Innovations with YAML/CABAC in H.264/AVC software decoding

Thibault Raffaillac, Ph.D.

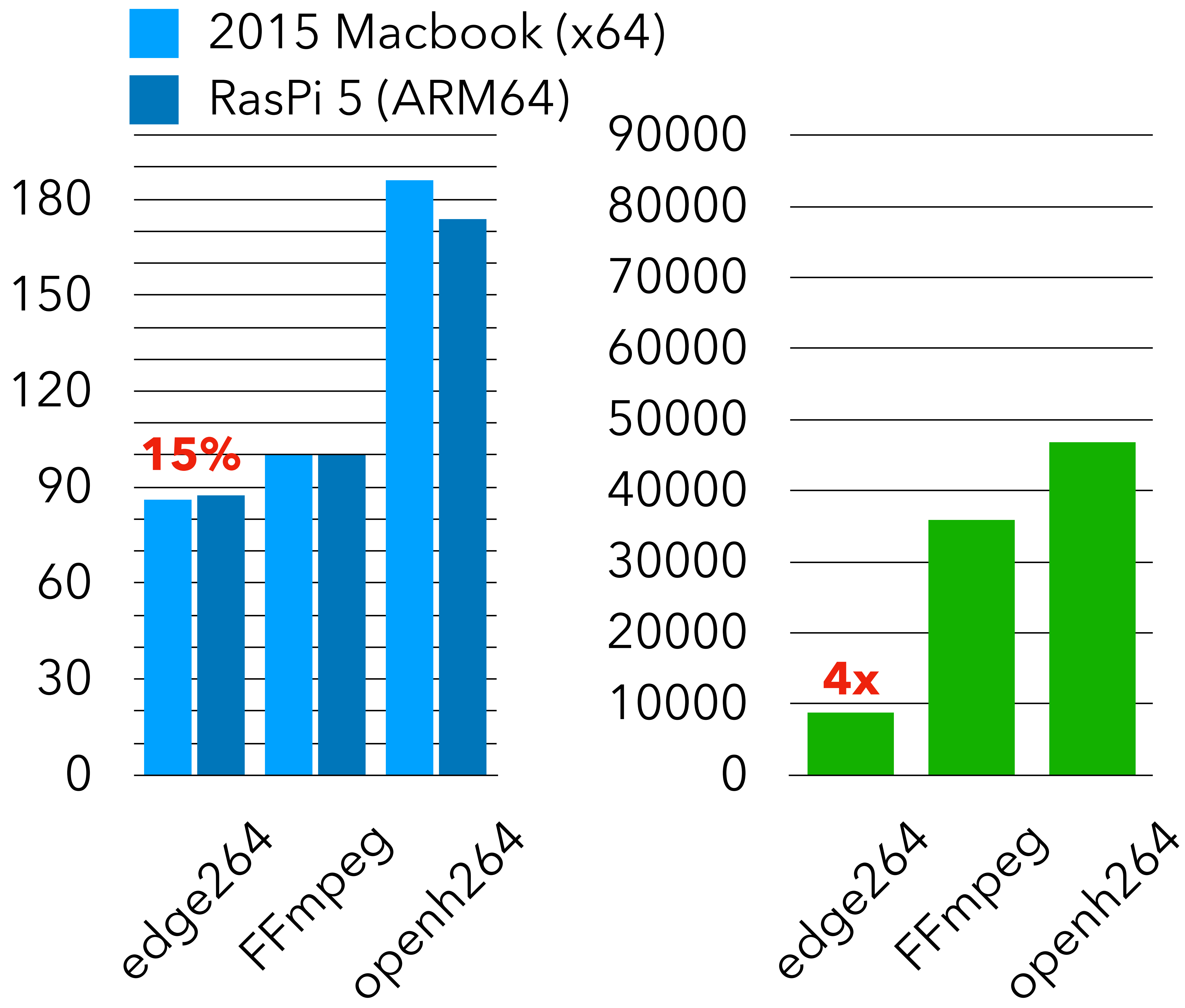
Software engineer at Quantum Surgical (Montpellier, France)

FOSDEM'26 – 31 January 2026

edge264

H.264/AVC decoder for Progressive High & MVC 3D profiles

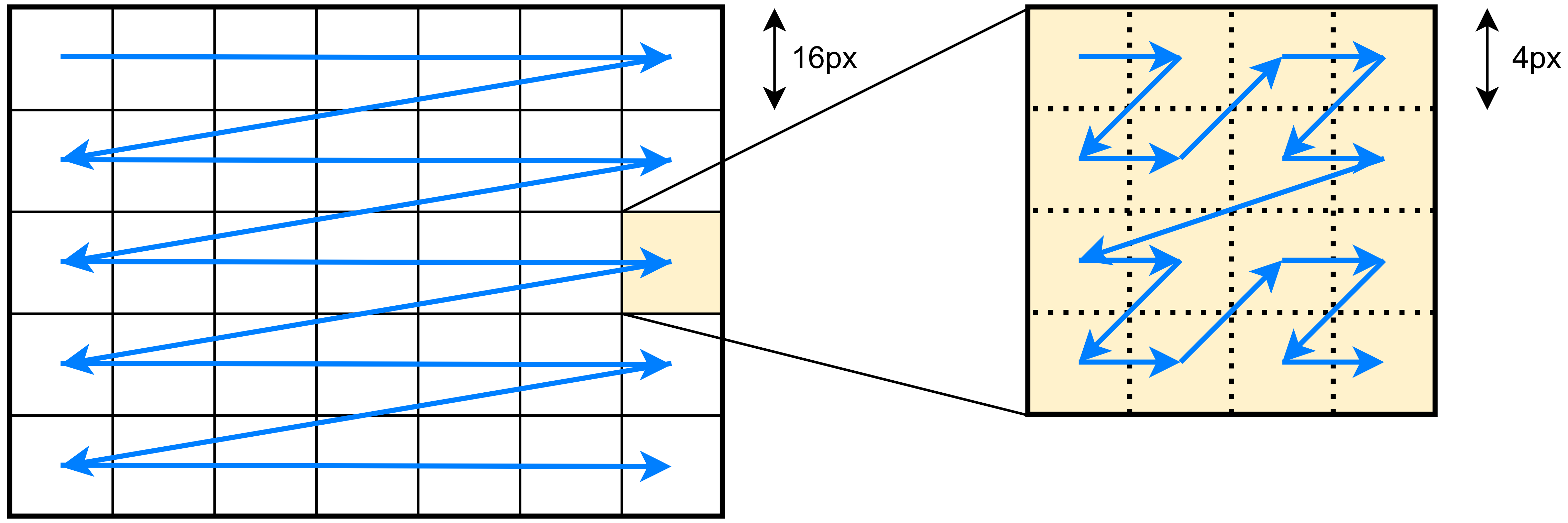
- BSD-3-Clause license
- x86/x64, **ARM32**/64, **WASM (in progress)**
- Linux, Windows, Mac
- **Prefetching**
- **CI/CD**
- **Custom bitstream encoder**
- **Stress testing (in progress)**
- **Netflix support**

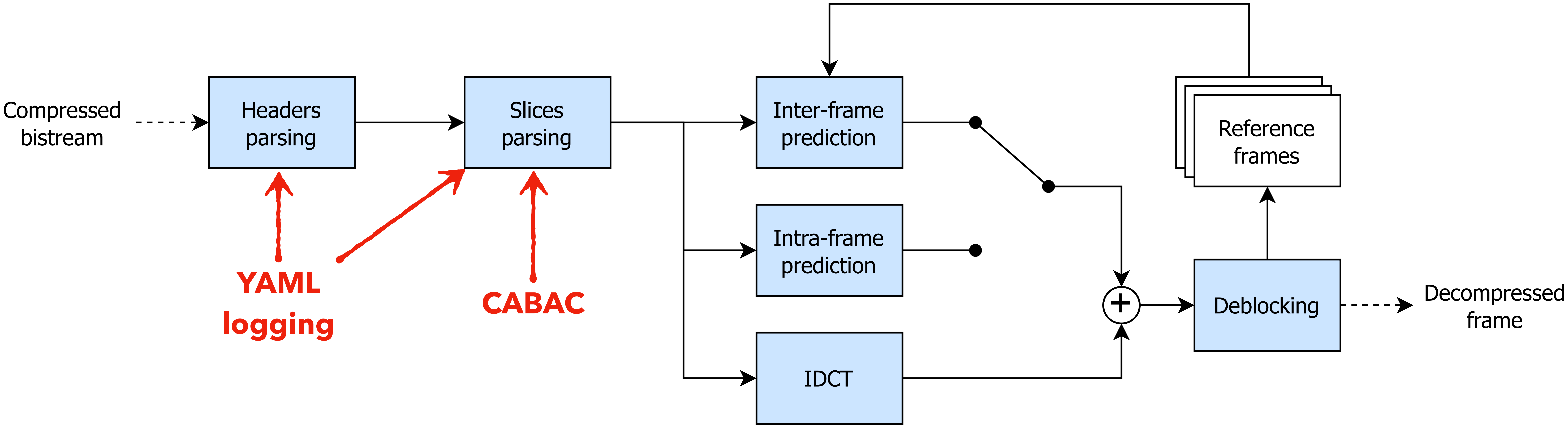


Speed (%)

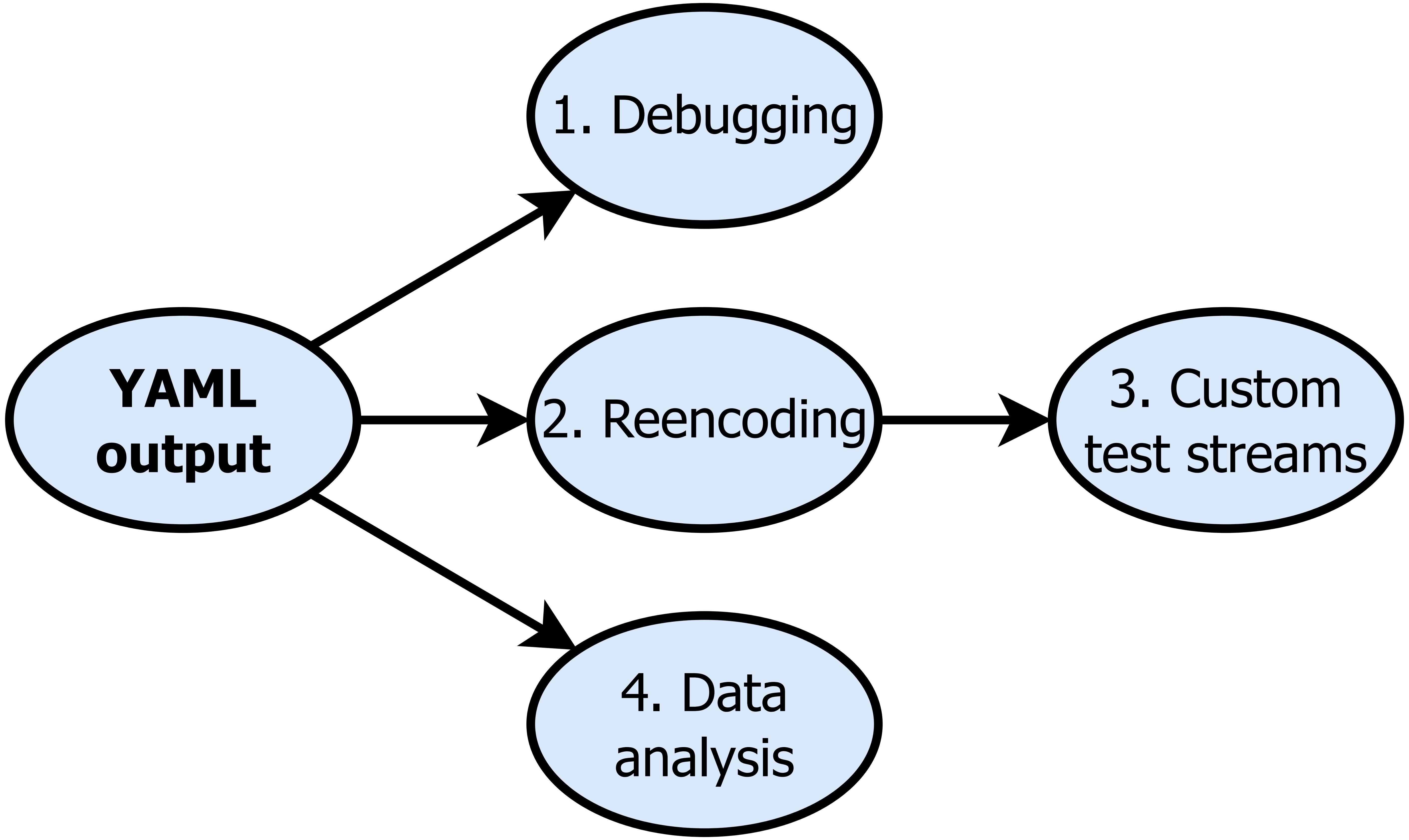
Code size (loc)

H.264/AVC decoder structure





1. YAML logging output

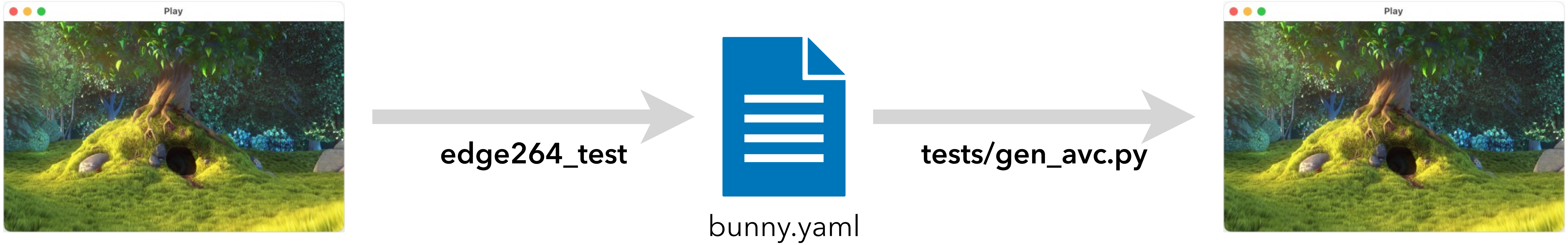


1-1. Debugging

```
- nal_ref_idc: 1
  nal_unit_type: 7 # Sequence parameter set
  profile_idc: 66 # Baseline
  constraint_set_flags: [1,1,1,0,0,0]
  level_idc: 1.2
  chroma_format_idc: 1 # 4:2:0 # inferred
  bit_depth: {luma: 8, chroma: 8} # inferred
  log2_max_frame_num: 16
  pic_order_cnt_type: 0
  log2_max_pic_order_cnt_lsb: 16
  max_num_ref_frames: 1
  gaps_in_frame_num_value_allowed_flag: 0
  pic_size_in_mbs: {width: 11, height: 9}
  frame_mbs_only_flag: 1
  direct_8x8_inference_flag: 1
  max_num_reorder_frames: 16 # inferred
  max_dec_frame_buffering: 16 # inferred
  decode_NAL_result: 0
```

- **Readability** by slim syntax (vs. XML/JSON), coloring, comments (vs. JSON)
- **Compactness** by recursive format (vs. CSV), inline structs
- **Scriptability** by design (vs. TXT, HTML)

1-2. Reencoding



bunny.264 **edge264_test** bunny.yaml **tests/gen_avc.py** bunny.264

Use a high-level language!

Include context from past parameter sets → low overhead, easier to reencode

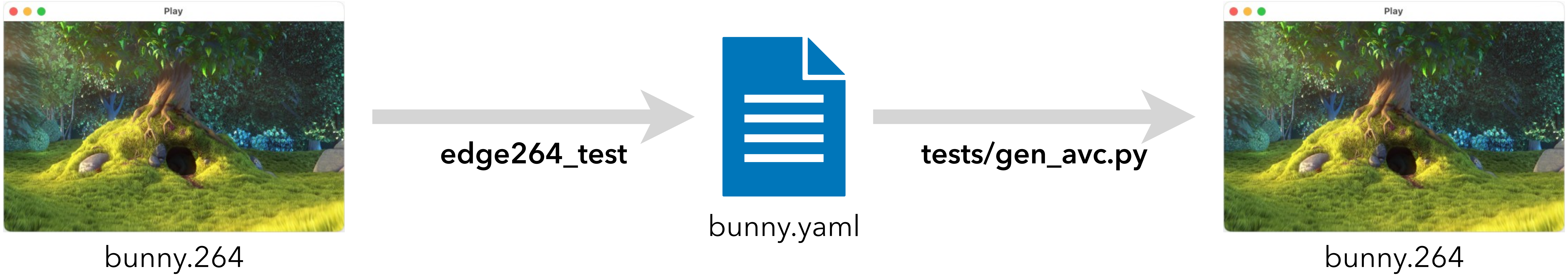
```
frame_num: 0
```

↓

```
frame_num: {bits: 4, value: 0}
```

- Versus lib h264bitstream: **9400** loc with C → **500** loc with Python

1-2. Reencoding



bunny.264

edge264_test

bunny.yaml

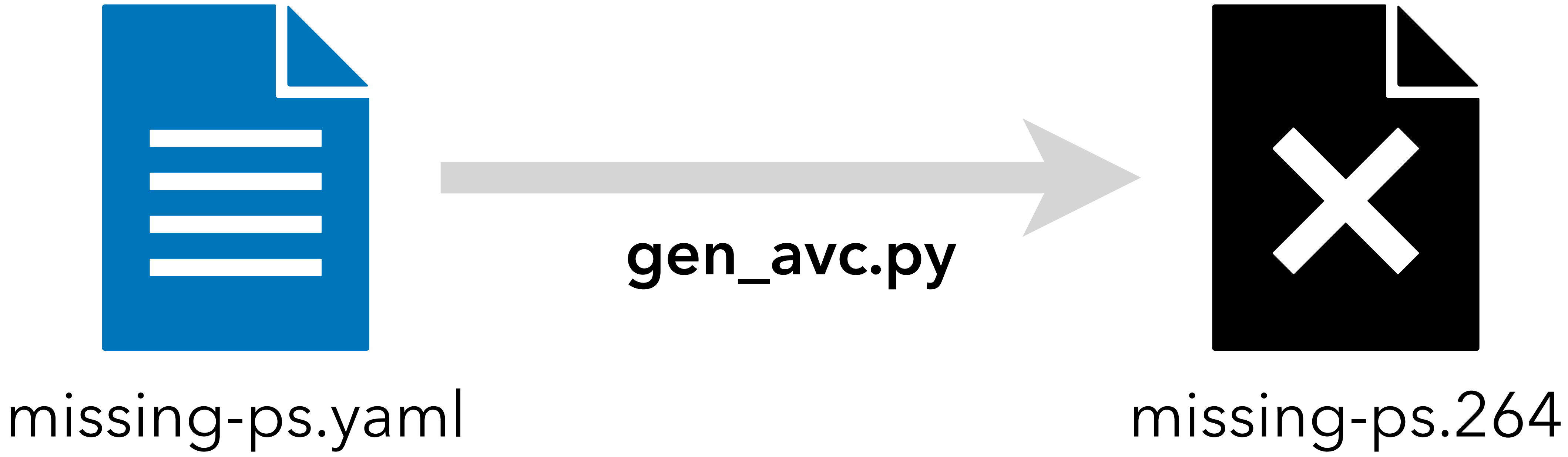
tests/gen_avc.py

bunny.264

Python **integers** = infinite big-endian **bit buffers** (with leading set bit)

- Initialize empty bit buffer `buf = 1`
- Insert n-bit value `buf = buf<<n | value`
- Get buffer size `nbits = buf.bit_length()-1`
- Remove leading bit & pad to bytes `buf = (buf ^ 1<<nbits) << (-nbits%8)`
- Write to file `f.write(buf.to_bytes((nbits+7)//8, byteorder="big"))`

1-3. Custom test streams

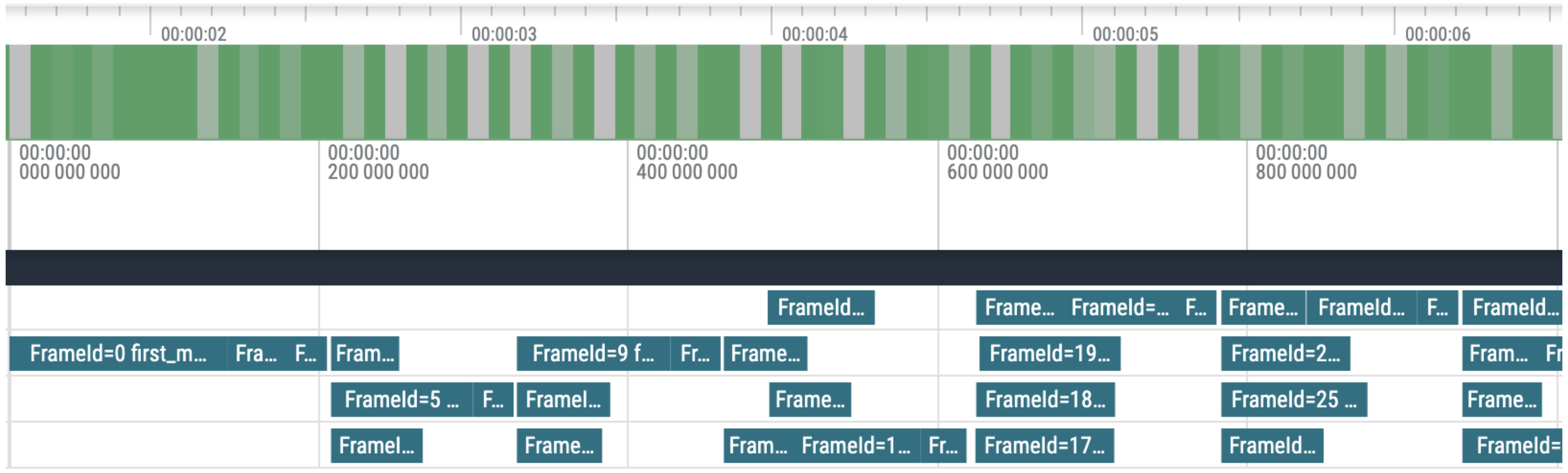


YAML encoder → create bitstreams for **stress testing**

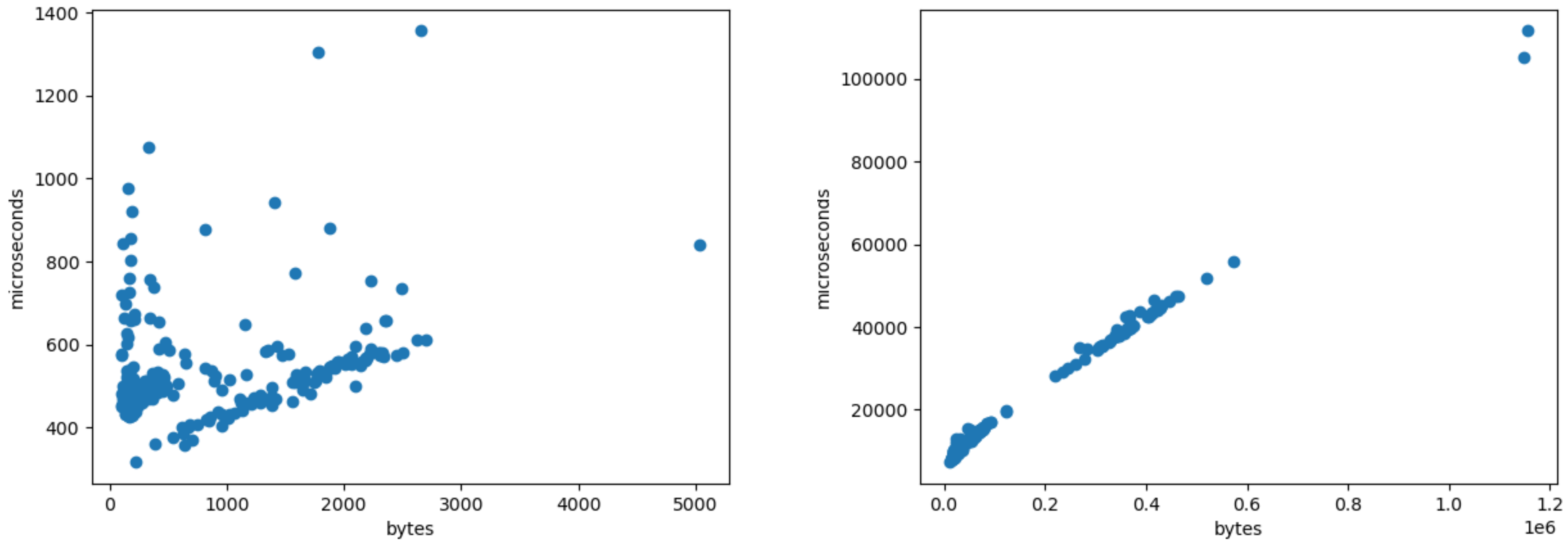
- Catch up after ffmpeg/openh264 on 10+ years of field testing
- Search for *shall* clauses in spec! (e.g. « *A sequence parameter set RBSP, with that particular value of seq_parameter_set_id, shall be available to the decoding process prior to its activation.* »)
- Can be included in public repository & CI/CD since I own them
- Much smaller than conformance bitstreams → much faster to run test suite

1-4. Data analysis

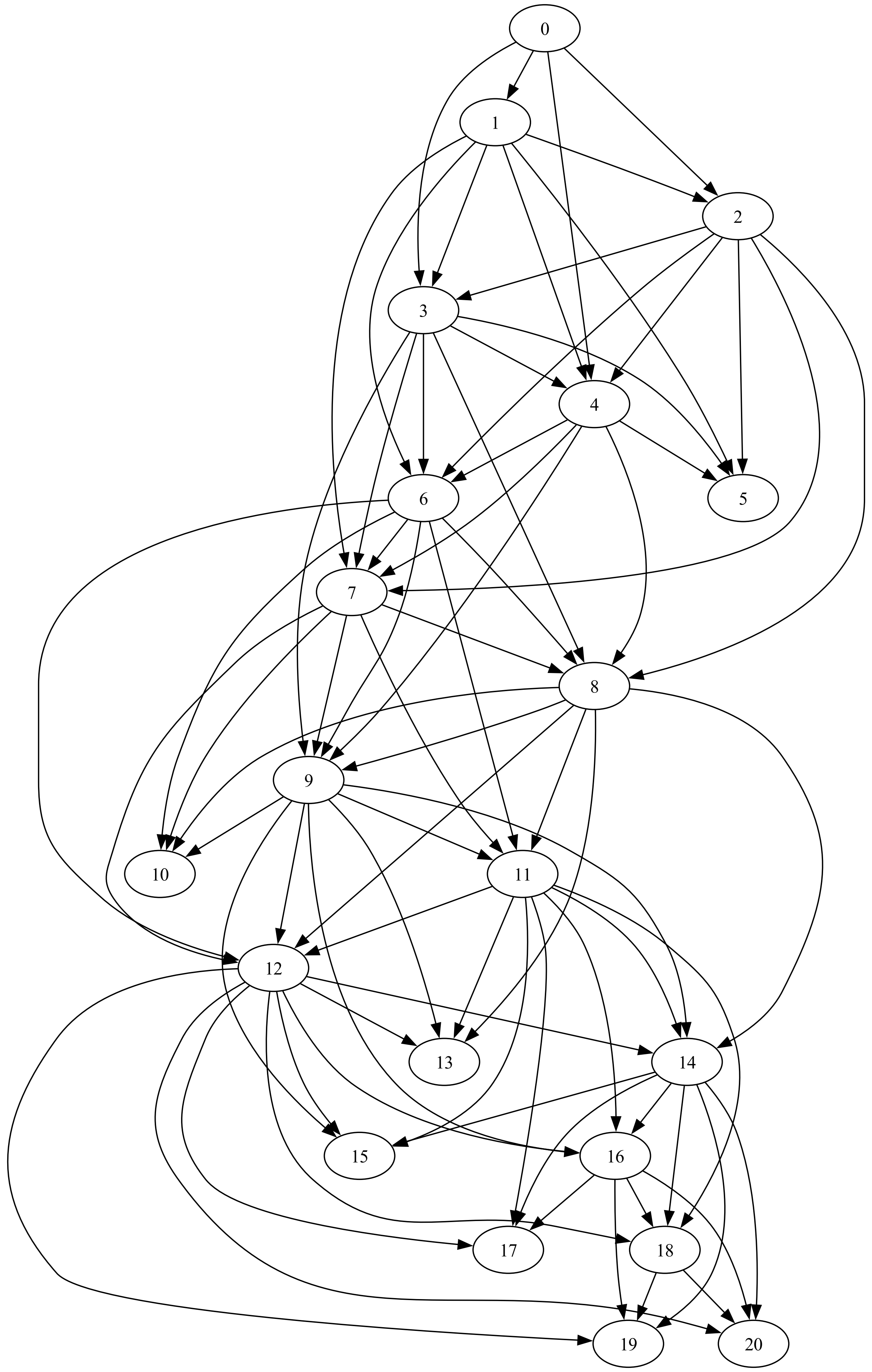
Current thread scheduling



Decoding time per size

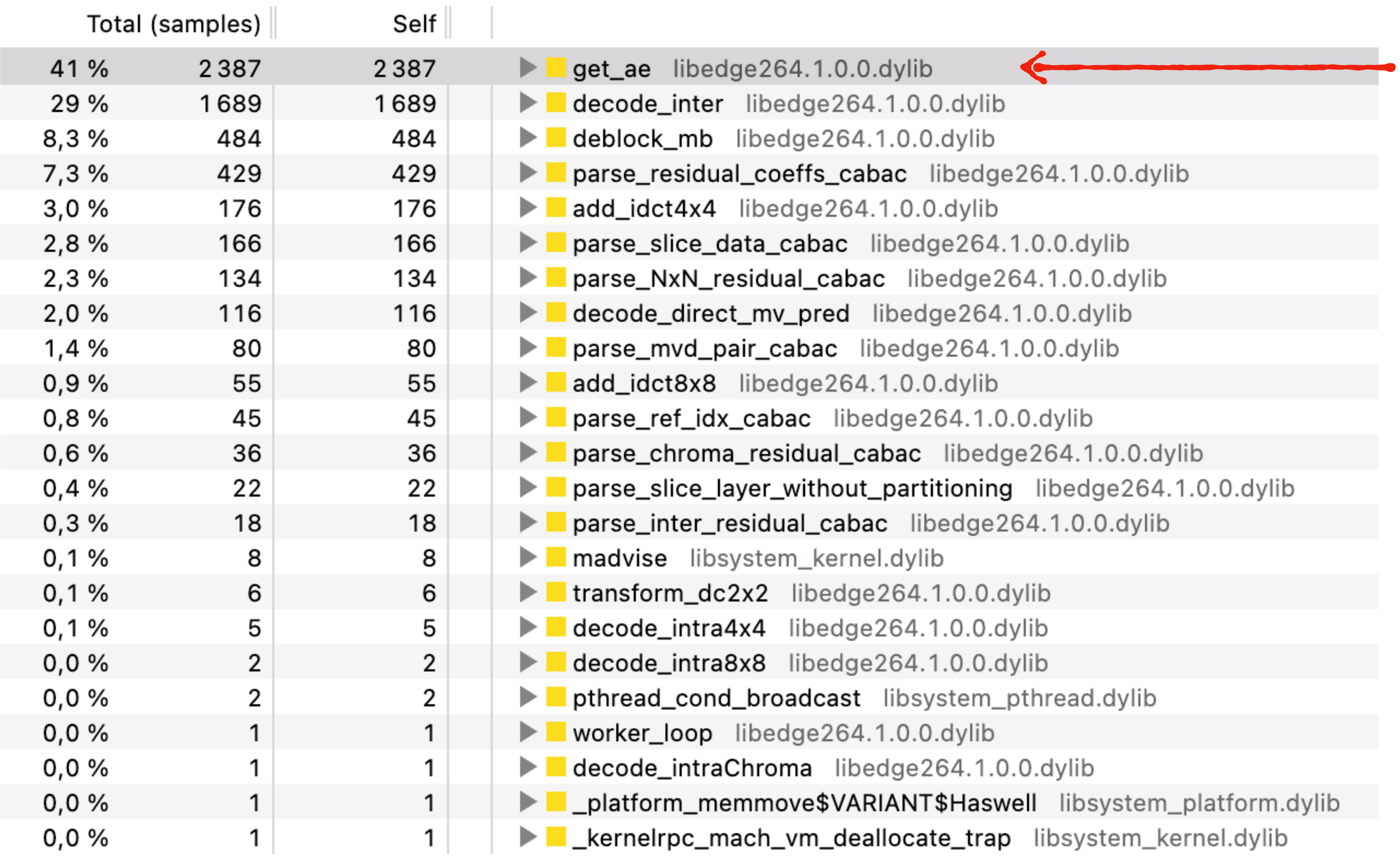


Frame dependencies



2. CABAC decoding

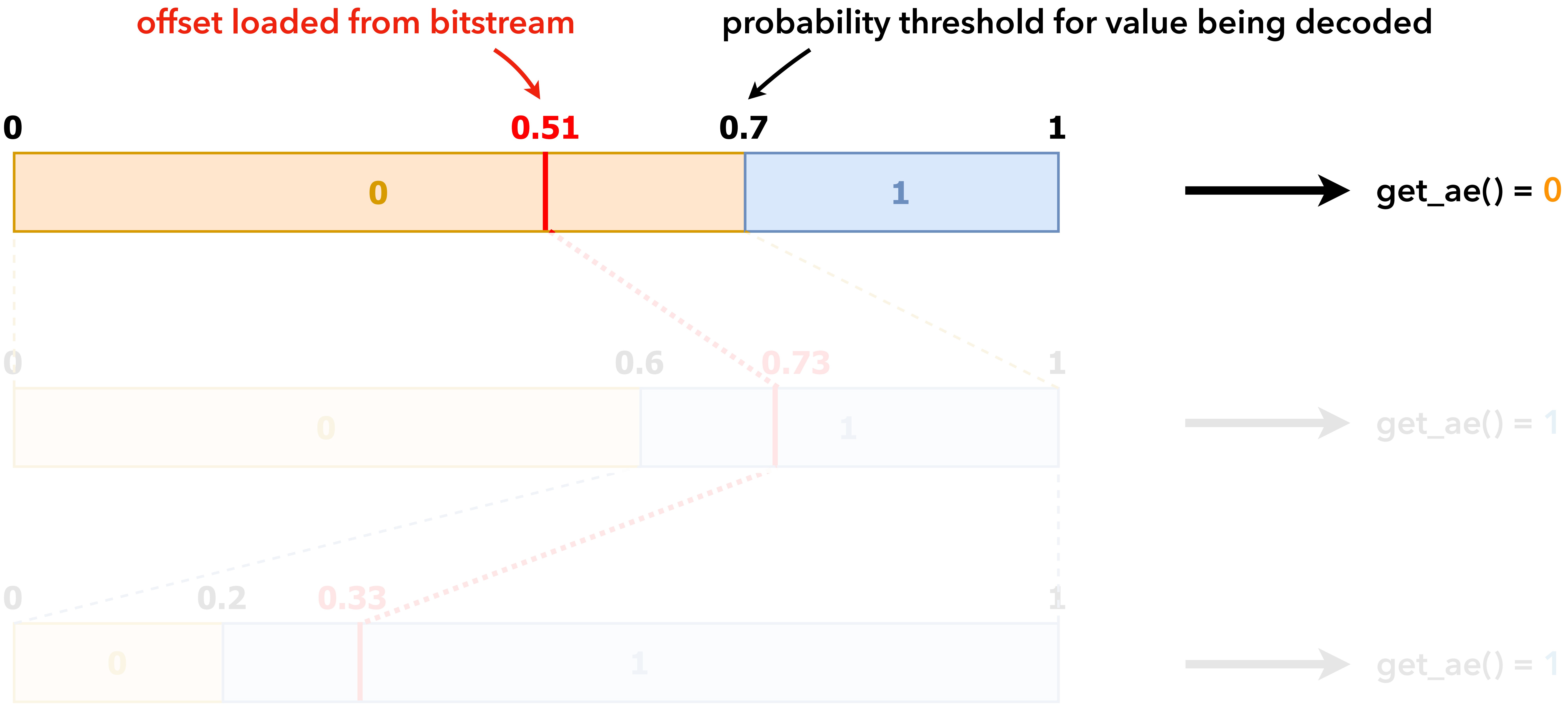
Screenshot of Firefox Profiler (collected with samply)



Total (samples)		Self		
41 %	2 387	2 387	get_ae	libedge264.1.0.0.dylib
29 %	1 689	1 689	decode_inter	libedge264.1.0.0.dylib
8,3 %	484	484	deblock_mb	libedge264.1.0.0.dylib
7,3 %	429	429	parse_residual_coeffs_cabac	libedge264.1.0.0.dylib
3,0 %	176	176	add_idct4x4	libedge264.1.0.0.dylib
2,8 %	166	166	parse_slice_data_cabac	libedge264.1.0.0.dylib
2,3 %	134	134	parse_NxN_residual_cabac	libedge264.1.0.0.dylib
2,0 %	116	116	decode_direct_mv_pred	libedge264.1.0.0.dylib
1,4 %	80	80	parse_mvd_pair_cabac	libedge264.1.0.0.dylib
0,9 %	55	55	add_idct8x8	libedge264.1.0.0.dylib
0,8 %	45	45	parse_ref_idx_cabac	libedge264.1.0.0.dylib
0,6 %	36	36	parse_chroma_residual_cabac	libedge264.1.0.0.dylib
0,4 %	22	22	parse_slice_layer_without_partitioning	libedge264.1.0.0.dylib
0,3 %	18	18	parse_inter_residual_cabac	libedge264.1.0.0.dylib
0,1 %	8	8	madvise	libsystem_kernel.dylib
0,1 %	6	6	transform_dc2x2	libedge264.1.0.0.dylib
0,1 %	5	5	decode_intra4x4	libedge264.1.0.0.dylib
0,0 %	2	2	decode_intra8x8	libedge264.1.0.0.dylib
0,0 %	2	2	pthread_cond_broadcast	libsystem_pthread.dylib
0,0 %	1	1	worker_loop	libedge264.1.0.0.dylib
0,0 %	1	1	decode_intraChroma	libedge264.1.0.0.dylib
0,0 %	1	1	_platform_memmove$VARIANT$Haswell	libsystem_platform.dylib
0,0 %	1	1	_kernelrpc_mach_vm_deallocate_trap	libsystem_kernel.dylib

get 1 bit from CABAC

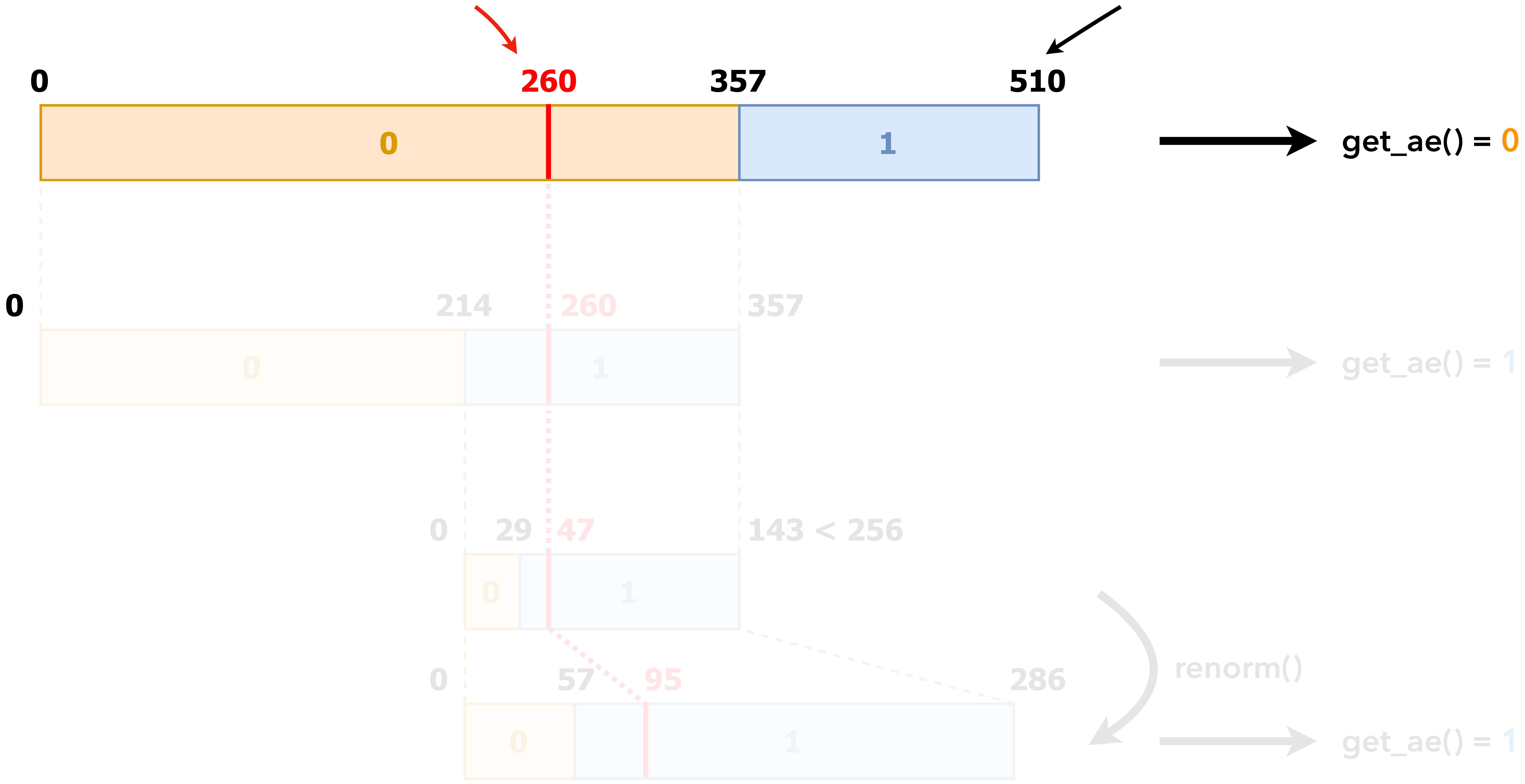
2. CABAC decoding


offset loaded from bitstream
probability threshold for value being decoded
0
0.51
0.7
1
0
1
get_ae() = 0
0
0.6
0.73
1
0
1
get_ae() = 1
0
0.2
0.33
1
0
1
get_ae() = 1

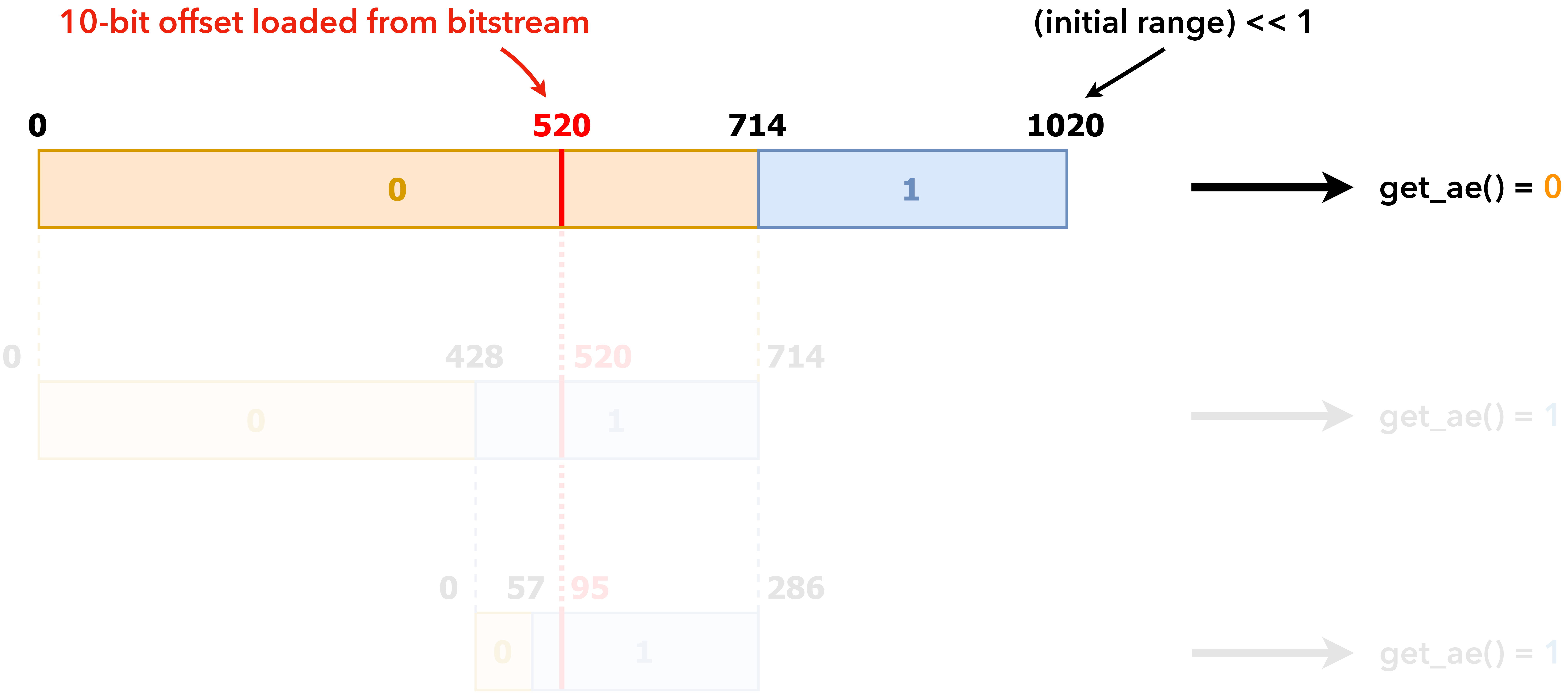
2. CABAC decoding

9-bit offset loaded from bitstream

9-bit range initialized at start



2. CABAC decoding



2-1. Extending CABAC state to size_t

Initially **load 64 bits in offset**, and **shift range up 55 bits** (64 - 9)

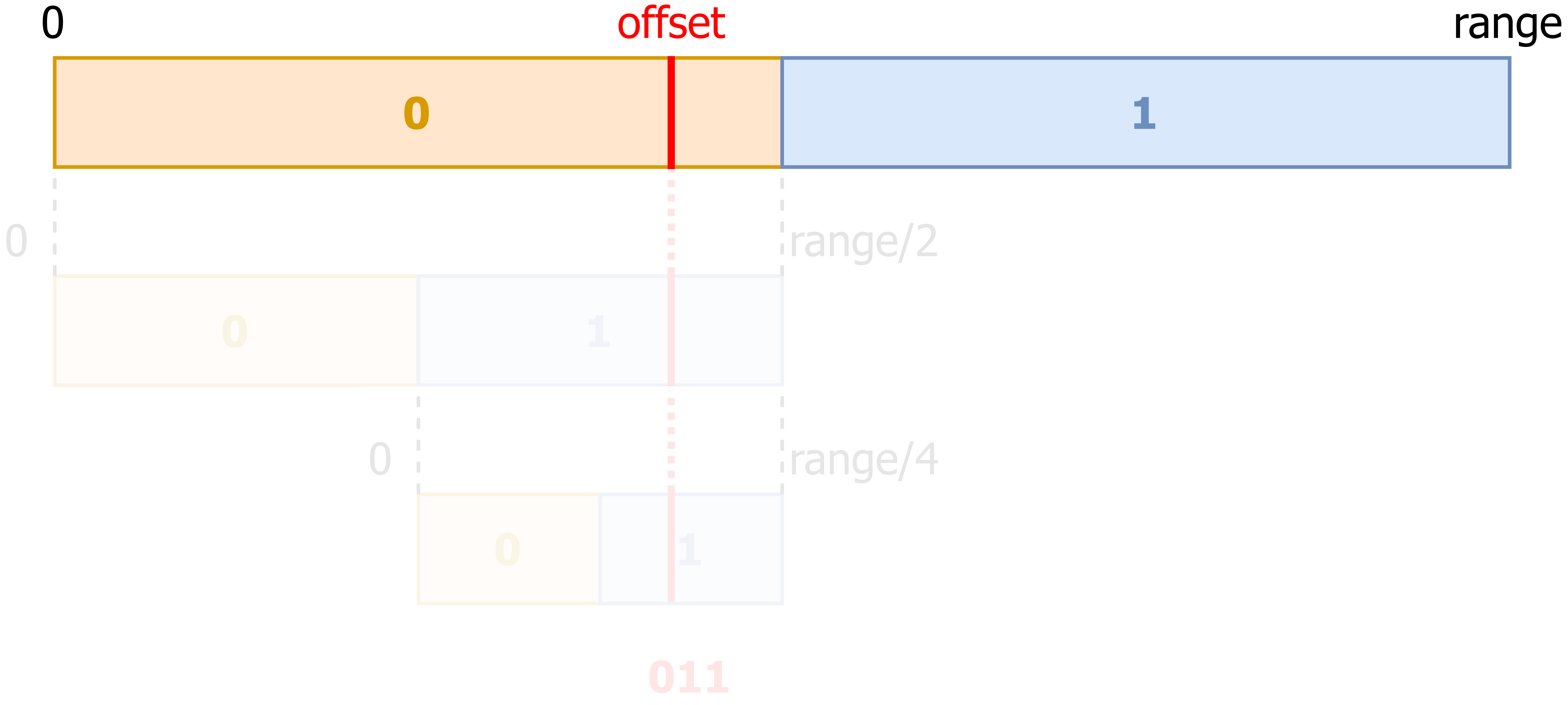
At each renormalization, offset & range are ≤ 8-bit, refill 56 bits (7 bytes) into offset & shift range up the same

- Trades less frequent renormalizations with new counting of extra bits
- Wide loads work well with on-the-fly unescaping (c.f. last year)

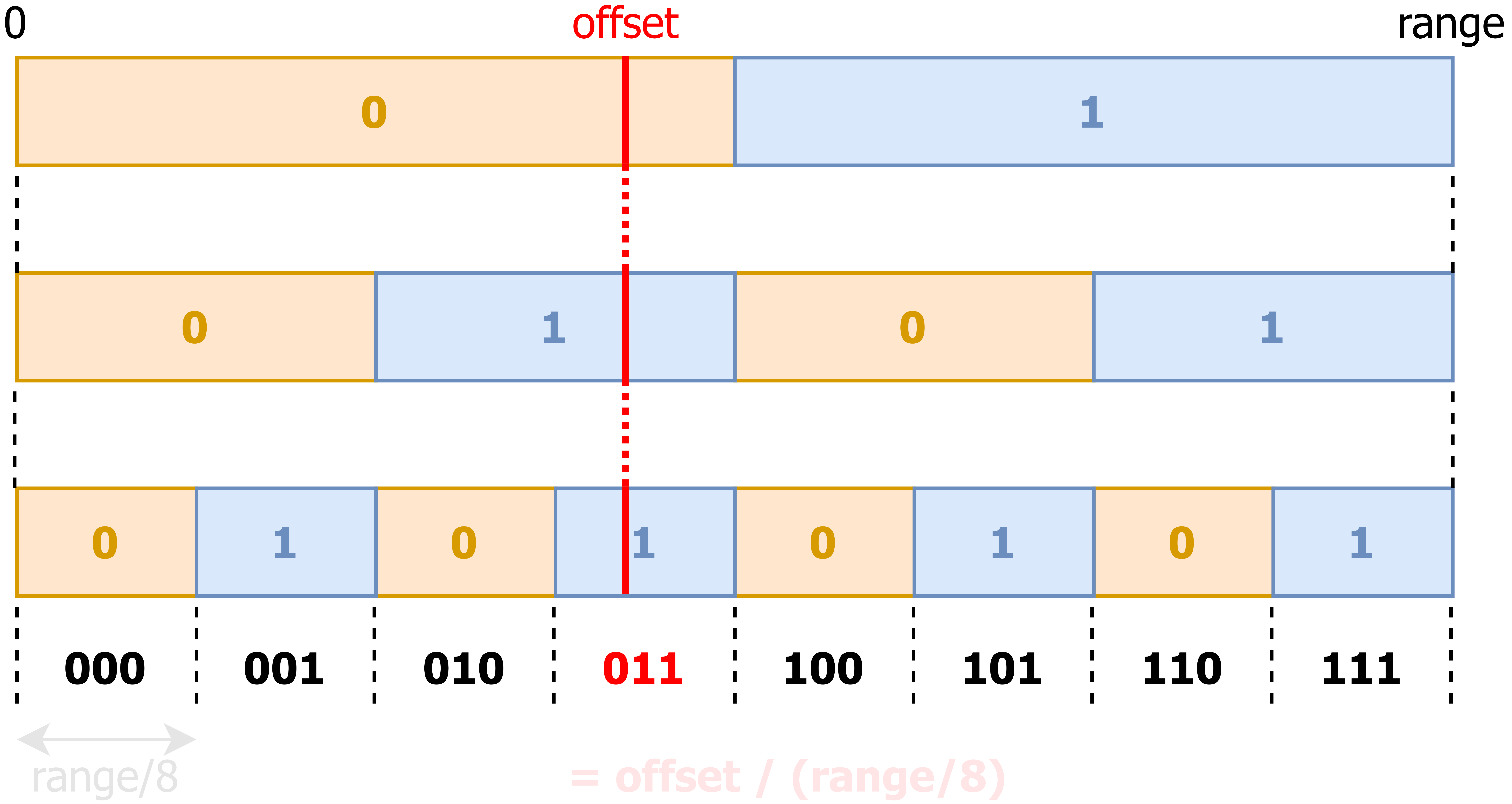
range ∈ [256;511] → range's 9th bit is set → **bit count = 64 - clz(range)**

- Allows keeping bit count without an extra variable or extra set bit (ffmpeg)

2-2. Batch-decoding CABAC bypass



2-2. Batch-decoding CABAC bypass



= offset / (range/8)

2-2. Batch-decoding CABAC bypass

To get N bypass bits (*with size_t format*):

- Ensure offset & range have ≥ N extra bits (renorm otherwise)
- Shift range down N bits
- **offset / range** → N bypass bits
- offset % range → new offset

To return M unconsumed bits:

- offset + range * unconsumed → new offset
- Shift range up M bits

Thank you for your attention!

https://github.com/tvlabs/edge264

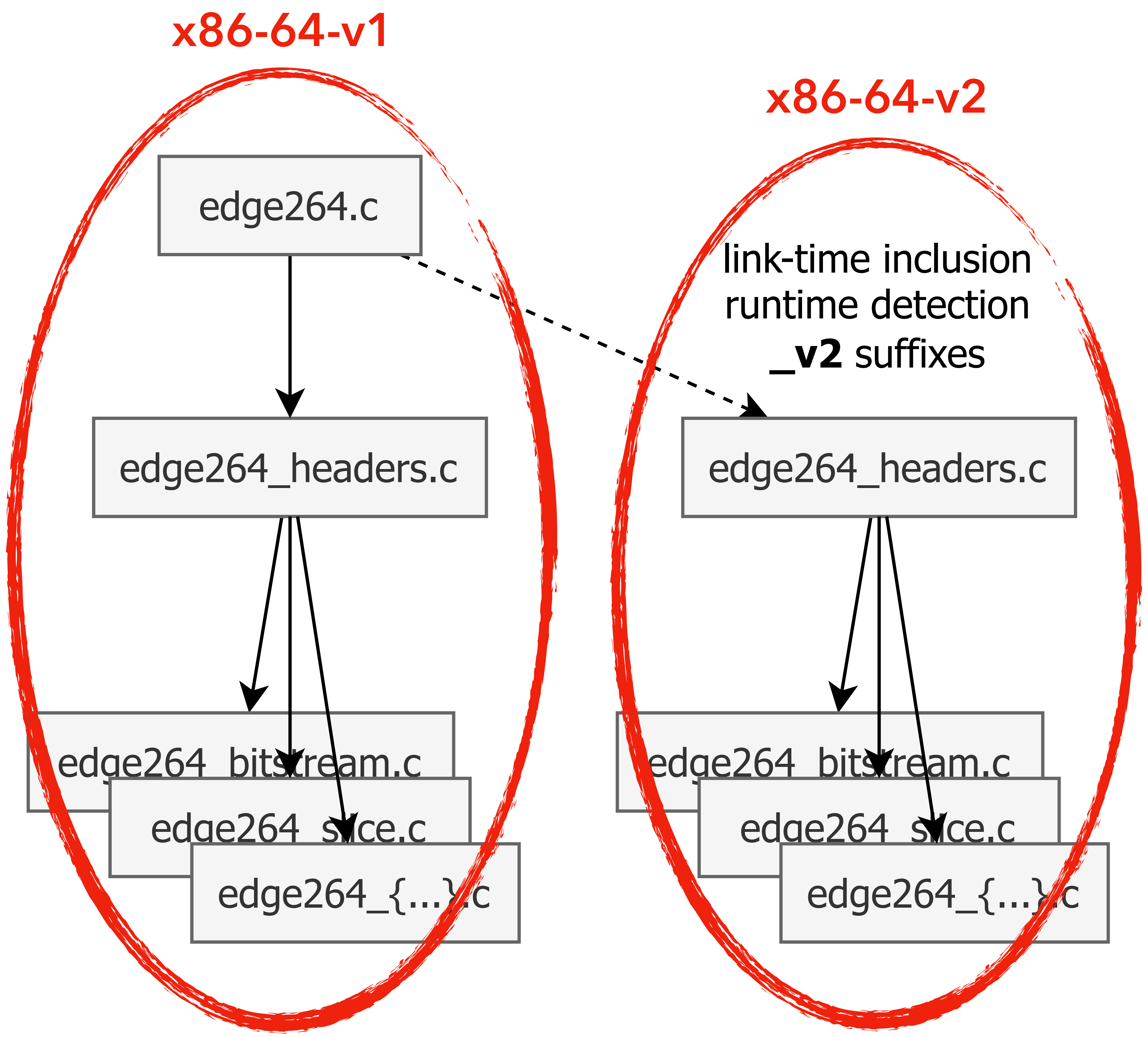
traf@ik.me

1. YAML logging output
   1. Decoding
   2. Reencoding (*Python* 🫶)
   3. Custom test streams (*shall* 🔎)
   4. Data analysis
2. CABAC decoding
   1. Extending state to size_t
   2. Batch-decoding bypass (*div* %)

1. x86-64 microarchitecture runtime variants

Compile entire lib for x86-64-v1, compile lib minus top-level functions for v2 and v3, then branch at runtime

- Easy `make` selection of optional variants
- Allows SIMD everywhere (except top-level functions)
- Used for logging support too


x86-64-v1
x86-64-v2
edge264.c
edge264_headers.c
edge264_bitstream.c
edge264_slice.c
edge264_{...}.c
link-time inclusion
runtime detection
_v2 suffixes
edge264_headers.c
edge264_bitstream.c
edge264_slice.c
edge264_{...}.c