Bit modes

The station to correlator data format

André Offringa -- Kapteyn Astronomical Institute

Lofar status meeting 23-02-2011

About bit modes

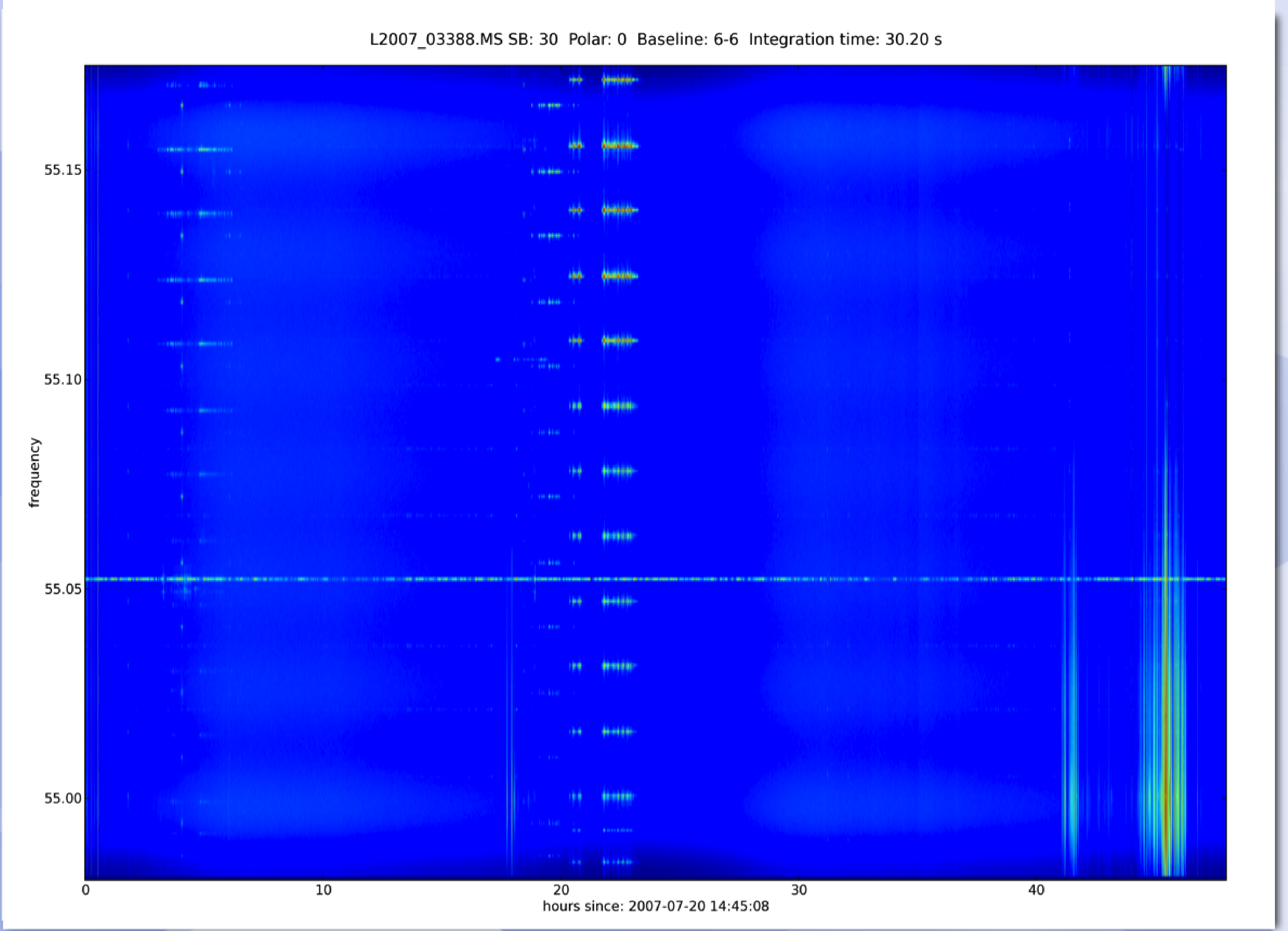
- The nr of beams and bandwidth are limited by the transport from station to BG.
- We can send 244 beamlets: for example 48 beams with 1 MHz or 48 MHz in 1 beam.
- We currently send 16 bit samples
- If samples can be send with 8 or 4 bits, we can send 2 resp 4 times as many
- But what is the effect on the data?

Current RCP data format

- Stations send a stream of “beamlets”
- Each beamlet contains a stream of sub-band data (1024 clock ticks / sample)
- Each time step consists of
  - Two polarizations with
  - real and imaginary values of
  - 16 bit integers (in C: “signed short”)
  - Thus 8 bytes/time step/beamlet
  - 8 x (200Ms/1024) x #beamlets(=244) B/s
  - ≈ 380 MB/s (=3.04 gbit)

Data used for analysis

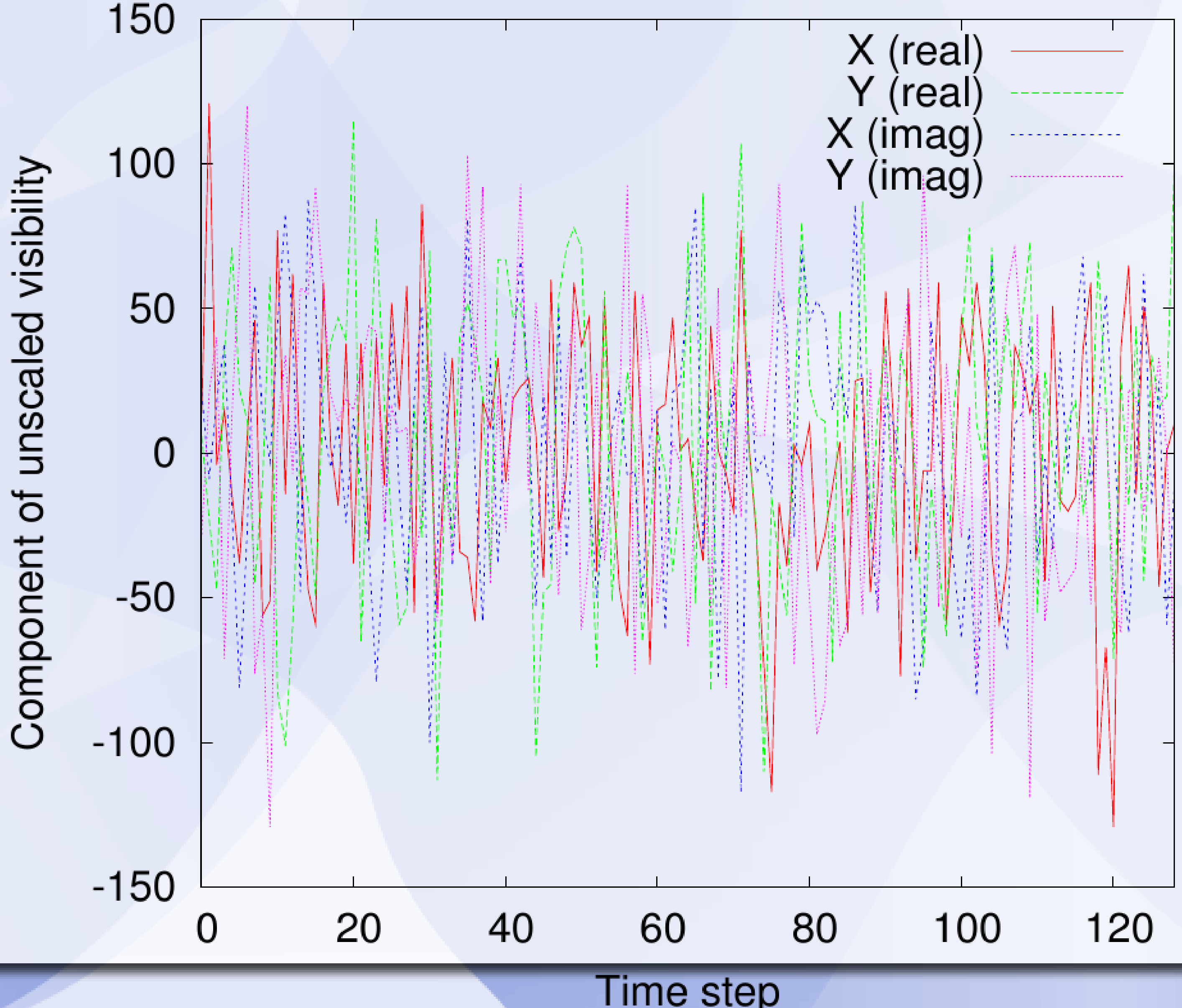
- Set recorded for RFI analysis by Rob van Nieuwpoort and John Romein
- “Raw” RCP data
- The set:
  - 5 LBA sub-bands
    - 27 MHz: 27 MC
    - 36 MHz: Model airplanes
    - 50 MHz: “clean” band
    - 55.1, 55.3 MHz: TV
  - 3 stations: CS004, RS205, RS208


L2007_03388.MS SB: 30 Polar: 0 Baseline: 6-6 Integration time: 30.20 s
frequency
55.15
55.10
55.05
55.00
0
10
20
30
40
hours since: 2007-07-20 14:45:08

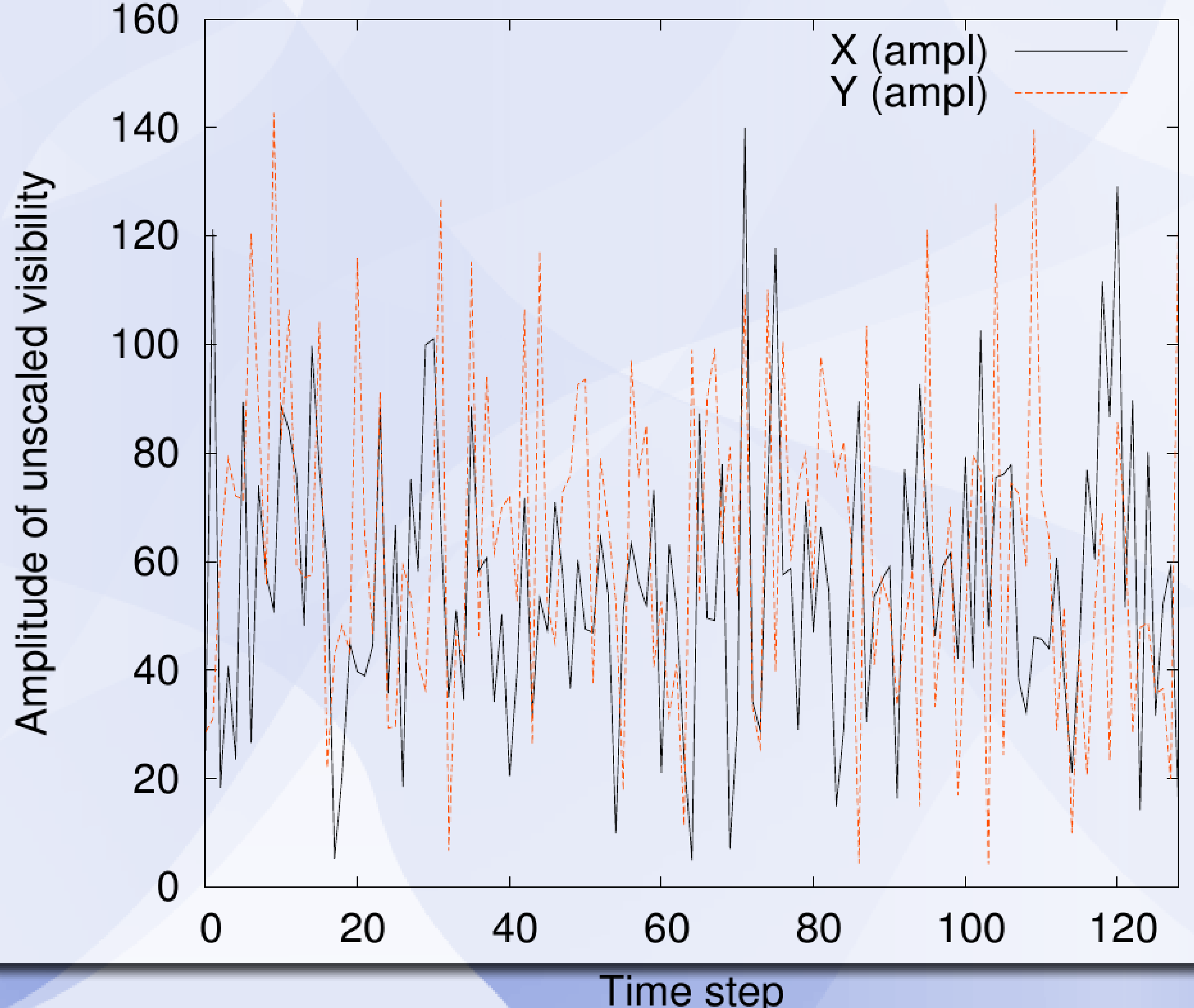
Beamlet stream

- What does such a stream look like?

Beamlet stream



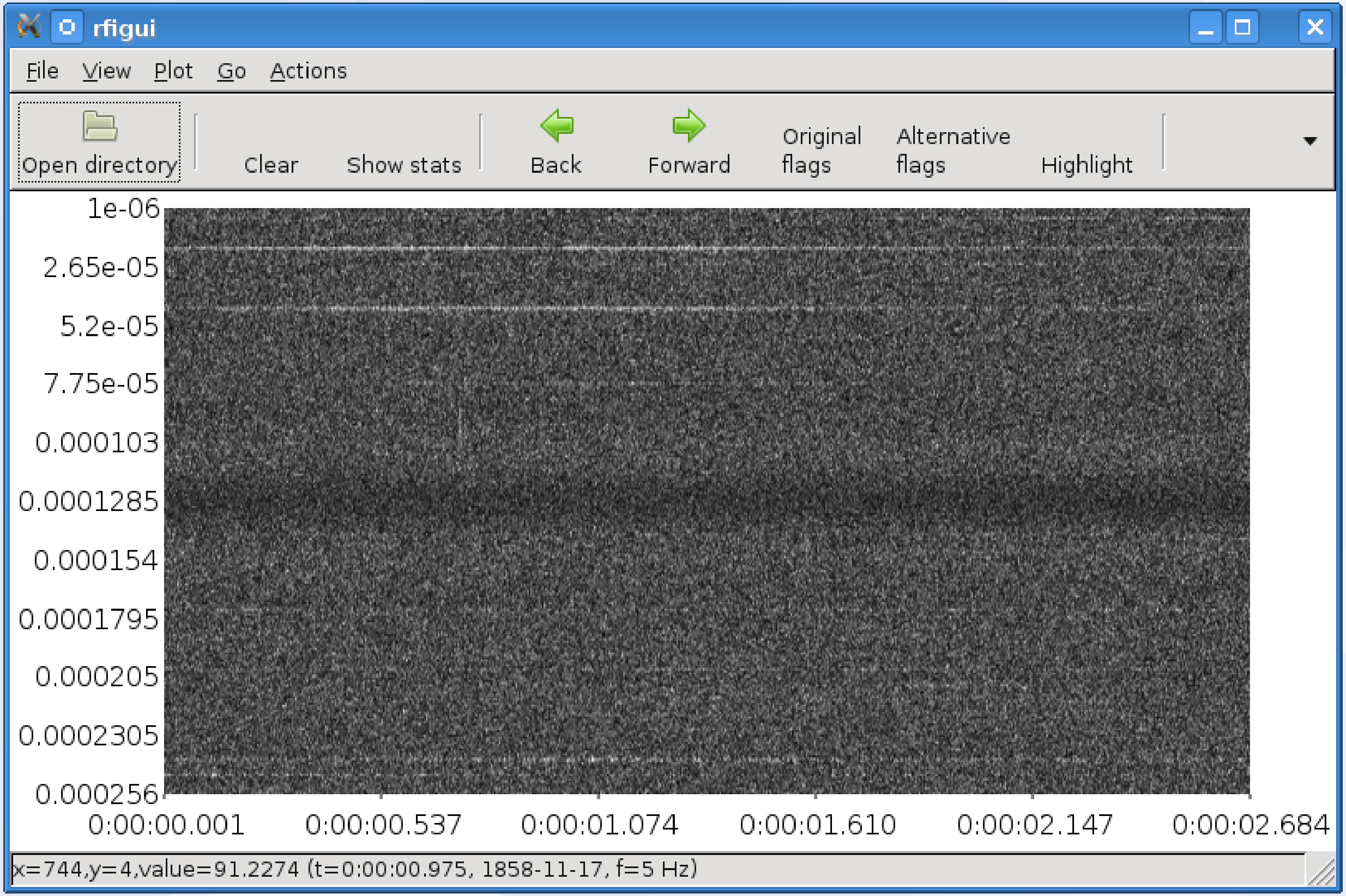
Beamlet stream



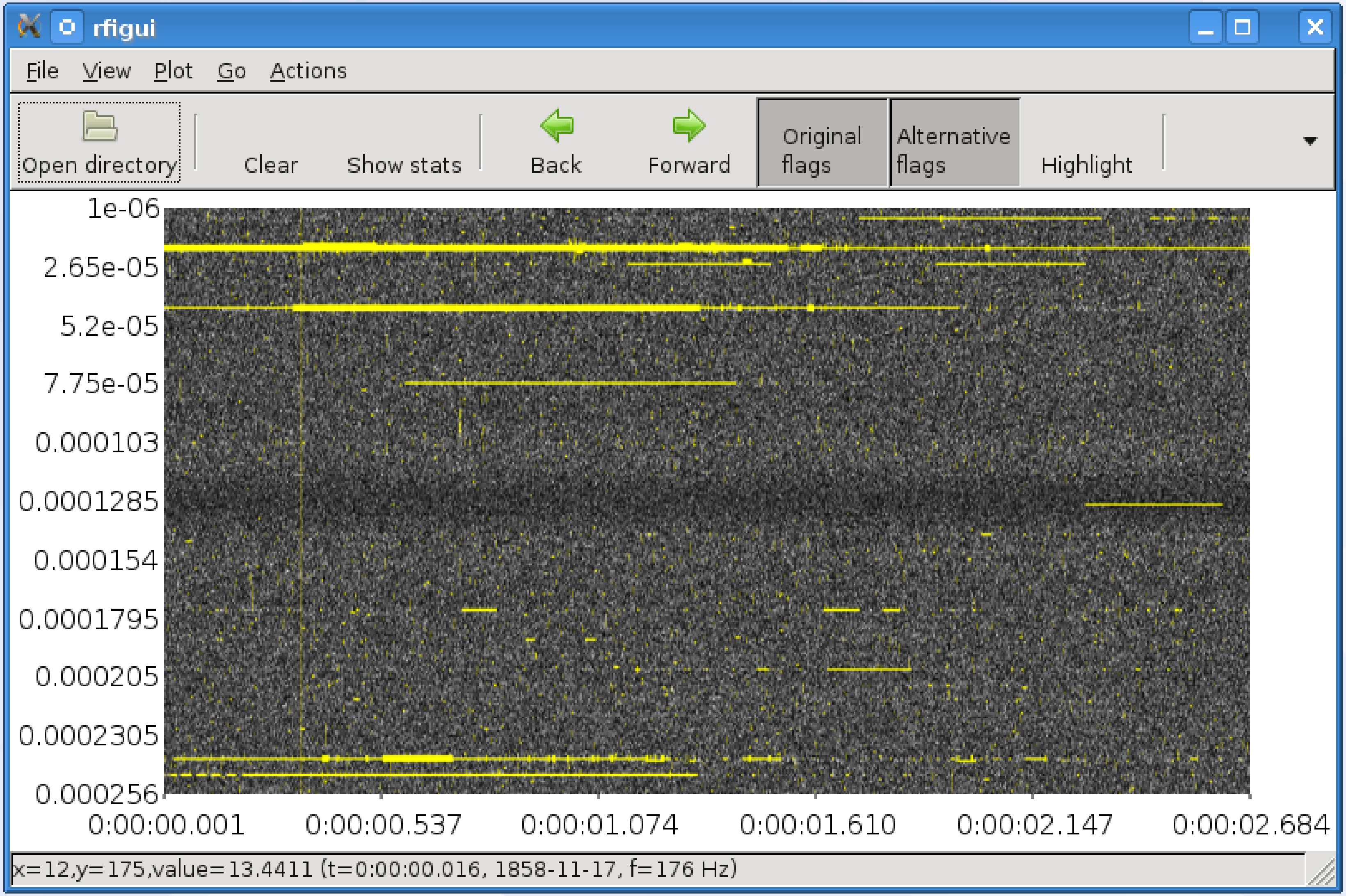
Implemented in “RFI Gui”

- The RCP format was implemented in the RFI Gui (File → Open file → pick file with extension “.raw”)
- Different modes to open RCP set:
  - A single beamlet stream
  - Concatenate beamlets as channels
  - FFT a single beamlet
- All from a single station

Implemented in “RFI Gui”



Testing flagging strategies...

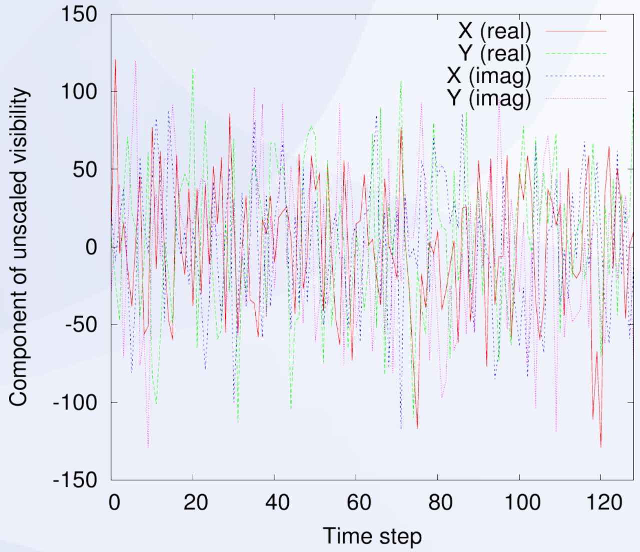


How to discard 8 or 12 bits?

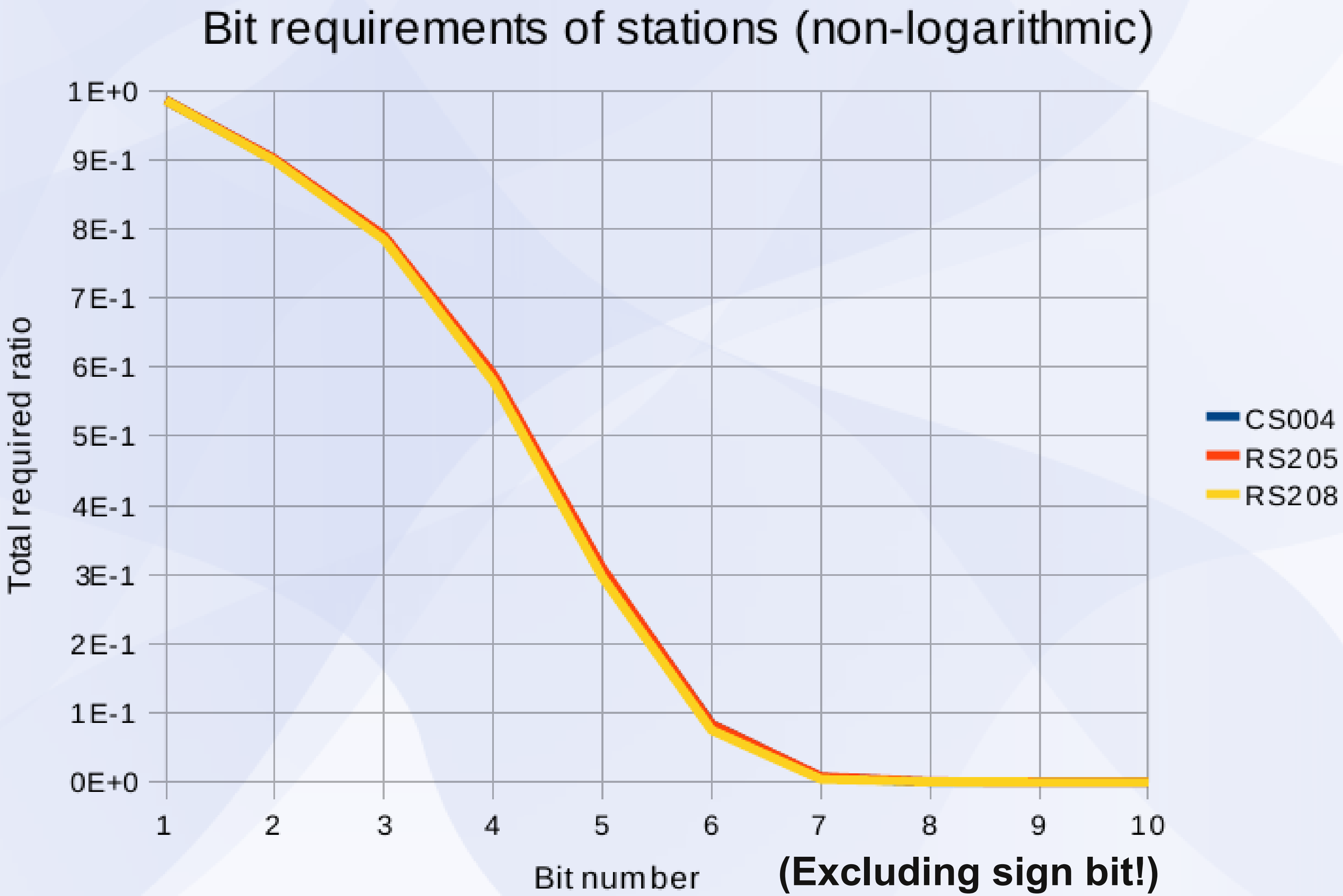
- Common ways from information theory:
  - Encoding
    - E.g., Hofman or Rice encoding
  - Prediction
    - E.g. as in FLAC: linear prediction
  - Quantization
    - (Possibly Non-linear) scaling and truncation
  - Encapsulation
    - Dynamicly changing above parameters

How to discard 8 or 12 bits?

- First impression:
  - Encoding not trivial
  - Values are not easily predictable – near Nyquist rate
  - Distribution is not uniform → uniform quantization (“signed short”) not ideal.
  - Values “seldom” use more then 8 bits
- Proper quantization most trivial solution
  - Allow dynamic changes?

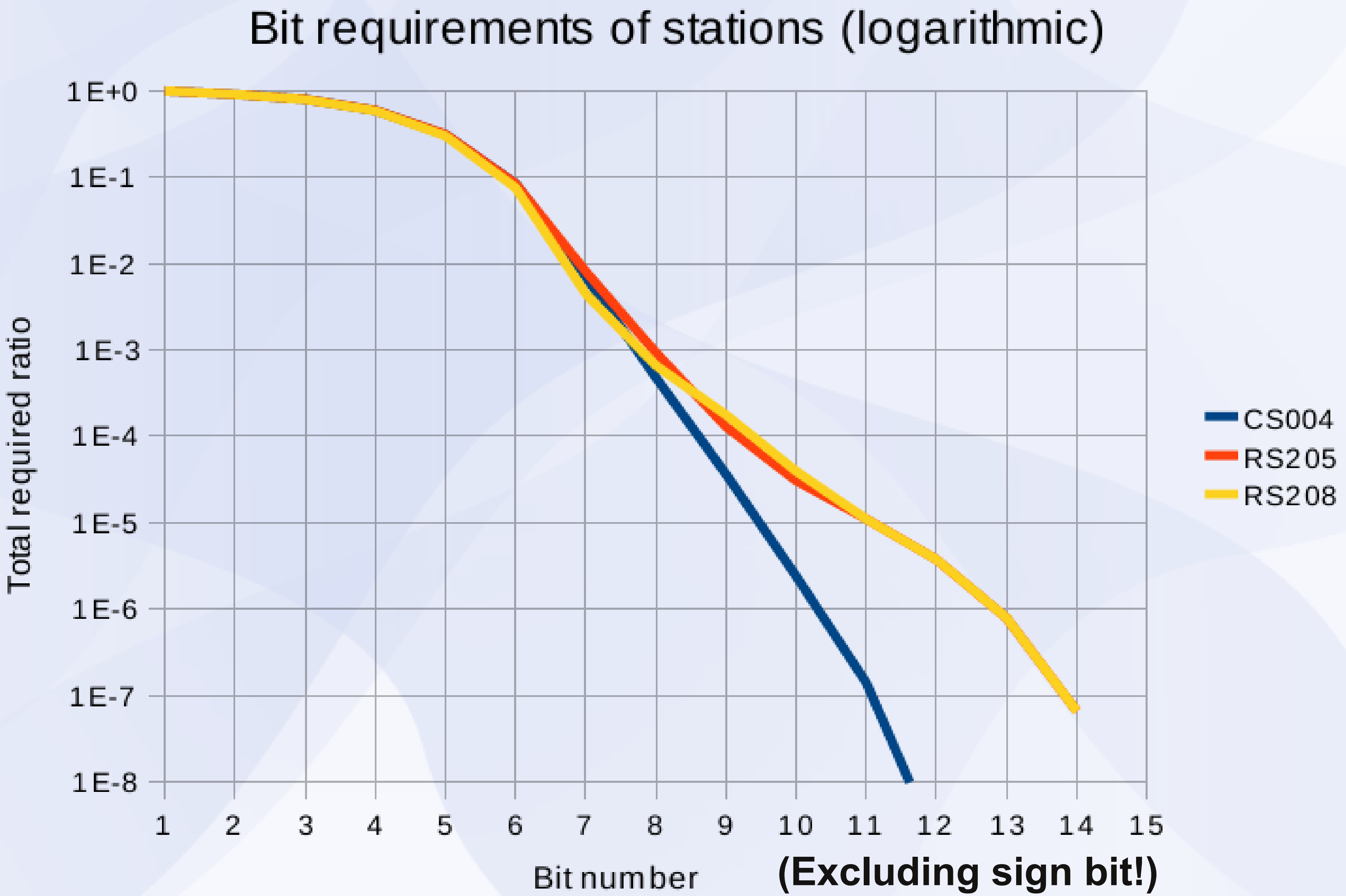


How to quantize?



(Excluding sign bit!)

How to quantize?



(Excluding sign bit!)

How to quantize to 8 bits?

- 1 out of 1000 samples uses > 8 bits
- Rough est error of clipping to 8 bits:
  - Single real sample with value of v:

    $$\epsilon(v) = max(0, |v| - 128)$$

    (assuming if |v| > 127 then $|\bar{v}|$ = 127)
  - Total absolute error:

    $$E(\epsilon) = \int P(v)\epsilon(v)\,dv \approx 1/1000 \times 64 \approx 0.06$$
  - Total relative error: (~SNR loss)

    $$Q(\epsilon) = E(\epsilon)/E(|v|) \approx 0.06/10 \approx \textbf{1\%}$$

How to quantize to 8 bits?

- 1 out of 10000 samples uses > 9 bits
- Error of truncation of bit 10-16:

$$\epsilon(v) = max(0, |v| - 256)$$
$$E(\epsilon) = \int P(v)\epsilon(v)\,dv \approx 1/10000 \times 128 \approx 0.013$$
$$Q_\epsilon = E(\epsilon)/E(|v|) \approx 0.013/10 \approx 0.1\,\%$$

- Error of removing bit 1:

$$\epsilon(v) = |v| \, mod \, 2$$
$$E(\epsilon) \approx 0.5$$
$$Q_\epsilon \approx 0.5/10 \approx \textbf{5\%}$$

- Hence, bit 1 is more important than bit 8

How to quantize to 8 bits?

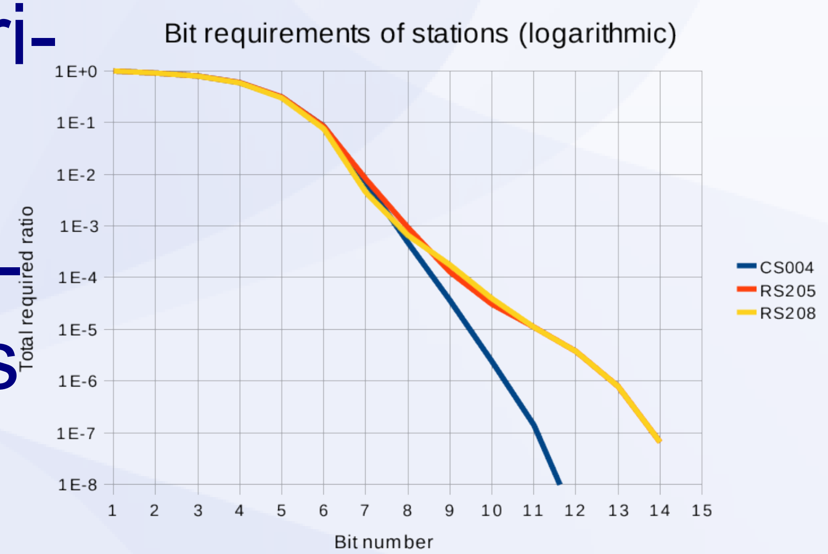
- Fourier transform is uniform, thus total error in real domain = total error in Fourier domain
- Quantization / clipping high values effect all channels
- Clipping high values compares to slightly non-linear system: RFI “harmonics” (→flagging will lower total error)

How to quantize to 8 bits?

- Clipping values to least significant 8 bits seems a good first approach.
- This results in a **1%** expected error (thus increase of noise).
- Hence, would make LOFAR almost twice as efficient.

How to quantize to 8 bits?

- The minimum total error can be achieved by integrating the distribution.



- E.g.: exponential quantization in higher values

$$v \rightarrow \bar{v}$$

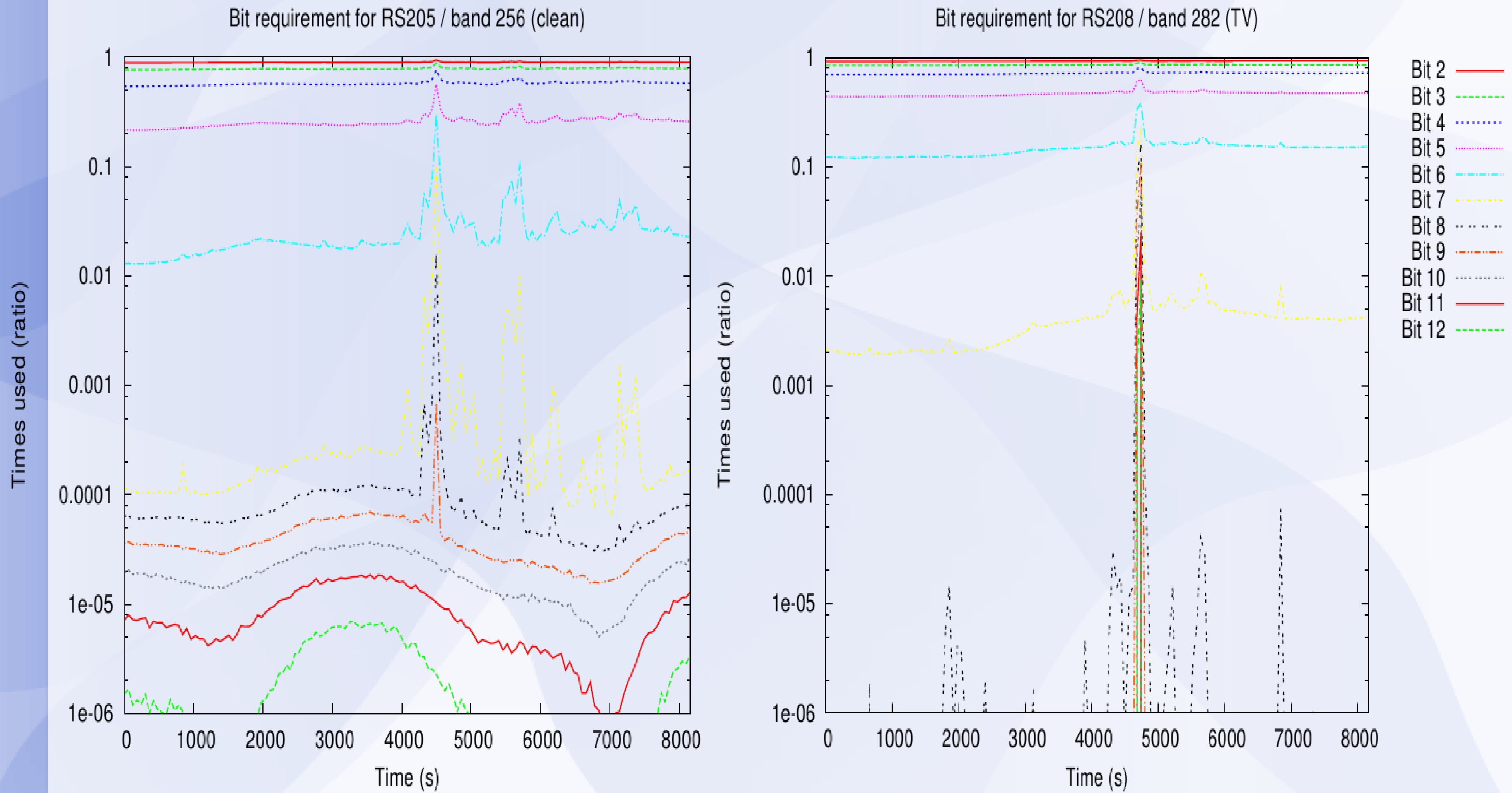
$$0\text{-}64 \rightarrow v$$

$$65\text{-}65536 \rightarrow {}^{d}\log(v-64 + c)+64-{}^{d}\log(c)$$

approx good values: d=1.045, c=20

- Total error now ~ **0.1%**
- Technologically feasible (by table lookup)

More on quantization error



4 bit mode

- Most trivial solution: remove bits
- One solution is to “keep” bits 3, 4 and 5 and the sign bit.
- This results in a **20%-30%** avg error/sample.
- This does imply it is most “efficient” to look at four fields at the same time
- Note that a *signed short* can represent more negative values then positive values

4 bit mode

- More advanced solution: table lookup

..-257	-8	0..1	0
-256..-33	-7	2..4	1
-32..-21	-6	5..8	2
-20..-14	-5	9..13	3
-13..-9	-4	14..20	4
-8..-5	-3	21..32	5
-4..-2	-2	33..256	6
-1..0(**!**)	-1	257 ..	7

- Little better, about **10-20%** expected error

Bit mode implementation status

- Who is doing what?
- Info from Stefan Wijnholds:
  - Arie Doorduin is working on implementation of bit modes.
  - Recent document by Eric Kooistra on required changes to FPGA pipeline
  - Where and how to round numbers is being discussed, Stefan is also involved.

Conclusions

- Trivial clipping to 8 bits results in **1%** error
- Mapping function with table lookup can improve this to **0.1%** error.
- 4 bits is more complicated, but still efficient, leading to about **20%** error.
- (Dynamic) optimization per station, sub-bands might improve error further.
- These are all preliminary results: further testing is needed for optimal strategies.