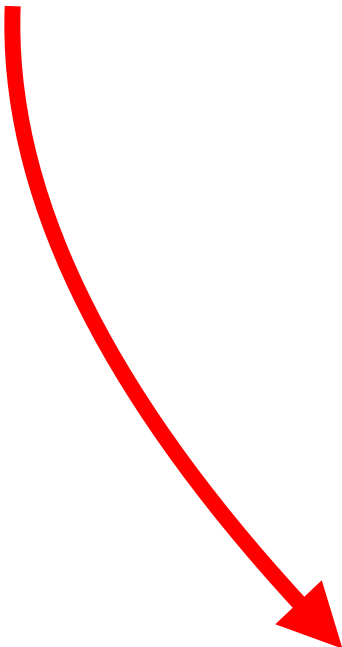
() YMMV: higher clocked CPUs can do 20+ Gbps*

```
net_port = [ip1@]port1 : [ip2@]port2 : ... ;  
    record = on : <scan> ;
```



```
net_port = [ip1@]port1 : [ip2@]port2 : ... ;  
record = on : <scan> ;
```

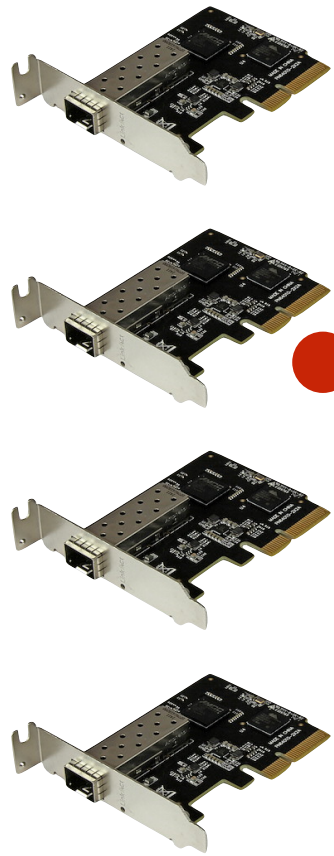
```
net_port = [ip1@]port1 : [ip2@]port2 : ... ;  
record = on : <scan> ;
```



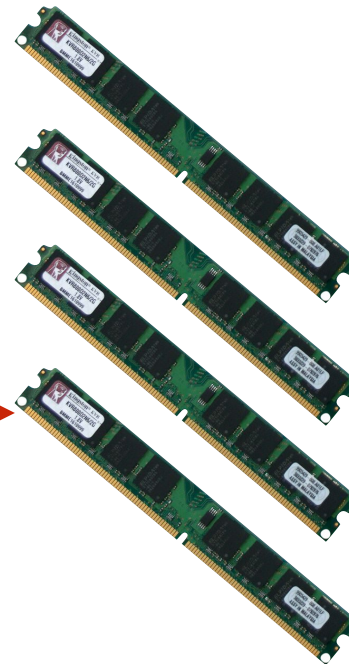
creates **n_{port}** threads
reading from network addresses
[ip1:]port1,
[ip2:]port2,
etc.

Server chassis (Mark6/FlexBuff)

ethernet cards



memory modules



hard disk drives



Network reader thread #1

```
s = socket(AF_INET, PF_DGRAM);  
bind(s, IP1:PORT_NR1);  
while( true ) {  
    read(socket, buf, N);  
    /* ... */  
}
```

Network reader thread #2

```
s = socket(AF_INET, PF_DGRAM);  
bind(s, IP2:PORT_NR2);  
while( true ) {  
    read(socket, buf, N);  
    /* ... */  
}
```

Network reader thread #3

```
s = socket(AF_INET, PF_DGRAM);  
bind(s, IP3:PORT_NR3);  
while( true ) {  
    read(socket, buf, N);  
    /* ... */  
}
```

Disk writer thread #1

```
fd = open("/mnt/disk0/...");  
while( true ) {  
    write(fd, buf, N);  
    /* ... */  
}
```

Disk writer thread #2

```
fd = open("/mnt/disk0/...");  
while( true ) {  
    write(fd, buf, N);  
    /* ... */  
}
```

Disk writer thread #3

```
fd = open("/mnt/disk0/...");  
while( true ) {  
    write(fd, buf, N);  
    /* ... */  
}
```

```
00:53:20;mk5=record? nthread
```

```
00:53:20/mk5/!record? : 4 : 32 ;
```

```
00:53:31; mk5=net_protocol?
```

```
00:53:31/mk5/net protocol? 0 : udpsnor: 768000000 : 128000000 : 16 ;
```

```
00:53:39; mk5=net_port?
```

```
00:53:39/mk5/!net_port? 0 : 2630 : 2631 : 2632 : 2634 ;
```

```
00:53:20;mk5=record? nthread
```

```
00:53:20/mk5/!record? : 4 : 32 ;
```

```
00:53:31; mk5=net_protocol?
```

```
00:53:31/mk5/net protocol? 0 : udpsnor: 768000000 : 128000000 : 16 ;
```

```
00:53:39; mk5=net_port?
```

```
00:53:39/mk5/ !net_port? 0 : 2630 : 2631 : 2632 : 2634 ;
```



```
00:53:20;mk5=record? nthread
```

```
00:53:20/mk5/!record? : 4 : 32 ;
```

```
00:53:31; mk5=net_protocol?
```

```
00:53:31/mk5/net protocol? 0 : udpsnor: 768000000 : 128000000 : 16 ;
```

```
00:53:39; mk5=net_port?
```

```
00:53:39/mk5/ !net_port? 0 : 2630 : 2631 : 2632 : 2634 ;
```

2024.009.15:46:35.17/mk5=version?

2024.009.15:46:35.17/mk5/!version? 0 : jive5ab :
3.1.0-dev : 64bit : Debug :
koll : 08-Jan-2024 :
12h26m00s : nossapi :
/usr/local/src/etransfer ;

2024.009.15:48:37.54;mk5=scan_check?

2024.009.15:48:37.59/mk5/scan_check? 0 :?:
xxx_xx_no0001 : **VDIF : 128** :
2024y009d15h43m13.3117s :
5.9812s : 256Mbps :
1060224 : 8032 ;

2024.009.15:48:42.56;mk5=evlbi?

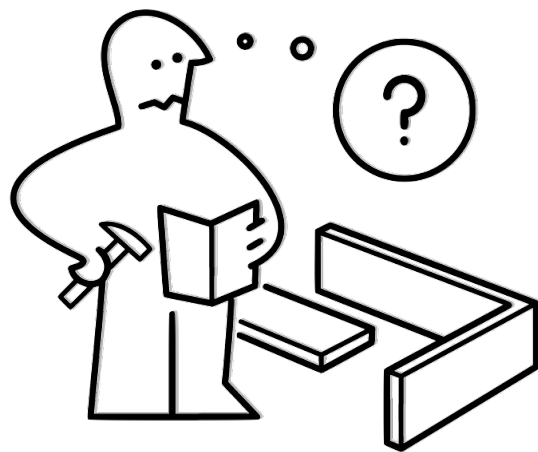
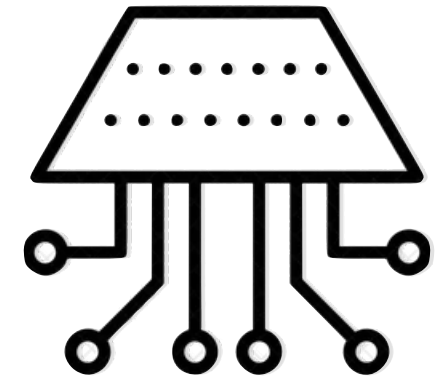
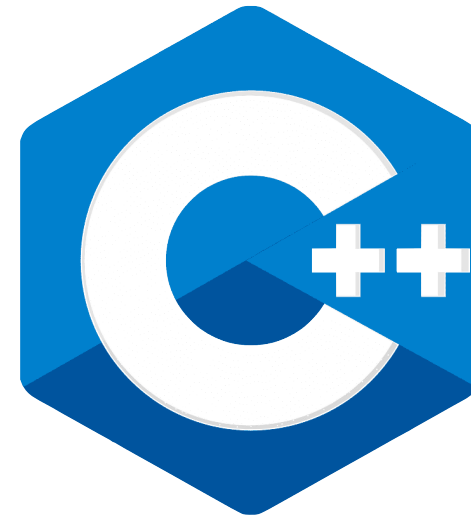
2024.009.15:48:42.56/mk5/evlbi? 0 : total: 3062244:
loss : 0 (0.00%) :
out-of-order: 0 (0.00%) :
extent: 0 seqnr/pkt ;

128 x 256 Mbps = 32 Gbps

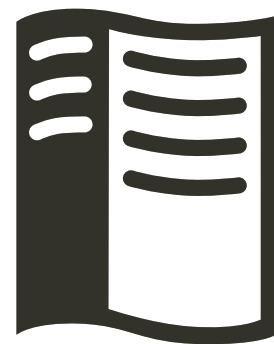
on a standard Mark6

with

32 x 10 TB HDD (or larger)



CMake



Read *the* Docs

d (s)


data stream #0

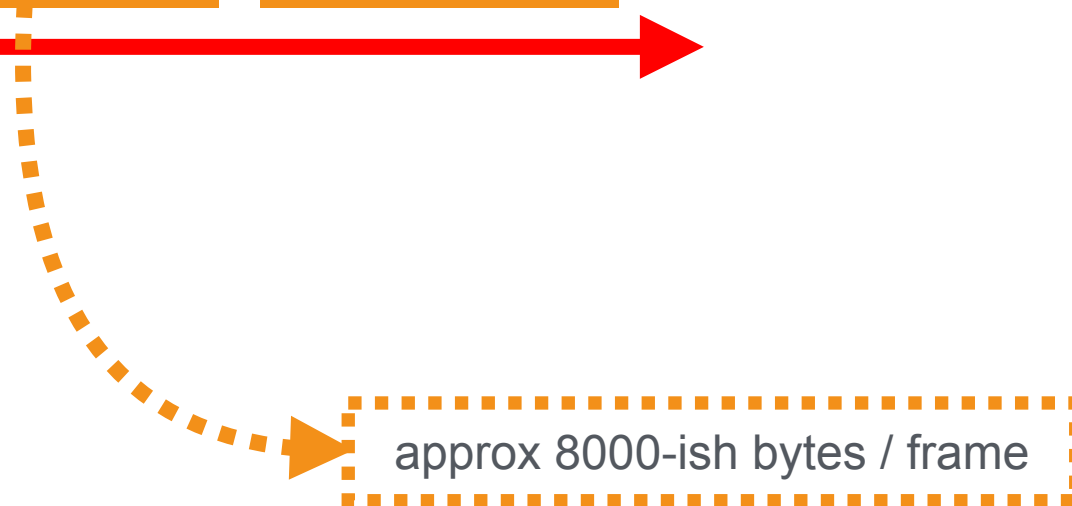
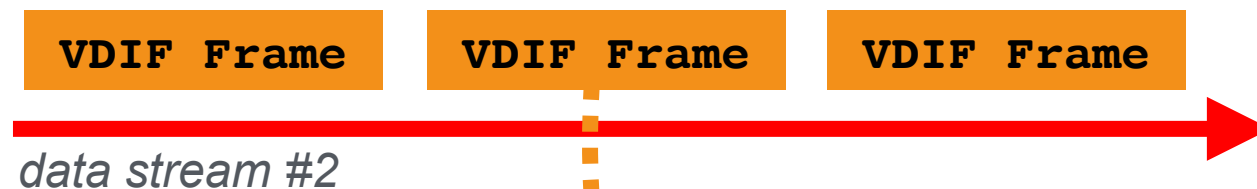

data stream #1

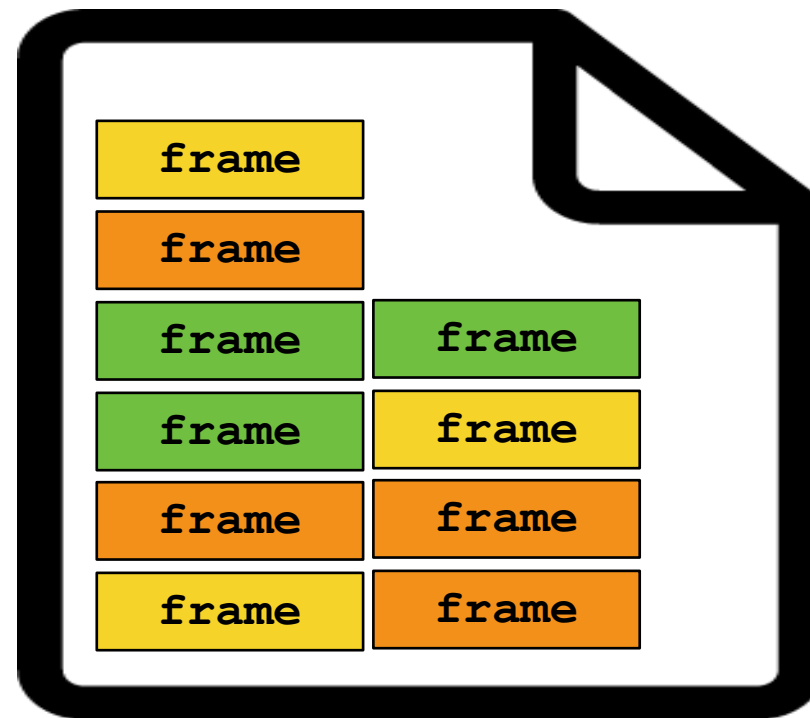

data stream #2

VLI

d (s)

VLI



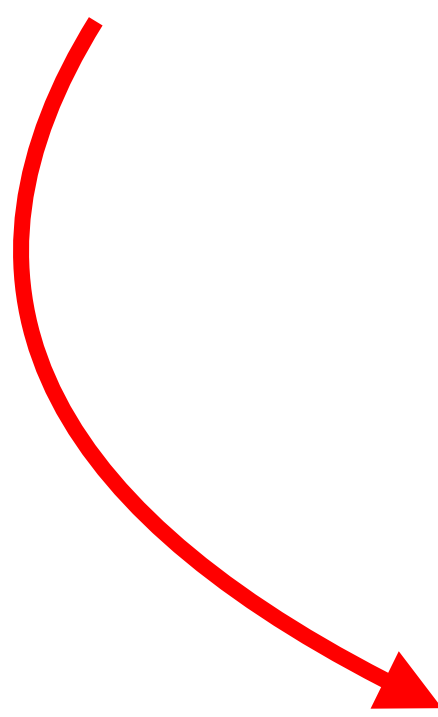


`EXP_STA_SCAN(vbs/mk6)`

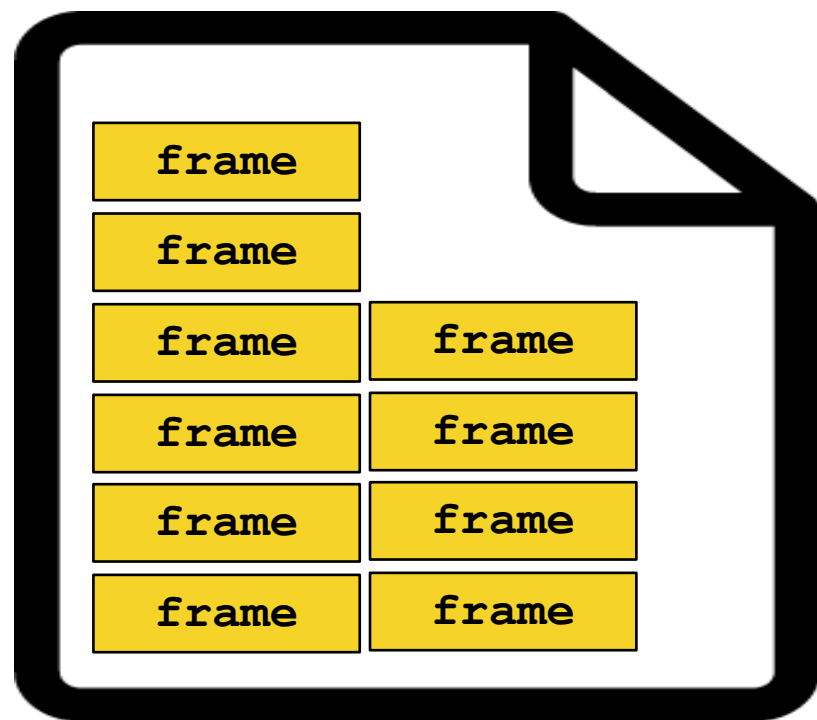
```
net_port = [ip1@]port1 [=suffix1] :  
           [ip2@]port2 [=suffix2] : ... ;  
  
record = on : <scan> ;
```



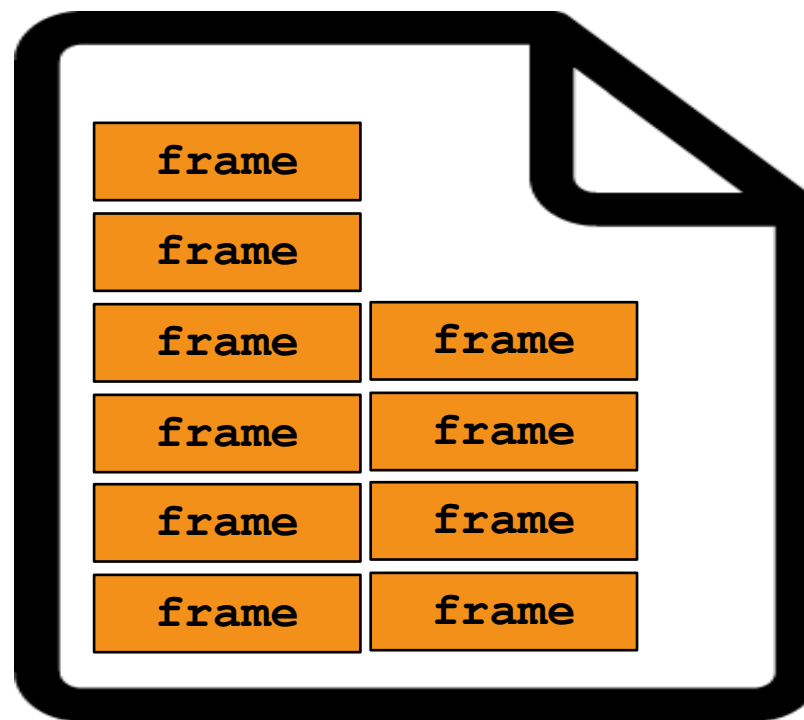
```
net_port = [ip1@]port1[=suffix1] :  
           [ip2@]port2[=suffix2] : ... ;  
  
record = on : <scan> ;
```



records **all frames arriving on**
[IP1:]PORT1 into **<scan>_ds<suffix1>**
[IP2:]PORT2 into **<scan>_ds<suffix2>**
etc.



EXP_STA_SCAN_ds<suffix1>



EXP_STA_SCAN_ds<suffix2>

...

2024.010.12:10:11.26;fb=scan_set=junyang_on_no0001_dsds0
2024.010.12:10:15.66/fb/!scan_check? 0 :?: junyang_on_no0001_dsds0 : VDIF : 32 : 2024y010d12h08m29.27995 : 5.98402s : 256Mbps: 0 : 8032 ;

2024.010.12:10:18.45;fb=scan_set=junyang_on_no0001_dsds1
2024.010.12:10:19.98/fb/!scan_check? 0 :?: junyang_on_no0001_dsds1: VDIF : 32 : 2024y010d12h08m29.28045 : 5.983615 : 256Mbps: 0 : 8032 ;

2024.010.12:10:28.62;fb=scan_set=junyang_on_no0001_dsds2
2024.010.12:10:30.07/fb/!scan_check? 0 :?: junyang_on_no0001_dsds2 : VDIF : 32 : 2024y010d12h08m29.2802s : 5.983885 : 256Mbps: 0 : 8032 ;

2024.010.12:10:34.39;fb=scan_set=junyang_on_no0001_dsds3
2024.010.12:10:35.85/fb/!scan_check? 0: ? : junyang_on_no0001_dsds3 : VDIF : 32 : 2024y010d12h08m29.2803s : 5.98348s : 256Mbps : 0 : 8032 ;

2024.010.12:10:11.26;fb=scan_set=junyang_on_no0001_dsds0
2024.010.12:10:15.66/fb/!scan_check? 0 :?: junyang_on_ **no0001_dsds0** : VDIF : 32 : 2024y010d12h08m**29.27995 : 5.98402s** : 256Mbps: 0 : 8032 ;

2024.010.12:10:18.45;fb=scan_set=junyang_on_no0001_dsds1
2024.010.12:10:19.98/fb/!scan_check? 0 :?: junyang_on_ **no0001_dsds1** : VDIF : 32 : 2024y010d12h08m**29.28045 : 5.983615** : 256Mbps: 0 : 8032 ;

2024.010.12:10:28.62;fb=scan_set=junyang_on_no0001_dsds2
2024.010.12:10:30.07/fb/!scan_check? 0 :?: junyang_on_ **no0001_dsds2** : VDIF : 32 : 2024y010d12h08m**29.2802s : 5.983885** : 256Mbps: 0 : 8032 ;

2024.010.12:10:34.39;fb=scan_set=junyang_on_no0001_dsds3
2024.010.12:10:35.85/fb/!scan_check? 0: ? : junyang_on_ **no0001_dsds3** : VDIF : 32 : 2024y010d12h08m**29.2803s : 5.98348s** : 256Mbps : 0 : 8032 ;

2024.010.12:10:11.26;fb=scan_set=junyang_on_no0001_dsds0
2024.010.12:10:15.66/fb/!scan_check? 0 :?: junyang_on_no0001_dsds0 : VDIF : 32 : 2024y010d12h08m29.27995 : 5.98402s : 256Mbps: 0 : 8032 ;

2024.010.12:10:18.45;fb=scan_set=junyang_on_no0001_dsds1
2024.010.12:10:19.98/fb/!scan_check? 0 :?: junyang_on_no0001_dsds1 : VDIF : 32 : 2024y010d12h08m29.28045 : 5.983615 : 256Mbps: 0 : 8032 ;

2024.010.12:10:28.62;fb=scan_set=junyang_on_no0001_dsds2
2024.010.12:10:30.07/fb/!scan_check? 0 :?: junyang_on_no0001_dsds2 : VDIF : 32 : 2024y010d12h08m29.2802s : 5.983885 : 256Mbps: 0 : 8032 ;

2024.010.12:10:34.39;fb=scan_set=junyang_on_no0001_dsds3
2024.010.12:10:35.85/fb/!scan_check? 0: ? : junyang_on_no0001_dsds3 : VDIF : 32 : 2024y010d12h08m29.2803s : 5.98348s : 256Mbps: 0 : 8032 ;



2024.009.15:46:35.17/mk5=version?

2024.009.15:46:35.17/mk5/!version? 0 : jive5ab :
3.1.0-dev : 64bit : Debug :
koll : 08-Jan-2024 :
12h26m00s : nossapi :
/usr/local/src/etransfer ;

2024.009.15:48:37.54;mk5=scan_check?

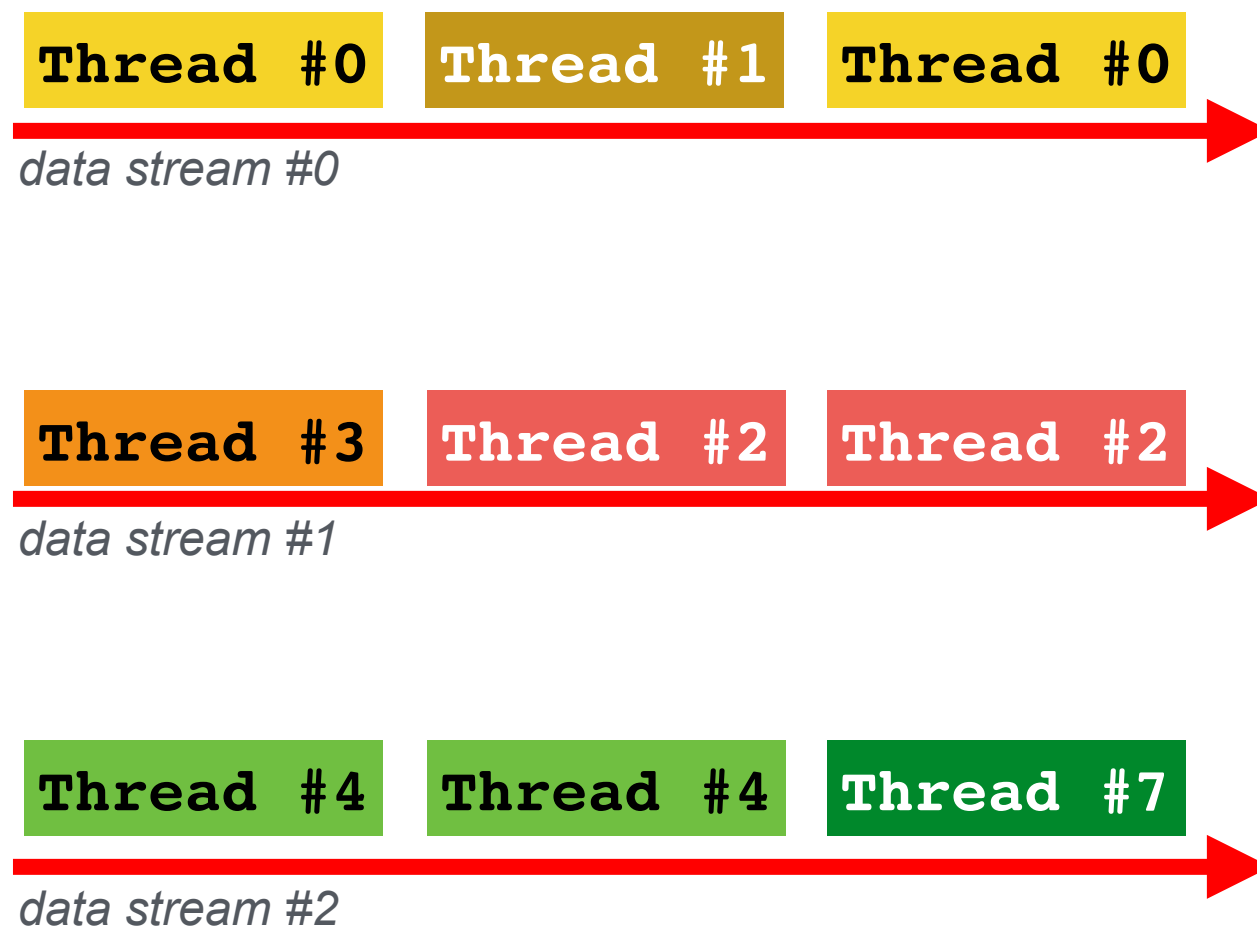
2024.009.15:48:37.59/mk5/scan_check? 0 :?:
xxx_xx_no0001 : VDIF : 128 :
2024y009d15h43m13.3117s :
5.9812s : 256Mbps :
1060224 : 8032 ;

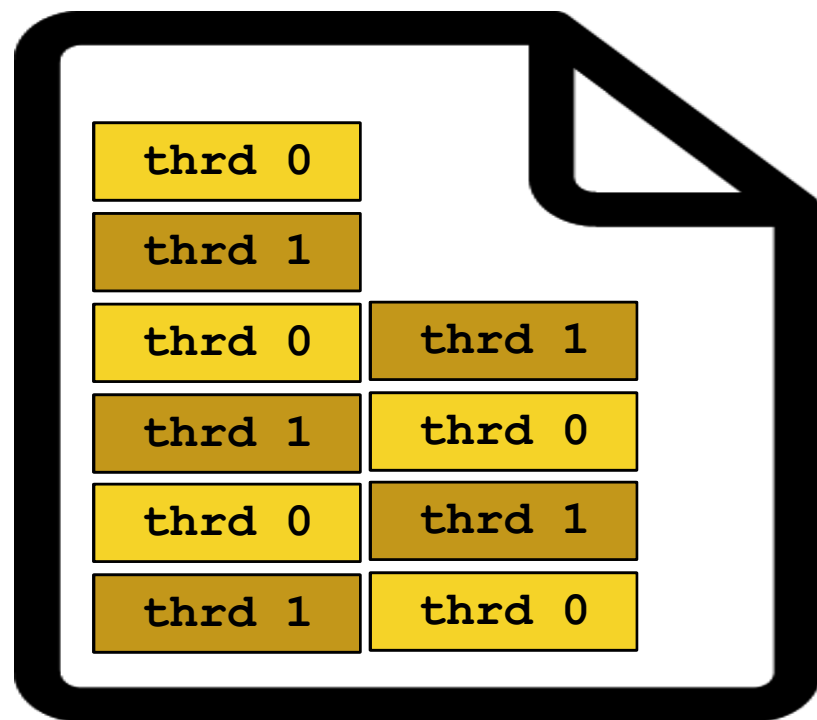
2024.009.15:48:42.56;mk5=evlbi?

2024.009.15:48:42.56/mk5/evlbi? 0 : **total: 3062244:**
loss : 0 (0.00%) :
out-of-order: 0 (0.00%) :
extent: 0 seqnr/pkt ;

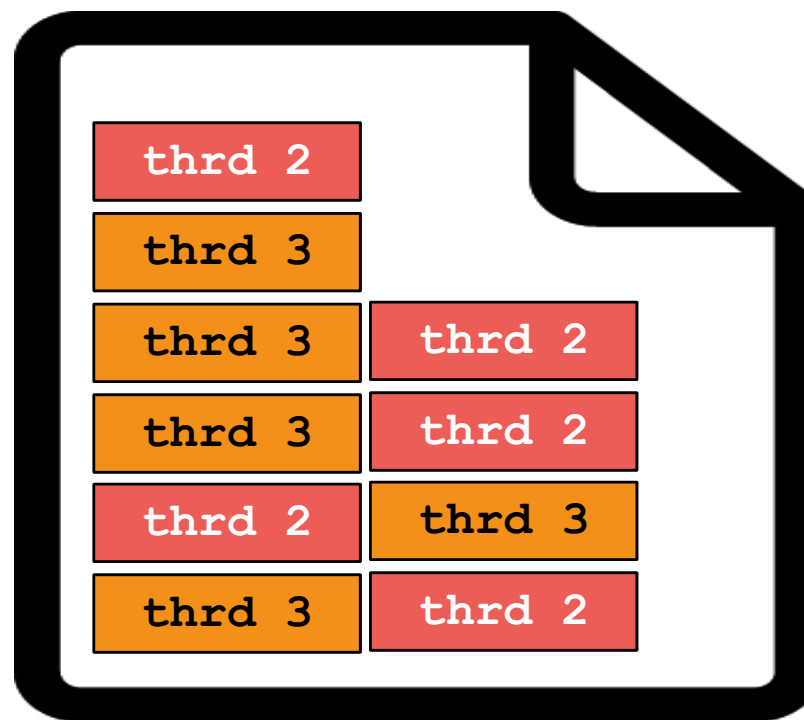
d (s)

VLI





EXP_STA_SCAN_ds<suffix1>



EXP_STA_SCAN_ds<suffix2>

...

The **datastream=** command

See this commit log for brief usage/help/explanation:

<https://github.com/jive-vlbi/jive5ab/commit/74c3f1b35bd258f344def8a50cae805381420549>

VDIF {threadId=0, stationId="Wf"}
sent by 192.168.178.1:5000

VDIF {threadId=1, stationId="Wf"}
sent by 192.168.178.2:5000

data stream #0

VDIF {**threadId**=0, **stationId**="Wf"}
sent by 192.168.178.1:5000

VDIF {**threadId**=1, **stationId**="Wf"}
sent by 192.168.178.2:5000

data stream #0

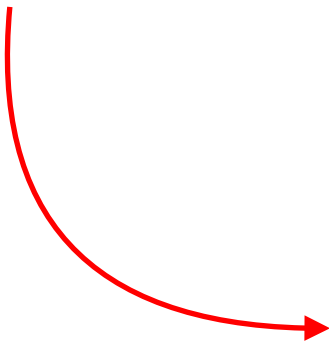
VDIF {threadId=0, stationId="Wf"}
sent by 192.168.178.1:5000

VDIF {threadId=1, stationId="Wf"}
sent by 192.168.178.2:5000

data stream #0


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

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



Match on any/all of:

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

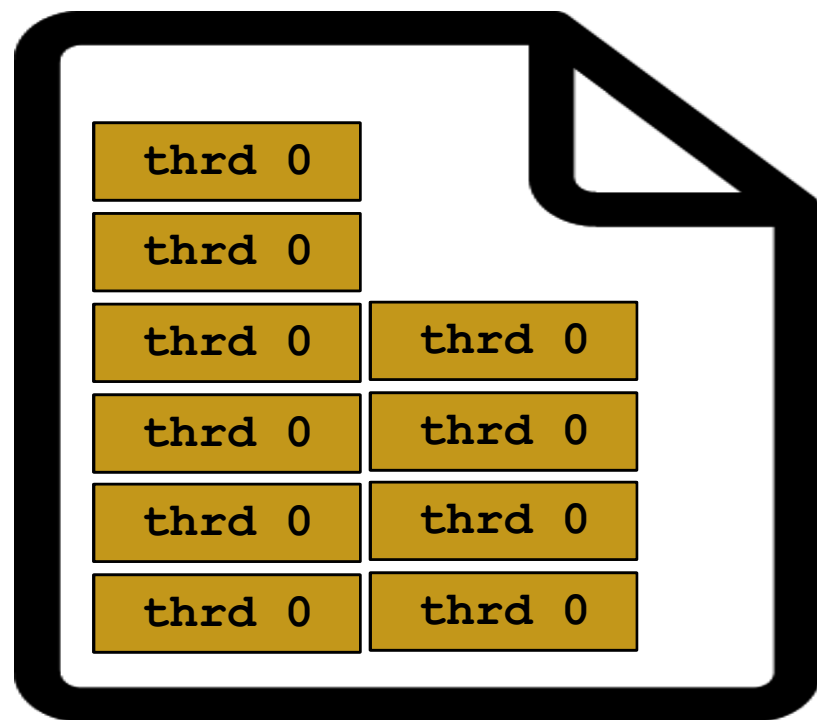
See this commit log for brief usage/help/explanation:

<https://github.com/jive-vlbi/jive5ab/commit/74c3f1b35bd258f344def8a50cae805381420549>

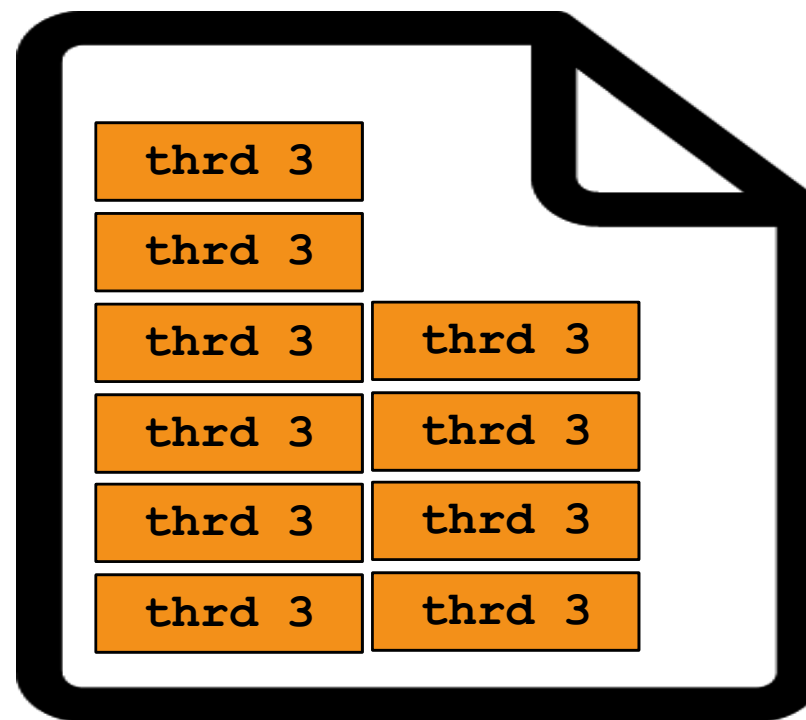
```
datastream = add : <suffix> : <match specification> [: <more matches> ] ;
```

See this commit log for brief usage/help/explanation:

<https://github.com/jive-vlbi/jive5ab/commit/74c3f1b35bd258f344def8a50cae805381420549>



EXP_STA_SCAN_ds<suffix1>



EXP_STA_SCAN_ds<suffix2>

...


```
scan_set = <scan name> ;
```



```
scan_set = <scan name> ;
```

```
↳ automatically find : <scan name>_dsA
```

```
↳ automatically find : <scan name>_dsB
```

```
↳ automatically find : <scan name>_ds...
```

scan_check?

↳ returns result for the *collection* of sub-recordings

```
disk2{file|net}
```

↳ operate on the *collection* of sub-recordings

the **scan_check?** algorithm:

- re-invented + re-implemented
 - handle >1 VDIF threads in one “recording”
 - handle (very) high data rates

the **scan_check?** algorithm:

- re-invented + re-implemented
 - handle >1 VDIF threads in one “recording”
 - handle (very) high data rates

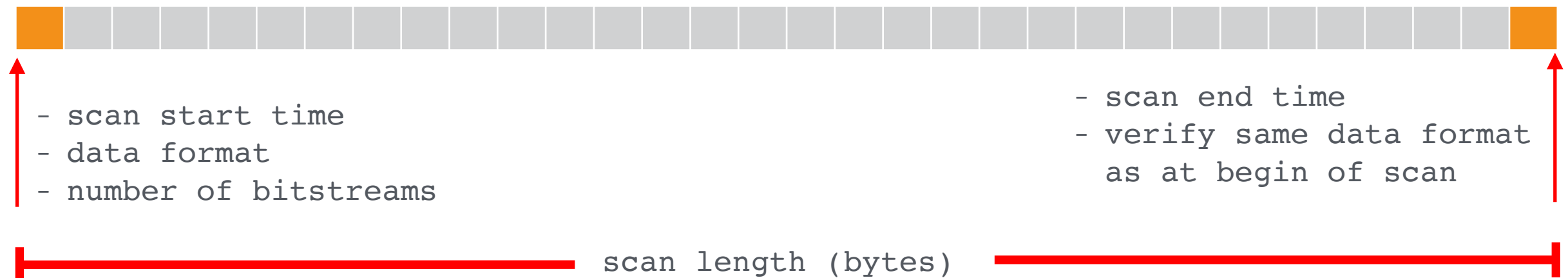
EXP_STA_SCAN (vbs/mk6)



the **scan_check?** algorithm:

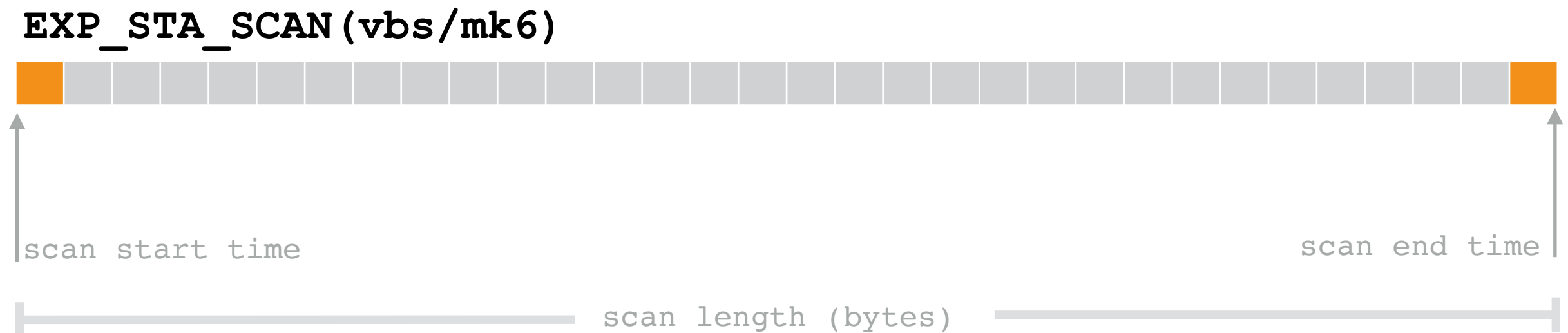
- re-invented + re-implemented
 - handle >1 VDIF threads in one “recording”
 - handle (very) high data rates

EXP_STA_SCAN(vbs/mk6)



the **scan_check?** algorithm:

- re-invented + re-implemented
 - handle >1 VDIF threads in one “recording”
 - handle (very) high data rates



$$\hookrightarrow \text{recorded data rate} = \frac{\delta \text{byte}}{\delta t}$$

the **scan_check?** algorithm:

- re-invented + re-implemented
 - handle >1 VDIF threads in one “recording”
 - handle (very) high data rates

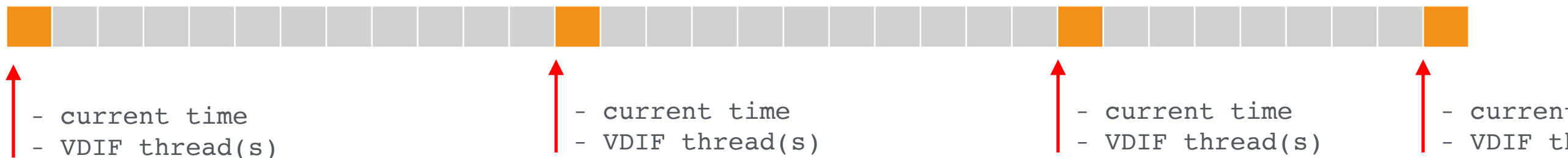
EXP_STA_SCAN (vbs/mk6)



the **scan_check?** algorithm:

- re-invented + re-implemented
 - handle >1 VDIF threads in one “recording”
 - handle (very) high data rates

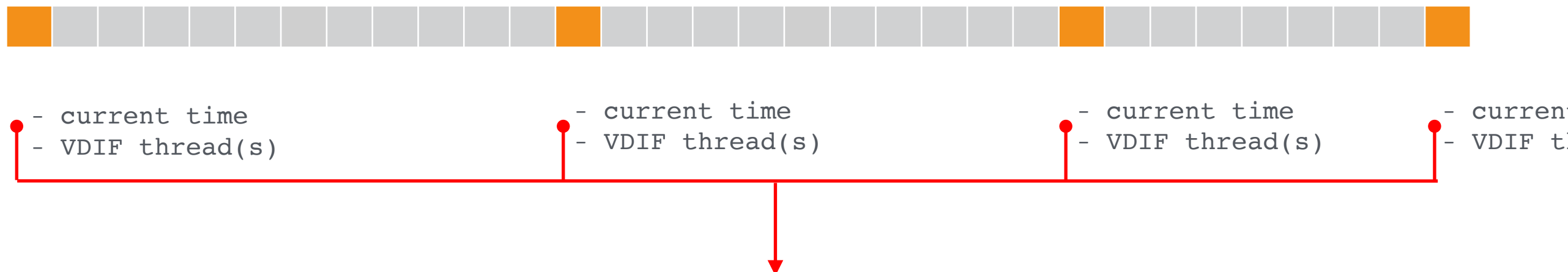
EXP_STA_SCAN (vbs/mk6)



the **scan_check?** algorithm:

- re-invented + re-implemented
 - handle >1 VDIF threads in one “recording”
 - handle (very) high data rates

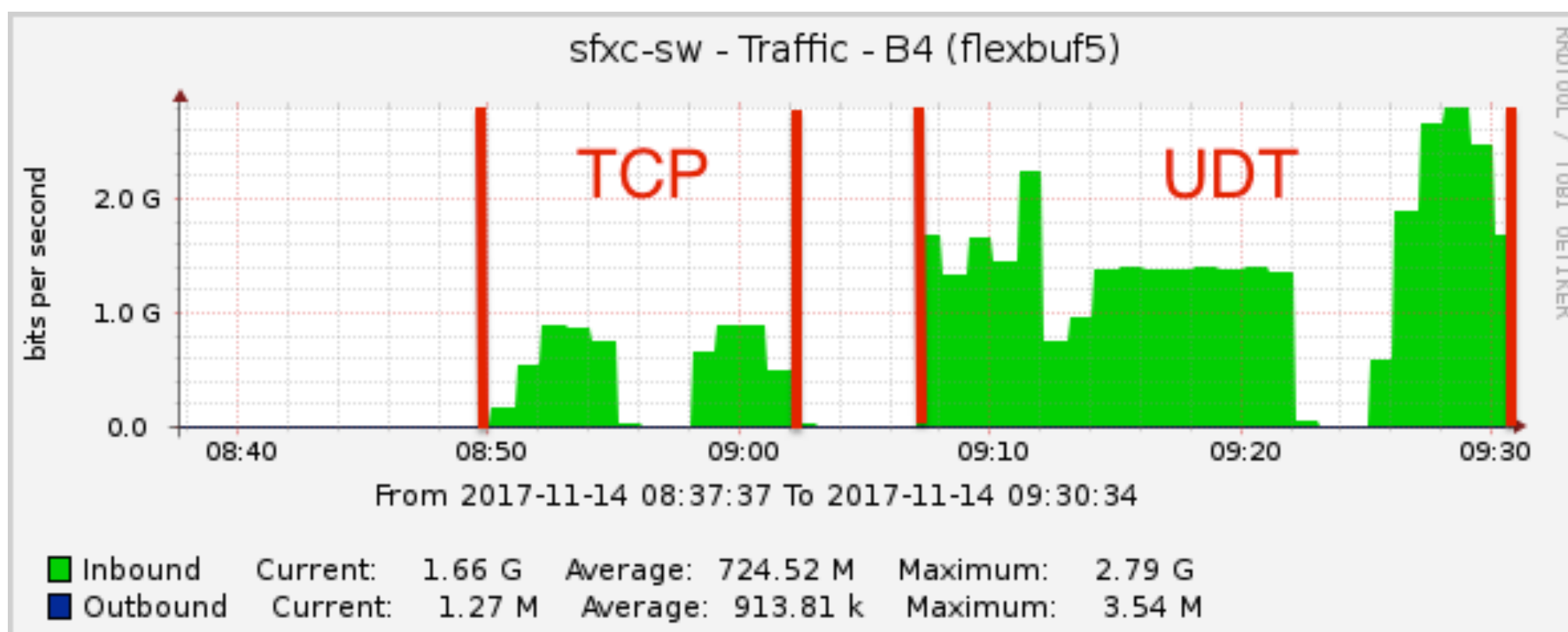
EXP_STA_SCAN(vbs/mk6)



Combined result:

- VDIF threads
- data rate per thread

Very hot off
the press ...



Private

github.com/Haivision/srt

iptables - Fail2banSFXC FringesJIVE ZoomRoom#1Private ZoomSharePointJIVESharePointsZoomsSlacks

READMEMPL-2.0 license

Secure Reliable Transport (SRT) Protocol

[About SRT](#) | [Features](#) | [Getting Started](#) | [Build Instructions](#) | [Sample Apps and Tools](#) | [Contribute](#) | [License](#) | [Releases](#)

SRT

SECURE
RELIABLE
TRANSPORT

LicenseMPLv2.0releasev1.5.3quality gatefailedcodecov65%

?

Windowspassing

Ubuntu 23.04 package1.5.1Fedora 37 package1.5.1Debian Testing1.5.3-1+b2Homebrew package1.5.3Vcpkg package1.5.3ConanCenter package1.5.3

What is SRT?

Secure Reliable Transport (SRT) is a transport protocol for ultra low (sub-second) latency live video and audio streaming, as well as for generic bulk data transfer^[1]. SRT is available as an open-source technology with the code on GitHub, a published [Internet Draft](#), and a growing [community of SRT users](#).

SRT is applied to contribution and distribution endpoints as part of a video stream workflow to deliver the best quality and lowest latency video at all times.

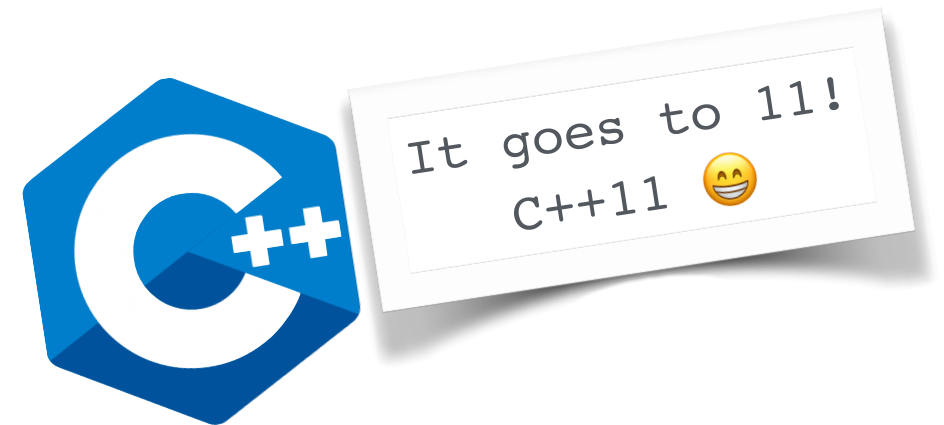
Secure	Encrypts video streams
Reliable	Recovers from severe packet loss
Transport	Dynamically adapts to changing network conditions

In live streaming configurations, the SRT protocol maintains a constant end-to-end latency. This allows the live stream's signal characteristics to be recreated on the receiver side, reducing the need for buffering. As packets are streamed from source to destination, SRT detects and adapts to real-time network conditions between the two endpoints. It helps compensate for jitter and bandwidth

https://github.com/Haivision/srt

“Some people have asked us why we're using the UDT library within our SRT protocol. Actually, some people claimed that SRT is just a slightly modified version of UDT and that UDT is known to be useless for live video transmission. Guess what, the latter is true. UDT has been designed for high throughput file transmission over public networks. However, SRT is far from being a slightly modified version of UDT. I'll get into the details, but will start with a little bit of history.”

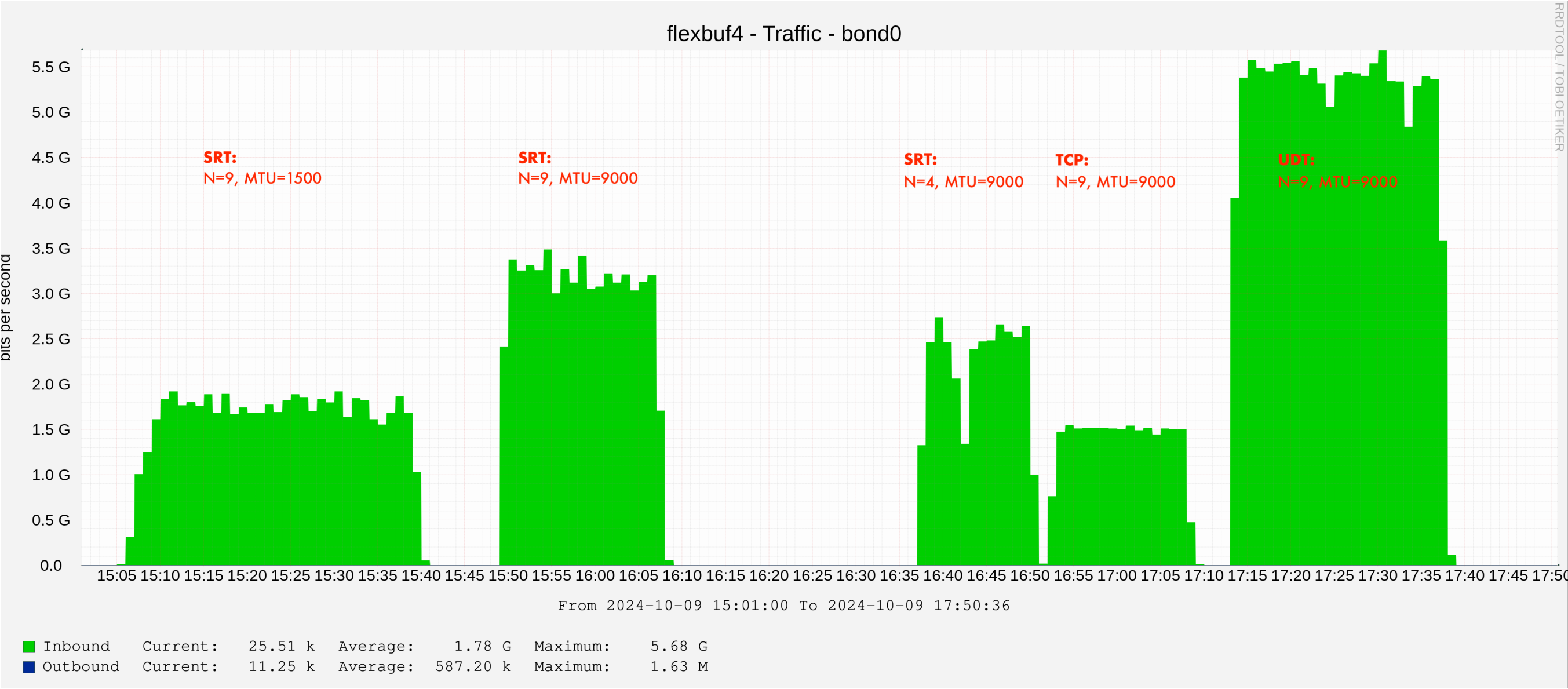
*“Some people have asked us **why** we're using the **UDT library within our SRT protocol**. Actually, some people claimed that SRT is just a slightly modified version of UDT and that UDT is known to be useless for live video transmission. Guess what, the latter is true. UDT has been designed for high throughput file transmission over public networks. However, **SRT is far from being a slightly modified version of UDT**. I'll get into the details, but will start with a little bit of history.”*



```
$> git clone --branch srt_protocol https://github.com/jive-vlbi/jive5ab
```

```
net_protocol = srt [ : options ] ;
```

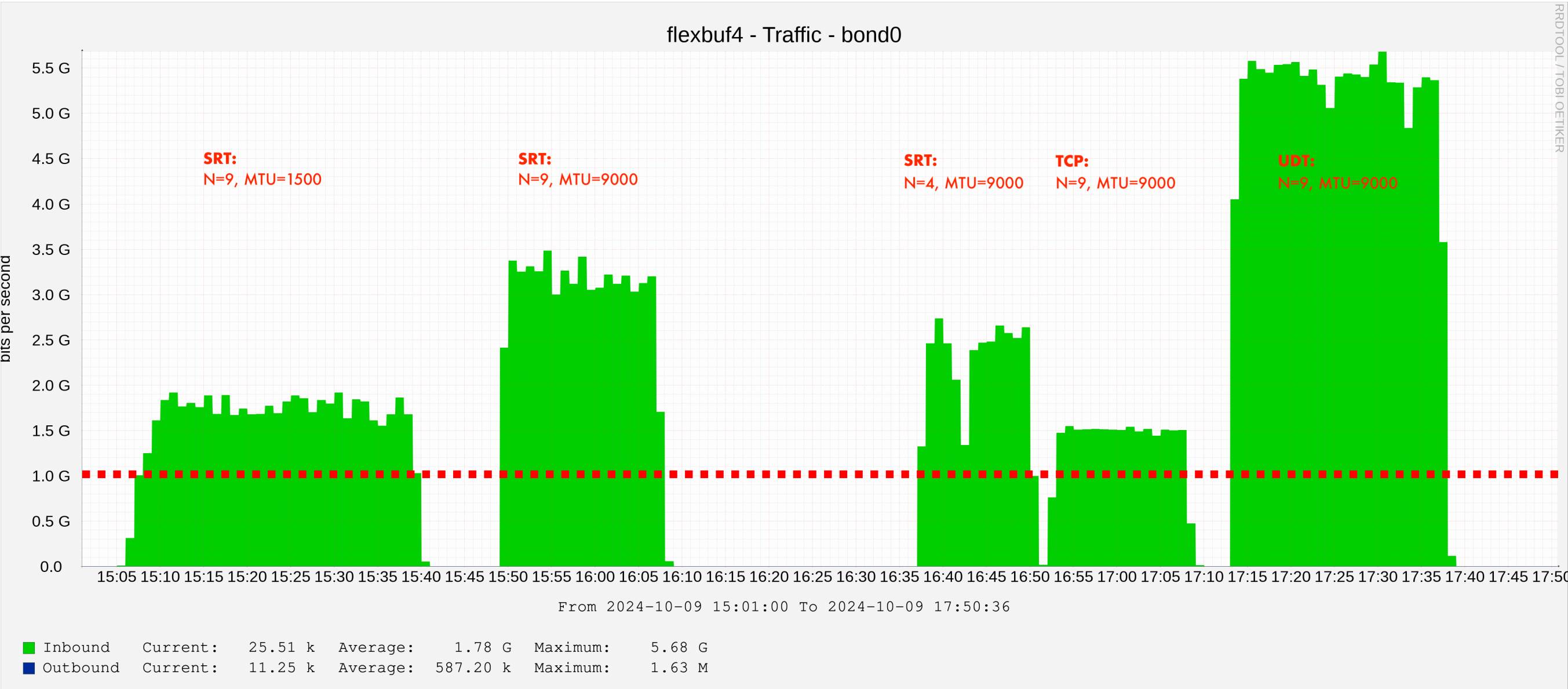
All transfers **vbs2vbs**, HartRAO(SA) ⇒ JIVE(NL)



Protocol	SRT		SRT		SRT	TCP		UDT
N parallel	9		9		4	9		9
MTU	1500		9000		9000	9000		9000

```
$> m5copy -<proto> -m <MTU> -n <N parallel> vbs://evlbi-251.hartrao.ac.za/v626co* vbs://flexbuf4.jive.nl/null/
```

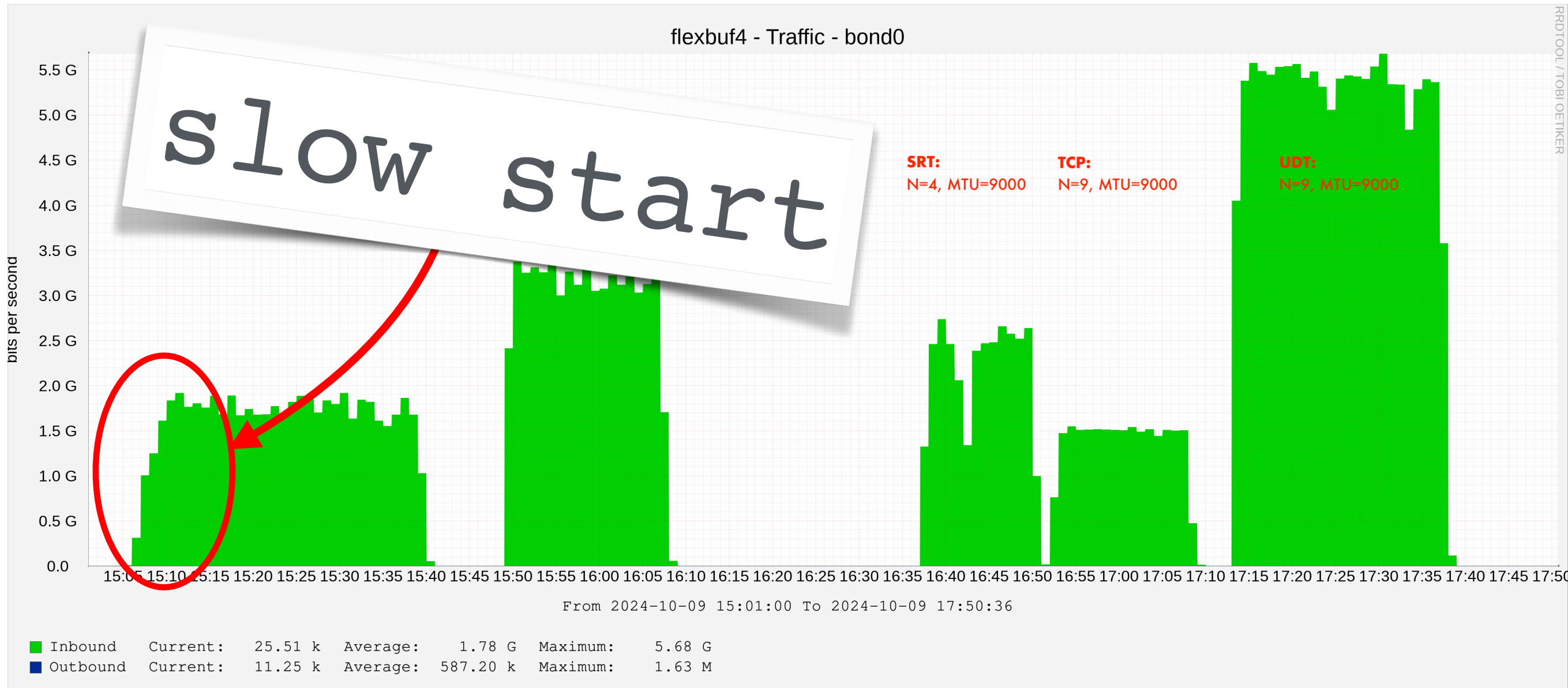
All transfers **vbs2vbs**, HartRAO(SA) ⇒ JIVE(NL)



Protocol	SRT		SRT		SRT	TCP		UDT
N parallel	9		9		4	9		9
MTU	1500		9000		9000	9000		9000

```
$> m5copy -<proto> -m <MTU> -n <N parallel> vbs://evlbi-251.hartrao.ac.za/v626co* vbs://flexbuf4.jive.nl/null/
```

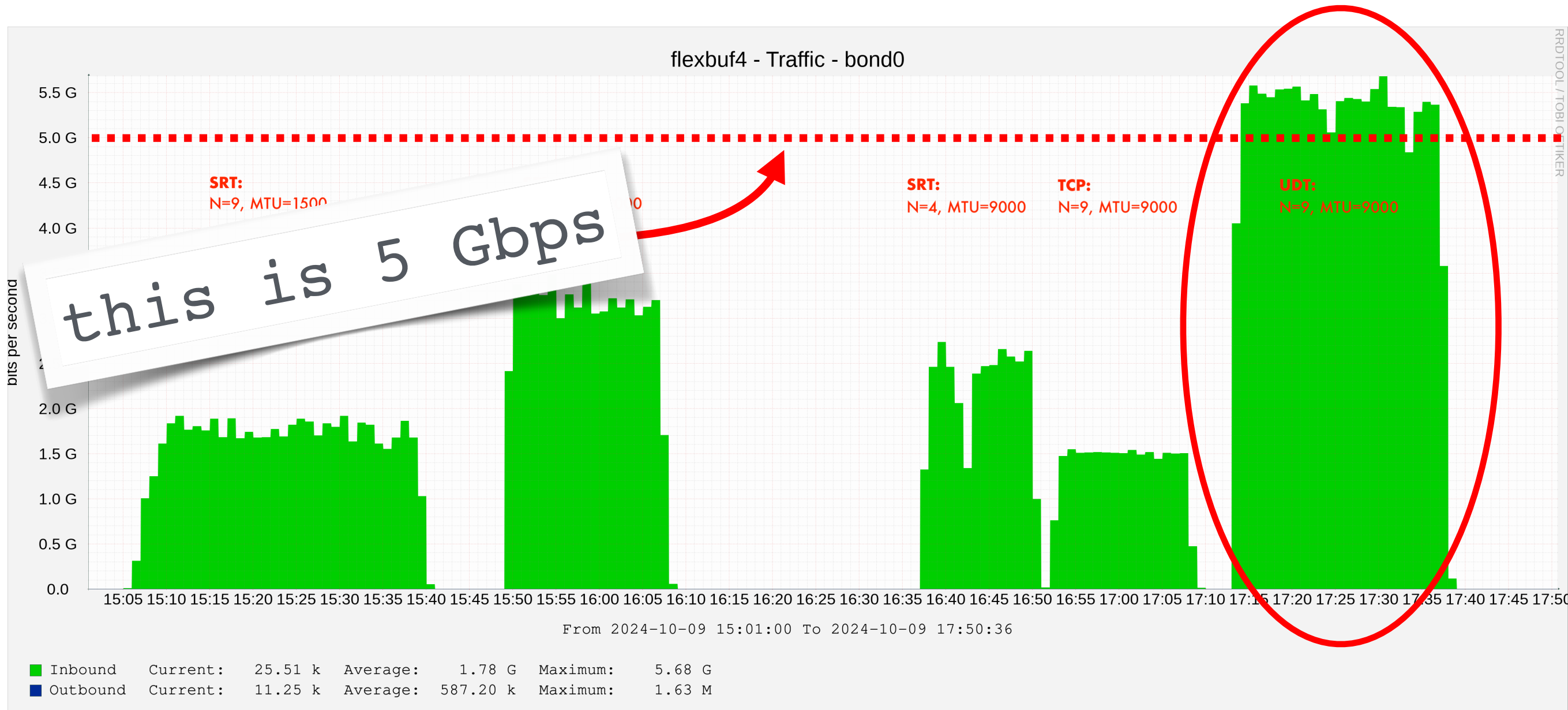
All transfers **vbs2vbs**, HartRAO(SA) ⇒ JIVE(NL)



Protocol	SRT		SRT		SRT	TCP		UDT
N parallel	9		9		4	9		9
MTU	1500		9000		9000	9000		9000

```
$> m5copy -<proto> -m <MTU> -n <N parallel> vbs://evlbi-251.hartrao.ac.za/v626co* vbs://flexbuf4.jive.nl/null/
```

All transfers vbs2vbs, HartRAO(SA) ⇒ JIVE(NL)



```
$> m5copy -<proto> -m <MTU> -n <N parallel> vbs://evlbi-251.hartrao.ac.za/v626co* vbs://flexbuf4.jive.nl/null/
```

```
# create temporary mountpoint
```

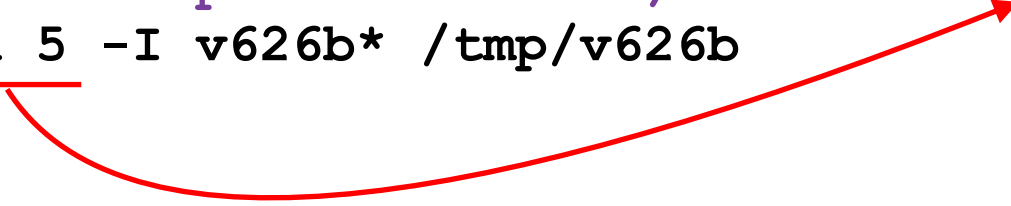
```
$> mkdir /tmp/v626b
```

```
# index + mount only subset of recordings
```

```
 #(the scans for experiment v626b)
```

```
$> vbs_fs -n 5 -I v626b* /tmp/v626b
```

this makes vbs_fs go faster



```
# now we have BIG files only
```

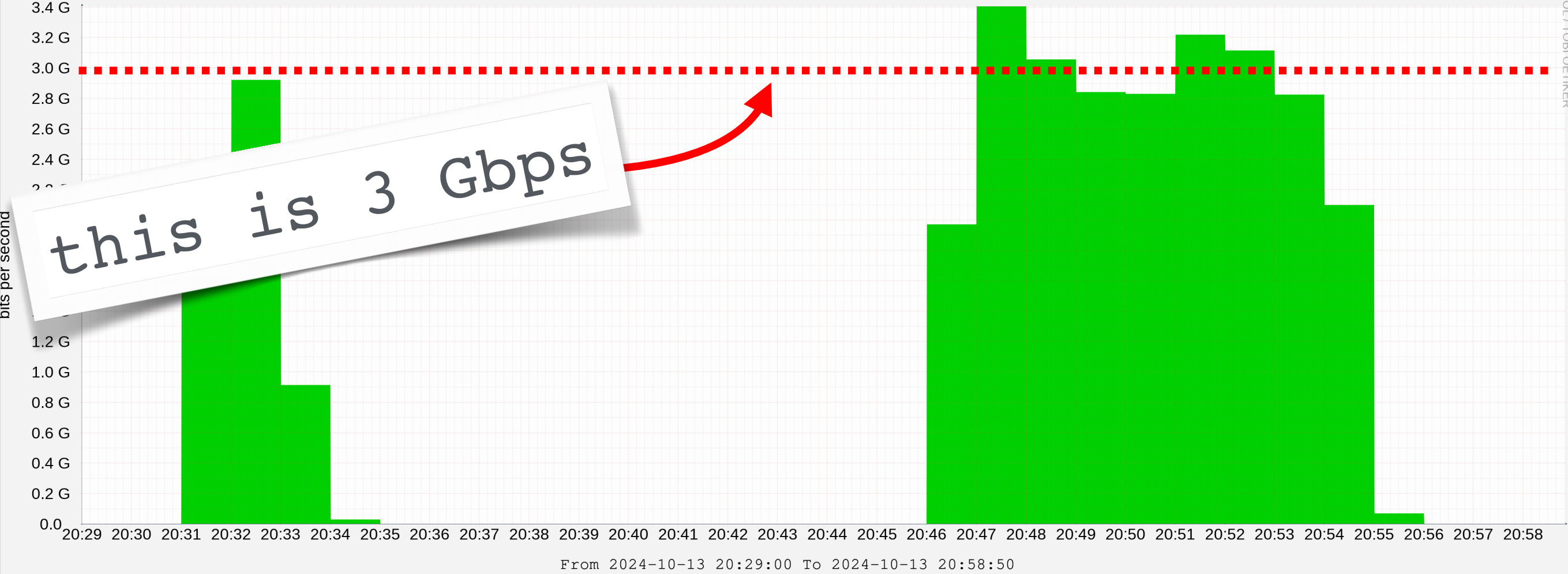
```
$> ls -lh /tmp/v626b
```

```
-rw-r--r-- 0 oper oper 12G May 26 05:30 v626b_hh_no0145
-rw-r--r-- 0 oper oper 27G May 26 05:34 v626b_hh_no0146
-rw-r--r-- 0 oper oper 12G May 26 05:36 v626b_hh_no0147
-rw-r--r-- 0 oper oper 27G May 26 05:40 v626b_hh_no0148
-rw-r--r-- 0 oper oper 12G May 26 05:42 v626b_hh_no0149
-rw-r--r-- 0 oper oper 27G May 26 05:46 v626b_hh_no0150
```

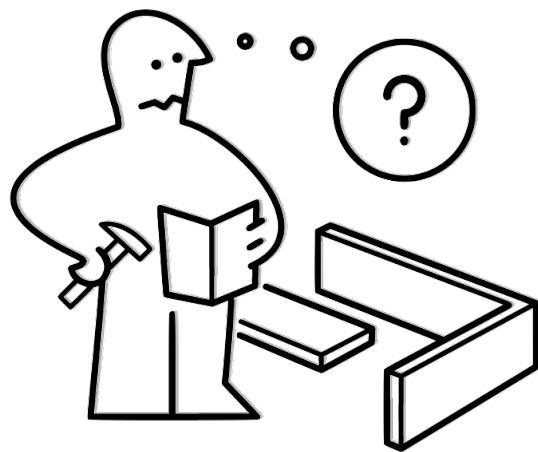
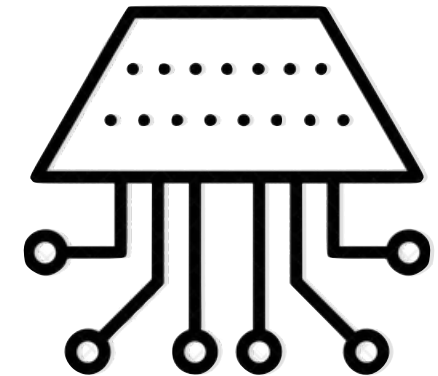
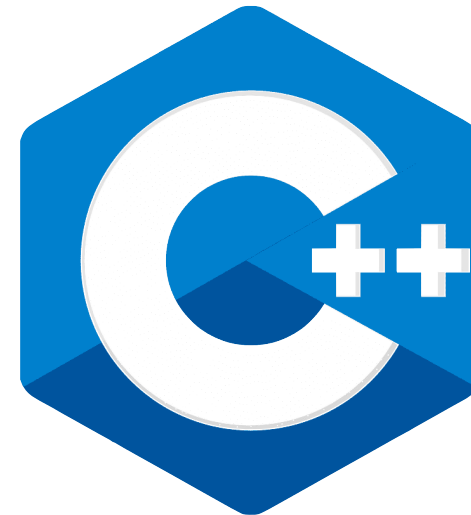
```
# let jive5ab read from the vbs_fs mountpoint
```

```
$> m5copy -srt -m 9000 file://localhost/tmp/v626b/* file://flexbuf4/dev/null
```

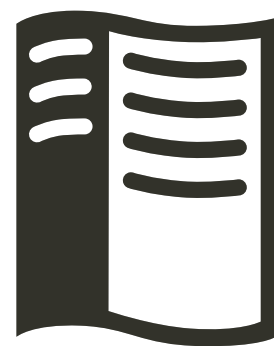

flexbuf4 - Traffic - bond0



Inbound	Current:	27.23 k	Average:	1.03 G	Maximum:	3.40 G
Outbound	Current:	12.09 k	Average:	141.11 k	Maximum:	423.61 k



CMake



Read *the* Docs

```
2024.009.15:46:35.17/mk5=version?
2024.009.15:46:35.17/mk5/!version? 0 : jive5ab :
                                3.1.0-dev : 64bit : Debug :
                                koll : 08-Jan-2024 :
                                12h26m00s : nossapi :
                                /usr/local/src/etransfer ;

2024.009.15:48:37.54;mk5=scan_check?
2024.009.15:48:37.59/mk5/scan_check? 0 :?:
                                xxx_xx_no0001 : VDIF : 128 :
                                2024y009d15h43m13.3117s :
                                5.9812s : 256Mbps :
                                1060224 : 8032 ;

2024.009.15:48:42.56;mk5=evlbi?
2024.009.15:48:42.56/mk5/evlbi? 0 : total: 3062244:
                                loss : 0 (0.00%) :
                                out-of-order: 0 ( 0.00%) :
                                extent: 0 seqnr/pkt ;
```

Thanks
for
attention!